ISTQB® Certified Tester
Lehrplan
Testen mit generativer KI
(CT-GenAI)

v1.0D

International Software Testing Qualifications Board

Deutschsprachige Ausgabe, zur Verfügung gestellt von
Austrian Testing Board, German Testing Board e. V. und
Swiss Testing Board

Übersetzung des englischsprachigen Lehrplans des International Software Testing Qualifications Board (ISTQB®), Originaltitel: Certified Tester Specialist Level – Testing with Generative AI (CT-GenAI) Syllabus, Version 1.0

Urheberschutzvermerk

Zur besseren Lesbarkeit wird im gesamten Dokument auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen verzichtet. Es wird das generische Maskulinum verwendet, wobei unterschiedliche Geschlechter gleichermaßen gemeint sind.

Änderungsübersicht der deutschsprachigen Ausgabe

Version	Datum	Anmerkungen
v1.0D	11.11.2025	CT-GenAI, Version 1.0D (Deutschsprachige Fassung der ISTQB Version 1.0)
v1.0	25.07.2025	CT-GenAI, Version 1.0 (Englische Originalversion)

Inhaltsverzeichnis

Danksagung

Das englischsprachige Dokument wurde am 25.07.2025 von der Generalversammlung des ISTQB® offiziell veröffentlicht.

Es wurde von einem Team des International Software Testing Qualifications Board erstellt: Abbas Ahmad (Produktverantwortlicher), Gualtiero Bazzana, Alessandro Collino, Olivier Denoo und Bruno Legeard (technischer Leiter).

Das Team dankt Anne Kramer, Jedrzej Kwapinski, Samuel Ouko und Ina Schieferdecker für ihr Technisches Review sowie dem Reviewteam und den nationalen Mitgliedsboards für ihre Vorschläge und Beiträge.

Die folgenden Personen haben am Review, an der Kommentierung und der Abstimmung dieses Lehrplans teilgenommen:

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 Ágnecz, Horst Pohlmann, Ina Schieferdecker, Ingvar Nordström, Jan Sabak, Jaroslaw Hryszko, Jedrzej 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 Medzihradszky, Ferenc Hamori, Giorgio Pisani, Giancarlo Tomasig, Young jae Choi, Arnika Hryszko, Andrei Brovko, Ilia kulakov, Praveen, Kostas Pashalidis, Ferdinand Gramsamer, A. Berfin Öztaş, Abdullah Gök, Abdurrahman Akin, Aleyna Zuhal Işik, 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 Yalçınkaya, Efe Can Yemez, Elif Cerav, Emine Tekiner, Emre Aman, Emre Can Akgül, Esra Kücük, Gençay Genç, Gül Çalışır Açan, Gül Nihal Singil, Güler Gök, Gulhanim Anulur, Hakan Güvez, Haktan Bilgehan Dilber, Halil Ibrahim Tasdemir, Hasan Küçükayar, Hatice Erdoğan, Hatice Kübra Daşdoğan, Hüseyin Sevki Ari, Hyulya Gyuler, İlknur Neşe Tuncal, Kaan Eminğlu, 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.

Als Reviewer der lokalisierten deutschsprachigen Fassung waren aktiv:

Jürgen Beniermann, Raphael Dumhart, Jörn Münzel, Ina Schieferdecker, Georg Sehl

Unser herzlicher Dank geht an Ursula Zimpfer für ihre wertvolle Unterstützung bei der Bearbeitung der deutschsprachigen Fassung des vorliegenden Lehrplans.

0 Einleitung

0.1 Zweck des Lehrplans Testen mit generativer KI

Dieser Lehrplan bildet die Grundlage der Qualifikation Testen mit generativer KI (CT-GenAI) der Aufbaustufe „Specialist“ im Softwaretest-Qualifizierungsprogramm des International Software Testing Qualifications Board (im Folgenden ISTQB® genannt). Das German Testing Board e. V. (im Folgenden GTB genannt) hat diesen Lehrplan in Zusammenarbeit mit dem Austrian Testing Board (ATB) und dem Swiss Testing Board (STB) in die deutsche Sprache übersetzt. Das ISTQB® stellt den Lehrplan folgenden Adressaten zur Verfügung:

1. Nationalen Mitgliedsboards, die den Lehrplan in ihre Sprache(n) übersetzen und Schulungsanbieter akkreditieren dürfen. Die nationalen Mitgliedsboards dürfen den Lehrplan an die Anforderungen ihrer nationalen Sprache anpassen und Referenzen hinsichtlich lokaler Veröffentlichungen berücksichtigen.
2. Zertifizierungsstellen zur Ableitung von Prüfungsfragen in ihrer nationalen Sprache, die an die Lernziele dieses Lehrplans angepasst sind.
3. An die Ausbildungsanbieter, um Kursunterlagen zu erstellen und geeignete Lehrmethoden festzulegen.
4. Für Kandidaten der Zertifizierung, um sich auf die Prüfung der Zertifizierung vorzubereiten (entweder als Teil eines Schulungskurses oder unabhängig davon).
5. Für die internationale Software- und Systemtechnik-Community zur Förderung des Berufsbildes des Software- und Systemtesters und als Grundlage für Bücher und Artikel.

0.2 Testen mit generativer KI

Die Qualifikation Testen mit generativer KI richtet sich an alle, die mit generativer KI-Technologie (GenAI) für Softwaretests zu tun haben. Dazu gehören Personen in Funktionen wie Tester, Testanalyst, Testautomatisierungsentwickler, Testmanager, Abnahmetester und Softwareentwickler. Die Qualifikation Testen mit generativer KI eignet sich auch für alle, die ein grundlegendes Verständnis für den Einsatz von GenAI für Softwaretests erwerben möchten, wie Projektmanager, Qualitätsmanager, Softwareentwicklungsmanager, Businessanalysten, IT-Direktoren und Unternehmensberater.

0.3 Karriereweg für Tester

Das ISTQB®-Schema unterstützt Testexperten in allen Stufen ihrer Karriere und bietet ihnen sowohl eine breite als auch eine tiefe Wissensbasis. Personen, die die ISTQB®-Zertifizierung Testen mit generativer KI erwerben, könnten auch an den Core Advanced Levels (Testanalyst, Technischer Testanalyst, Testmanager und Testingenieur) und anschließend am Expert Level (Testmanagement oder Testprozessverbesserung) interessiert sein. Wer Fähigkeiten in der Testtätigkeit in einer agilen Umgebung entwickeln möchte, könnte die Zertifizierung Agile Test Leadership at Scale (ATLaS) in Betracht ziehen. Die Aufbaustufe Specialist bietet einen tiefen Einblick in Bereiche, die spezifische Testansätze und Testaktivitäten beinhalten (z. B. KI-Tests oder Testen mobiler Apps) oder die das Test-Know-how für bestimmte Branchendomänen bündeln (z. B. Automotive oder Gaming).

Aktuelle Informationen über das ISTQB®-Certified-Tester-Schema finden Sie unter www.istqb.org oder auf den Seiten der nationalen Boards, wie z. B. https://www.gtb.de (Deutschland), https://www.austriantestingboard.at (Österreich) oder www.swisstestingboard.org (Schweiz).

0.4 Geschäftlicher Nutzen

In diesem Abschnitt werden die fünf geschäftlichen Nutzen (Business Outcomes, BO) aufgeführt, die von Kandidaten erwartet werden, welche die Zertifizierung Certified Tester „Testen mit generativer KI“ erworben haben.

Kandidaten, die die Zertifizierung „Testen mit generativer KI“ erworben haben, können Folgendes:

GenAI-BO1	Die grundlegenden Konzepte, Fähigkeiten und Grenzen von generativer KI verstehen.
GenAI-BO2	Praktische Fähigkeiten zur Verwendung von Large Language-Modellen für den Softwaretest entwickeln.
GenAI-BO3	Einblicke in die Risiken und Abhilfemaßnahmen beim Einsatz von generativer KI für den Softwaretest gewinnen.
GenAI-BO4	Einblicke in die Anwendungsmöglichkeiten von Lösungen mit generativer KI für den Softwaretest gewinnen.
GenAI-BO5	Effektiv zur Definition und Umsetzung einer generativen KI-Strategie und -Roadmap für den Softwaretest innerhalb eines Unternehmens beitragen.

0.5 Prüfbare Lernziele, Ziele der praktischen Übungen und kognitive Wissensstufen

Lernziele (*learning objectives*, LO) und Ziele der praktischen Übungen (*hands-on objectives*, HO) unterstützen den geschäftlichen Nutzen und dienen zur Ausarbeitung der Prüfungen für die Zertifizierung Certified Tester „Testen mit generativer KI“.

Im Allgemeinen sind alle Inhalte dieses Lehrplans auf den Stufen K1, K2 und K3 (siehe unten) prüfbar, mit Ausnahme der Einleitung, der Ziele der praktischen Übungen und der Anhänge. Die Prüfungsfragen überprüfen die Kenntnisse der Schlüsselbegriffe auf der Stufe K1 oder die Lernziele auf allen K-Stufen.

Die spezifischen Lernziele werden zu Beginn jedes Kapitels aufgeführt und wie folgt klassifiziert:

- K1: Erinnern
- K2: Verstehen
- K3: Anwenden

Weitere Details und Beispiele für Lernziele finden Sie in Anhang A – Lernziele/Kognitive Wissensstufen.

Alle Begriffe, die als Schlüsselbegriffe direkt unter den Kapitelüberschriften aufgeführt sind, müssen bekannt sein (K1), auch wenn sie nicht ausdrücklich in den Lernzielen erwähnt werden.

Die spezifischen Ziele der praktischen Übungen (HO) werden am Anfang jedes Kapitels aufgeführt. Jedes HO ist mit einem LO der Stufe K2 oder K3 verknüpft, mit dem Ziel, das Lernen durch praktische Übungen zu verbessern. Die Stufe eines HO-Ziels wird wie folgt klassifiziert:

- H0: Dies kann eine Live-Demonstration einer Übung oder ein aufgezeichnetes Video umfassen. Da die Übung nicht von den Lernenden durchgeführt wird, handelt es sich nicht um eine Übung im engeren Sinne.
- H1: Angeleitete Übung. Die Lernenden folgen einer Abfolge von Schritten, die von der Lehrperson durchgeführt werden.

- H2: Übung mit Hinweisen. Die Lernenden erhalten eine Aufgabe mit entsprechenden Hinweisen, sodass die Aufgabe innerhalb des vorgegebenen Zeitrahmens gelöst werden kann

0.6 Die Zertifizierungsprüfung für Testen mit generativer KI

Die Zertifizierungsprüfung für Testen mit generativer KI basiert auf diesem Lehrplan. Die Beantwortung der Prüfungsfragen kann die Nutzung von Inhalten aus mehr als einem Abschnitt dieses Lehrplans erfordern. Alle Abschnitte des Lehrplans sind prüfungsrelevant, mit Ausnahme der Einführung, der Ziele der praktischen Übungen und der Anhänge. Standards, Bücher und Artikel sind als Referenzen genannt (Kapitel 6), ihr Inhalt ist jedoch nicht prüfungsrelevant, abgesehen von dem, was im Lehrplan selbst aus diesen Standards und Büchern zusammengefasst ist.

Weitere Einzelheiten entnehmen Sie bitte dem Dokument “Exam Structures and Rules Compatible with Syllabus Foundation and Advanced Levels and Specialists Modules”.

Hinweis zur Zulassungsvoraussetzung: Das ISTQB®-Zertifikat Certified Tester Foundation Level muss vor der Prüfung zum ISTQB® Certified Tester – Testen mit generativer KI erworben werden.

0.7 Akkreditierung

Ein nationales ISTQB®-Mitgliedsboard kann Schulungsanbieter akkreditieren, deren Lehrmaterial diesem Lehrplan entspricht. Die Akkreditierungsrichtlinien können bei diesem nationalen Board (in Deutschland: German Testing Board e. V.; in der Schweiz: Swiss Testing Board; in Österreich: Austrian Testing Board) oder bei einer der Organisationen bezogen werden, die die Akkreditierung im Auftrag des nationalen Boards durchführt. Eine akkreditierte Schulung ist als konform mit diesem Lehrplan anerkannt und darf eine ISTQB®-Prüfung als Teil der Schulung enthalten.

Die Akkreditierungsrichtlinien für diesen Lehrplan folgen den allgemeinen Akkreditierungsrichtlinien, die von der ISTQB®-Arbeitsgruppe "Processes Management and Compliance" veröffentlicht wurden.

0.8 Umgang mit Normen und Standards

Es gibt Standards, die sich auf Qualitätsmerkmale und den Softwaretest beziehen, u. a. die im Foundation-Level-Lehrplan referenzierten (z. B. ISO, IEC und IEEE). Auch im Lehrplan für Testen mit generativer KI wird auf einige Normen und Standards verwiesen. Diese Verweise dienen als Rahmen oder als Quelle für zusätzliche Informationen, falls der Leser dies wünscht. Beachten Sie, dass der Lehrplan diese Normen und Standards als Referenzen verwendet. Die Inhalte der Normen und Standards sind nicht prüfungsrelevant. Weitere Informationen über Normen und Standards sind in Kapitel 6 nachlesbar.

0.9 Detaillierungsgrad

Der Detaillierungsgrad dieses Lehrplans erlaubt international einheitliche Schulungen und Prüfungen. Um dieses Ziel zu erreichen, enthält der Lehrplan:

- Allgemeine Lernziele, welche die Intention der ISTQB®-Zertifizierung Testen mit generativer KI beschreiben.
- Eine Liste von Schlüsselbegriffen, an die sich die Lernenden erinnern müssen.
- Lernziele für jeden Wissensbereich, die die zu erreichenden kognitiven Lernergebnisse beschreiben.
- Eine Beschreibung der Schlüsselkonzepte einschließlich Verweisen auf Quellen wie anerkannte Literatur oder Standards.

- Für jedes praktische Ziel eine Beschreibung der empfohlenen Vorgehensweise zur Unterstützung des Lernens.

Der Lehrplaninhalt ist keine Beschreibung des gesamten Wissensbereichs für das Testen mit generativer KI. Er spiegelt den Detaillierungsgrad wider, der in den Schulungen zum ISTQB® Certified Tester – Testen mit generativer KI behandelt wird. Der Schwerpunkt liegt auf Konzepten und Verfahren, die bei der Verwendung generativer KI für das Testen auf alle Softwareprojekte angewendet werden können.

Der Lehrplan verwendet die Terminologie (d. h. die Bezeichnungen und Bedeutungen) der Begriffe, die gemäß dem Glossar des ISTQB® im Bereich Testen und Qualitätssicherung von Software verwendet werden.

0.10 Aufbau des Lehrplans

Der Lehrplan umfasst fünf Kapitel mit prüfungsrelevantem Inhalt. Die Hauptüberschrift eines jeden Kapitels gibt die Schulungszeit für das Kapitel an. Für die Unterkapitel wird keine Zeitangabe gemacht. Für akkreditierte Schulungen fordert der Lehrplan mindestens 13,6 Unterrichtsstunden (815 Minuten), die sich wie folgt auf die fünf Kapitel verteilen:

- Kapitel 1: Einführung in generative KI für den Softwaretest – 100 Minuten
  - Der Tester lernt die Grundlagen von Large Language-Modellen (LLM), einschließlich Tokenisierung und multimodaler Fähigkeiten.
  - Der Tester untersucht Anwendungen generativer KI (GenAI) im Bereich Softwaretest, unterscheidet zwischen KI-Chatbots und LLM-gestützten Testwerkzeugen und experimentiert mit Tokenisierung, Kontextfenstern und multimodalen Prompts.
- Kapitel 2: Prompt-Engineering für effektives Softwaretesten – 365 Minuten
  - Der Tester lernt, effektive, strukturierte Prompts für GenAI im Testen zu erstellen.
  - Der Tester sammelt praktische Erfahrungen mit Verfahren des Prompt-Engineering für Software-Testaufgaben und wendet diese an.
- Kapitel 3: Management von Risiken bei generativer KI im Softwaretest – 160 Minuten
  - Der Tester lernt beim Testen mit GenAI, Halluzinationen, Reasoning-Fehler und Verzerrungen zu identifizieren und zu mindern.
  - Der Tester lernt, Datenschutz- und Sicherheitsprobleme von GenAI im Softwaretest zu adressieren.
  - Der Tester lernt den Energieverbrauch und die Umweltauswirkungen von GenAI im Softwaretest kennen.
  - Der Tester lernt KI-Vorschriften, Standards und Best Practices für den ethischen, transparenten und sicheren Einsatz von GenAI im Softwaretest kennen.
- Kapitel 4: LLM-gestützte Testinfrastruktur für den Softwaretest – 110 Minuten
  - Der Tester untersucht GenAI-Architekturen wie Retrieval-augmentierte Generierung (RAG) und GenAI-Agenten.
  - Der Tester lernt den Prozess zum Fein-Tuning von LLMs für Software-Testaufgaben kennen.
  - Der Tester lernt Konzepte von Large-Language-Model-Operations (LLMOps) für die Bereitstellung und Verwaltung von LLMs im Testen kennen.
- Kapitel 5: Bereitstellung und Integration generativer KI in Testorganisationen – 80 Minuten

- Der Tester lernt eine strukturierte Roadmap für die Integration von GenAI in Testprozesse kennen.
- Der Tester lernt die organisatorische Transformation für die Integration von GenAI in Testprozesse kennen.

1 Einführung in generative KI für den Softwaretest – 100 Minuten

Schlüsselbegriffe

Keine

Spezifische Schlüsselbegriffe für generative KI

Deep Learning, Einbettung, Foundation-LLM, generative KI, generativer vortrainierter Transformer, Instruction-Tuned-LLM, KI-Chatbot, Kontextfenster, Large Language-Modell (LLM), maschinelles Lernen, Merkmal, multimodales Modell, Reasoning-LLM, symbolische KI, Tokenisierung, Transformer

Lernziele und Ziele der praktischen Übungen für Kapitel 1: Die Lernenden können …

1.1 Grundlagen und Schlüsselkonzepte generativer KI

GenAI-1.1.1 (K1) … verschiedene Arten von KI unterscheiden: symbolische KI, klassisches maschinelles Lernen, Deep Learning und generative KI.

GenAI-1.1.2 (K2) … die Grundlagen von generativer KI und von Large Language-Modellen erläutern.

HO-1.1.2 (H1) … angeleitet die Tokenisierung und die Evaluierung der Tokenanzahl bei der Verwendung eines LLM für eine Software-Testaufgabe ausführen.

GenAI-1.1.3 (K2) … zwischen Foundation-, Instruction-Tuned- und Reasoning-LLMs unterscheiden.

GenAI-1.1.4 (K2) … Grundprinzipien von multimodalen LLMs und von Vision-Language-Modellen zusammenfassen.

HO-1.1.4 (H1) … angeleitet einen Prompt für ein multimodales LLM schreiben und ausführen, der sowohl Text- als auch Bildeingaben für eine Software-Testaufgabe verwendet.

1.2 Nutzung generativer KI im Softwaretest: Grundprinzipien

GenAI-1.2.1 (K2) … Beispiele für wichtige Funktionen von LLMs für Testaufgaben nennen.

GenAI-1.2.2 (K2) … Interaktionsmodelle bei der Verwendung von GenAI für Softwaretests vergleichen.

1.1 Grundlagen und Schlüsselkonzepte generativer KI

Generative künstliche Intelligenz (GenAI) ist ein Zweig der KI, der große, vortrainierte Modelle verwendet, um Ergebnisse wie Text, Bilder oder Code zu generieren, die von Menschen erstellten Ergebnissen ähneln. Large-Language-Modelle (LLM) sind GenAI-Modelle, die mit großen Textdatensätzen vortrainiert sind, sodass sie den Kontext bestimmen und relevante Antworten entsprechend den Benutzer-Prompts generieren können.

Zu den Schlüsselkonzepten gehören Tokenisierung (d. h. die Aufteilung von Text in Einheiten für eine effiziente Verarbeitung), Kontextfenster (die Begrenzung der gleichzeitig berücksichtigten Informationsmenge, um die Relevanz zu gewährleisten) und multimodale Modelle (die mehrere Datentypen wie Text, Bilder und Audio für reichhaltige Interaktionen verarbeiten können).

Im Softwaretest können diese LLMs Aufgaben wie die Überprüfung und Verbesserung von Akzeptanzkriterien, die Erstellung von Testfällen oder Testskripten, die Identifizierung potenzieller Fehler, die Analyse von Fehlermustern, die Generierung synthetischer Testdaten oder die Dokumentationserstellung während des gesamten Testprozesses unterstützen.

1.1.1 KI-Spektrum: symbolische KI, klassisches maschinelles Lernen, Deep Learning und generative KI

Künstliche Intelligenz (KI) ist ein weites Feld, das verschiedene Arten von Technologien umfasst, von denen jede ihre eigene einzigartige Weise der Problemlösung hat, wie z. B. symbolische KI, klassisches maschinelles Lernen, Deep Learning und GenAI (neben anderen Technologien, die nicht Gegenstand dieses Lehrplans sind):

- Symbolische KI verwendet ein regelbasiertes System, um menschliche Entscheidungsfindung nachzuahmen. Im Wesentlichen repräsentiert symbolische KI Wissen mithilfe von Symbolen und logischen Regeln.
- Das klassische maschinelle Lernen ist ein datengesteuerter Ansatz, der Datenvorbereitung, Merkmalsauswahl und Modelltraining erfordert und für Aufgaben wie die Kategorisierung von Fehlerzuständen und die Vorhersage von Softwareproblemen verwendet werden kann.
- Deep Learning verwendet Strukturen des maschinellen Lernens, sogenannte neuronale Netze, um Merkmale automatisch aus Daten zu lernen. Deep Learning-Modelle können Muster in sehr großen und komplexen Datensätzen wie Bildern, Videos, Audiodateien oder Texten finden, ohne dass Benutzer Merkmale manuell definieren müssen. In der Praxis kann jedoch weiterhin menschliches Eingreifen bei Aufgaben wie Datenannotation, Modelloptimierung oder Ergebnisvalidierung erforderlich sein.
- Generative KI nutzt Deep Learning-Techniken, um neue Inhalte (Text, Bilder, Code) zu erstellen, indem sie Muster aus ihren Trainingsdaten lernt und nachahmt. Modelle wie LLMs können Texte generieren, Code schreiben und logisches Denken oder Problemlösen simulieren, sofern es im Rahmen ihres Trainings liegt.

