KI in Qualitätssicherung und Testautomatisierung

Generative KI systematisch für Softwarequalität und Test einsetzen

01 INTRO

Generative KI kann zahlreiche Aufgaben in Qualitätssicherung und Test unterstützen: Anforderungen analysieren, Testideen entwickeln, Testfälle erzeugen, Testdaten vorbereiten, Automatisierungscode erstellen, Fehlerbilder untersuchen und bestehende Tests bewerten. Der eigentliche Nutzen entsteht jedoch nicht dadurch, möglichst viel Testarbeit an ein Sprachmodell zu delegieren, sondern durch die kontrollierte Integration von KI in einen belastbaren Quality-Engineering-Prozess.

Der Kurs verbindet klassische Methoden des Softwaretests mit AI-gestützten Verfahren und praktischer Testautomatisierung. Die Teilnehmer entwickeln und bewerten AI-gestützte Testprozesse von der Anforderungsanalyse über Testdesign und Automatisierung bis zur Auswertung. Ein besonderer Schwerpunkt liegt auf der Qualität der erzeugten Ergebnisse: Vollständigkeit, Nachvollziehbarkeit, Reproduzierbarkeit und die Verifikation von AI-generierten Tests sind selbst Gegenstand der Qualitätssicherung.

DAUER

5 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Praktische Erfahrung in Softwaretest, Qualitätssicherung oder Testautomatisierung

02 TRAININGSPROGRAMM

Inhalte

KI im Quality Engineering einordnen

- klassische Test- und QS-Prozesse
- generative KI als Werkzeug im Testprozess
- LLMs und Coding Agents
- geeignete und ungeeignete QS-Aufgaben
- Assistenz, Generierung und autonome Bearbeitung
- deterministische und probabilistische Verfahren
- Chancen und neue Fehlerquellen
- Verantwortung für das Testergebnis

Was verändert sich durch AI-generierte Software?

- steigende Geschwindigkeit der Codeerzeugung
- mehr Änderungen in kürzerer Zeit
- AI-generierte Implementierungen
- plausible, aber fehlerhafte Lösungen
- neue Anforderungen an Reviews und Tests
- Verschiebung des Engpasses
- unabhängige Verifikation
- Quality Engineering als Kontrollinstanz

Anforderungen mit KI analysieren

- Anforderungen strukturieren
- funktionale Anforderungen identifizieren
- Qualitätsanforderungen erkennen
- Mehrdeutigkeiten aufdecken
- Widersprüche suchen
- fehlende Informationen identifizieren
- Annahmen sichtbar machen
- Analyseergebnisse gegen die Originalquelle prüfen

Anforderungen auf Testbarkeit untersuchen

- überprüfbare Aussagen
- unklare Akzeptanzkriterien
- implizite Randbedingungen
- fehlende Fehlerfälle
- Daten- und Zustandsabhängigkeiten
- Vor- und Nachbedingungen
- Testbarkeit verbessern
- KI als Review-Unterstützung

Akzeptanzkriterien entwickeln

- Anforderungen in überprüfbare Kriterien überführen
- positive und negative Kriterien
- Grenzbedingungen
- Geschäftsregeln
- Zustandsübergänge
- Given-When-Then
- Beispiele als Spezifikation
- menschliche Prüfung der erzeugten Kriterien

Testideen mit KI entwickeln

- Testgegenstand beschreiben
- Risiken und Qualitätsmerkmale berücksichtigen
- positive Tests
- negative Tests
- Grenzfälle
- Ausnahmefälle
- Kombinationen
- ungewöhnliche und adversariale Testideen

Klassische Testentwurfsverfahren und KI

- Äquivalenzklassen
- Grenzwertanalyse
- Entscheidungstabellen
- Zustandsübergangstests
- Use-Case-basierte Tests
- kombinatorische Verfahren
- exploratives Testen
- KI zur Unterstützung statt Ersatz der Testmethodik

Testfälle generieren

- aus Anforderungen
- aus Akzeptanzkriterien
- aus User Stories
- aus Geschäftsregeln
- aus vorhandener Dokumentation
- aus bestehendem Code
- aus Schnittstellendefinitionen
- aus Fehlerhistorien

Qualität AI-generierter Testfälle

- fachliche Richtigkeit
- Relevanz
- Redundanz
- Vollständigkeit
- Nachvollziehbarkeit
- Wartbarkeit
- falsche Annahmen
- scheinbare Testabdeckung

Coverage jenseits von Code Coverage

- Requirements Coverage
- Risk Coverage
- Feature Coverage
- Datenabdeckung
- Zustandsabdeckung
- Regelabdeckung
- Traceability
- Lücken systematisch identifizieren

KI zur Analyse bestehender Tests

- Testsuite erschließen
- Testabsichten erkennen
- redundante Tests
- fehlende Testfälle
- schwache Assertions
- problematische Testdaten
- fragiler Testcode
- Wartbarkeitsprobleme

Testdaten mit KI entwickeln

- Testdaten aus Anforderungen ableiten
- gültige und ungültige Daten
- Grenzwerte
- Kombinationen
- strukturierte Datensätze
- synthetische Daten
- Abhängigkeiten zwischen Feldern
- Plausibilität überprüfen

Datenschutz bei Testdaten

- personenbezogene Daten
- Produktionsdaten
- vertrauliche Informationen
- Pseudonymisierung und Anonymisierung
- synthetische Daten
- Daten an externe Modelle übertragen
- organisatorische Vorgaben
- sichere Testdatenprozesse

KI und API-Testing

- OpenAPI und andere Schnittstellenbeschreibungen analysieren
- Endpunkte und Operationen erfassen
- Request- und Response-Strukturen
- positive und negative API-Tests
- Schema-Verletzungen
- Statuscodes
- Fehlerbehandlung
- API-Testcode generieren und überprüfen

KI und UI-Testing

- Benutzerabläufe beschreiben
- Testfälle aus UI-Anforderungen
- vorhandene Oberflächen analysieren
- Selektoren und Locators
- Page Objects und andere Abstraktionen
- Assertions
- Synchronisation und Wartebedingungen
- Grenzen AI-generierter UI-Tests

Testautomatisierung mit generativer KI

- Testfall zu Automatisierungscode
- vorhandenes Framework berücksichtigen
- Projektkonventionen übernehmen
- Setup und Fixtures
- Assertions
- Parametrisierung
- Testdaten
- erzeugten Testcode kritisch überprüfen

Bestehende Automatisierungsframeworks erschließen

- Projektstruktur analysieren
- Testarchitektur erkennen
- Framework-Konventionen
- Hilfsfunktionen
- Fixtures
- Testdatenkonzepte
- Reporting
- neue Tests konsistent integrieren

AI-generierter Testcode und Wartbarkeit

- Lesbarkeit
- Duplikate
- unnötige Abstraktionen
- falsche Abstraktionen
- überkomplexer Testcode
- stabile Schnittstellen
- Testcode als Produktionscode behandeln
- Refactoring AI-generierter Tests

Assertions und Oracles

- was soll tatsächlich geprüft werden?
- schwache Assertions
- übermäßig spezifische Assertions
- implizite Annahmen
- Test Oracles
- Referenzdaten
- Invarianten
- KI kann Erwartungswerte nicht beliebig selbst bestimmen

Das Oracle-Problem bei generativer KI

- erwartetes Ergebnis ohne eindeutige Referenz
- Modell bewertet eigenes Ergebnis
- Zirkelschlüsse
- unabhängige Quellen
- deterministische Prüfungen
- Regeln und Invarianten
- menschliche Bewertung
- geeignete Kombination verschiedener Oracles

Fehleranalyse mit KI

- Fehlermeldungen
- Stacktraces
- Logs
- Screenshots
- Testreports
- reproduzierbare Fehler
- Hypothesen über Fehlerursachen
- Hypothesen experimentell überprüfen

Fehlgeschlagene Tests analysieren

- Produktfehler oder Testfehler
- Umgebungsprobleme
- Timing-Probleme
- Datenprobleme
- Flaky Tests
- wiederkehrende Fehlermuster
- Clustering von Fehlern
- KI-Vermutung und tatsächliche Ursache trennen

Flaky Tests

- typische Ursachen
- Timing
- Parallelität
- externe Abhängigkeiten
- instabile Testdaten
- Logs und historische Ergebnisse analysieren
- Reparaturvorschläge
- Symptome nicht einfach durch Retries verstecken

Testreports und große Ergebnismengen

- Testergebnisse zusammenfassen
- Fehlermuster erkennen
- ähnliche Fehler gruppieren
- relevante von Folgefehlern unterscheiden
- Trends
- Regressionen
- Management Summary
- Verlust wichtiger Details durch Zusammenfassung vermeiden