Mittlerweile hat sich die Disziplin KI in mehrere Richtungen entwickelt, die jeweils unterschiedliche Stärken und Schwächen aufweisen. Der entscheidende Vorteil der Verwendung von GenAI für den Softwaretest besteht darin, dass vortrainierte Modelle direkt auf Testaufgaben angewendet werden können, ohne dass eine zusätzliche Trainingsphase erforderlich ist, obwohl dies mit einigen Risiken verbunden ist (siehe Abschnitt 3.1).

1.1.2 Grundlagen von generativer KI und LLMs

Basierend auf dem Deep Learning-Modell „Generativer Vortrainierter Transformer“ (*generative pre-trained transformer*, GPT) werden LLMs auf sehr großen Datensätzen trainiert, darunter Bücher, Artikel und Websites. Small-Language-Modelle (SLMs) sind kompakte Modelle mit weniger Parametern als

Large Language-Modelle, die entworfen wurden, um leichtgewichtige und fokussierte GenAI-Lösungen bereitzustellen.

LLMs können sprachliche Nuancen verarbeiten und kohärente Inhalte generieren. Zwei Schlüsselkonzepte, die LLMs bei der Verarbeitung und Generierung von Inhalten unterstützen, sind Tokenisierung und Einbettungen. Tokenisierung und Einbettungen wandeln Sprache in eine numerische Form um, die das Modell effektiver verarbeiten kann.

- Tokenisierung in Sprachmodellen ist der Prozess, Text in kleinere Einheiten, die als Token bezeichnet werden, aufzuteilen. Token können so klein wie ein Zeichen oder so groß wie ein Wortteil oder ein ganzes Wort sein. Wenn ein LLM einen Satz verarbeitet, tokenisiert es zunächst die Eingabe, sodass jedes Token einzeln verarbeitet werden kann, während der Gesamtkontext erhalten bleibt.
- Einbettungen sind numerische Darstellungen von Token, die ihre semantischen, syntaktischen und kontextuellen Beziehungen in einem Format kodieren, das für die Verarbeitung durch generative KI-Modelle geeignet ist. Jedes Token wird in einen Vektor in einem hochdimensionalen Raum umgewandelt, der nuancierte Informationen über seine Bedeutung und Verwendung erfasst. Token mit ähnlichen Bedeutungen oder kontextuellen Rollen haben Einbettungen, die in diesem Raum nahe beieinanderliegen. Diese Nähe ermöglicht es LLMs, Wortbeziehungen zu verstehen, den Kontext beizubehalten und kohärente und kontextuell angemessene Antworten zu generieren.

LLMs nutzen eine neuronale Netzarchitektur, die als Transformer-Modell bekannt ist. Transformer-Modelle zeichnen sich bei Sprachaufgaben dadurch aus, dass sie den Kontext umfangreicher Textsequenzen verarbeiten und lernen, wie Token miteinander in Beziehung stehen. Während der Inferenz[1] sagen LLMs das nächste Token in einer Sequenz voraus und nutzen diese gelernten Beziehungen, um kohärente und kontextuell angemessene Texte zu generieren. Das Transformer-Modell kann verwendet werden, um neue Texte zu generieren, die auf der Grundlage von Trainingsdaten und dem Prompt statistisch plausibel sind. Jedoch bedeutet „plausibel“ nicht notwendigerweise „richtig“.

LLMs zeigen vor allem aufgrund der probabilistischen Natur ihrer Inferenzmechanismen und Hyperparametereinstellungen ein nicht-deterministisches Verhalten. Diese inhärente Zufälligkeit kann zu Abweichungen in den Ausgaben führen, selbst wenn dieselbe Eingabe mehrfach bereitgestellt wird.

Im Bereich der LLMs bezieht sich das Kontextfenster auf die Menge des vorangehenden Textes, gemessen in Token, die das Modell bei der Generierung von Antworten berücksichtigen kann. Ein größeres Kontextfenster ermöglicht es dem Modell, die Kohärenz über längere Passagen aufrechtzuerhalten, beispielsweise bei der Analyse großer Testprotokolle. Eine Erhöhung der Anzahl der Token im Kontextfenster erhöht jedoch auch die Rechenkomplexität und die Verarbeitungszeit, die das Modell benötigt, um effektiv zu arbeiten.

Praktische Übung zum Ziel HO-1.1.2 (H1): … angeleitet die Tokenisierung und die Evaluierung der Tokenanzahl bei der Verwendung eines LLM für eine Software-Testaufgabe ausführen.

Diese praktische Übung soll den Teilnehmenden helfen, ein praktisches Verständnis der Tokenisierung und ihrer Auswirkungen bei der Arbeit mit LLMs zu entwickeln. Die Übung gliedert sich in zwei wesentliche Teile:

[1] Hier: Vorgang der Text-Generierung.

- Tokenisierung: Verwenden Sie einen Tokenisierer, um einen Beispieltext in einzelne Token zu zerlegen. Untersuchen Sie die Ausgabe, um zu sehen, wie Wörter, Satzzeichen und Phrasen dargestellt werden, und identifizieren Sie Muster und Feinheiten in der Tokenisierung.
- Evaluierung der Tokenanzahl: Messen Sie die Anzahl der Token, die aus verschiedenen Eingabetexten generiert wurden. Analysieren Sie, wie die Tokenanzahl die Modellleistung beeinflusst, insbesondere in Bezug auf die Kontextfenster des Modells und seine Effizienz.

Am Ende dieser Übung werden die Teilnehmer besser einschätzen können, wie sich unterschiedliche Textstrukturen und Eingabelängen auf die Interaktion mit LLMs auswirken können.

1.1.3 Foundation-, Instruction-Tuned- und Reasoning-LLMs

Large Language-Modelle werden in zunehmend spezialisierten Trainingsphasen entwickelt, um ihre Effektivität bei einer Vielzahl von Aufgaben zu verbessern. Diese Phasen führen zu drei Hauptkategorien: Foundation-LLMs, Instruction-Tuned-LLMs und Reasoning-LLMs.

- Foundation-LLMs: Hierbei handelt es sich um Allzweckmodelle, die auf umfangreichen und vielfältigen Datensätzen trainiert wurden, die Text, Code, Bilder und andere Modalitäten umfassen. Dank ihres umfangreichen Vortrainings können sie verschiedene Aufgaben in Bereichen wie der Verarbeitung natürlicher Sprache, der Bildverarbeitung und der Spracherkennung unterstützen. Obwohl sie leistungsstark und flexibel sind, müssen Foundation-LLMs in der Regel weiter angepasst werden, um spezifische Aufgabenanforderungen zu erfüllen.
- Instruction-Tuned-LLMs: Instruction-Tuned-LLMs leiten sich von Foundation-Modellen ab und werden mithilfe von Datensätzen, die Prompts mit erwarteten Antworten verknüpfen, getuned. Diese Phase verbessert ihre Übereinstimmung mit menschlichen Instruktionen und damit ihre Gebrauchstauglichkeit in realen Anwendungen. Der Abstimmungsprozess umfasst die Optimierung der Aufgabenkonformität, der Befolgung von Instruktionen und der Kohärenz der Antworten, wodurch die Fähigkeit des Modells verbessert wird, die Absichten der Benutzer effektiv zu interpretieren und darauf zu reagieren.
- Reasoning-LLMs: Reasoning-Modelle erweitern Instruction-Tuned-Modelle, indem sie strukturierte kognitive Fähigkeiten wie logische Schlussfolgerungen, mehrstufige Problemlösung und Argumentation mit Gedankenketten (*chain-of-thought*, CoT) in den Vordergrund stellen. Diese Modelle werden anhand sorgfältig ausgewählter Aufgaben, die Kontextverständnis, intermediäre Argumentationsschritte und die Synthese komplexer Informationen erfordern, weiter trainiert oder feingetunt. Dadurch eignen sie sich besser für Aufgaben mit hoher kognitiver Belastung, einschließlich solcher in technischen Bereichen.

Im Zusammenhang mit GenAI-Anwendungen für Softwaretests werden sowohl Instruction-Tuned- (manchmal auch als Non-Reasoning-LLM bezeichnet) als auch Reasoning-LLMs verwendet. Die Auswahl hängt von der Komplexität und den Reasoning-Anforderungen der jeweiligen Testaufgabe ab.

1.1.4 Multimodale LLMs und Vision-Language-Modelle

Multimodale LLMs erweitern das traditionelle Transformer-Modell um die Möglichkeit, mehrere Datenmodalitäten zu verarbeiten, z. B. Text, Bilder, Ton und Video. Diese Modelle werden anhand großer und vielfältiger Datensätze trainiert, wodurch sie Beziehungen zwischen verschiedenen Datentypen lernen können. Um verschiedene Modalitäten zu verarbeiten, wird die Tokenisierung für jeden Datentyp angepasst – beispielsweise werden Bilder mithilfe von Vision-Language-Modellen in Einbettungen umgewandelt, bevor sie im Transformer-Modell verarbeitet werden.

Vision-Language-Modelle, eine Untergruppe multimodaler LLMs, integrieren speziell visuelle und textuelle Informationen, um Aufgaben wie Bildbeschriftung, visuelle Beantwortung von Fragen und Analyse der Konsistenz zwischen textuellen und visuellen Eingaben durchzuführen.

Im Bereich des Softwaretests bieten multimodale LLMs, insbesondere LLMs, die mit Vision-Language-Modellen angereichert wurden, erhebliche Möglichkeiten. Sie können visuelle Elemente von Anwendungen, wie Screenshots und GUI-Wireframes, zusammen mit zugehörigen Textbeschreibungen, wie Fehlerberichten oder User-Storys, analysieren. Diese Fähigkeit ermöglicht es Testern, Diskrepanzen zwischen dem erwarteten Ergebnis und den tatsächlichen visuellen Elementen auf einem Screenshot zu identifizieren. Darüber hinaus können LLMs, die mit Vision-Language-Modellen angereichert wurden, reichhaltige, realistische Testfälle generieren, die sowohl Textdaten als auch visuelle Hinweise enthalten und so die Überdeckung erhöhen.

Praktische Übung zum Ziel HO-1.1.4 (H1): … angeleitet einen Prompt für ein multimodales LLM schreiben und ausführen, der sowohl Text- als auch Bildeingaben für eine Software-Testaufgabe verwendet.

Diese Übung umfasst das Review und die Ausführung eines vorgegebenen Prompts für ein multimodales LLM-Modell unter Verwendung von Text- und Bildeingaben, um eine Testaufgabe in zwei Schritten zu lösen:

- Review der Eingabe: Review der Eingabe und der Eingabedaten (Text und Bild).
- Ausführen des Prompts und Überprüfen des Ergebnisses: Verwenden Sie ein multimodales LLM, um sowohl Bilder als auch Text einzugeben, und überprüfen Sie die Antwort des LLMs.

Diese Übung zeigt, wie multimodale LLMs für eine Aufgabe verwendet werden können, die sowohl Text- als auch Bildeingaben in Anwendungsfällen für Softwaretests umfasst, einschließlich des Erkennens der damit verbundenen Vorteile und potenziellen Herausforderungen.

1.2 Nutzung generativer KI im Softwaretest: Grundprinzipien

GenAI bietet transformative Fähigkeiten für verschiedene Testaktivitäten. LLMs zeichnen sich aus durch die Verarbeitung natürlicher Sprache und von Code, die Generierung kohärenter Texte und Codes, die Beantwortung von Fragen, die Zusammenfassung von Informationen, die Übersetzung von Sprachen sowie die Analyse von Bildern in einem multimodalen Kontext.

Testfachleute in allen Funktionen können GenAI auf zwei sich ergänzende Arten nutzen: durch GenAI-Chatbots, die sofort Antworten auf Fragen liefern, und durch LLM-gestützte Anwendungen, die in Testwerkzeuge integriert sind.

1.2.1 Wichtige LLM-Funktionen für Testaufgaben

LLMs können Anforderungen, Spezifikationen, Screenshots, Code, Testfälle und Fehlerberichte interpretieren und sind somit Werkzeuge zum Verstehen und Klären der während des gesamten Testprozesses verwendeten Informationen und zum Generieren von Elementen der Testmittel. Im Folgenden sind einige der wichtigsten Funktionen von LLM aufgeführt, die für den Softwaretest relevant sind:

- Anforderungsanalyse und -verbesserung: LLMs können bei der Analyse von Anforderungen und anderen Elementen der Testbasis helfen, indem sie Unklarheiten, Inkonsistenzen oder fehlende Informationen identifizieren. Sie können aussagekräftige Fragen generieren, um Anforderungen während der Diskussionen mit den Stakeholdern zu klären.

- Unterstützung bei der Erstellung von Testfällen: LLMs können dabei helfen, Testfälle zu generieren und Testziele auf der Grundlage von Systemanforderungen, User-Storys oder anderen Elementen der Testbasis vorzuschlagen.
- Generierung von Testorakeln: LLMs können dabei helfen, erwartete Ergebnisse zu generieren.
- Generierung von Testdaten: LLMs können Datensätze generieren, Grenzwerte festlegen und verschiedene Kombinationen von Testdaten erstellen.
- Unterstützung bei der Testautomatisierung: LLMs können dabei helfen, Testskripte aus der Beschreibung von Testfällen zu generieren und bestehende Testskripte zu verbessern, indem sie Änderungen vorschlagen und geeignete Verfahren für den Testentwurf identifizieren.
- Analyse der Testergebnisse: LLMs können bei der Analyse von Testergebnissen helfen, indem sie Zusammenfassungen erstellen und Anomalien nach Fehlerschweregrad und Priorität klassifizieren.
- Erstellung von Testmitteln: LLMs können bei der Erstellung verschiedener Dokumente helfen, darunter Testkonzepte, Testberichte und Fehlerberichte, und diese im Laufe des Projekts auf dem neuesten Stand halten.

Diese Funktionen zeigen, wie LLMs verschiedene Aspekte des Softwaretestens während des gesamten Testprozesses beeinflussen können.

1.2.2 KI-Chatbots und LLM-gestützte Testanwendungen für den Softwaretest

KI-Chatbots und LLM-gestützte Testanwendungen können Tester gleichermaßen helfen, unterscheiden sich jedoch in Bezug auf Funktionalität, Flexibilität und Integrationsansätze.

KI-Chatbots bieten eine benutzerfreundliche, dialogorientierte Schnittstelle, über die Tester direkt mit LLMs kommunizieren können. Diese natürlichsprachliche Interaktion ermöglicht es Testern, Fragen, Befehle oder Prompts einzugeben und unmittelbar kontextbezogene Antworten zu erhalten. Durch Verfahren wie Prompt-Verkettung können Tester die Ergebnisse iterativ verfeinern, wodurch KI-Chatbots besonders geeignet sind für Routineaufgaben, exploratives Testen und sogar für die Einarbeitung neuer Tester, da sie einen schnellen Zugang zu Testwissen und -praktiken bieten.

Diese KI-Chatbots sind besonders vorteilhaft in Szenarien, die schnelles Feedback, die Klärung von Testkonzepten oder die dynamische Erforschung von Anforderungen und potenziellen Testfällen erfordern. Dank ihrer intuitiven Schnittstelle sind sie auch für nicht-technische Stakeholder zugänglich, was die potenzielle Benutzergruppe erweitert und eine breitere Akzeptanz fördert.

LLM-gestützte Testanwendungen hingegen beinhalten die Integration von LLM-Funktionen über APIs, um klar definierte und oft automatisierte Testaufgaben auszuführen. Diese Anwendungen bieten eine größere Anpassungsfähigkeit und Skalierbarkeit, sodass Unternehmen und Werkzeuganbieter generative KI in bestehende Testframeworks einbetten können. Dies ermöglicht die Automatisierung sich wiederholender oder komplexer Aufgaben wie die Generierung von Testfällen, die Fehleranalyse oder die Synthese von Testdaten. In fortgeschritteneren Implementierungen können Unternehmen KI-Agenten erstellen, die speziell für bestimmte Rollen im Test entworfen wurden (siehe Kapitel 4).

Unabhängig davon, wie Tester mit LLMs interagieren – ob über Chatbots oder integrierte LLM-gestützte Anwendungen –, erfordert die erfolgreiche Implementierung generativer KI im Test ein starkes Prompt-Engineering (siehe Kapitel 2). Sorgfältig entworfene Prompts und klare, spezifische Instruktionen sind unerlässlich, um sicherzustellen, dass die von LLMs generierten Ergebnisse genau, relevant und auf das Testziel abgestimmt sind. Diese Vorgehensweise trägt dazu bei, den Wert generativer KI zu maximieren und eine konsistente, zuverlässige Unterstützung für eine Vielzahl von Testaktivitäten zu gewährleisten.

2 Prompt-Engineering für effektives Softwaretesten – 365 Minuten

Schlüsselbegriffe

Akzeptanzkriterien, Testbedingung, Testbericht, Testdaten, Testentwurf, Testfall, Testskript

Spezifische Schlüsselbegriffe für generative KI

Benutzer-Prompt, Few-Shot-Prompting, Meta-Prompting, One-Shot-Prompting, Prompt, Prompt-Engineering, Prompt-Verkettung (*prompt chaining*), System-Prompt, Verarbeitung natürlicher Sprache (*natural language processing*, NLP), Zero-Shot-Prompting

Lernziele und Ziele der praktischen Übungen für Kapitel 2: Die Lernenden können …

2.1 Effektive Prompt-Entwicklung

GenAI-2.1.1	(K2)	… Beispiele für die Struktur von Prompts nennen, die in generativer KI für das Softwaretesten verwendet werden.
HO-2.1.1	(H0)	… vorgegebene Prompts für Software-Testaufgaben untersuchen und in jedem die Komponenten Rolle, Kontext, Instruktion, Eingabedaten, Einschränkungen und Ausgabeformat identifizieren.
GenAI-2.1.2	(K2)	… zwischen den wichtigsten Prompting-Verfahren für das Softwaretesten unterscheiden.
HO-2.1.2a	(H0)	… Demonstrationen von Prompt-Verkettung, Few-Shot-Prompting und Meta-Prompting erläutern, die auf Software-Testaufgaben angewendet werden.
HO-2.1.2b	(H1)	… in gegebenen Beispielen die verwendeten Verfahren des Prompt-Engineering identifizieren.
GenAI-2.1.3	(K2)	… zwischen System-Prompts und Benutzer-Prompts unterscheiden.

2.2 Anwendung von Prompt-Engineering-Verfahren auf Software-Testaufgaben

GenAI-2.2.1	(K3)	… generative KI für Aufgaben bei der Testanalyse anwenden.
HO-2.2.1a	(H2)	… selbstständig strukturiertes, multimodales Prompting anwenden, um Akzeptanzkriterien für eine User-Story auf der Grundlage eines GUI-Wireframes zu generieren.
HO-2.2.1b	(H2)	… selbstständig Prompt-Verkettung und die Verifizierung durch Menschen anwenden, um eine bestimmte User-Story schrittweise zu analysieren und die Akzeptanzkriterien zu verfeinern.
GenAI-2.2.2	(K3)	… generative KI für Testentwurf und Testrealisierung anwenden.
HO-2.2.2a	(H2)	… selbstständig mithilfe von Prompt-Verkettung, strukturierten Prompts und Meta-Prompting aus User-Storys funktionale Testfälle erstellen.
HO-2.2.2b	(H2)	… selbstständig Few-Shot-Prompting anwenden, um aus User-Storys Testbedingungen und Testfälle im Gherkin-Stil zu generieren.
HO-2.2.2c	(H2)	… selbstständig Prompt-Verkettung anwenden, um Testfälle innerhalb einer bestimmten Testsuite unter Berücksichtigung ihrer spezifischen Prioritäten und Abhängigkeiten zu priorisieren.
GenAI-2.2.3	(K3)	… generative KI für automatisierte Regressionstests anwenden.
HO-2.2.3a	(H2)	… selbstständig Few-Shot-Prompting zur Erstellung und Verwaltung von schlüsselwortgetriebenen Testskripten anwenden.
HO-2.2.3b	(H2)	… selbstständig strukturiertes Prompt-Engineering für die Analyse von Testberichten im Regressionstest anwenden.

GenAI-2.2.4	(K3)	… generative KI für Aufgaben bei der Teststeuerung und -überwachung anwenden.
HO-2.2.4	(H0)	… die von generativer KI aus Testdaten erstellten Testüberwachungsmetriken erläutern.
GenAI-2.2.5	(K3)	… geeignete Prompting-Verfahren für einen bestimmten Kontext und einen bestimmten Test auswählen und anwenden.
HO-2.2.5	(H1)	… angeleitet kontextgerechte Prompting-Verfahren für eine vorgegebene Testaufgabe auswählen und anwenden.

2.3 Bewertung generativer KI-Ergebnisse und Verfeinerung von Prompts für Software-Testaufgaben

GenAI-2.3.1	(K2)	… Metriken zur Bewertung der Ergebnisse generativer KI bei Testaufgaben verstehen.
HO-2.3.1	(H0)	… erläutern, wie Metriken zur Bewertung des Ergebnisses generativer KI bei einer Testaufgabe verwendet werden können.
GenAI-2.3.2	(K2)	… Beispiele für Verfahren zur Bewertung und iterativen Verfeinerung von Prompts nennen.
HO-2.3.2	(H1)	… angeleitet einen Prompt für eine vorgegebene Testaufgabe bewerten und optimieren.

2.1 Effektive Prompt-Entwicklung

Ein effektiver Prompt-Entwurf stellt sicher, dass GenAI-Werkzeuge Software-Testaufgaben genau und effizient ausführen und Tester nützliche Ergebnisse aus den LLMs erhalten. Ein strukturierter Prompt umfasst verschiedene Komponenten (siehe Abschnitt 2.1.1). Jede dieser Komponenten trägt zur Klarheit und Präzision eines Prompts bei, der Anforderungen und Erwartungen effektiv an LLMs kommuniziert.

Verschiedene Verfahren des Prompt-Engineering verbessern die Effektivität von Prompts im Softwaretest. Verfahren wie Prompt-Verkettung, Few-Shot-Prompting und Meta-Prompting helfen bei der Bewältigung komplexer Testherausforderungen (siehe Abschnitt 2.1.2).

Die Kombination von strukturierten Prompts (siehe Abschnitt 2.1.1) mit Kern-Verfahren des Prompt-Engineering zielt darauf ab, gute Ergebnisse bei der Abfrage eines LLMs für Softwaretesten zu erzielen (siehe Abschnitt 2.1.3).

2.1.1 Struktur von Prompts für generative KI im Testen

Ein strukturierter Prompt für den Softwaretest umfasst in der Regel sechs Komponenten:

- Rolle: Die Rolle definiert die Perspektive oder Persona, die das GenAI-Modell bei der Generierung einer Antwort einnehmen soll. Die Angabe der Rolle hilft dem LLM, seine Aufgaben zu bestimmen und einen angemessenen Ton oder Ansatz zu wählen, z. B. als Tester, Testmanager oder Testautomatisierungsingenieur zu agieren.
- Kontext: Der Kontext liefert die Hintergrundinformationen, die das GenAI-Modell benötigt, um die Testbedingungen zu bestimmen. Dazu gehören Details zum Testobjekt, die zu testende spezifische Funktionalität und alle relevanten Kontextinformationen.
- Instruktion: Instruktionen sind Richtlinien für das GenAI, die die auszuführende Aufgabe beschreiben. Klare, imperativ formulierte und prägnante Instruktionen enthalten eine Aufgabenbeschreibung und alle relevanten Anforderungen für die Aufgabe.
- Eingabedaten: Zu den Eingabedaten gehören alle Informationen, die zur Ausführung der Aufgabe erforderlich sind, wie z. B. User-Storys, Akzeptanzkriterien, Screenshots, Code, vorhandene Testfälle oder Beispiele für die Ausgabe. Die Bereitstellung detaillierter und strukturierter Eingabedaten hilft dem LLM, genauere und kontextbezogene Ergebnisse zu generieren.
- Einschränkungen: Einschränkungen umfassen alle Beschränkungen oder besonderen Überlegungen, die das LLM berücksichtigen sollte. Einschränkungen helfen dabei, festzulegen, wie Instruktionen auf Eingabedaten angewendet werden sollen.
- Ausgabeformat: Ausgabespezifikationen bezeichnen das erwartete Format, die Struktur oder die Eigenschaften der Antwort. Diese Indikatoren helfen dabei, die Ausgabe des LLMs zu gestalten.

Diese Komponenten bilden die Grundstruktur des Prompts. Diese Struktur sollte je nach der auszuführenden Aufgabe und des zu verwendenden LLMs mit der Implementierung von Prompting-Verfahren (siehe Abschnitt 2.1.2) kombiniert werden.

Praktische Übung zum Ziel HO-2.1.1 (H0): … vorgegebene Prompts für Software-Testaufgaben untersuchen und in jedem die Komponenten Rolle, Kontext, Instruktion, Eingabedaten, Einschränkungen und Ausgabeformat identifizieren.

In einer Demonstration werden mehrere strukturierte Prompts auf einem KI-Chatbot-System getestet, die jeweils auf bestimmte Software-Testaufgaben zugeschnitten sind. Diese Prompts folgen einem strukturierten Format, das aus sechs Schlüsselkomponenten besteht: Rolle, Kontext, Instruktion, Eingabedaten, Einschränkungen und Ausgabeformat. Mit dieser Demonstration soll gezeigt werden, wie ein LLM für eine Software-Testaufgabe verwendet wird, um genaue, relevante und umsetzbare Erkenntnisse zu liefern.

2.1.2 Kern-Verfahren für das Prompting im Softwaretest

In den letzten Jahren wurden viele LLM-Prompting-Verfahren für verschiedene Anwendungsfälle von GenAI-Prompts vorgeschlagen (Schulhoff 2024). Unter diesen werden in Verbindung mit der oben beschriebenen 6-Komponenten-Prompt-Struktur häufig drei Kern-Verfahren des Promptings für Testaufgaben mit GenAI verwendet (siehe Abschnitt 2.1.1): Prompt-Verkettung, Few-Shot-Prompting und Meta-Prompting.

- Bei der Prompt-Verkettung wird eine Aufgabe in eine Reihe von Zwischenschritten (mehrere Prompts) unterteilt. Das Ergebnis eines jeden Schritts wird manuell oder automatisch überprüft und verfeinert, bevor mit dem nächsten Schritt fortgefahren wird. Dieser Ansatz führt zu einer höheren Genauigkeit, da jede Antwort in den nächsten Prompt einfließt. Prompt-Verkettung ist besonders nützlich in Testprozessen, in denen die Aufgaben kompliziert sind und eine Aufteilung in Teilaufgaben sowie eine systematische Überprüfung der Zwischenergebnisse des LLMs erfordern. Dieses Verfahren ermöglicht auch dynamische Interaktionen in Testprozessen.
- Beim Few-Shot-Prompting werden dem LLM Beispiele im Prompt bereitgestellt. Während Zero-Shot-Prompting (ohne Beispiele) auf dem bereits vorhandenen Wissen des Modells basiert, um eine Antwort zu generieren, liefert One-Shot-Prompting ein Beispiel, um das gewünschte Ergebnis für eine bestimmte Eingabe zu demonstrieren. Few-Shot-Prompts enthalten mehr als ein Beispiel (einige wenige), um das gewünschte Antwortverhalten des Modells weiter zu konsolidieren.

  Dieses Verfahren hilft dabei, das Modell zu steuern, indem es eine klare Referenz liefert und sicherstellt, dass die Ergebnisse konsistent sind und den Erwartungen entsprechen. Few-Shot-Prompting ist besonders effektiv für Aufgaben, bei denen Beispiele das erforderliche Verhalten veranschaulichen können, sodass das Modell effektiv verallgemeinern und zuverlässige Ergebnisse liefern kann.
- Meta-Prompting nutzt die Fähigkeit der KI, eigene Prompts zu generieren oder zu verfeinern. In einem iterativen Zyklus kann das LLM Prompts generieren, die vom Tester bewertet und verfeinert werden können. Dieser Ansatz optimiert die Qualität der Prompts, indem er das Wissen des LLMs über optimierte Prompts nutzt. Meta-Prompting ist besonders vorteilhaft, wenn Effizienz und Prompt-Optimierung entscheidend sind, da es den manuellen Aufwand für das Entwerfen effektiver Prompts reduziert. Ein weiterer Vorteil von Meta-Prompting besteht darin, dass der Tester, wenn er sich nicht sicher ist, wie er einen effektiven Prompt erstellen soll, mit dem LLM zusammenarbeiten kann, um diesen gemeinsam zu erstellen. Dies spiegelt eine Form der Verwendung des GenAI-Werkzeugs wider, bei der der Tester und die KI interaktiv zusammenarbeiten, um ein gemeinsames Ziel zu erreichen. Dieses Konzept der Paarbildung

verdeutlicht eine neue Art der Zusammenarbeit mit KI-Werkzeugen, die sowohl die Produktivität als auch das Lernen nicht nur im Bereich des Prompt-Engineering, sondern auch im Programmieren und Testen in Paaren verbessert.

Diese Prompting-Verfahren können sehr effektiv in Kombination genutzt werden, um die Ergebnisse des LLMs zu verbessern (siehe Abschnitt 2.2.5).

Praktische Übung zum Ziel HO-2.1.2a (H0): … Demonstrationen von Prompt-Verkettung, Few-Shot-Prompting und Meta-Prompting erläutern, die auf Software-Testaufgaben angewendet werden.

Die Teilnehmer werden auf einem KI-Chatbot Erfahrungen mit Prompt-Verkettung, Few-Shot-Prompting und Meta-Prompting sammeln, während die Verfahren jeweils auf bestimmte Software-Testaufgaben angewendet werden. Die Demonstration zielt darauf ab, diese Prompting-Verfahren im Kontext von Softwaretests zu untersuchen und zu diskutieren, wobei der Schwerpunkt darauf liegt, wie jedes Verfahren zur Genauigkeit und Vollständigkeit der LLM-Ausgaben beiträgt.

Praktische Übung zum Ziel HO-2.1.2b (H1): … in gegebenen Beispielen die verwendeten Verfahren des Prompt-Engineering identifizieren.

Die Teilnehmer verwenden eine Reihe von Beispielen für Prompting-Verfahren im Zusammenhang mit Softwaretests, um die wichtigsten angewandten Prompting-Verfahren zu identifizieren. Der Schwerpunkt liegt auf dem Erkennen von Verfahren wie Prompt-Verkettung, Few-Shot-Prompting und Meta-Prompting, wobei deren Merkmale und praktische Anwendungen hervorgehoben werden.

Diese Aktivität zielt darauf ab, das Verständnis der Teilnehmer dafür zu vertiefen, wie verschiedene Prompting-Verfahren die effektive Nutzung von GenAI im Softwaretest verbessern.

2.1.3 System-Prompt und Benutzer-Prompt

System-Prompts und Benutzer-Prompts dienen verschiedenen Zwecken bei Interaktionen mit LLMs und spielen jeweils eine unterschiedliche Rolle bei der Gestaltung der Kommunikation. Der System-Prompt wird in der Regel vom Entwickler oder Tester definiert, um das Gesamtverhalten des LLMs zu steuern, und ist für den Benutzer des Chatbots in den meisten Schnittstellen nicht sichtbar oder editierbar.

Ein System-Prompt fungiert als vordefinierter Befehlssatz, der das Verhalten, die Persönlichkeit und die Betriebsparameter des LLMs definiert. Die Betriebsparameter bestimmen, wie das LLM reagiert – beispielsweise durch Verwendung eines formellen Tonfalls, prägnante Antworten, Einhaltung domänenspezifischer Regeln oder Vermeidung bestimmter Verhaltensweisen. Der System-Prompt legt die Regeln für die gesamte Konversation fest. Er kann Teile eines strukturierten Prompts enthalten, wie z. B. die Rolle, den Kontext und Einschränkungen.

Der System-Prompt bleibt während der gesamten Interaktionssitzung konstant und legt den grundlegenden Rahmen für die Reaktion des LLMs fest. Ein System-Prompt könnte beispielsweise lauten: „Sie sind ein professioneller Softwaretester. Antworten Sie immer klar, verwenden Sie eine formelle Sprache und konzentrieren Sie sich auf ISTQB®-konforme Praktiken. Vermeiden Sie Annahmen und zitieren Sie gegebenenfalls Testprinzipien.“

Der Benutzer-Prompt hingegen stellt die tatsächliche Eingabe oder Frage des Chatbot-Benutzers dar. Er ändert sich mit jeder Interaktion und kann spezifische Instruktionen, Fragen oder Aufgaben enthalten, die der Chatbot-Benutzer vom LLM verarbeitet haben möchte. Im Gegensatz zum System-Prompt sind Benutzer-Prompts direkt sichtbar und bilden den unmittelbaren Kontext für jede Antwort.

Ein Beispiel für einen Benutzer-Prompt könnte lauten: „Nennen Sie die wichtigsten Unterschiede zwischen Black-Box-Test und White-Box-Test mit Beispielen."

Typischerweise wird zu Beginn der Konversation einmalig der System-Prompt festgelegt und anschließend für jede Interaktion nacheinander ein Benutzer-Prompt gesendet. Das LLM generiert Antworten, indem es sowohl den unveränderlichen System-Prompt als auch den aktuellen Benutzer-Prompt gemeinsam berücksichtigt. Für eine effektive Implementierung sollten System-Prompts klar und spezifisch in Bezug auf die Rolle des LLMs und mögliche Einschränkungen sein. Sie können auch einen Kontext und allgemeine Instruktionen enthalten, z. B. hinsichtlich der erwarteten Ausgabe.

Benutzer-Prompts müssen fokussiert und gut strukturiert sein und explizite Instruktionen sowie zusätzliche relevante Instruktionen zu Kontext und Ausgabeformat enthalten.

2.2 Anwendung von Prompt-Engineering-Verfahren auf Software-Testaufgaben

Die Anwendung von Verfahren des Prompt-Engineering im Softwaretest ermöglicht es GenAI, Testaufgaben wie Testanalyse, Testentwurf, Testautomatisierung, Testfallpriorisierung, Fehlerfindung, Überdeckungsanalyse sowie Testüberwachung und Teststeuerung zu unterstützen. Durch die Verwendung und Kombination von Verfahren wie Prompt-Verkettung, Few-Shot-Prompting und Meta-Prompting können Teams KI-Prompts auf die spezifischen Testziele zuschneiden und so die Ergebnisse präziser, relevanter und effektiver gestalten. Hochwertige Eingaben sind entscheidend für aussagekräftige KI-Ergebnisse.

2.2.1 Testanalyse mit generativer KI

GenAI kann Testanalyseaufgaben unterstützen, indem sie Testbedingungen definiert und priorisiert, Fehlerzustände in der Testbasis identifiziert und eine Überdeckungsanalyse bereitstellt. Die Eingabedaten umfassen Anforderungen, User-Storys, technische Spezifikationen, GUI-Wireframes und andere relevante Informationen. Die Ausgabe besteht aus typischen Testanalyse-Arbeitsergebnissen, wie priorisierten Testbedingungen (z. B. Akzeptanzkriterien).

Im Folgenden sind einige typische Aufgaben der Testanalyse aufgeführt, die von GenAI unterstützt werden können:

- **Identifizieren potenzieller Fehlerzustände in der Testbasis:** GenAI kann dabei helfen, die Testbasis auf Inkonsistenzen, Unklarheiten oder unvollständige Informationen zu analysieren, die zu Fehlerzuständen führen könnten. Durch den Vergleich ähnlicher Anforderungsmuster oder die Anwendung von Wissen aus früheren Fehlerberichten kann das LLM potenzielle Anomalien kennzeichnen und Verbesserungen vorschlagen.
- **Definieren von Testbedingungen auf der Grundlage der Testbasis,** z. B. auf der Grundlage von Anforderungen/User-Storys: LLMs können Anforderungen und User-Storys analysieren, um Testbedingungen zu definieren. Mithilfe der Verarbeitung natürlicher Sprache können sie die Bedeutung von Anforderungen interpretieren und in messbare und testbare Aussagen zerlegen. Dies kann dabei helfen, Anforderungen in spezifische Testbedingungen zu übersetzen.
- **Priorisierung von Testbedingungen aufgrund der Risikostufe:** Mit Informationen über die Eintrittswahrscheinlichkeit des Risikos und das Schadensausmaß des Risikos für jede Testbedingung kann ein LLM dabei helfen, den Testaufwand zu priorisieren. Unter Berücksichtigung von Aspekten wie der Einhaltung gesetzlicher Vorschriften, benutzerseitigen Merkmalen (z. B. Anmeldefunktionalität oder Zahlungsabwicklung) und historischen Fehlerzuständen kann das LLM Prioritätsstufen empfehlen.

- **Unterstützung der Abdeckungsanalyse:** Durch die Zuordnung von Anforderungen und User-Storys zu Testbedingungen kann ein LLM eine Überdeckungsanalyse durchführen, um festzustellen, ob alle Aspekte der Testbasis abgedeckt sind. Dies ist besonders nützlich für Projekte mit komplexen Anforderungen, bei denen Lücken in der Überdeckung dazu führen können, Fehlerzustände zu übersehen.
- **Vorschläge für Testverfahren:** GenAI kann relevante Testverfahren (z. B. Grenzwertanalyse, Äquivalenzklassenbildung) basierend auf der Art der zu testenden Anforderung oder User-Story vorschlagen. Dies kann Testern helfen, die effektivsten Testverfahren für bestimmte Testbedingungen anzuwenden.

Die Qualität und Relevanz der Eingaben, die dem LLM in Bezug auf die zu erledigende Aufgabe zur Verfügung gestellt werden, wirken sich direkt auf die Genauigkeit und Präzision der vom LLM generierten Ausgabe aus.

Praktische Übung zum Ziel HO-2.2.1a (H2): … selbstständig strukturiertes, multimodales Prompting anwenden, um Akzeptanzkriterien für eine User-Story auf der Grundlage eines GUI-Wireframes zu generieren.

Dies ist eine Übung zum Schreiben strukturierter Prompts unter Verwendung multimodaler Eingaben (Text und Bild). Das Ziel besteht darin, aus einer User-Story und einem GUI-Wireframe hochwertige (d. h. gut strukturierte, klare und vollständige) Akzeptanzkriterien zu erstellen. Es können weitere Textelemente hinzugefügt werden, um den Kontext zu verdeutlichen, z. B. Einschränkungen für Eingabefelder oder Geschäftsregeln, die auf die Datenverarbeitung anzuwenden sind.

Die vom LLM erzielten Ergebnisse werden verglichen, um die Auswirkungen verschiedener Formulierungen des strukturierten Prompts (Rolle, Kontext, Instruktion, Text- und Bild-Eingabedaten, Einschränkungen und Ausgabeformat) für eine Testanalyseaufgabe zu bewerten.

Diese Übung vermittelt praktische Erfahrungen hinsichtlich der Bedeutung der Prompt-Strukturierung, des Beitrags präziser Instruktionen und der Bedeutung sowohl textueller als auch visueller Kontextdaten für die Erzielung genauer und relevanter Ergebnisse aus dem LLM.

Praktische Übung zum Ziel HO-2.2.1b (H2): … selbstständig Prompt-Verkettung und die Verifizierung durch Menschen anwenden, um eine bestimmte User-Story schrittweise zu analysieren und die Akzeptanzkriterien zu verfeinern.

Diese Übung dient dazu, Prompt-Verkettung anzuwenden, um eine vorgegebene User-Story zu analysieren und die Akzeptanzkriterien zu verfeinern, indem zunächst Unklarheiten identifiziert und dann sowohl die Testbarkeit als auch schließlich die Vollständigkeit bewertet werden. Diese Übung fördert einen schrittweisen Ansatz, bei dem die Analyse in jedem Schritt verfeinert wird, um sicherzustellen, dass die Akzeptanzkriterien gut formuliert und umsetzbar sind, um die Testziele zu erreichen. Bei jedem Schritt werden die vom LLM gelieferten Ergebnisse manuell überprüft und gegebenenfalls korrigiert, entweder durch Anpassung der Ausgabe oder durch einen Prompt-Verkettungsprozess mit dem LLM. Auf diese Weise wird in der nächsten Stufe ein sauberes Ergebnis aus der vorherigen Stufe verwendet, um einen weiteren Aspekt der Verbesserung der Akzeptanzkriterien anzugehen.

Diese Übung vermittelt praktische Erfahrungen mit den Vorteilen, die sich daraus ergeben, eine komplexe Aufgabe in Teilaufgaben aufzuteilen, wobei die Ergebnisse jeder Stufe von Menschen verifiziert werden.

2.2.2 Testentwurf und Testrealisierung mit generativer KI

Wie in [ISTQB_CTFL_SYL] beschrieben, umfasst der Testentwurf die Ausarbeitung und Verfeinerung von Testbedingungen, die dann in Testfälle und andere Testmittel übersetzt werden. Die Testrealisierung beinhaltet die Erstellung oder Beschaffung der erforderlichen Testmittel zur Durchführung der Tests.

Sowohl manuelle Tests als auch automatisierte Testskripte können mit Unterstützung von GenAI erstellt, priorisiert und in einem Testausführungsplans berücksichtigt werden. GenAI kann eine Vielzahl dieser Testaktivitäten in großem Maße bei der Erstellung und Bewertung verschiedener Testmittel, darunter Testfälle, Testdaten, Testskripte und Testumgebungen, unterstützen.

Hier sind einige typische Aufgaben im Bereich Testentwurf und Testrealisierung aufgelistet, die von GenAI unterstützt werden können:

- **Testfallgenerierung:** Durch die Verarbeitung natürlicher Sprache kann GenAI Testfallentwürfe auf der Grundlage funktionaler und nicht-funktionaler Anforderungen erstellen. Bei Eingabe geeigneter Informationen kann ein LLM Testvorbedingungen und -eingaben, erwartete Ergebnisse und Überdeckungskriterien vorschlagen und so Testfälle erstellen, die unterschiedliche Testziele erfüllen, von der grundlegenden Funktionsüberprüfung bis hin zu komplexen Ende-zu-Ende-Tests.
- **Testdatensynthese:** GenAI kann repräsentative, datenschutzkonforme synthetische Testdaten erstellen, die den Produktionsdaten ähneln und extreme Situationen und unterschiedliche Testbedingungen abdecken. Diese synthetischen Testdaten können für funktionale und nicht-funktionale Tests verwendet werden. KI-generierte Testdaten können auf die Anwendungsanforderungen zugeschnitten werden und realistische Szenarien simulieren, ohne sensible Informationen preiszugeben.
- **Automatisierte Testskriptgenerierung:** GenAI kann manuelle Testabläufe und automatisierte Testskripte aus strukturierten Testfällen generieren, indem sie Testschritte interpretiert und in Code übersetzt, der mit verschiedenen Testautomatisierungsframeworks kompatibel ist. Diese Testskripte können aufgrund neuer Anforderungen aktualisiert oder erweitert werden.
- **Planung und Priorisierung der Testdurchführung:** GenAI kann Testfälle und ihre gegenseitigen Abhängigkeiten analysieren und die Testausführungspläne auf der Grundlage von Prioritäten, damit verbundenen Risiken, Ressourcenverfügbarkeit und Testzielen optimieren.

Praktische Übung zum Ziel HO-2.2.2a (H2): … selbstständig mithilfe von Prompt-Verkettung, strukturierten Prompts und Meta-Prompting aus User-Storys funktionale Testfälle erstellen.

Diese Übung konzentriert sich auf die Entwicklung funktionaler Testfälle aus User-Storys mit GenAI unter Verwendung von Prompt-Verkettung, strukturierten Prompts und Meta-Prompting, um eine umfassende Überdeckung sicherzustellen. Der erste Schritt besteht darin, einen Prompt zu erstellen, der die KI anweist, funktionale Testfälle auf der Grundlage vorgegebener Akzeptanzkriterien in einem bestimmten Ausgabeformat zu generieren. In einem zweiten Schritt ist die Vollständigkeit der generierten Testfälle zu verifizieren. Hier überprüft der Prompt, ob jedes Akzeptanzkriterium abgedeckt ist, indem die KI eine Tabelle generiert, die die Überdeckung zusammenfasst. Der dritte Schritt besteht schließlich darin, einen Meta-Prompt zu erzeugen, um die Erstellung von Ende-zu-Ende-Testabläufen zu unterstützen. Dieses Meta-Prompting hilft dabei, den Prompt zu verfeinern, um umfassende Ende-zu-Ende-Tests zu generieren, und unterstützt iterative Verbesserungen, um die Effektivität zu maximieren.

Diese Übung verbessert das Verständnis für die Verwendung von LLMs zur Generierung von Testfällen, zur Validierung der Überdeckung und für Ende-zu-Ende-Tests.

Praktische Übung zum Ziel HO-2.2.2b (H2): … selbstständig Few-Shot-Prompting anwenden, um aus User-Storys Testbedingungen und Testfälle im Gherkin-Stil zu generieren.