Exploratives Testen mit KI

- Testcharter entwickeln
- Risiken identifizieren
- alternative Nutzungsszenarien
- ungewöhnliche Eingaben
- neue Testideen während der Session
- Beobachtungen strukturieren
- KI als Sparringspartner
- menschliche Exploration nicht durch Skripte ersetzen

Risikobasiertes Testen mit KI

- Produktrisiken identifizieren
- Schadensausmaß und Eintrittswahrscheinlichkeit
- technische und fachliche Risiken
- Änderungen priorisieren
- Testintensität ableiten
- AI-Unterstützung bei der Risikoanalyse
- fehlendes Domänenwissen berücksichtigen
- Priorisierung nachvollziehbar halten

Regressionstests auswählen

- Änderungen analysieren
- betroffene Komponenten
- Abhängigkeiten
- historische Fehler
- vorhandene Tests
- risikobasierte Auswahl
- AI-basierte Empfehlungen
- deterministische Kriterien ergänzen

Testen AI-basierter Systeme

- probabilistische Ergebnisse
- nicht deterministisches Verhalten
- unterschiedliche Formulierungen
- semantische statt rein syntaktische Bewertung
- Referenzdatensätze
- Toleranzen
- wiederholte Ausführung
- statistische Betrachtung

LLM-Anwendungen testen

- Prompt und System Instructions
- Kontext
- Retrieval
- Tool-Aufrufe
- strukturierte Ausgaben
- Halluzinationen
- unerwünschtes Verhalten
- Regressionen nach Modell- oder Promptänderungen

Evaluations für LLM-Systeme

- Testdatensätze
- erwartete Eigenschaften
- automatische Evaluatoren
- regelbasierte Prüfungen
- LLM-as-a-Judge
- menschliche Bewertung
- Metriken
- reproduzierbare Evaluation

Grenzen von LLM-as-a-Judge

- Modellbias
- instabile Bewertungen
- Abhängigkeit vom Bewertungs-Prompt
- Selbstbewertung
- Referenzmodelle
- Kalibrierung
- Stichproben durch Menschen
- keine einzelne Metrik als Wahrheit behandeln

RAG-Systeme testen

- Retrieval und Generation getrennt prüfen
- relevante Dokumente
- Precision und Recall im Retrieval
- Kontextqualität
- Quellenbezug
- unbeantwortbare Fragen
- Halluzination trotz korrektem Retrieval
- Regressionstests für Wissenssysteme

Agentische Systeme testen

- Agentenentscheidungen
- Tool-Auswahl
- Tool-Parameter
- mehrstufige Abläufe
- Zwischenzustände
- Abbruchbedingungen
- unerwartete Aktionen
- Endergebnis und Prozess getrennt bewerten

Testen von Tool- und MCP-Integrationen

- Tool-Schnittstellen
- Eingabevalidierung
- Rückgabewerte
- Fehlerfälle
- Berechtigungen
- unerwartete Tool-Aufrufe
- manipulierte Tool-Ergebnisse
- Integrationstest für agentische Workflows

Prompt Injection als Testgegenstand

- direkte Prompt Injection
- indirekte Prompt Injection
- manipulierte Dokumente
- Webseiten und externe Inhalte
- Tool-Ausgaben als untrusted Input
- Daten und Instruktionen trennen
- Security-Testfälle
- technische Schutzmechanismen überprüfen

AI-gestützte Testagenten

- Testaufgabe autonom bearbeiten
- Anwendung untersuchen
- Testfälle entwickeln
- Tests implementieren
- Tests ausführen
- Ergebnisse analysieren
- Fehler nacharbeiten
- Grenzen der Autonomie

Testagenten kontrollieren

- erlaubte Systeme
- erlaubte Aktionen
- Test- und Produktionsumgebung
- Credentials
- Datenzugriffe
- Quality Gates
- Protokollierung
- Human-in-the-Loop

KI im Testprozess integrieren

- Anforderungen
- Testanalyse
- Testdesign
- Automatisierung
- Ausführung
- Auswertung
- Defect Analysis
- Reporting
- geeignete AI-Unterstützung je Prozessschritt

Human und AI sinnvoll kombinieren

- repetitive Arbeit delegieren
- Varianten erzeugen lassen
- Analyse beschleunigen
- fachliche Entscheidungen beim Menschen
- unabhängige Verifikation
- Stichproben
- Vier-Augen-Prinzip
- Verantwortung bleibt zuordenbar