In dieser Übung geht es darum, mithilfe von Few-Shot-Prompting aus vorgegebenen User-Storys Testfälle im Gherkin-Stil zu erstellen. Ausgehend von einem Review vordefinierter Beispiele und der Gherkin-Syntax wird in einem ersten Schritt eine bestimmte Anzahl n an Beispielen ausgewählt, die in den Prompt aufgenommen werden sollen, jeweils mit einer User-Story, Testbedingungen und erwarteten Testfällen im Given-When-Then-Stil, um die gewünschte Ausgabe zu modellieren. Dieser Prompt wird dann auf eine neue User-Story angewendet, wodurch Gherkin-Szenarien generiert werden, die die ursprünglichen Testbedingungen widerspiegeln. Wenn die Ergebnisse ungenau sind, sollten der Prompt oder die Beispiele verfeinert werden.

Diese Übung hilft dabei, Erfahrungen mit der Anwendung von Few-Shot-Prompting-Verfahren auf realistische Testentwurf- und Testrealisierungsaufgaben zu sammeln.

Praktische Übung zum Ziel HO-2.2.2c (H2): … selbstständig Prompt-Verkettung anwenden, um Testfälle innerhalb einer bestimmten Testsuite unter Berücksichtigung ihrer spezifischen Prioritäten und Abhängigkeiten zu priorisieren.

Diese Übung konzentriert sich auf die Verwendung von GenAI, um die Priorisierung von Testfällen innerhalb einer bestimmten Testsuite mit zugehöriger Risikoanalyse und Abhängigkeiten zwischen Testfällen zu verbessern. Die Übung beginnt mit einem kurzen Überblick über verschiedene Testansätze, wie z. B. risikobasierte, überdeckungsbasierte und anforderungsbasierte Vorgehensweisen, sowie einer Überprüfung der vorgegebenen Testsuite. Anschließend erstellen die Teilnehmer Prompts, um umsetzbare Priorisierungspläne für verschiedene Testpriorisierungsstrategien zu erstellen. Die Ergebnisse des LLMs, die auf dem Prompt und den vorgegebenen Eingabedaten basieren, sollten manuell überprüft werden, um etwaige Fehler in der logischen Schlussfolgerung des LLMs zu entdecken.

Das Ziel dieser Übung besteht darin, mit GenAI bei Testaufgaben zu experimentieren, die Fähigkeiten zu einer Abwägung mehrerer Kriterien bei der Entscheidungsfindung (*multi-criteria reasoning*) erfordern (hier: die Berücksichtigung verschiedener Risiken und Abhängigkeiten, die bei der Priorisierung von Testfällen zu berücksichtigen sind).

2.2.3 Automatisierte Regressionstests mit generativer KI

Mit jeder neuen Iteration oder jedem neuen Release steigt oft die Anzahl der auszuführenden Regressionstests. Dies macht sie zu idealen Kandidaten für die Automatisierung, insbesondere in Continuous Integration/Continuous Delivery (CI/CD-)Pipelines aufgrund der hohen Häufigkeit der Testdurchführung. GenAI kann diesen Prozess rationalisieren, indem sie bei der Erstellung, Wartung und Optimierung automatisierter Regressionstestsuiten hilft. Durch die dynamische Anpassung an Änderungen im Codebestand und die Durchführung von Auswirkungsanalysen kann GenAI ermitteln, welche Bereiche der Software am ehesten von den jüngsten Änderungen betroffen sind, und die Regressionstests auf die Stellen fokussieren, an denen sie am dringendsten benötigt werden.

Nachfolgend sind einige typische automatisierte Regressionstests und Testberichterstattungsaktivitäten aufgeführt, die durch GenAI-Prompting unterstützt werden können:

- **Implementierung automatisierter Testskripte mit schlüsselwortgetriebener Automatisierung:** LLMs können zur Implementierung von Testskripten auf Basis schlüsselwortgetriebener Testautomatisierungsframeworks verwendet werden, wobei vordefinierte Schlüsselworte gängige Testschritte darstellen. GenAI kann diese Schlüsselworte bestimmten Testfällen zuordnen, Testskripte generieren und dadurch Tester sowie Testautomatisierungsentwickler bei ihrer Arbeit unterstützen.

- **Auswirkungsanalyse und Testoptimierung:** GenAI kann zur Analyse von Codeänderungen verwendet werden, um Bereiche mit hohem Risiko zu identifizieren und so gezielte Regressionstests dort zu ermöglichen, wo sie am dringendsten benötigt werden.
- **Selbstheilende und adaptive Tests:** GenAI kann verwendet werden, um Testskripte automatisch anzupassen, um kleinere Änderungen an der Benutzungsoberfläche oder der API zu bewältigen. Dadurch werden unnötige Fehler aufgrund kleiner Änderungen vermieden und es wird sichergestellt, dass die Testsuite über einen längeren Zeitraum stabil bleibt.
- **Automatisierte Testberichterstattung und Erkenntnisse:** GenAI ermöglicht die Erstellung detaillierter, zeitnah verfügbarer Testberichte mit Erfolgsmetriken, Fehlerwirkungen und wichtigen Erkenntnissen. Somit stellt es Stakeholdern Dashboards zur Verfügung, die Testtrends hervorheben und vorausschauenden Einblicke in potenzielle Fehlerquellen liefern.
- **Verbesserte Fehlerberichterstattung und Grundursachenanalyse:** GenAI kann die automatische Erstellung umfassender Fehlerberichte mit Testprotokollen, Screenshots und Testumgebungsdaten unterstützen.

Diese Aktivitäten können auf eine Vielzahl von Regressionstests angewendet werden, darunter funktionale und nicht-funktionale Regressionstests. Die Tester müssen sich jedoch bewusst sein, dass GenAI Fehler machen kann. Die generierten Ergebnisse müssen daher abhängig von dem damit verbundenen Risiko sorgfältig überprüft werden (siehe Kapitel 3).

Darüber hinaus kann GenAI Ende-zu-Ende-GUI-Tests und API-basierte automatisierte Regressionstests unterstützen, die jeweils ihre eigenen Herausforderungen und Lösungen mit sich bringen. GUI-Tests werden häufig durch wiederkehrende Änderungen an der Benutzungsschnittstelle instabil. GenAI kann Testskripte automatisch anpassen, um mit Änderungen wie dynamischen Identifikatoren und geänderten Interaktionen umzugehen, wodurch der Bedarf an manuellen Eingriffen reduziert wird. API-Regressionstests stehen vor Herausforderungen wie den sich ändernden Anfrage-/Antwortformaten, Endpunkten und Authentifizierungen. GenAI kann Testskripte automatisch an sich ändernde API-Spezifikationen anpassen und vielfältige Testdaten generieren, wodurch eine umfassende Überdeckung gewährleistet und der Bedarf an manuellen Aktualisierungen reduziert wird.

Praktische Übung zum Ziel HO-2.2.3a (H2): … selbstständig Few-Shot-Prompting zur Erstellung und Verwaltung von schlüsselwortgetriebenen Testskripten anwenden.

Diese Übung konzentriert sich auf die Entwicklung und Automatisierung von Testskripten für eine bestimmte Webanwendung unter Verwendung eines GUI-Testautomatisierungsframeworks. Die Übung gliedert sich in zwei Hauptabschnitte: Testautomatisierung und Debugging von Testskripten. Der erste Teil der Übung enthält Anleitungen zum Erstellen einer Dokumentation für eine Schlüsselwortbibliothek, zum Generieren initialer Testskripte, zum Validieren dieser Testskripte durch KI und zum Erweitern der Überdeckung durch zusätzliche Testskripte. Der zweite Teil konzentriert sich auf die Debugging-Unterstützung, wobei mithilfe von System-Prompts ein KI-Assistent erstellt wird, der Testskripte überprüfen und korrigieren kann.

Diese Übung kombiniert traditionelle Testautomatisierung mit KI-gestützter Validierung und zeigt, wie Few-Shot-Prompting effektiv zum Erstellen, Warten und Debugging von schlüsselwortgetriebenen Testskripten eingesetzt werden kann.

Praktische Übung zum Ziel HO-2.2.3b (H2): … selbstständig strukturiertes Prompt-Engineering für die Analyse von Testberichten im Regressionstest anwenden.

Diese Übung veranschaulicht einen methodischen Ansatz zur Analyse von Regressionstestberichten unter Verwendung strukturierter Prompts. Der Prozess beginnt mit einer Analyse der bereitgestellten Testergebnisse und einem Vergleich mit der Testspezifikation. Anschließend erfolgt die Gruppierung

ähnlicher Fehler, die Pflege einer Liste bekannter Anomalien und eine Gegenüberstellung der Befunde. Jeder Schritt ist mit dem nächsten in einer einzigen LLM-Konversation verknüpft.

Der schrittweise Ansatz zeigt, wie strukturierte Prompts verwendet werden können, um Regressionstestergebnisse und Testprotokolle in umsetzbare Erkenntnisse umzuwandeln und so eine effektive Analyse von Testberichten im Rahmen von Regressionstests zu unterstützen.

2.2.4 Testüberwachung und Teststeuerung mit generativer KI

Testüberwachungsaufgaben erfordern das Abrufen großer Mengen von (teilweise unstrukturierten) Daten, die oft bereits in Testmanagement-Werkzeugen verfügbar sind und von GenAI analysiert und aufbereitet werden können.

GenAI erleichtert eine Reihe von Testüberwachungs- und Teststeuerungsaufgaben, darunter:

- **Testüberwachung und Metrikanalyse:** GenAI kann die Automatisierung der Testüberwachung sowie die Analyse von Trends erleichtern, um potenzielle Risiken vorherzusagen und Teams auf Abweichungen vom Plan aufmerksam zu machen. So bleiben Teams auf dem Laufenden und können Maßnahmen zur Aufrechterhaltung der Qualität ergreifen.
- **Teststeuerung:** GenAI kann bei der Teststeuerung unterstützen, indem sie Erkenntnisse für die Neupriorisierung von Tests, die Anpassung von Testplänen und die Umverteilung von Ressourcen nach Bedarf bereitstellt. Dadurch wird sichergestellt, dass die Tests ihre Flexibilität beibehalten und sich auf Bereiche mit hoher Priorität konzentrieren.
- **Einblicke in den Testabschluss und kontinuierliches Lernen:** GenAI kann durch die Erstellung von Testabschlussberichten, in denen erfolgreiche Abschlussergebnisse und gewonnene Erkenntnisse hervorgehoben werden, hilfreich sein. Auf diese Weise können Teams Teststrategien verfeinern und zukünftige Testprozesse verbessern.
- **Verbesserte Visualisierung und Berichterstattung von Testmetriken:** GenAI kann bei der Erstellung dynamischer Dashboards und Zusammenfassungen in natürlicher Sprache helfen und so sicherstellen, dass alle Beteiligten Zugriff auf die relevanten Metriken haben. Diese Unterstützung liefert die Informationen, die für schnelle Entscheidungen erforderlich sind, und bietet einen klaren Blick auf den Testfortschritt.

Praktische Übung zum Ziel HO-2.2.4 (H0): … die von generativer KI aus Testdaten erstellten Testüberwachungsmetriken erläutern.

Diese Demonstration zeigt, wie GenAI Testteams unterstützen kann, indem sie Testdaten in umsetzbare Testüberwachungsmetriken umwandelt und so fundierte Entscheidungen ermöglicht. Ein LLM verarbeitet Testdaten, die aus Testwerkzeugen extrahiert wurden, um wichtige Metriken wie Testfortschritt, Fehlertrends oder Überdeckung zu generieren und potenzielle Risiken hervorzuheben. Diese KI-generierten Metriken können dann auf einem Dashboard angezeigt und in natürlicher Sprache zusammengefasst werden, damit sie für alle Stakeholder leicht verständlich sind.

Diese Vorführung veranschaulicht, wie GenAI Testdaten in praktische Erkenntnisse umwandelt und Testteams dabei unterstützt, den Testfortschritt zu überwachen, Qualität sicherzustellen und sich schnell an Änderungen anzupassen.

2.2.5 Auswahl von Prompting-Verfahren für Softwaretests

Die folgende Tabelle zeigt die Eignung der drei in Abschnitt 2.1.2 genannten Prompting-Verfahren entsprechend den Merkmalen der Testaufgabe.

Prompting-Verfahren	Empfohlener Anwendungsfall	Zentrale Funktionen und Einsatzmöglichkeiten
Prompt-Verkettung	Komplexe Aufgaben, die Präzision erfordern und bei denen jeder Schritt von einem Menschen verifiziert werden muss	Teilt Aufgaben in kleinere Schritte auf, nützlich für Testanalyse, Testentwurf und Testautomatisierung, bei denen jeder Testschritt auf Genauigkeit überprüft wird.
Few-Shot-Prompting	Wiederholende oder auf ein bestimmtes Ausgabeformat beschränkte Aufgaben	Stellt GenAI Beispiele für die wiederholte Generierung mit einem bestimmten Muster zur Verfügung, beispielsweise Testfälle im Gherkin-Format (z. B. szenariobasierter Test), schlüsselwortgetriebener Test oder Testberichterstattung mit einem bestimmten Ausgabeformat.
Meta-Prompting	Flexible, dynamische Aufgaben, nützlich für die Erstellung von Prompts für neue Aufgaben	Allgemeine Beschreibung des Ziels und der auszuführenden Aufgabe, die das LLM bei der Erstellung des Prompts anleitet. Nützlich für alle Arten komplexer Aufgaben wie Testberichtanalyse und Anomalieerkennung.

Es ist sogar möglich, mehrere Prompting-Verfahren für einen einzigen Anwendungsfall zu verwenden. Beispielsweise kann Meta-Prompting verwendet werden, um einen ersten Prompt zu erstellen. Dieser generierte Prompt kann Beispiele enthalten, die angepasst und verbessert werden müssen (Few-Shot-Prompting). Schließlich kann es sinnvoll sein, die Aufgabe in kleinere Teilaufgaben zu unterteilen, um die Validierung der Zwischenschritte zu ermöglichen (Prompt-Verkettung).

Praktische Übung zum Ziel HO-2.2.5 (H1): … angeleitet kontextgerechte Prompting-Verfahren für eine vorgegebene Testaufgabe auswählen und anwenden.

Diese Übung konzentriert sich auf die Auswahl geeigneter Prompting-Verfahren für verschiedene Testaufgaben. Die Teilnehmer erhalten mehrere Testaufgaben mit unterschiedlichen Herausforderungen. Für jede Testaufgabe sollten die Teilnehmer die Art der Aufgabe bewerten – ob sie Präzision oder eine wiederholende Struktur erfordert – und das bzw. die Prompting-Verfahren(en) vorschlagen, die am besten zum Kontext passen und den spezifischen Anforderungen der Aufgabe entsprechen. Die Auswahlmöglichkeiten werden in der Gruppe diskutiert.

Diese Übung soll das Verständnis dafür vertiefen, wie verschiedene Prompting-Verfahren im praktischen Testeinsatz effektiv eingesetzt werden können.

2.3 Bewertung generativer KI-Ergebnisse und Verfeinerung von Prompts für Software-Testaufgaben

Die Bewertung der Leistung von GenAI im Softwaretest erfordert klare Metriken, um die Qualität, Relevanz und Effektivität der generierten Ergebnisse zu beurteilen (Li 2024). Diese Metriken, ob allgemein oder aufgabenspezifisch, helfen bei der Optimierung des LLM-Prompting.

2.3.1 Metriken zur Bewertung der Ergebnisse von generativer KI bei Testaufgaben

Zur Bewertung der Qualität und Effizienz der Ergebnisse von GenAI bei einer Testaufgabe können mehrere Metriken herangezogen werden:

Metrik	**Beschreibung**	**Beispiel**
Genauigkeit (*accuracy*)	Misst die allgemeine Korrektheit der generierten Ausgabe anhand von Experten verfassten Testfällen, Anforderungen, oder anderen Standards.	Der Grad, in dem die generierten Testfälle alle spezifizierten Anforderungen abdecken.
Präzision (*precision*)	Bewertet die Korrektheit der generierten Ausgabe in Bezug auf ein bestimmtes Ziel.	Der Grad, in dem die generierten Testfälle Anomalien korrekt identifizieren.
Sensitivität (*recall*)	Misst die Fähigkeit eines Modells, alle relevanten Instanzen innerhalb eines Datensatzes zu identifizieren.	Der Grad, in dem generierte Testfälle gültige und ungültige Äquivalenzklassen einer Datenkategorie abdecken.
Relevanz und Kontextangemessenheit	Bestimmt, ob die generierte Ausgabe für einen bestimmten Kontext anwendbar und geeignet ist.	Der Grad, in dem die generierten Testfälle mit der Testbasis übereinstimmen und die domänenspezifischen Anforderungen integrieren.
Vielfalt (*diversity*)	Stellt sicher, dass eine Vielzahl von Eingaben und Szenarien abgedeckt wird, wobei Wiederholungen vermieden werden.	Der Grad, in dem die generierten Testfälle verschiedene Benutzerverhalten abdecken und Grenzfälle untersuchen.
Ausführungserfolgsrate	Misst den Anteil der generierten Testfälle oder Testskripte, die erfolgreich ausgeführt werden können.	Ermittelt, wie viele der generierten Testskripte ohne Syntaxfehler oder Probleme im Ausgabeformat in einer ansonsten funktionierenden Testumgebung ausgeführt werden können.
Zeiteffizienz	Bewertet die Zeitersparnis im Vergleich zum manuellen Testaufwand.	Die von der KI benötigte Zeit zur Generierung von Testfällen im Vergleich zu der Zeit, die ein Mensch für die manuelle Erstellung gleichwertiger Tests benötigen würde.

Zusätzlich zu diesen allgemeinen Metriken können aufgabenspezifische Metriken zugeschnitten werden, um zu bewerten, wie gut die GenAI bestimmte Testaktivitäten unterstützt.

Um diese Metriken effektiv zu bewerten, können Tester manuelle Reviews durchführen oder diese automatisieren, z. B. durch den Vergleich der LLM-Ausgabe mit einer vordefinierten Referenz. Angesichts der nicht-deterministischen Natur von GenAI müssen die Metriken auf statistisch relevanten Daten basieren.

Praktische Übung zum Ziel HO-2.3.1 (H0): … erläutern, wie Metriken zur Bewertung des Ergebnisses generativer KI bei einer Testaufgabe verwendet werden können.

Während der Vorführung einer vorgegebenen Testaufgabe werden aufgabenangepasste Metriken zur Bewertung der GenAI-Ergebnisse vorgestellt und deren konkrete Anwendung auf die mit einem LLM erzielten Ergebnisse für diese Testaufgabe gezeigt.

Diese Vorführung veranschaulicht die Bedeutung von Evaluationsmetriken, um das Vertrauen in die Ergebnisse generativer KI für Softwaretesten zu schaffen.

2.3.2 Verfahren zur Bewertung und iterativen Verfeinerung (Refinement) von Prompts

Aufbauend auf den oben dargestellten Metriken werden spezifische Verfahren zur Bewertung und Verfeinerung von Prompts eingesetzt, um die KI-Ergebnisse zu verbessern:

- **Iterative Prompt-Änderung:** Ausgangspunkt ist ein Basis-Prompt, der schrittweise auf Grundlage der beobachteten Ergebnisse angepasst wird. Dabei kann nach und nach mehr Kontext hinzugefügt oder die Formulierung (z. B. hinsichtlich der Terminologie) angepasst werden, um Spezifität und Relevanz zu erhöhen.
- **A/B-Tests von Prompts:** Es werden mehrere Versionen eines Prompts erstellt und anhand vordefinierter Metriken wird bewertet, welche Version die besseren Ergebnisse liefert. Dieser Ansatz hilft dabei, zu ermitteln, welche Formulierung oder Struktur eines Prompts die genauesten und relevantesten Ergebnisse liefert.
- **Analyse der Ausgabe:** Die von der KI generierten Ausgaben werden auf Unrichtigkeiten oder Inkonsistenzen untersucht, z. B. in Bezug auf die Testbasis. Das Verständnis der Arten von Fehlern und Inkonsistenzen kann dazu beitragen, die Prompts zu verfeinern, um ähnliche Fehlerzustände in zukünftigen Iterationen zu vermeiden.
- **Einbindung von Benutzerfeedback:** Rückmeldungen von Testern zur Zweckmäßigkeit und Verständlichkeit der generierten Ausgaben, z. B. hinsichtlich des Detaillierungsgrads der generierten Tests, werden gesammelt und analysiert. Diese Erkenntnisse dienen dazu, um Prompts zu verfeinern und die realen Testanforderungen besser zu erfüllen.
- **Anpassung der Prompt-Länge und -Spezifität:** Durch Experimentieren mit unterschiedlichen Prompt-Längen und Detailstufen lässt sich die Antwortqualität optimieren. In manchen Fällen verbessert zusätzlicher Kontext die Qualität der Ergebnisse, während in anderen Situationen kürzere Prompts zu einer besseren Verallgemeinerung führen können.

Mithilfe dieser Verfahren können Testteams Sitzungen zur Bewertung und Optimierung von Prompts organisieren, um eine kontinuierliche Verbesserung der GenAI-Prompts sicherzustellen. Der Austausch von Praktiken innerhalb des Testteams oder der Testorganisation trägt nicht nur zur Standardisierung der Prompting-Verfahren und zur Aufrechterhaltung einer gleichbleibenden Qualität bei, sondern fördert auch eine Kultur des Lernens und der iterativen Verbesserung. Dieser kollaborative Ansatz trägt zur

Evolution der GenAI-Testmethodiken bei, indem er es den Testteams ermöglicht, auf kollektiven Erkenntnissen aufzubauen, wiederholte Fehlhandlungen zu vermeiden und den Einsatz von GenAI-Werkzeugen im Laufe der Zeit effektiver zu verfeinern, z. B. durch das Teilen von Prompt-Bibliotheken.

Praktische Übung zum Ziel HO-2.3.2 (H1): … angeleitet einen Prompt für eine vorgegebene Testaufgabe bewerten und optimieren.

Diese Übung konzentriert sich auf die Anwendung von Verfahren zur Prompt-Optimierung für eine vorgegebene Testaufgabe. Die Teilnehmer beginnen mit einem initialen Prompt und verfeinern diesen iterativ, um die von der KI generierten Ergebnisse zu verbessern. Dabei verwenden sie Verfahren wie A/B-Test und manuelle Verifizierung, um die Prompt-Qualität zu bewerten und zu verbessern.

Ziel ist es, dass die Teilnehmer erleben, wie iterative Verfeinerung zu einer effektiveren und kontextuell relevanteren Testfallgenerierung führt.

Am Ende der Übung haben die Teilnehmer mehrere Iterationen der Prompt-Verfeinerung durchgeführt und jede Iteration anhand der besprochenen Metriken bewertet, um die Qualität der KI-Ausgabe zu verbessern.