Reproduzierbarkeit und Traceability

- verwendete Eingaben
- Modell und Version
- relevante Konfiguration
- erzeugte Artefakte
- Quellen
- Änderungen
- Testresultate
- nachvollziehbare Entstehungskette

AI-generierte Testartefakte verwalten

- Testfälle
- Testcode
- Testdaten
- Prompts und Instruktionen
- Evaluationsdatensätze
- Versionierung
- Reviews
- AI-generierte Artefakte in bestehende Prozesse integrieren

Quality Gates für AI-gestützte Testprozesse

- keine ungeprüfte Übernahme
- Syntax und Build
- Testausführung
- statische Analyse
- fachliche Review
- Coverage-Prüfung
- Traceability
- verbindliche Freigabekriterien

Datenschutz, Security und Governance

- Quellcode
- Anforderungen
- Testdaten
- Logs
- vertrauliche Projektdaten
- externe AI-Dienste
- Berechtigungen
- organisatorische Regeln

Nutzen von KI im Test messbar machen

- Zeitersparnis
- zusätzliche Testideen
- Fehlerfindungsrate
- Wartungsaufwand
- False Positives
- Nachbearbeitungsaufwand
- Kosten
- Produktivitätsgewinn nicht nur behaupten, sondern messen

AI-gestützte QS-Strategie entwickeln

- geeignete Einsatzfelder identifizieren
- Risiken bewerten
- vorhandenen Testprozess analysieren
- Werkzeuge auswählen
- Kontrollmechanismen festlegen
- Pilotbereich definieren
- Erfolgskriterien
- schrittweise Einführung

Praktische Übungen

- Anforderungen mit einem LLM analysieren
- Testideen und Testfälle systematisch erzeugen
- klassische Testentwurfsverfahren AI-gestützt anwenden
- AI-generierte Testfälle auf Lücken und Fehler prüfen
- Testdaten entwickeln
- API-Tests generieren und verifizieren
- UI-Testautomatisierung erweitern
- bestehende Testsuite analysieren
- fehlgeschlagene Tests und Logs untersuchen
- Testfälle für eine LLM-Anwendung entwickeln
- Evaluation für nichtdeterministische Ergebnisse aufbauen
- agentischen Testworkflow untersuchen
- vollständigen AI-gestützten QS-Workflow entwerfen und bewerten

03 DURCHFÜHRUNG

Rahmen und Organisation

Dauer	5 Tage
Durchführung	Präsenz oder Live Online, vorzugsweise als Inhouse-Schulung
Schwerpunkt	Inhouse-Schulungen für Unternehmen und Teams
Sprache	Deutsch, englische Fachbegriffe und Dokumentation
Unterlagen	Kursunterlagen, Beispiele und Übungsaufgaben
Technik	Eigener Rechner mit Entwicklungsumgebung, Testframework und Zugang zu einem aktuellen Sprachmodell

> **Inhouse-Schulungen**
> Vorhandene Testframeworks, Anwendungen, APIs, CI/CD-Prozesse, Testmanagementsysteme und unternehmensspezifische Qualitätsanforderungen können in die Übungen einbezogen werden.

04 ZIELGRUPPE

Für wen ist der Kurs gedacht?

Der Kurs richtet sich an Softwaretester, Testanalysten, Testautomatisierer, Quality Engineers, Testmanager, QA Leads und Entwickler mit Verantwortung für Softwarequalität.

Er eignet sich insbesondere für erfahrene Test- und QA-Teams, die generative KI nicht nur punktuell ausprobieren, sondern systematisch in Testanalyse, Testdesign, Automatisierung und Qualitätsbewertung integrieren möchten.

> **Voraussetzungen**
> Praktische Erfahrung in Softwaretest oder Qualitätssicherung wird vorausgesetzt. Für die praktischen Anteile zur Testautomatisierung sind Grundkenntnisse einer Programmiersprache hilfreich; Kenntnisse klassischer Testentwurfsverfahren und Erfahrung mit automatisierten Tests erleichtern den Einstieg. Spezielle Kenntnisse in Machine Learning oder der Entwicklung von LLM-Systemen sind nicht erforderlich.