3 Management von Risiken bei generativer KI im Softwaretest – 160 Minuten

Schlüsselbegriffe

Datenschutz, Sicherheit, Sicherheitslücke

Spezifische Schlüsselbegriffe für generative KI

Halluzination, Reasoning-Fehler (*reasoning error*), Temperatur, Verzerrung (*bias*)

Lernziele und Ziele der praktischen Übungen für Kapitel 3: Die Lernenden können …

3.1 Halluzinationen, Reasoning-Fehler und Verzerrungen

GenAI-3.1.1	(K1)	… Definitionen von Halluzinationen, Reasoning-Fehlern und Verzerrungen in Systemen mit generativer KI wiedergeben.
GenAI-3.1.2	(K3)	… Halluzinationen, Reasoning-Fehler und Verzerrungen in der LLM-Ausgabe identifizieren.
HO-3.1.2a	(H1)	… angeleitet mit Halluzinationen beim Testen mit GenAI experimentieren.
HO-3.1.2b	(H1)	… angeleitet mit Reasoning-Fehlern beim Testen mit GenAI experimentieren.
GenAI-3.1.3	(K2)	… Verfahren zur Minderung von GenAI-Halluzinationen, Reasoning-Fehlern und Verzerrungen bei Software-Testaufgaben zusammenfassen.
GenAI-3.1.4	(K1)	…Verfahren zur Risikominderung für nicht-deterministisches Verhalten von LLMs wiedergeben.

3.2 Datenschutz- und Sicherheitsrisiken generativer KI im Softwaretest

GenAI-3.2.1	(K2)	… die wichtigsten Risiken für den Datenschutz und die Sicherheit (security) im Zusammenhang mit der Verwendung von generativer KI im Softwaretest erläutern.
GenAI-3.2.2	(K2)	… Beispiele für Datenschutz und Sicherheitslücken bei der Verwendung von generativer KI im Softwaretest nennen.
GenAI-3.2.3	(K2)	… Strategien zur Minderung von Risiken zusammenfassen, um beim Einsatz von generativer KI im Softwaretest den Datenschutz zu wahren und die Sicherheit zu erhöhen.
HO-3.2.3	(H0)	… in einer vorgegebenen Fallstudie zum Einsatz von generativer KI Datenschutz- und Sicherheitsrisiken im Test erkennen.

3.3 Energieverbrauch und Umweltauswirkungen von generativer KI im Softwaretest

GenAI-3.3.1	(K2)	… die Auswirkungen von Aufgabenmerkmalen und Modellnutzung auf den Energieverbrauch von generativer KI im Softwaretest erläutern.
HO-3.3.1	(H1)	… angeleitet einen Simulator verwenden, um den Energieverbrauch und die CO_2-Emissionen für vorgegebene Testaufgaben mit generativer KI zu berechnen.

3.4 KI-Vorschriften, Standards und Best-Practice-Rahmenwerke

GenAI-3.4.1	(K1)	… Beispiele für KI-Vorschriften, Standards und Best-Practice-Rahmenwerke nennen, die für den Einsatz von generativer KI im Softwaretest relevant sind.

3.1 Halluzinationen, Reasoning-Fehler und Verzerrungen

GenAI-Systeme, insbesondere LLMs, sind anfällig für bestimmte Fehlerzustände, darunter Halluzinationen, Reasoning-Fehler und Verzerrungen. Diese Fehlerzustände mindern die Qualität des GenAI-Ergebnisses bei Testaufgaben, was dazu führt, dass die generierten Testmittel nicht den Erwartungen der Tester entsprechen. Diese Halluzinationen, Reasoning-Fehler und Verzerrungen müssen von den Testern im LLM-Ergebnis identifiziert werden, und es sollten Maßnahmen ergriffen werden, um diese Risiken zu mindern.

Das nicht-deterministische Verhalten von LLMs (siehe Abschnitt 1.1.2) macht es schwierig, diese Art von Fehlerzuständen zu beheben. Oft scheinen sie für ein LLM-Ergebnis behoben zu sein, tauchen dann aber in einer anderen Konversation mit demselben LLM wieder auf.

3.1.1 Halluzinationen, Reasoning-Fehler und Verzerrungen bei generativer KI

Halluzinationen treten auf, wenn ein LLM eine Ausgabe generiert, die sachlich falsch oder für eine bestimmte Aufgabe irrelevant erscheint. Im Softwaretest können sich Halluzinationen darin äußern, dass LLMs fiktive oder irrelevante Testfälle erstellen, fehlerhafte oder nicht funktionierende Testskripte generieren oder Testfälle vorschlagen, die nicht existierende Akzeptanzkriterien überprüfen. Dies kann Tester in die Irre führen und die Gültigkeit der Testergebnisse beeinträchtigen.

Reasoning-Fehler treten auf, wenn LLMs logische Strukturen falsch interpretieren, z. B. Ursache-Wirkungs-Beziehungen, bedingte Logik oder schrittweise Problemlösungsprozesse, was zu falschen Schlüssen führt. Im Gegensatz zu Menschen verfügen LLMs nicht über echtes logisches Denken, sondern verlassen sich auf Erkennung von Mustern, was zu fehlerhaften logischen Schritten bei der Lösung von Aufgaben wie mathematischem Schlussfolgern führen kann (Mirzadeh 2024). Testplanung und Testfallpriorisierung sind Beispiele für Testaufgaben, die logisches Denken erfordern und bei denen LLMs zu Reasoning-Fehlern neigen können.

LLM-Verzerrungen (Gallegos 2024) stammen aus den Daten, mit denen das Modell trainiert wurde. Diese Verzerrungen können zu Ergebnissen führen, die bestimmte Arten von Informationen, Ansätzen oder Annahmen bevorzugen. Beispielsweise können LLMs, die hauptsächlich mit englischsprachigen Daten trainiert wurden, nicht-englische Perspektiven unterrepräsentieren. Im Softwaretest können Verzerrungen die LLM-Antworten beeinflussen, wenn beispielsweise Testdaten generiert oder Akzeptanzkriterien für Testfälle verfeinert werden.

Die Halluzinationen, Reasoning-Fehler und Verzerrungen in den Ausgaben von GenAI resultieren aus der Beschaffenheit ihrer Trainingsdaten und den inhärenten Einschränkungen des Transformer-Modells (siehe Kapitel 1). Das Erkennen und Bewältigendieser Herausforderungen erhöht die Qualität der Ergebnisse generativer KI in Testprozessen.

3.1.2 Identifikation von Halluzinationen, Reasoning-Fehlern und Verzerrungen in LLM-Ergebnissen

Die effektive Integration von GenAI-Systemen in Softwaretests erfordert die Fähigkeit, Halluzinationen, Reasoning-Fehler und Verzerrungen in LLM-Ergebnissen zu erkennen. Je nach Art des Problems können unterschiedliche Ansätze zur Erkennung angewendet werden. Im Folgenden sind gängige Ansätze aufgeführt, die mittels Review oder einer Kombination aus Review und automatisierter Verifizierung angewendet werden:

Erkennung von Halluzinationen:

- Cross-Verifizierung: Vergleich von KI-generierten Ergebnissen mit vorhandenen Dokumentationen, Anforderungen und bekanntem Systemverhalten. Automatisierte Werkzeuge können dabei helfen, die Ergebnisse mit etablierten Datenquellen abzugleichen, um Abweichungen zu kennzeichnen.

- Konsultation von Fachexperten: Einbeziehung von Fachexperten, um die Genauigkeit der generierten Inhalte zu überprüfen. Das Fachwissen dieser Experten ist unerlässlich, um präzise Erkenntnisse zu gewinnen, die automatisierte Systeme möglicherweise übersehen.
- Konsistenzprüfungen: Überprüfung, ob die generierten Ergebnisse untereinander und mit bekannten Informationen übereinstimmen. Automatisierte Systeme können dabei helfen, Muster zu erkennen und Inkonsistenzen zu markieren.

Erkennung von Reasoning-Fehlern:

- Logische Validierung: Bewertung des logischen Ablaufs (z. B. die Konsistenz, Kohärenz und strukturierten Argumentationen innerhalb des generierten Textes) von KI-generierten Inhalten hinsichtlich Kohärenz und Korrektheit durch Reviewzyklen. Automatisierte Werkzeuge können dabei helfen, aber komplexe Fälle erfordern möglicherweise menschliches Urteilsvermögen.
- Testen der Ausgaben: Zum Beispiel Ausführung der generierten Testfälle oder Testskripte an den Testobjekten, um die Testergebnisse zu überprüfen. Dies kann je nach Art der generierten Testmittel teilweise oder vollständig automatisiert erfolgen.

Erkennung von Verzerrungen:

- Überprüfung, ob generierte Testmittel wie z. B. synthetische Testdaten bezogen auf die Teststrategie fair und genau sind.
- Bewertung von Verzerrungen in Bezug auf Testarten, wie z. B. unterrepräsentierte nicht-funktionale Tests in der generierten Ausgabe des LLMs.

Die tatsächliche Umsetzung dieser Nachweismethoden hängt von der geschätzten Risikostufe von Halluzinationen, Reasoning-Fehlern oder Verzerrungen bei der mit GenAI durchgeführten Testaufgabe ab.

Praktische Übung zum Ziel HO-3.1.2a (H1): … angeleitet mit Halluzinationen beim Testen mit GenAI experimentieren.

Diese Übung konzentriert sich auf das Experimentieren mit Beispielen für Halluzinationen von GenAI in Bezug auf das Wissen über Softwaretests. Die Teilnehmenden versuchen, mindestens zwei LLMs mit einer Situation zu konfrontieren, in der die LLMs irrelevante Elemente erfinden, z. B. indem sie nicht vorhandene Kriterien hinzufügen, die in den gegebenen Kontextdaten nicht vorkommen. Variationen im Prompting werden getestet, um den Einfluss des Promptings auf Halluzinationen zu untersuchen.

Diese Übung verbessert das Verständnis für die Identifizierung von GenAI-Halluzinationen im Softwaretest.

Praktische Übung zum Ziel HO-3.1.2b (H1): … angeleitet mit Reasoning-Fehlern beim Testen mit GenAI experimentieren.

Diese Übung konzentriert sich auf die Darstellung eines Beispiels für einen Reasoning-Fehler einer GenAI. Ein Beispiel für ein zu lösendes Problem im Bereich der Testplanung, wie z. B. die Schätzung des Testaufwands und die Priorisierung von Testfällen (siehe [ISTQB_CTFL_SYS], Kapitel 5). Die Übung ist mit einer gewissen Komplexität der Eingabedaten entworfen, die Problemlösungsfähigkeiten erfordert und die Grenzen von LLMs für diesen Zweck aufzeigt. Das Ergebnis des LLMs wird mit dem exakten Ergebnis verglichen, das erreicht werden sollte. Es werden drei verschiedene LLM-Typen ausprobiert (LLM, SLM und Reasoning-Modell), und es werden Variationen des Prompts verwendet, um zu versuchen, die Ergebnisse zu verbessern.

Diese Übung verbessert das Verständnis dafür, wie man Reasoning-Fehler von GenAI in Software-Testaufgaben identifiziert, die logische Problemlösungsfähigkeiten erfordern.

3.1.3 Verfahren zur Minderung von Halluzinationen, Reasoning-Fehlern und Verzerrungen bei generativer KI in Software-Testaufgaben

Um unerwünschte Ergebnisse der GenAI im Softwaretest zu minimieren, können verschiedene Strategien eingesetzt werden, um Halluzinationen, Reasoning-Fehler und Verzerrungen zu reduzieren. Diese Probleme treten überwiegend auf, wenn die Prompts nicht richtig entworfen sind (siehe Kapitel 2) oder wenn für eine bestimmte Testaufgabe relevante, kontextbezogene Eingabedaten fehlen. Zu den wichtigsten Verfahren zur Minderung des Risikos im Zusammenhang mit KI-Halluzinationen, Reasoning-Fehlern und Verzerrungen gehören:

- Vollständigen Kontext bereitstellen: Sicherstellen, dass der Prompt alle relevanten Informationen enthält (siehe Abschnitt 2.1.1) und einen umfassenden Kontext bietet, um die KI bei der Erstellung genauer Ergebnisse zu unterstützen.
- Aufteilung von Prompts in handhabbare Segmente: Zerlegung von komplexen Prompts mithilfe von Verfahren zur Prompt-Verkettung (siehe Abschnitt 2.1.2) in kleinere Schritte und systematische Überprüfung jeder Ausgabe vor dem nächsten Schritt. Dieser schrittweise Ansatz kann dazu beitragen, Reasoning-Fehler frühzeitig im Generierungsprozess zu erkennen.
- Verwendung von klaren, interpretierbaren Datenformaten: Vermeidung von Formaten, die für die GenAI möglicherweise mehrdeutig oder schwer zu interpretieren sind. Strukturierte, übersichtliche Formate helfen dem Modell, sich auf die wesentlichen Aspekte der Aufgabe zu konzentrieren.
- Auswahl des für die Aufgabe geeigneten GenAI-Modells: Verwendung eines LLMs, das speziell für die jeweilige Aufgabe trainiert wurde (siehe Abschnitt 5.1.3).
- Vergleichen der Ergebnisse verschiedener Modelle: Gegebenenfalls hilft die Bewertung eines Prompts mit mehreren LLMs und der Vergleich der Ergebnisse dabei, Ausgabefehler zu erkennen und die zuverlässigsten Ergebnisse auszuwählen.

In Kapitel 4 werden zwei sich ergänzende Verfahren zur Verbesserung der LLM-Ergebnisse vorgestellt: Retrieval-augmentierte Generierung und Fein-Tuning.

3.1.4 Minderung des nicht-deterministischen LLM-Verhaltens

Das charakteristische nicht-deterministische Verhalten von LLMs (Ouyang 2023) kann zu Variationen in den Ergebnissen führen, selbst wenn dieselben Eingaben verwendet werden. Dies ist auf die probabilistischen Stichprobenverfahren zurückzuführen, die bei der Inferenz verwendet werden. Folglich kann es schwierig sein, bei der LLM-Verwendung konsistente und reproduzierbare Ergebnisse zu erzielen, insbesondere bei langen Ausgaben, was das Risiko von Abweichungen erhöht.

Eine vollständige Reproduzierbarkeit kann zwar nicht garantiert werden, aber bestimmte Strategien können dazu beitragen, die Variabilität zu verringern:

- Anpassen der Temperatur-Parameter des LLMs: Durch Verringern der Temperatur während der Antwortgenerierung (Inferenz) wird die Wahrscheinlichkeitsverteilung verengt, wodurch die Zufälligkeit reduziert wird und konsistentere Ausgaben erzielt werden. Dies schränkt jedoch auch die Kreativität und Vielfalt der Antworten ein, wodurch die Ausgaben repetitiver oder übermäßig deterministisch werden.
- Festlegen eines Zufallsstartwerts (*random seed*): Einige LLM-Implementierungen ermöglichen die Festlegung des Startwerts für den Zufallszahlengenerator, wodurch sichergestellt wird,

dass dieselbe Pseudozufallssequenz (d. h. deterministisch erzeugte Zufallswerte) verwendet wird, was die Reproduzierbarkeit verbessert.

Um das Risiko von Halluzinationen und Reasoning-Fehlern in LLM-Ausgaben zu verringern, ist deren nicht-deterministisches Verhalten zu adressieren, z. B. durch die Automatisierung bestimmter Aspekte der Ergebnisverifizierung, um einen strukturierten und konsistenten Bewertungsprozess sicherzustellen.

3.2 Datenschutz- und Sicherheitsrisiken generativer KI im Softwaretest

Der Einsatz von generativer KI (GenAI) beim Testen birgt Risiken in Bezug auf Datenschutz und Sicherheit (*security*) aufgrund des Umgangs mit sensiblen Informationen und potenziellen Sicherheitslücken in der LLM-gestützten Testinfrastruktur. Ein robuster Datenschutz ist unerlässlich, um Verstöße, unbefugten Zugriff und die Offenlegung vertraulicher Daten zu verhindern.

3.2.1 Risiken für Datenschutz und Datensicherheit im Zusammenhang mit der Verwendung von generativer KI

Generative KI kann große Datenmengen verarbeiten, die sensible oder personenbezogene Daten enthalten können. Dies wirft folgende Datenschutzbedenken auf:

- Unbeabsichtigte Offenlegung von Daten: Generative KI-Modelle können Ergebnisse generieren, die versehentlich sensible Informationen preisgeben.
- Mangelnde Kontrolle über die Datennutzung: GenAI-Werkzeuge können sensible Daten ohne ausdrückliche Zustimmung oder Kontrolle des Benutzers speichern und verarbeiten. Dies kann zu potenziellem Missbrauch oder unbefugtem Zugriff führen.
- Konformitätsrisiken: Die Verwendung von GenAI-Werkzeugen ohne Einhaltung der Datenschutzbestimmungen, wie beispielsweise der Datenschutz-Grundverordnung (DSGVO, Verordnung (EU) 2016/679), kann zu Rechtsstreitigkeiten führen.

Darüber hinaus entstehen beim Testen mit generativer KI spezifische Sicherheitsrisiken, wie z. B.:

- LLM-basierte Testinfrastruktur kann anfällig für Sicherheitsangriffe sein, wie z. B. Datenpannen oder unbefugten Zugriff.
- Böswillige Akteure können Sicherheitslücken in LLMs ausnutzen, beispielsweise durch manipulative Angriffe (siehe Abschnitt 3.2.2), um deren Verhalten zu verändern oder sensible Informationen zu extrahieren.
- Angreifer können absichtlich böswillige bzw. schädliche Eingabedaten einführen, um LLMs irrezuführen und deren Genauigkeit oder Sicherheit zu beeinträchtigen.

3.2.2 Datenschutz und Sicherheitslücken in generativer KI für Testprozesse und -werkzeuge

Die folgende Tabelle enthält einige Beispiele für Angriffsvektoren in GenAI-Testprozessen und Testwerkzeugen.

Angriffsvektor	**Beschreibung**	**Beispiel**

Datenexfiltration	Senden von Anfragen, um vertrauliche Trainingsdaten zu extrahieren.	Das Überschreiten des Kontextfensters von LLMs mit langen Prompts, um den Speicher der KI zu überlasten, könnte dazu führen, dass sie zufällige Ausschnitte ihrer Trainingsdaten preisgibt und möglicherweise sensible Informationen offenlegt.
Manipulation von Anfragen	Einführung von Daten, die die Ausgabe der KI stören.	Bilder, die die KI in einen anderen Kontext locken und so Halluzinationen hervorrufen, z. B. in Bezug auf Akzeptanzkriterien.
Datenverunreinigung	Manipulation von Trainingsdaten	Bereitstellung gefälschter Bewertungen bei der Beurteilung der Ergebnisse eines KI-generierten Testberichts
Generierung von bösartigem Code	Manipulation eines LLMs, um während der Nutzung Hintertüren (z. B. externe Befehlsaufrufe) zu generieren.	Generierung von Code zum Öffnen eines Kommunikationskanals mit einer bestimmten bösartigen IP-Adresse

3.2.3 Strategien zum Sichern des Datenschutzes und zur Verbesserung der Sicherheit beim Testen mit generativer KI

Da generative KI immer mehr zum Mainstream wird und damit gewisse Risiken einhergehen, entstehen Vorschriften und Standards, um diese Risiken zu mindern (siehe Abschnitt 3.4.1).

Datenschutzbestimmungen, wie die DSGVO, schränken die Anwendungen von generativer KI nicht ausdrücklich ein, bieten jedoch Schutzmaßnahmen, die die Einsatzmöglichkeiten einschränken können, insbesondere in Bezug auf die Rechtmäßigkeit und die Beschränkungen bei der Erhebung, Verarbeitung und Speicherung von Daten.

Um diese Risiken zu mindern, sollten Unternehmen robuste Maßnahmen zum Datenschutz implementieren, darunter:

- Datenminimierung: Vermeidung der Verarbeitung sensibler Daten, sofern dies nicht gesetzlich zulässig ist, und Verwendung nur der erforderlichen Menge an nicht sensiblen Daten beim KI-Testen, um Risiken für den Datenschutz zu verringern.
- Anonymisierung und Pseudonymisierung von Daten: Maskierung oder Ersetzen sensibler Informationen durch nicht identifizierbare Daten.
- Sichere Datenspeicherung und -übertragung: Implementierung starker Verschlüsselung und Zugriffskontrollen.
- Schulung zu Ressourcen: Unternehmen sollten klare Schulungsprogramme und Richtlinien einführen, um den verantwortungsvollen Umgang mit GenAI-Werkzeugen sicherzustellen, ethische Praktiken zu fördern und potenzielle Risiken zu mindern.

Bei der Implementierung von generativer KI für Testzwecke können zusätzliche Risikominderungsstrategien in Betracht gezogen werden:

- Systematische Überprüfung der generierten Ergebnisse: Die Bewertung durch Menschen ist unerlässlich, um die Qualität und Genauigkeit von Testaufgaben zu gewährleisten, die auf generativer KI basieren.
- Bewertung durch Vergleich mit einem anderen LLM: Dabei werden mehrere LLMs für eine bestimmte Aufgabe verwendet, um die Ergebnisse durch Vergleich ihrer Antworten zu bewerten.
- Auswahl einer sicheren, betriebsbereiten Umgebung: Je nach erforderlicher Vertraulichkeit können Unternehmen zwischen verschiedenen sicheren Lösungen wählen: Nutzung eines kommerziellen, sicheren Angebots eines LLM-Anbieters, Betrieb des LLM in einer sicheren Cloud oder Installation des LLM in der Infrastruktur des Unternehmens.
- Regelmäßige Sicherheitsaudits und Bewertungen der Sicherheitslücken: Identifizierung und Behebung von Schwachstellen in generativen KI-Systemen.
- Auf dem Laufenden bleiben mit Best Practices für Sicherheit: Sich über die neuesten Sicherheitsrichtlinien und -technologien auf dem Laufenden halten.

Die Strategien ergänzen sich oft gegenseitig, und eine Kombination dieser Strategien ist erforderlich, um die Datensicherheit bei der Verwendung von generativer KI zu gewährleisten. Es wird dringend empfohlen, leitende Sicherheitsingenieure, Rechtsberater, den Chief Technology Officer (CTO) oder den Chief Information Security Officer (CISO) einzubeziehen, sofern diese im Unternehmen vorhanden sind.

Praktische Übung zum Ziel HO-3.2.3 (H0): … in einer vorgegebenen Fallstudie zum Einsatz von generativer KI Datenschutz- und Sicherheitsrisiken im Test erkennen.

Diese Demonstration veranschaulicht, wie Risiken für den Datenschutz und die Sicherheit (*security*) bei der Verwendung von generativer KI im Softwaretest entstehen können. Die Teilnehmer untersuchen Fallstudien, um potenzielle Bedrohungen wie Modellschwachstellen, unbefugten Datenzugriff oder böswillige Verwendung generierter Ergebnisse zu identifizieren. Sie untersuchen Strategien zur Risikominderung, darunter sichere Datenverarbeitung, robuste Zugriffskontrollen und KI-Überwachungspraktiken, und reflektieren dabei die ethischen und praktischen Implikationen.

Am Ende werden die Teilnehmer die Grundsätze des Datenschutzes verstehen und lernen, Sicherheitsrisiken in Testbedingungen der generativen KI zu erkennen und zu bewältigen.

3.3 Energieverbrauch und Umweltauswirkungen von generativer KI im Softwaretest

Studien wie (Luccioni 2024a) zeigen, dass das Training von und die Verarbeitung durch LLMs eine intensive Nutzung einer großen Anzahl spezialisierter Rechenressourcen erfordern. LLMs sind als webbasierte Dienste verfügbar, und ihre Nutzung erhöht die Belastung von Geräten, Netzwerken und Rechenzentren, was zu einem höheren Energieverbrauch führt.

3.3.1 Die Auswirkungen der Verwendung von generativer KI auf den Energieverbrauch und die CO_2-Emissionen

Die Umweltauswirkungen von generativer KI sollten nicht unterschätzt werden, da der Energieverbrauch mit zunehmender Nutzung stark ansteigt. Die Komplexität der Aufgabe und die erforderlichen Rechenressourcen beeinflussen den Energieverbrauch. So kann beispielsweise die Erzeugung eines einzelnen Bildes mit einem leistungsstarken KI-Modell genauso viel Energie verbrauchen wie das vollständige Aufladen eines Smartphones, während die Erzeugung von Text nur einen kleinen Prozentsatz der Ladung eines Smartphones benötigt (Heikkilä 2023).

Auch wenn es schwierig ist, genaue Daten über die Umweltauswirkungen von generativer KI zu erhalten (Luccioni 2024b), ist es offensichtlich, dass diese energieintensiven Vorgänge insgesamt zu erheblichen CO□-Emissionen beitragen (Berthelot 2024). Eine einzelne Such- oder Textgenerierungsaufgabe mag zwar vernachlässigbar erscheinen, doch ihre kumulative Wirkung bei Millionen von Benutzern weltweit führt zu einer erheblichen Belastung der Umwelt.

Die Einführung von Best Practices, wie z. B. die Begrenzung unnötiger Modellinteraktionen, ist entscheidend für die Minderung der von der generativen KI ausgehenden Umweltrisiken.

Praktische Übung zum Ziel HO-3.3.1 (H1): … angeleitet einen Simulator verwenden, um den Energieverbrauch und die CO_2-Emissionen für vorgegebene Testaufgaben mit generativer KI zu berechnen.

Diese Übung konzentriert sich auf die Bewertung des Energieverbrauchs und der damit verbundenen CO_2-Emissionen verschiedener generativen KI-Aufgaben im Rahmen von Softwaretests. Die Teilnehmer verwenden Simulationen, um diese Metriken zu berechnen und zu untersuchen, wie sich unterschiedliche Aufgabenmerkmale und die Modellnutzung auf die Umweltbelastung auswirken.

Durch die Beobachtung, wie verschiedene Faktoren den Energieverbrauch und die Emissionen beeinflussen, verstehen die Teilnehmer die Treiber des Energieverbrauchs mit LLMs.

3.4 KI-Vorschriften, Standards und Best-Practice-Rahmenwerke

Generative KI verändert das Testen von Software, indem es Tester bei einer Vielzahl von Testaufgaben unterstützt (siehe Kapitel 2). Diese Möglichkeiten bergen jedoch auch erhebliche Risiken, wie z. B. Reasoning-Fehler, Datenschutz, Sicherheitslücken und Umweltauswirkungen (siehe Abschnitte 3.1, 3.2 und 3.3). Bei der Bewältigung dieser Risiken sollten allgemeine Vorschriften, Standards und Best-Practice-Rahmenwerke für KI berücksichtigt werden.

3.4.1 KI-Vorschriften, Standards und Rahmenwerke, die für den Einsatz von generativer KI in Softwaretests relevant sind

Nachfolgend finden Sie eine Übersicht über die wichtigsten Richtlinien, die für den Einsatz von generativer KI im Bereich des Testens relevant sind:

Name/Typ	Beschreibung	Anwendung im Testen
ISO/IEC 42001:2023 Informationstechnologie – Künstliche Intelligenz – Managementsystem Typ: Norm	Legt Anforderungen für das Management von KI-Systemen innerhalb eines Unternehmens fest.	Stellt sicher, dass die generative KI bei Tests den empfohlenen Praktiken entspricht, wodurch Konsistenz und Zuverlässigkeit gefördert werden.
ISO/IEC 23053:2022 Rahmenwerk für künstliche Intelligenz(KI)-Systeme, die maschinelles Lernen verwenden Typ: Norm	Bietet ein Rahmenwerk für KI-Lebenszyklusprozesse, wobei der Schwerpunkt auf Fehlertoleranz und Transparenz liegt.	Bietet einen Rahmen für Datenqualität, Transparenz und Fehlertoleranz bei der Verwendung von generativer KI für Tests.

EU AI Act Typ: Verordnung	Schafft einen Rechtsrahmen für KI-Risiken und klassifiziert Anwendungen nach Risikostufen. Quelle: (AI Act 2024)	Schreibt die Einhaltung von Transparenz, Zurechenbarkeit und Verringerung von Verzerrungen für den Einsatz von generativer KI beim Testen vor.
NIST-Rahmenwerk für das KI-Risikomanagement (USA) Typ: Rahmenwerk	Bietet Leitlinien für das Risikomanagement von KI mit Schwerpunkt auf Fairness, Transparenz und Sicherheit (*security*). Quelle: (NIST AI RMF 1.0)	Gewährleistet Fairness und mindert Risiken bei der generativen KI, wodurch verzerrte Testergebnisse verhindert werden.

Für Testunternehmen ist es unerlässlich, sich über die Entwicklung von Vorschriften, Standards, nationalen Gesetzen und Best-Practice-Rahmenwerken, wie denen in dieser Tabelle, auf dem Laufenden zu halten.

4 LLM-gestützte Testinfrastruktur für den Softwaretest – 110 Minuten

Schlüsselbegriffe

Testinfrastruktur

Spezifische Schlüsselbegriffe für generative KI

Fein-Tuning, Large-Language-Model-Operations (LLMOps), LLM-gestützter Agent, Retrieval-augmentierte Generierung (RAG), Vektordatenbank

Lernziele und Ziele der praktischen Übungen für Kapitel 4: Die Lernenden können …

4.1 Architekturansätze für LLM-gestützte Testinfrastrukturen

GenAI-4.1.1	(K2)	… die wichtigsten Architekturkomponenten und Konzepte einer LLM-gestützten Testinfrastruktur erläutern.
GenAI-4.1.2	(K2)	… die Retrieval-augmentierte Generierung zusammenfassen.
HO-4.1.2	(H1)	… angeleitet mit Retrieval-augmentierter Generierung für eine bestimmte Testaufgabe experimentieren.
GenAI-4.1.3	(K2)	… die Rolle und Anwendung von LLM-gestützten Agenten bei der Automatisierung von Testprozessen erläutern.
HO-4.1.3	(H0)	… erläutern, wie ein LLM-gestützter Agent bei der Automatisierung einer wiederkehrenden Testaufgabe unterstützt.

4.2 Fein-Tuning und LLMOps: Operationalisierung von generativer KI für den Softwaretest

GenAI-4.2.1	(K2)	… das Fein-Tuning von Sprachmodellen für bestimmte Testaufgaben erläutern.
HO-4.2.1	(H0)	… ein Beispiel für einen Fein-Tuning-Prozess für eine bestimmte Testaufgabe und ein Sprachmodell erläutern.
GenAI-4.2.2	(K2)	… LLMOps und dessen Rolle bei der Bereitstellung und Verwaltung von LLMs für Testaufgaben erläutern.

4.1 Architekturansätze für LLM-gestützte Testinfrastrukturen

KI-Chatbots und LLM-gestützte Testwerkzeuge sind zwei Arten von Testinfrastrukturen, die LLMs verwenden (siehe Abschnitt 1.2.2).

Über die grundlegende Architektur einer LLM-gestützten Testinfrastruktur hinaus (siehe Abschnitt 4.1.1) erweitern Retrieval-augmentierte Generierung (siehe Abschnitt 4.1.2) und LLM-gestützte Agentenarchitekturen (siehe Abschnitt 4.1.3) die Funktionalität und den Nutzen des Einsatzes von LLMs im Softwaretest.

4.1.1 Wichtige Architekturkomponenten und Konzepte einer LLM-gestützten Testinfrastruktur

Eine LLM-gestützte Testinfrastruktur bezeichnet ein System, das ein LLM in den Softwaretestprozess integriert, um Automatisierung, Reasoning und Entscheidungsfindung zu verbessern. Im Gegensatz zu einem herkömmlichen KI-Chatbot, der sich in erster Linie auf Dialoginteraktionen konzentriert, dient ein LLM-gestütztes Testwerkzeug dazu, Softwaretesten zu unterstützen, indem es testbezogene Anfragen verarbeitet, Anforderungen analysiert, Testfälle generiert und Ergebnisse bewertet.

Die typische Architektur einer LLM-gestützten Testinfrastruktur folgt einem Multi-Komponenten-Entwurf, der eine sichere und effiziente Interaktion mit dem LLM ermöglicht. Die Architektur umfasst Frontend- und Backend-Komponenten sowie externe Datenquellen und ein integriertes LLM:

- Das Frontend dient als Benutzungsschnittstelle, über die Tester durch Eingabe von Abfragen oder Befehlen mit dem System interagieren.
- Das Backend verarbeitet Benutzereingaben und steuert wichtige Funktionen wie Authentifizierung, Datenabruf, Prompt-Vorbereitung sowie die Interaktion mit dem LLM.
- Das LLM, das entweder als Drittanbieterdienst (über eine API zugänglich) oder als benutzerdefiniertes internes Modell gehostet werden kann, generiert Antworten auf der Grundlage strukturierter Prompts.

Diese Architektur geht über ein traditionelles Client-Server-Modell hinaus, indem sie intelligente Verarbeitungskomponenten wie LLMs und Backends mit mehreren Datenquellen einbezieht:

1. Das LLM ist nicht nur ein Server, sondern eine intelligente Verarbeitungskomponente, die auf der Grundlage von Testmitteln und Testarbeitsergebnissen Interpretationen vornimmt und Schlussfolgerungen zieht.
2. Im Gegensatz zu regelbasierten Chatbots, die vorgefertigte Antworten geben, generiert eine LLM-gestützte Testinfrastruktur dynamisch Testinformationen aus dem Kontext – beispielsweise aus Anforderungen, Quellcode oder Testergebnissen.
3. Das Backend integriert mehrere Datenquellen, darunter:
    - Relationale Datenbanken (für strukturierte Daten, die beim Testen verwendet werden, wie z. B. Testfälle).
    - Vektordatenbanken (für die semantische Suche nach inhaltlich verwandten Informationen mittels Einbettungen; siehe Abschnitt 4.1.2).
4. Das Backend verbessert die unbearbeiteten Ausgaben des LLMs durch Nachbearbeitung und stellt sicher, dass die Antworten mit den Testbedingungen des Testprozesses übereinstimmen, bevor sie an das Frontend weitergeleitet werden.

4.1.2 Retrieval-augmentierte Generierung

Retrieval-augmentierte Generierung (*retrieval-augmented generation*, RAG) verbessert LLMs durch die Einbindung zusätzlicher Datenquellen in ihren Antwortgenerierungsprozess (Zhao 2024) und erhöht so die Relevanz und Genauigkeit ihrer Ergebnisse.

RAG kombiniert Retrieval-Systeme mit Sprachmodellen, um kontextbezogene Antworten zu generieren. Während der Vorverarbeitung werden große Dokumente in kleinere Abschnitte (*chunks*, z. B. 256–512 Token) unterteilt, um ein fokussiertes Retrieval und die Kompatibilität mit dem Kontextfenster des Modells zu gewährleisten. Jeder Abschnitt wird bereinigt, verarbeitet und mithilfe vortrainierter Modelle in einen hochdimensionalen Vektor (Einbettung) kodiert. Diese Einbettungen, die in Vektordatenbanken gespeichert werden können, ermöglichen ein effizientes, auf Ähnlichkeit basierendes Retrieval zur Laufzeit (Inferenz). Eine Benutzerabfrage wird kodiert, relevante Abschnitte werden anhand semantischer Ähnlichkeit abgerufen und diese Abschnitte werden als Kontext für das Sprachmodell verwendet, um eine fundierte Antwort zu generieren.

Eine relevante Antwort ist im Wesentlichen eine vom Sprachmodell generierte Ausgabe, die tief in den relevanten, zutreffenden und kontextuell geeigneten Informationen verankert ist, die während des Retrieval-Prozesses gesammelt wurden. Dadurch wird sichergestellt, dass die Antwort nicht nur auf dem vortrainierten Wissen des Modells basiert, sondern zusätzlich mit präzisen, Prompt-spezifischen Daten angereichert ist. Diese Synergie zwischen Retrieval und Generierung steigert die Genauigkeit und Relevanz der Antworten und macht sie für den Benutzer zuverlässiger und informativer.

Ein RAG-System verarbeitet den Benutzer-Prompt in einem zweistufigen Prozess:

1. Retrieval: Basierend auf einer Benutzeranfrage ruft das System relevante Informationen aus den zuvor erstellten Vektordatenbanken ab. Das Retrieval erfolgt typischerweise auf der Grundlage der semantischen Ähnlichkeit zwischen den Einbettungen des Prompts und denen der Abschnitte.
2. Generierung: Die abgerufenen Informationen werden anschließend an das LLM weitergeleitet, das eine Antwort generiert, die vorhandenes Wissen des Modells mit den neu gewonnenen Daten kombiniert. Dies führt zu einer genaueren und kontextuell passenderen Ausgabe.

Der Einsatz von RAG im Softwaretest ermöglicht es LLM-gestützten Testinfrastrukturen, auf unternehmensinterne Datenquellen wie Datenbanken, Dokumentationen und Repositorien zuzugreifen, um kontextbezogene Informationen in Echtzeit abzurufen. Dadurch wird sichergestellt, dass Testaufgaben wie Testanalyse oder Testentwurf stets auf den aktuellen Spezifikationen, Anforderungen und bestehenden Testdaten basieren.

Praktische Übung zum Ziel HO-4.1.2 (H1): … angeleitet mit Retrieval-augmentierter Generierung für eine bestimmte Testaufgabe experimentieren.

Diese praktische Übung konzentriert sich auf die Anwendung von RAG-Verfahren für eine bestimmte Testaufgabe. Die Teilnehmer experimentieren mit einem RAG-System unter Einbeziehung von Dokumenten und beobachten, wie es auf der Grundlage komplexer Informationen mehr oder weniger genaue Antworten generiert. Die Teilnehmer vergleichen die Ergebnisse des LLMs mit und ohne RAG für die jeweilige Testaufgabe. Diese Übung zielt darauf ab, die Stärken und Grenzen des RAG-Systems bei der Bearbeitung unterschiedlicher Testaufgaben zu identifizieren.

Durch die Analyse der abgerufenen Daten und der generierten Ergebnisse erhalten die Teilnehmer Einblicke in die Rolle von RAG bei der Verbesserung von LLM-gestützten Testprozessen.

4.1.3 Die Rolle LLM-gestützter Agenten bei der Automatisierung von Testprozessen

LLM-gestützte Agenten (Wang 2024) sind spezialisierte Anwendungen der generativen KI, die von LLMs gestützt werden und für die semi-autonome oder autonome Verarbeitung definierter Aufgaben entworfen wurden. Im Kern stützen sich diese Agenten auf LLMs, um natürliche Sprache zu verstehen und zu generieren. Zusätzlich haben sie die Fähigkeit, Instruktionen zu verarbeiten, Kontext abzurufen und intelligente Aktionen durchzuführen.

Im Gegensatz zu herkömmlichen KI-Chatbots, die sich ausschließlich auf Frage-Antwort-Interaktionen fokussieren, können LLM-gestützte Agenten Aufgaben ausführen oder „handeln“, indem sie einen vordefinierten Satz von Funktionen, allgemein als „Tools“ bezeichnet, aufrufen. Diese Fähigkeit ermöglicht es ihnen, mit externen Systemen zu interagieren und diese zu steuern, wodurch sie bei der Ausführung von Aufgaben äußerst vielseitig sind. Der Autonomiegrad von LLM-gestützten Agenten kann variieren:

- Autonome Agenten arbeiten unabhängig und führen Aufgaben mit minimaler menschlicher Intervention aus, unter Verwendung vordefinierter Regeln, bestärkendem Lernen und adaptiver Feedbackschleifen.
- Semi-autonome Agenten führen Aufgaben unter regelmäßiger menschlicher Aufsicht aus, um sicherzustellen, dass die Ergebnisse den von Benutzern definierten Zielen entsprechen.

Multi-Agenten-Architekturen umfassen ein kollaboratives System, bei dem mehrere Agenten mit jeweils spezialisierten Rollen miteinander kommunizieren und sich abstimmen, um komplexe Probleme effizienter zu lösen als ein einzelner Agent. Dieses koordinierte Vorgehen zwischen mehreren KI-Agenten wird als „Orchestrierung“ bezeichnet.

In Testprozessen können LLM-gestützte Agenten Testaufgaben automatisieren, indem sie menschliches Schlussfolgern (*reasoning*) und Entscheidungsfindung nachahmen. Allerdings sind diese Agenten denselben Problemen ausgesetzt, die auch bei der Nutzung von LLMs auftreten, wie mögliche Halluzinationen, Reasoning-Fehler und Verzerrungen (siehe Abschnitt 3.1). Diese Agenten können falsche oder irreführende Ergebnisse liefern, was die Zuverlässigkeit automatisierter Testprozesse beeinträchtigen kann. Diese Risiken lassen sich durch automatisierte Verifikationsverfahren für die Ergebnisse der Agenten oder durch den Einsatz semi-autonomer Agenten für kritische Aufgaben mindern.

Praktische Übung zum Ziel HO-4.1.3 (H0): … erläutern, wie ein LLM-gestützter Agent bei der Automatisierung einer wiederkehrenden Testaufgabe unterstützt.

Die Demonstration konzentriert sich auf eine Testaufgabe, die von einem LLM-gestützten Agenten ausgeführt wird. Es werden die an den Agenten übergebenen Eingabedaten, sein Verhalten und die

Ergebnisse seiner Aktionen gezeigt, um die verschiedenen Aspekte der Integration agentenbasierter Lösungen in einen Testprozess zu veranschaulichen.

Diese Demonstration zeigt ein konkretes Beispiel für den Einsatz eines LLM-gestützten Agenten im Kontext einer Testaufgabe.

4.2 Fein-Tuning und LLMOps: Operationalisierung von generativer KI für den Softwaretest

Zwei wichtige Praktiken für die Operationalisierung einer LLM-gestützten Testinfrastruktur im Softwaretest sind das Fein-Tuning von LLMs sowie die Verwaltung der operativen Pipeline durch LLMOps (Mailach 2024).

4.2.1 Fein-Tuning von LLMs für Testaufgaben

Fein-Tuning passt ein vortrainiertes Sprachmodell (LM), wie z. B. ein LLM oder ein SLM (siehe Abschnitt 1.1.2), an die Durchführung spezifischer Aufgaben oder an bestimmte Branchen an (Parthasarathy 2024). Dabei wird das Modell mit einem gezielten Datensatz weitertrainiert, sodass es branchenspezifisches Wissen und Besonderheiten erlernen kann. Durch das Fein-Tuning wird die Leistungsfähigkeit des Modells für spezialisierte Anwendungen verbessert, wodurch es präzisere und relevantere Ergebnisse für den jeweiligen Anwendungsfall liefert.

In der Praxis eignet sich das Fein-Tuning dazu, generische LLMs mit spezialisierten Reasoning-Fähigkeiten für eine bestimmte Branche auszustatten oder ein für dieses Fachgebiet spezifisches Vokabular zu übernehmen. Fein-Tuning kann ebenso auf kleinere Modelle (SLMs) angewendet werden, die weniger ressourcenintensiv sind. Durch das Fein-Tuning eines SLMs lässt sich für spezifische Aufgaben ein höheres Leistungsniveau erzielen, ohne dass der gleiche Rechenaufwand wie bei LLMs erforderlich ist. Dieser Vergleich unterstreicht die Flexibilität und Effizienz bei der Verwendung von LLMs und SLMs je nach den spezifischen Anforderungen der Aufgabe.

Im Softwaretest kann z. B. ein LLM oder SLM durch Fein-Tuning Testfälle aus User-Storys in einem für den Unternehmenskontext spezifischen Ausgabeformat erzeugen. Durch das Training des Modells mit den User-Storys und den entsprechenden Testfällen des Unternehmens wird das Modell an den spezifischen Testprozess und die Terminologie des Unternehmens angepasst.

Das Fein-Tuning eines generativen KI-Modells für den Softwaretest ist mit mehreren Herausforderungen verbunden:

- Vermeidung von Verzerrungen oder ungenauen Ergebnissen durch die Verwendung qualitativ hochwertiger und aufgabenspezifischer Trainingsdatensätze.
- Vermeidung der Überanpassung (d. h., das Modell wird zu sehr auf die Trainingsdaten spezialisiert, was sich negativ auf seine Leistung bei neuen, unbekannten Daten auswirkt), um die Generalisierung über verschiedene Szenarien hinweg aufrechtzuerhalten.
- Bewältigung der Opazität (mangelnde Transparenz hinsichtlich der Art und Weise, wie ein LLM seine Entscheidungen trifft oder seine Ergebnisse erzeugt) im Reasoning-Prozess des Modells, was das Debugging und die Validierung erschwert.
- Handhabung des erheblichen Bedarfs an Rechenressourcen, die insbesondere beim Fein-Tuning von LLMs erforderlich sind.

Praktisches Lernziel HO-4.2.1 (H0): … ein Beispiel für einen Fein-Tuning-Prozess für eine bestimmte Testaufgabe und ein Sprachmodell erläutern.

Diese Demonstration zeigt die verschiedenen Schritte, die beim Fein-Tuning eines LLMs für eine bestimmte Testaufgabe erforderlich sind. Sie beginnt mit der Auswahl eines geeigneten LLMs oder SLMs. Als Nächstes wird ein Datensatz vorgestellt, der auf die jeweilige Testaufgabe zugeschnitten ist. Anschließend wird eine beispielhafte Lösung für den Fein-Tuning-Prozess gezeigt (z. B. ein Framework für maschinelles Lernen). Schließlich wird ein Prompt an das feingetunte Modell gesendet und die Qualität der generierten Ausgabe diskutiert.

Diese Demonstration des Fein-Tuning-Prozesses eines LLMs/SLMs für eine Testaufgabe zeigt mehrere wichtige Aspekte dieses Prozesses und hebt insbesondere die Bedeutung der Qualität der Trainingsdaten hervor.

4.2.2 LLMOps bei der Bereitstellung und Verwaltung von LLMs für den Softwaretest

LLMOps oder Large-Language-Model-Operations bezeichnet die Gesamtheit an Praktiken, Werkzeugen und Prozessen, die darauf abzielen, die Entwicklung, Bereitstellung und Wartung von LLMs in Produktionsumgebungen zu optimieren (Sinha 2024).

Der Einsatz generativer KI in den Testprozessen eines Unternehmens kann auf verschiedene Weise erfolgen, was sich auf die zu treffenden LLMOps-Entscheidungen auswirkt. Hier sind drei mögliche Ansätze:

- Einsatz eines KI-Chatbots: Zu den wichtigsten Überlegungen bei diesem Ansatz gehören das Management von Datenschutz- und Sicherheitsrisiken bei gleichzeitiger Kostenoptimierung. Organisationen können LLM-as-a-Service-Plattformen nutzen, wenn die erforderlichen Garantien gegeben sind, oder eine interne Infrastruktur mit Open-Source-lizenzierten LLMs einsetzen, um eine bessere Kontrolle zu haben. Eine strenge Bewertung der Garantien des Anbieters oder der internen Fähigkeiten ist entscheidend, um Datenschutz- und Sicherheitsrisiken zu minimieren (siehe Abschnitt 3.2) und die Betriebseffizienz sicherzustellen.
- Einsatz eines Testwerkzeugs mit generativen KI-Funktionen: Dieser Ansatz beinhaltet ähnliche Überlegungen wie beim Einsatz von KI-Chatbots, wie etwa Datenschutz, Sicherheit und Betriebskosten. Darüber hinaus müssen Unternehmen die vom Anbieter des Testwerkzeugs angebotenen Garantien hinsichtlich Datensicherheit und Leistung prüfen. Diese Testwerkzeuge ergänzen in der Regel bestehende Testprozesse, was eine gründliche Kosten-Nutzen-Analyse und Risikobewertung erfordert.
- Eigenentwicklung eines Testwerkzeugs auf Basis generativer KI: Dieser Ansatz legt den Schwerpunkt auf eine umfassende Kontrolle von Datenschutz- und Sicherheitsrisiken sowie auf eine sorgfältige Planung der Betriebskosten für KI, wie z. B. Rechenressourcen, Datenspeicherung und Mitarbeiterschulungen. Unternehmen müssen außerdem strukturierte Prozesse zur Validierung und Wartung von GenAI-spezifischen Entwicklungen einrichten. Die Eigenentwicklung setzt Expertise in der Implementierung und Bereitstellung einer LLM-gestützten Testinfrastruktur voraus.

Diese Ansätze schließen sich nicht gegenseitig aus, da ein Unternehmen für einige Aufgaben einen KI-Chatbot einsetzen und für andere Aufgaben maßgeschneiderte Werkzeuge entwickeln kann. Daher können sie je nach den spezifischen Testaktivitäten gleichzeitig implementiert werden. Darüber hinaus können sie zusätzliche Technologien wie RAG und Fein-Tuning von LLMs/SLMs integrieren, um die Effektivität und Anpassbarkeit der Testprozesse mit generativer KI zu erhöhen.

5 Bereitstellung und Integration generativer KI in Testorganisationen – 80 Minuten

Schlüsselbegriffe

Keine

Spezifische Schlüsselbegriffe für generative KI

Schatten-KI

Lernziele und Ziele der praktischen Übungen für Kapitel 5: Die Lernenden können …

5.1 Roadmap für die Einführung von generativer KI im Softwaretest

GenAI-5.1.1	(K1)	… sich an die Risiken von Schatten-KI erinnern.
GenAI-5.1.2	(K2)	… die wichtigsten Aspekte erläutern, die bei der Definition einer Strategie für den Einsatz von generativer KI im Softwaretest zu berücksichtigen sind.
GenAI-5.1.3	(K2)	… die wichtigsten Kriterien für die Auswahl von LLMs/SLMs für Software-Testaufgaben in einem gegebenen Kontext zusammenfassen.
HO-5.1.3	(H1)	… angeleitet die wiederkehrenden Kosten für den Einsatz von generativer KI bei einer bestimmten Testaufgabe schätzen.
GenAI-5.1.4	(K1)	… sich an die wichtigsten Phasen bei der Einführung von generativer KI in einer Testorganisation erinnern.

5.2 Umgang mit Veränderungen bei der Einführung von generativer KI im Softwaretest

GenAI-5.2.1	(K2)	… die wesentlichen Fähigkeiten und Wissensbereiche erläutern, die Tester benötigen, um effektiv mit generativer KI in Testprozessen zu arbeiten.
GenAI-5.2.2	(K1)	… sich an Strategien zur Entwicklung von KI-Fähigkeiten innerhalb von Testteams erinnern, um die Einführung von generativer KI in Testaktivitäten zu unterstützen.
GenAI-5.2.3	(K1)	… erkennen, wie sich Testprozesse und Verantwortlichkeiten innerhalb einer Testorganisation bei der Einführung von generativer KI verändern.

5.1 Roadmap für die Einführung von generativer KI im Softwaretest

Eine Teststrategie mit generativer KI muss wichtige Aspekte sorgfältig berücksichtigen, wie z. B. die zu erreichenden Testziele, die Auswahl geeigneter LLMs, Probleme in Bezug auf die für Prompts verwendeten Eingabedaten sowie die Einhaltung von KI-Standards und regulatorischen Vorgaben. Auf der Grundlage dieser Strategie kann die Organisation eine Roadmap erstellen und den Fortschritt bei der Integration von generativer KI in die Testprozesse überwachen.

5.1.1 Risiken von Schatten-KI

Schatten-KI kann zu Risiken in Bezug auf Sicherheit (*security*), Konformität und Datenschutz führen:

- Schwachstellen in Sicherheit (*security*) und Datenschutz: Persönliche KI-Werkzeuge verfügen möglicherweise nicht über eine robuste Sicherheit, was zu potenziellen Datenpannen führen kann.
- Konformitäts- und regulatorische Probleme: Die Verwendung nicht genehmigter KI-Werkzeuge kann zur Nichteinhaltung von Branchenstandards und regulatorischen Vorgaben führen (siehe Abschnitt 3.4.1) und möglicherweise rechtliche Konsequenzen nach sich ziehen.
- Unklare Urheberrechte: Die Nutzung von KI-Werkzeugen mit unklaren Lizenzvereinbarungen kann Anwender von LLMs Urheberrechtsstreitigkeiten aussetzen, insbesondere wenn urheberrechtlich geschützte Daten ohne entsprechende Berechtigung verarbeitet werden.

Eine klare Strategie sowie konkrete Schritte für die Integration und den Einsatz von generativer KI können Testorganisationen dabei unterstützen, die Risiken durch Schatten-KI zu vermeiden.

5.1.2 Schlüsselaspekte einer generativen KI-Strategie im Softwaretest

Für eine erfolgreiche Umsetzung einer generativen KI-Strategie im Testen müssen Organisationen mehrere Schlüsselfaktoren sorgfältig berücksichtigen, um eine reibungslose Integration und optimale Ergebnisse sicherzustellen. Dies beginnt mit der Definition messbarer Testziele für den Einsatz von generativer KI, beispielsweise die Steigerung der Testproduktivität, die Verkürzung von Testzyklen oder die Verbesserung der Testqualität. Die Auswahl geeigneter LLMs ist dabei von entscheidender Bedeutung (siehe Abschnitt 5.1.3) und sollte auf diese Testziele abgestimmt sein. Gleichzeitig muss die Kompatibilität mit der bestehenden Testinfrastruktur gewährleistet werden, ebenso wie die Erfüllung von Anforderungen an die Skalierbarkeit des Systems.

Die Datenqualität spielt eine entscheidende Rolle, da die Effektivität des LLM-gestützten Testens maßgeblich von hochqualitativen, relevanten Eingabedaten abhängt, die durch robuste Sicherheitsmaßnahmen geschützt werden. Eine dauerhaft hohe Qualität der Eingabedaten ist entscheidend, um vertrauenswürdige und korrekte Ergebnisse zu erzielen.

Darüber hinaus sollten umfassende Schulungsprogramme angeboten werden, um sicherzustellen, dass Testteams über die technischen und ethischen Fähigkeiten verfügen, die für den effektiven Einsatz von GenAI-Werkzeugen erforderlich sind. Zusätzlich zu den Schulungen sollten spezifische Metriken erhoben werden, um die Effektivität der generativen KI-Ergebnisse zu messen (siehe Abschnitt 2.3.1).

Um die Konformität mit gesetzlichen Standards und ethischen Richtlinien zu gewährleisten, sollten Organisationen prozessuale Leitlinien für die Nutzung von generativer KI etablieren. Dazu gehören Regeln für die Verwendung sensibler Daten, Transparenzpflichten (z. B. Kennzeichnung, welche Inhalte durch generative KI erzeugt wurden) sowie Quality Gates mit Review der generierten Testmittel.

5.1.3 Auswahl von LLMs/SLMs für Software-Testaufgaben

Es gibt eine große Auswahl an LLMs/SLMs, die sich hinsichtlich ihrer funktionalen Fähigkeiten (z. B. multimodale Eingabe, Reasoning-Fähigkeiten), ihrer technischen Merkmale (z. B. Größe des Kontextfensters) und ihrer Lizenztypen (z. B. kommerziell vs. Open Source) unterscheiden. Während viele Benchmarks zur Bewertung von LLMs/SLMs für Aufgaben wie Verarbeitung natürlicher Sprache, Codegenerierung oder Bildanalyse verfügbar sind, existieren bislang nur wenige, die sich spezifisch auf Software-Testaufgaben konzentrieren (Wenhan 2024). Daher erfordert die Auswahl von LLMs/SLMs für Testaufgaben die sorgfältige Berücksichtigung mehrerer wichtiger Kriterien:

- Modellleistung: Bewerten der Leistung des Modells für die angestrebten Testaufgaben anhand der Benchmarks der Organisation unter Verwendung geeigneter Metriken (siehe Abschnitt 2.3.1).
- Potenzial für Fein-Tuning: Prüfen, ob und inwieweit das Sprachmodell (LLM oder SLM) mit domänenspezifischen Daten angepasst werden kann, um die Leistung für einen bestimmten Anwendungsfall zu verbessern und die Genauigkeit und Relevanz in speziellen Kontexten zu erhöhen.
- Wiederkehrende Kosten: Berücksichtigung der wiederkehrenden Kosten für die Nutzung des LLMs/SLMs, einschließlich Lizenzgebühren und Betriebsausgaben, um sicherzustellen, dass diese mit dem Budget der Organisation für die angestrebten Testaufgaben vereinbar sind.
- Community und Support: Wahl von Modellen mit aktiver Community-Unterstützung und ausführlicher Dokumentation, um die Implementierung und Fehlerbehebung zu erleichtern.

Durch sorgfältige Bewertung dieser Kriterien können Testorganisationen ein oder mehrere LLMs /SLMs auswählen, die ihren spezifischen Anforderungen und organisatorischen Rahmenbedingungen entsprechen.

Praktische Übung zum Ziel HO-5.1.3 (H1): … angeleitet die wiederkehrenden Kosten für den Einsatz von generativer KI bei einer bestimmten Testaufgabe schätzen.

Diese Übung konzentriert sich auf die Schätzung der wiederkehrenden Kosten für den Einsatz von generativer KI bei einer bestimmten Testaufgabe auf der Grundlage verschiedener Annahmen. Dazu gehören Faktoren wie die Anzahl der Token in den Eingabe- und Ausgabedaten, die verwendeten Prompts sowie die Häufigkeit der Durchführung der Aufgabe. Es werden Preismodelle mehrerer LLM/SLM-Anbieter betrachtet und miteinander verglichen, darunter mindestens eine kommerzielle Lösung und ein Open-Source-lizenziertes Modell.

Diese Übung bietet die Möglichkeit, unter praktischen Testbedingungen die wiederkehrenden Kosten von generativer KI zu berechnen und damit zu experimentieren. Dadurch können die finanziellen Auswirkungen unterschiedlicher Ansätze und Anbieter besser verstanden werden.

5.1.4 Phasen bei der Einführung von generativer KI im Softwaretest

Die Einführung von generativer KI in einer Testorganisation umfasst drei wichtige Phasen:

1. Erkundung: Die erste Phase konzentriert sich auf die Sensibilisierung und den Aufbau von Fähigkeiten. Zu den Aktivitäten gehören die Schulung von Testteams zu den Konzepten generativer KI, die Bereitstellung eines Zugangs zu LLMs /SLMs sowie erste Experimente mit Anwendungsfällen, um Tester mit generativer KI vertraut zu machen und Vertrauen aufzubauen.

2. Einführung und Definition der Nutzung: Sobald ein grundlegendes Verständnis vorhanden ist, konzentriert sich die zweite Phase auf die Identifizierung und Priorisierung praxisnaher Anwendungsfälle für generative KI im Softwaretest. Dazu gehören die Evaluierung einer LLM-gestützten Testinfrastruktur, der Aufbau von Fachwissen sowie die Sicherstellung der Anpassung an die Bedürfnisse der Organisation (siehe [ISTQB_CTFL_SYL], Kapitel 6).
3. Nutzung und Iteration: In dieser fortgeschrittenen Phase integrieren Organisationen generative KI vollständig in ihre Testprozesse. Eine kontinuierliche Überwachung des Fortschritts von generativer KI für Softwaretests und zugehörige Werkzeuge sind vorhanden, ebenso wie die Messung und das Management der Transformation, um nachhaltige Vorteile und Skalierbarkeit sicherzustellen.

Diese Phasen können für verschiedene Anwendungsfälle parallel verlaufen. So kann beispielsweise die Analyse von Testberichten bereits weiter fortgeschritten sein, während sich die Testautomatisierung noch in einer frühen Phase befindet. Es ist auch wichtig, frühzeitig Bedenken wie die Angst vor dem Verlust des Arbeitsplatzes zu erkennen und anzugehen, da diese sich auf die Akzeptanz und die Moral des Teams auswirken können.

5.2 Management von Veränderungen bei der Einführung von generativer KI im Softwaretest

Die erfolgreiche Einführung von generativer KI in einer Testorganisation erfordert einen strukturierten Ansatz für die Veränderungsmanagement-Prozesse. Zu den wichtigsten Aspekten gehören der Aufbau grundlegender generativer KI-Kompetenzen und die Weiterentwicklung traditioneller Rollen im Test, um KI-gestützte Testprozesse zu integrieren. Die Transformation umfasst sowohl technische Fähigkeiten als auch organisatorische Aspekte.

5.2.1 Wesentliche Fähigkeiten und Kenntnisse für das Testen mit generativer KI

Die erfolgreiche Integration von generativer KI in das Testen erfordert die Beherrschung von Prompt-Engineering-Verfahren, das Verständnis von Kontextfenstern der Modelle und die Entwicklung von Methoden zum Review von Tests. Tester müssen Fach- und Testkenntnisse mit KI-Fähigkeiten kombinieren, um LLM-getriebenes Testen bei Aufgaben wie der Testfallgenerierung, der Analyse von Fehlerberichten und der Testdatengenerierung zu bewerten.

Zu den Schlüsselkompetenzen zählen die Bewertung der LLM-Fähigkeiten, das Verständnis von Verfahren zur Verfeinerung von Prompts sowie die Evaluierung von KI-generierten Testmitteln. Zu den wesentlichen Kenntnissen gehören das Verständnis der mit generativer KI verbundenen Risiken sowie die Kenntnis gängiger Strategien zu deren Minderung. Tester sollten die Auswirkungen der gemeinsamen Nutzung von Testmitteln mit LLMs auf die Datensicherheit verstehen, eine ordnungsgemäße Datenbereinigung (Entfernen oder Maskieren sensibler, persönlicher oder vertraulicher Informationen) durchführen und die Praktiken des Prompt-Engineering befolgen, die den Datenschutz gewährleisten. Zu den Umweltaspekten gehören die Optimierung der Modellauswahl und der Nutzungsmuster zur Reduzierung des Rechenaufwands, die Auswahl von Modellen in der richtigen Größe für Testaufgaben und die Abwägung der Vorteile der generativen KI-Automatisierung gegenüber den Auswirkungen auf Kosten und Energieverbrauch.

5.2.2 Aufbau generativer KI-Fähigkeiten in Testteams

Ein praxisorientierter Ansatz ist unerlässlich, um Testteams strategisch im Einsatz von generativer KI für das Testen zu schulen. Dazu gehören das Üben mit verschiedenen LLMs/SLMs, das Befolgen strukturierter Lernpfade und der schrittweise Wissensaufbau durch den Austausch innerhalb der Organisation. Der Schwerpunkt der Schulungen liegt auf der Entwicklung praktischer Fähigkeiten durch angeleitete Übungen, Peer-Learning und die schrittweise Integration von KI in die täglichen Testaufgaben.

Mitglieder von Testteams entwickeln sich weiter ausgehend von der Beherrschung grundlegender Prompt-Erstellung hin zur Verwendung gezielterer Verfahren, wie z. B. testspezifischer Prompts. Ein Prompt-Muster ist hierbei eine wiederverwendbare Vorlage, die dazu dient, wirksame Prompts zu formulieren und generative KI zu konsistenten und zuverlässigen Ergebnissen zu führen. Interne praxisbezogene Gemeinschaften (*communities of practice*, CoP) unterstützen den kontinuierlichen Wissensaustausch mit regelmäßigen Treffen, um erfolgreiche generative KI-Anwendungen hervorzuheben, Herausforderungen zu diskutieren und Best Practices zu verfeinern. Diese Gemeinschaften fördern die kontinuierliche Verbesserung, indem sie Prompt-Muster-Bibliotheken austauschen und die gewonnenen Erfahrungen aus Implementierungen von generativer KI im Testprozess aus verschiedenen Projekten und Domänen dokumentieren.

5.2.3 Weiterentwicklung von Testprozessen in KI-gestützten Testorganisationen

Die Integration von generativer KI verändert die traditionellen Testprozesse von Testern und Testmanagern innerhalb von Testorganisationen.

Tester entwickeln sich von Spezialisten für Testentwurf und Testdurchführung zu KI-gestützten Testspezialisten, die ihr Fachwissen in Testverfahren mit Fähigkeiten zur Steuerung und Überprüfung von KI-generierten Testmitteln kombinieren. Ihre Testaufgaben erweitern sich um die Überprüfung der gesamten KI-basierten Ausgabe, die Verfeinerung von Prompts und die Pflege testspezifischer Prompt-Bibliotheken.

Die Aufgabenbereiche von Testmanagern werden um die Entwicklung einer KI-basierten Teststrategie, KI-basiertes Risikomanagement sowie die Überwachung und Steuerung von KI-basierten Testprozessen erweitert. Testmanager konzentrieren sich darauf, menschliche und KI-Fähigkeiten in Einklang zu bringen, KI-Governance-Rahmenwerke für Anwendungsfälle zu etablieren und sicherzustellen, dass ihre Testteams sowohl traditionelle Testkompetenzen als auch KI-Kenntnisse beibehalten. Testmanager sind nicht nur Führungskräfte für menschliche Tester, sondern koordinieren auch den Einsatz von generativer KI gestützten Testagenten, was neue Managementfähigkeiten für die Leitung hybrider Teams aus Menschen und generativen KI-Tools erfordert.

6 Referenzen

Normen

- **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®-Dokumente

- **[ISTQB_CTFL_SYL]** ISTQB® Foundation Level Lehrplan v4.02D, 2025

Glossar-Referenzen

- **ISTQB®-Glossar** https://glossary.istqb.org/

Bücher

- **Winteringham M.** (2025). Software Testing with Generative AI. Manning Publications (5 März 2025), ISBN-13: 978-1633437364, 304 Seiten.

Artikel

- **(Berthelot 2024)** Berthelot, Adrien et al. "Estimating the environmental impact of Generative-AI services using an LCA-based methodology." *Procedia CIRP* 122 (2024): 707-712.
- **(Gallegos 2024)** Gallegos, Isabel O. et al. "Bias and fairness in large language models: A survey." *Computational Linguistics* (2024): 1-79.
- **(Li 2024)** Li, Yihao, Pan Liu, Haiyang Wang, Jie Chu, W. Eric Wong. Evaluating Large Language Models for Software Testing, Computer Standards & Interfaces (2024), doi: https://doi.org/10.1016/j.csi.2024.103942.
- **(Luccioni 2024a)** Luccioni, Sasha, Yacine Jernite, Emma Strubell. "Power hungry processing: Watts driving the cost of AI deployment?." *The 2024 ACM Conference on Fairness, Accountability, and Transparency*. 2024.
- **(Mailach 2024)** Mailach, Alina et al. "Practitioners' Discussions on Building LLM-based Applications for Production." arXiv preprint arXiv:2411.08574 (2024).
- **(Mirzadeh 2024)** Mirzadeh, Iman et al. “GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models.” ArXiv abs/2410.05229 (2024).
- **(NIST AI RMF 1.0)** National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1, U.S. Department of Commerce, 2023, https://doi.org/10.6028/NIST.AI.100-1.
- **(Ouyang 2023)** Ouyang, Shuyin et al. "LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation." arXiv preprint arXiv:2308.02828 (2023).
- **(Parthasarathy 2024)** Parthasarathy, Venkatesh Balavadhani et al. "The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best

practices, applied research challenges and opportunities." arXiv preprint arXiv:2408.13296 (2024).

- **(Schulhoff 2024)** Schulhoff, S. “The Prompt Report: A Systematic Survey of Prompting Techniques”, Art. no. arXiv:2406.06608, 2024. doi:10.48550/arXiv.2406.06608.
- **(Sinha 2024)** Sinha, Megha, Sreekanth Menon, Ram Sagar. "LLMOps: Definitions, Framework and Best Practices." 2024 International Conference on Electrical, Computer and Energy Technologies (ICECET. IEEE, 2024.
- **(Wang 2024)** Wang, Yanlin et al. "Agents in Software Engineering: Survey, Landscape, and Vision." arXiv preprint arXiv:2409.09030 (2024).
- **(Wenhan 2024)** Wenhan, Wang et al. "TESTEVAL: Benchmarking Large Language Models for Test Case Generation." arXiv preprint arXiv:2406.04531 (2024).
- **(Zhao 2024)** Zhao, Penghao et al. "Retrieval-augmented generation for ai-generated content: A survey." arXiv preprint arXiv:2402.19473 (2024).

Webseiten

- **(AI Act 2024)** European Commission. "European Approach to Artificial Intelligence." *Shaping Europe’s Digital Future*, European Commission, https://digital-strategy.ec.europa.eu/en/policies/european-approach-artificial-intelligence. Letzter Zugriff 24. Nov. 2024.
- **(Heikkilä 2023)** Heikkilä, M. (2023, December 1). Making an image with generative AI uses as much energy as charging your phone. MIT Technology Review. Retrieved from https://www.technologyreview.com/2023/12/01/1084189/making-an-image-with-generative-ai-uses-as-much-energy-as-charging-your-phone/.
- **(Luccioni 2024b)** Luccioni, S. (2024, February 22). Generative AI’s environmental costs are soaring. Nature. Retrieved from https://www.nature.com/articles/d41586-024-00478-x.
- **(Google Dev Glossary 2024)** Google Developers. (n.d.). Machine learning glossary: Generative AI. Retrieved from https://developers.google.com/machine-learning/glossary/generative. Letzter Zugriff 24. Nov. 2024.
- **(MIT 2024)** "Glossary of Terms: Generative AI Basics." *MIT Sloan Teaching & Learning Technologies*, MIT Sloan School of Management, https://mitsloanedtech.mit.edu/ai/basics/glossary. Letzter Zugriff 24. Nov. 2024.

Die vorstehenden Quellenangaben verweisen auf Informationen, die im Internet und anderswo verfügbar sind. Auch wenn diese Verweise zum Zeitpunkt der Veröffentlichung dieses Lehrplans geprüft wurden, kann das ISTQB® nicht dafür verantwortlich gemacht werden, wenn die Verweise nicht mehr verfügbar sind.

7 Anhang A – Lernziele/Kognitive Wissensstufen

Die spezifischen Lernziele, die für diesen Lehrplan gelten, sind am Anfang jedes Kapitels aufgeführt. Jedes Thema im Lehrplan wird anhand des dafür festgelegten Lernziels geprüft.

Die Lernziele beginnen mit einem Aktionsverb, das der kognitiven Wissensstufe entspricht, wie unten aufgeführt.

Stufe 1: Erinnern (K1)

Der Kandidat erinnert sich an einen Begriff oder ein Konzept, erkennt ihn/es wieder und ruft ihn/es ab.

Aktionsverben: Erinnern, erkennen

Beispiele
Erinnern Sie sich an die Konzepte der Testpyramide.
Erkennen Sie die typischen Ziele von Testen.

Stufe 2: Verstehen (K2)

Der Kandidat kann die Gründe oder Erklärungen für Aussagen zum Thema auswählen und das Testkonzept zusammenfassen, vergleichen, klassifizieren und Beispiele dafür nennen.

Aktionsverben: Klassifizieren, vergleichen, unterscheiden, erklären, Beispiele nennen, interpretieren, zusammenfassen

Beispiele	**Hinweise**
Klassifizieren Sie Testwerkzeuge nach ihrem Zweck und den Testaktivitäten, die sie unterstützen.	
Vergleichen Sie die verschiedenen Teststufen.	Kann verwendet werden, um nach Ähnlichkeiten, Unterschieden oder beidem zu suchen.
Unterscheiden Sie zwischen Testen und Debugging.	Sucht nach Unterschieden zwischen Konzepten.
Unterscheiden Sie zwischen Projektrisiko und Produktrisiko.	Ermöglicht die separate Klassifizierung von zwei (oder mehr) Konzepten.
Erläutern Sie die Auswirkungen des Kontexts auf den Testprozess.	
Geben Sie Beispiele dafür, warum Testen notwendig ist.	
Fassen Sie die Aktivitäten des Arbeitsproduktprüfungsprozesses zusammen.	

Stufe 3: Anwenden (K3)

Der Kandidat kann ein Verfahren ausführen, wenn er mit einer vertrauten Aufgabe konfrontiert wird, oder das richtige Verfahren auswählen und es auf einen gegebenen Kontext anwenden.

Aktionsverben: Anwenden, implementieren, vorbereiten, nutzen

Beispiele	**Anmerkungen**
Wenden Sie die Grenzwertanalyse an, um aus vorgegebenen Anforderungen Testfälle abzuleiten.	Sollte sich auf ein Verfahren / eine Technik / einen Prozess usw. beziehen.
Implementieren Sie Methoden zur Metrikenerfassung, um technische und administrative Anforderungen zu unterstützen.	
Bereiten Sie Tests zur Installierbarkeit für mobile Apps vor.	
Nutzen Sie Verfolgbarkeit, um den Testfortschritt auf Vollständigkeit und Übereinstimmung mit den Testzielen, der Teststrategie und dem Testkonzept zu überwachen.	Könnte in einem LO verwendet werden, die vom Kandidaten die Anwendung einer Technik oder eines Verfahrens erwartet. Ähnlich wie „anwenden”.

Referenz

(Für die kognitiven Wissensstufen der Lernziele)

Anderson, L. W., Krathwohl, D. R. (Hrsg.) (2001). A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon.

8 Anhang B – Verfolgbarkeitsmatrix für geschäftliche Nutzen mit Lernzielen

In diesem Abschnitt wird die Verfolgbarkeit zwischen den geschäftlichen Nutzen und den Lernzielen von „Certified Tester – Testen mit generativer KI“ aufgeführt. Ziele der praktischen Übungen werden in dieser Tabelle nicht erwähnt, da jedes HO in Verbindung mit einem einzigen LO steht. Die Verfolgbarkeit zwischen einem HO und einem BO erfolgt über das LO, mit dem das HO verbunden ist.

Geschäftlicher Nutzen: Certified Tester für Testen mit generativer KI			BO1	BO2	BO3	BO4	BO5
GenAI-BO1	Die grundlegenden Konzepte, Fähigkeiten und Grenzen von generativer KI verstehen.		8				
GenAI-BO2	Praktische Fähigkeiten zur Verwendung von Large Language-Modellen für den Softwaretest entwickeln.			10			
GenAI-BO3	Einblicke in die Risiken und Abhilfemaßnahmen beim Einsatz von generativer KI für den Softwaretest gewinnen.				11		
GenAI-BO4	Einblicke in die Anwendungsmöglichkeiten von Lösungen mit generativer KI für den Softwaretest gewinnen.					19	
GenAI-BO5	Effektiv zur Definition und Umsetzung einer generativen KI-Strategie und -Roadmap für den Softwaretest innerhalb eines Unternehmens beitragen.						13
Prüfbare Lernziele	Lernziel – Die Lernenden können …	K-Level					
1	Einführung in generative KI für den Softwaretest – 100 Minuten						

Geschäftlicher Nutzen: Certified Tester für Testen mit generativer KI			BO1	BO2	BO3	BO4	BO5
1.1	Grundlagen und Schlüsselkonzepte generativer KI						
GenAI-1.1.1	… verschiedene Arten von KI unterscheiden: symbolische KI, klassisches maschinelles Lernen, Deep Learning und generative KI.	K1	X				
GenAI-1.1.2	… die Grundlagen von generativer KI und von Large Language-Modellen erläutern.	K2	X				
GenAI-1.1.3	… zwischen Foundation-, Instruction-Tuned- und Reasoning-LLMs unterscheiden.	K2	X				
GenAI-1.1.4	… Grundprinzipien von multimodalen LLMs und von Vision-Language-Modellen zusammenfassen.	K2	X				
1.2	Nutzung generativer KI im Softwaretest: Grundprinzipien						
GenAI-1.2.1	… Beispiele für wichtige Funktionen von LLMs für Testaufgaben nennen.	K2	X			X	
GenAI-1.2.2	… Interaktionsmodelle bei der Verwendung von GenAI für Softwaretests vergleichen.	K2	X			X	
2	Prompt-Engineering für effektives Softwaretesten – 365 Minuten						
2.1	Effektive Prompt-Entwicklung						

Geschäftlicher Nutzen: Certified Tester für Testen mit generativer KI			BO1	BO2	BO3	BO4	BO5
GenAI-2.1.1	… Beispiele für die Struktur von Prompts nennen, die in generativer KI für das Softwaretesten verwendet werden.	K2		X			
GenAI-2.1.2	… zwischen den wichtigsten Prompting-Verfahren für das Softwaretesten unterscheiden.	K2		X			
GenAI-2.1.3	… zwischen System-Prompts und Benutzer-Prompts unterscheiden.	K2		X			
2.2	Anwendung von Prompt-Engineering-Verfahren auf Software-Testaufgaben						
GenAI-2.2.1	… generative KI für Aufgaben bei der Testanalyse anwenden.	K3		X			
GenAI-2.2.2	… generative KI für Testentwurf und Testrealisierung anwenden.	K3		X			
GenAI-2.2.3	… generative KI für automatisierte Regressionstests anwenden.	K3		X			
GenAI– 2.2.4	… generative KI für Aufgaben bei der Teststeuerung und -überwachung anwenden.	K3		X			
GenAI-2.2.5	… geeignete Prompting-Verfahren für einen bestimmten Kontext und einen bestimmten Test auswählen und anwenden.	K3		X		X	
2.3	Bewertung generativer KI-Ergebnisse und Verfeinerung von Prompts für Software-Testaufgaben						

Geschäftlicher Nutzen: Certified Tester für Testen mit generativer KI			BO1	BO2	BO3	BO4	BO5
GenAI- 2.3.1	… Metriken zur Bewertung der Ergebnisse generativer KI bei Testaufgaben verstehen.	K2		X	X	X	
GenAI-2.3.2	… Beispiele für Verfahren zur Bewertung und iterativen Verfeinerung von Prompts nennen.	K2		X	X	X	
3	Management von Risiken bei generativer KI im Softwaretest – 160 Minuten						
3.1	Halluzinationen, Reasoning-Fehler und Verzerrungen						
GenAI-3.1.1	… Definitionen von Halluzinationen, Reasoning-Fehlern und Verzerrungen in Systemen mit generativer KI wiedergeben.	K1	X		X	X	
GenAI-3.1.2	… Halluzinationen, Reasoning-Fehler und Verzerrungen in der LLM-Ausgabe identifizieren.	K3			X	X	
GenAI-3.1.3	… Verfahren zur Minderung von GenAI-Halluzinationen, Reasoning-Fehlern und Verzerrungen bei Software-Testaufgaben zusammenfassen.	K2			X	X	
GenAI-3.1.4	…Verfahren zur Risikominderung für nicht-deterministisches Verhalten von LLMs wiedergeben.	K1	X		X	X	
3.2	Datenschutz- und Sicherheitsrisiken generativer KI im Softwaretest						
GenAI-3.2.1	… die wichtigsten Risiken für den Datenschutz und die Sicherheit (security) im Zusammenhang mit der Verwendung von generativer KI im Softwaretest erläutern.	K2			X	X	

Geschäftlicher Nutzen: Certified Tester für Testen mit generativer KI			BO1	BO2	BO3	BO4	BO5
GenAI-3.2.2	… Beispiele für Datenschutz und Sicherheitslücken bei der Verwendung von generativer KI im Softwaretest nennen.	K2			X	X	
GenAI-3.2.3	… Strategien zur Minderung von Risiken zusammenfassen, um beim Einsatz von generativer KI im Softwaretest den Datenschutz zu wahren und die Sicherheit zu erhöhen.	K2			X	X	
3.3	Energieverbrauch und Umweltauswirkungen von generativer KI im Softwaretest						
GenAI-3.3.1	… die Auswirkungen von Aufgabenmerkmalen und Modellnutzung auf den Energieverbrauch von generativer KI im Softwaretest erläutern.	K2			X	X	
3.4	KI-Vorschriften, Standards und Best-Practice-Rahmenwerke						
GenAI-3.4.1	… Beispiele für KI-Vorschriften, Standards und Best-Practice-Rahmenwerke nennen, die für den Einsatz von generativer KI im Softwaretest relevant sind.	K1			X	X	X
4	LLM-gestützte Testinfrastruktur für den Softwaretest – 110 Minuten						
4.1	Architekturansätze für LLM-gestützte Testinfrastrukturen						
GenAI-4.1.1	… die wichtigsten Architekturkomponenten und Konzepte einer LLM-gestützten Testinfrastruktur erläutern	K2				X	X
GenAI-4.1.2	… die Retrieval-augmentierte Generierung zusammenfassen.	K2				X	X

Geschäftlicher Nutzen: Certified Tester für Testen mit generativer KI			BO1	BO2	BO3	BO4	BO5
GenAI-4.1.3	… die Rolle und Anwendung von LLM-gestützten Agenten bei der Automatisierung von Testprozessen erläutern.	K2				X	X
4.2	Fein-Tuning und LLMOps: Operationalisierung von generativer KI für den Softwaretest						
GenAI-4.2.1	… das Fein-Tuning von Sprachmodellen für bestimmte Testaufgaben erläutern.	K2				X	X
GenAI-4.2.2	… LLMOps und dessen Rolle bei der Bereitstellung und Verwaltung von LLMs für Testaufgaben erläutern.	K2				X	X
5	Bereitstellung und Integration generativer KI in Testorganisationen – 80 Minuten						
5.1	Roadmap für die Einführung von generativer KI im Softwaretest						
GenAI-5.1.1	… sich an die Risiken von Schatten-KI erinnern.	K1					X
GenAI-5.1.2	… die wichtigsten Aspekte erläutern, die bei der Definition einer Strategie für den Einsatz von generativer KI im Softwaretest zu berücksichtigen sind.	K2					X
GenAI-5.1.3	… die wichtigsten Kriterien für die Auswahl von LLMs/SLMs für Software-Testaufgaben in einem gegebenen Kontext zusammenfassen.	K2					X
GenAI-5.1.4	… sich an die wichtigsten Phasen bei der Einführung von generativer KI in einer Testorganisation erinnern.	K1					X

Geschäftlicher Nutzen: Certified Tester für Testen mit generativer KI			BO1	BO2	BO3	BO4	BO5
5.2	Management von Veränderungen bei der Einführung von generativer KI im Softwaretest						
GenAI-5.2.1	… die wesentlichen Fähigkeiten und Wissensbereiche erläutern, die Tester benötigen, um effektiv mit generativer KI in Testprozessen zu arbeiten.	K2					X
GenAI-5.2.2	… sich an Strategien zur Entwicklung von KI-Fähigkeiten innerhalb von Testteams erinnern, um die Einführung von generativer KI in Testaktivitäten zu unterstützen.	K1					X
GenAI-5.2.3	… erkennen, wie sich Testprozesse und Verantwortlichkeiten innerhalb einer Testorganisation bei der Einführung von generativer KI verändern.	K1					X

9 Anhang C – Versionshinweise

Diese Version ist V1.0. Für diese erste Version gibt es keine Versionshinweise.

10 Anhang D – Spezifische Schlüsselbegriffe für generative KI

Begriff	Definition
Benutzer-Prompt	Eine Instruktion oder Abfrage, die ein Benutzer in ein Large Language-Modell (LLM) eingibt und die die Antwort des Modells so steuert, dass bestimmte Aufgaben erfüllt oder gewünschte Informationen bereitgestellt werden.
Deep Learning	ML unter Verwendung neuronaler Netze mit mehreren Schichten.
Einbettung (*embedding*)	Eine Technik, mit der Token als dichte Vektoren in einem kontinuierlichen Raum dargestellt werden, die während des Trainings gelernt werden, um semantische, syntaktische und kontextuelle Beziehungen zu erfassen.
Fein-Tuning	Ein überwachtes Lernen, bei dem ein Datensatz mit gekennzeichneten Beispielen verwendet wird, um die Gewichtung des LLMs zu aktualisieren und sie an bestimmte Aufgaben oder Domänen anzupassen.
Few-Shot-Prompting	Ein Verfahren, bei dem einem Modell innerhalb der Prompt-Eingabe einige Beispiele gegeben werden, um es bei der Generierung geeigneter Antworten anzuleiten.
Foundation-LLM	Allzweckmodelle, die auf einer Vielzahl von Textdaten vortrainiert wurden und in der Lage sind, das nächste Wort auf der Grundlage gelernter Sprachmuster vorherzusagen. Synonym: Basis-LLM
Generative KI (GenAI)	Eine Art von künstlicher Intelligenz, die mithilfe von maschinellem Lernen (neue) intellektuelle Inhalte generiert, die von Menschen geschaffenen Inhalten ähneln.
Generativer vortrainierter Transformer (generative pre-trained transformer, GPT)	Eine Art Deep Learning-Modell auf Basis eines Transformers, das auf einer Vielzahl von Textdaten vortrainiert wurde, um menschenähnlich klingende Texte zu verstehen und zu generieren.
Halluzination	Falsche Informationen, die von einem LLM erzeugt werden.
Instruction-Tuned-LLM	Ein Foundation-LLM (Large Language-Modell), das darauf trainiert ist, Instruktionen zu befolgen, oft verstärkt durch Feedback, um richtige Antworten zu fördern.
KI-Chatbot	Ein Dialogagent, der LLMs verwendet, um Anfragen zu verarbeiten und menschenähnlich klingende Textantworten zu generieren, wodurch eine interaktive Kommunikation mit Benutzern ermöglicht wird.

Begriff	Definition
Kontextfenster	Der Textabschnitt, gemessen in Token, den ein Sprachmodell bei der Generierung von Antworten berücksichtigt und der die Relevanz und Kohärenz seiner Ausgaben beeinflusst.
Large Language-Modell (LLM)	Ein Computerprogramm, das sehr große Sprachdatensammlungen verwendet, um Texte auf ähnliche Weise wie Menschen zu verstehen und zu produzieren.
LLM-gestützter Agent	Eine Anwendung, die logisches Denken, Entscheidungsfindung und Gedächtnis eines LLMs integriert und Tools zur Ausführung von Aufgaben verwendet.
LLMOps	Praktiken und Werkzeuge, die sich auf die Bereitstellung, Überwachung und Wartung von LLMs in Produktionsumgebungen konzentrieren.
Maschinelles Lernen (ML)	Der Prozess, bei dem mithilfe von Berechnungstechniken Systeme in die Lage versetzt werden, aus Daten oder Erfahrungen zu lernen (ISO/IEC TR 29119-11).
Merkmal (*feature*)	Ein individuelles messbares Attribut der Eingabedaten, die für das Training durch einen ML-Algorithmus und für die Vorhersage durch ein ML-Modell verwendet werden.
Meta-Prompting	Die Erstellung von Instruktionen auf höherer Ebene, die spezifische Prompts zum Erkunden oder Automatisieren von Funktionen generieren.
Multimodales Modell	GenAI-Modelle, die Inhalte aus verschiedenen Datentypen wie Text, Bildern und Audio verarbeiten und generieren können.
One-Shot-Prompting	Ein Verfahren zum Verfassen von Prompts, bei dem der Prompt ein Beispiel enthält, um die Antwort des LLMs zu steuern.
Prompt	Eine Eingabe in natürlicher Sprache, die bereitgestellt wird, um eine bestimmte Antwort in generativer KI und Large-Language-Modellen hervorzurufen.
Prompt-Engineering	Der Prozess des Entwerfens und der Verfeinerung (Refinement) von Prompts, um LLMs zur Erzeugung gewünschter Ausgaben anzuleiten.
Prompt-Verkettung	Eine Prompting-Verfahren, bei dem die Ausgabe eines Prompt-Engineering-Prozesses als Eingabe für einen anderen verwendet wird, wodurch eine Folge von Prompts entsteht.
Reasoning-Fehler	Falsche Interpretation logischer Strukturen, wie z. B. Ursache-Wirkungs-Beziehungen, bedingte Logik oder schrittweise Problemlösungsprozesse, was zu falschen Schlussfolgerungen des LLMs führt.

Begriff	Definition
Reasoning-LLM	Ein LLM, das auf Instruction-Tuned-LLMs aufbaut, indem es deren Fähigkeit zur Nachahmung menschlicher Schlussfolgerungsprozesse verfeinert.
Retrieval-augmentierte Generierung (RAG-Technik)	Eine Technik, die die Fähigkeiten von LLMs mit einem Retriever kombiniert, um relevante Daten für die Generierung präziser, kontextbezogener Antworten abzurufen.
Schatten-KI	Die Verwendung von GenAI-Werkzeugen oder -Systemen innerhalb eines Unternehmens ohne formelle Genehmigung oder Aufsicht.
Small-Language-Modell (SLM)	Sprachmodelle, die bewusst klein entworfen und trainiert sind und ein Gleichgewicht zwischen Effizienz und aufgabenspezifischem Sprachverständnis bieten.
Symbolische KI	Ein KI-Ansatz, der Symbole, Regeln und strukturiertes Wissen verwendet, um logische Denkprozesse zu modellieren.
System-Prompt	Ein vordefinierter Befehlssatz, der in der Regel für die Benutzer des Chatbots nicht sichtbar ist und der den Kontext, den Ton und die Grenzen für die Antworten eines LLMs konsistent festlegt und dessen Verhalten während der Interaktionen steuert.
Temperatur	Ein Parameter, der die Zufälligkeit oder Kreativität der Ausgaben eines LLM-Modells steuert.
Tokenisierung	Der Prozess der Aufteilung von Text in kleinere Einheiten zur Verarbeitung durch Sprachmodelle.
Transformer	Eine Deep Learning-Modellarchitektur, die Selbstaufmerksamkeitsmechanismen nutzt, um weitreichende Abhängigkeiten in Eingabesequenzen zu erfassen.
Vektordatenbank	Eine Datenbank, die für die Speicherung und Abfrage hochdimensionaler Vektordarstellungen von Daten optimiert ist.
Verarbeitung natürlicher Sprache (NLP)	Die Verarbeitung von in natürlicher Sprache kodierten Daten durch Computer zum Abrufen von Informationen und zur Wissensrepräsentation.
Vision-Language-Modell	Ein GenAI-System, das visuelle und textuelle Daten gemeinsam verarbeitet, um Aufgaben auszuführen, indem es Inhalte über beide Modalitäten hinweg verknüpft und generiert.
Zero-Shot-Prompting	Ein Verfahren zum Schreiben von Prompts, bei dem der Prompt keine Beispiele enthält und sich auf das bereits vorhandene Wissen des Modells stützt, um eine Antwort zu generieren.

11 Anhang E – Eingetragene Marken

ISTQB® ist eine eingetragene Marke des International Software Testing Qualifications Board.

12 Index

Alle Testbegriffe sind im Glossar des ISTQB® definiert (http://glossary.istqb.org/).