05 LERNZIELE

Was Sie nach dem Kurs können

Nach dem Kurs können die Teilnehmer:

- geeignete Einsatzfelder für generative KI in QS und Test identifizieren
- Anforderungen AI-gestützt auf Qualität und Testbarkeit analysieren
- Testideen und Testfälle systematisch mit KI entwickeln
- klassische Testentwurfsverfahren mit AI-Unterstützung kombinieren
- die Qualität AI-generierter Tests kritisch bewerten
- Testdaten kontrolliert erzeugen
- AI zur Unterstützung von API- und UI-Testautomatisierung einsetzen
- bestehende Tests und Automatisierungsframeworks analysieren
- Fehler, Logs und fehlgeschlagene Tests AI-gestützt untersuchen
- AI-generierte Testautomatisierung auf Wartbarkeit und Aussagekraft prüfen
- LLM-, RAG- und agentische Systeme angemessen testen
- Evaluationsverfahren für probabilistische Systeme entwickeln
- Grenzen von LLM-as-a-Judge einschätzen
- Testagenten kontrolliert einsetzen
- Prompt Injection und Tool-Nutzung als Testgegenstände berücksichtigen
- AI-generierte Testartefakte nachvollziehbar verwalten
- verbindliche Quality Gates für AI-gestützte Testprozesse etablieren
- Nutzen und Risiken von AI im Test messbar bewerten
- eine AI-gestützte QS-Strategie für ein Projekt oder Unternehmen entwickeln

06 FAQ

Häufige Fragen

Was unterscheidet diesen Kurs von AI-assisted Software Development?

AI-assisted Software Development betrachtet AI-Unterstützung über den gesamten Softwareentwicklungsprozess aus Sicht der Entwicklung. Testing ist dort ein wichtiger Bestandteil, aber nur einer von mehreren.

Dieser Kurs betrachtet den gesamten Themenkomplex aus Sicht von Quality Engineering und Test. Testanalyse, Testdesign, Testautomatisierung, Testdaten, Fehleranalyse, Coverage, Test Oracles und die Qualität AI-generierter Testartefakte werden wesentlich tiefer behandelt.

Was unterscheidet ihn von Agentic Software Development?

Agentic Software Development beschäftigt sich mit der Architektur und Kontrolle agentischer Entwicklungsprozesse. Dieser Kurs behandelt Agenten nur dort, wo sie für Test und Qualitätssicherung relevant sind – etwa als Testagenten oder als zu testendes System.

Ist der Kurs hauptsächlich eine Schulung für AI-generierte Testautomatisierung?

Nein. Testautomatisierung ist ein wesentlicher Teil, aber der Kurs beginnt deutlich früher im Testprozess: bei Anforderungen, Risiken, Testanalyse und Testdesign. Gute Automatisierung kann schlechte oder unvollständige Testideen nicht kompensieren.

Werden klassische Testentwurfsverfahren noch benötigt?

Ja. Gerade bei generativer KI werden systematische Testverfahren wichtiger, weil sie einen unabhängigen Maßstab liefern, anhand dessen sich AI-generierte Testideen beurteilen lassen.

Werden konkrete Testautomatisierungswerkzeuge verwendet?

Ja. Praktische Übungen können beispielsweise mit Playwright, Selenium, pytest oder vergleichbaren Werkzeugen durchgeführt werden. Der Kurs ist jedoch nicht als Produktschulung für ein bestimmtes Testframework konzipiert.

Werden auch AI-Anwendungen selbst getestet?

Ja. Der Kurs behandelt neben AI-Unterstützung im klassischen Softwaretest auch die besonderen Herausforderungen beim Test von LLM-, RAG- und agentischen Systemen.

Ist LLM-as-a-Judge eine zuverlässige Testmethode?

Es kann ein nützlicher Bestandteil einer Evaluation sein, ist aber kein unabhängiges und deterministisches Oracle. Deshalb werden Kombinationen aus regelbasierten Prüfungen, Referenzdaten, statistischen Verfahren und menschlicher Bewertung behandelt.

Werden Testagenten behandelt?

Ja. Testagenten werden sowohl praktisch als Werkzeug als auch aus QA-Sicht betrachtet. Der Schwerpunkt liegt dabei auf kontrollierter Autonomie, überprüfbaren Ergebnissen und klaren Grenzen.

Muss ich programmieren können?

Für Testanalyse, Testdesign und viele AI-gestützte QS-Aufgaben nicht zwingend. Für die praktischen Teile zur Testautomatisierung sind grundlegende Programmierkenntnisse jedoch sinnvoll.

Warum dauert der Kurs fünf Tage?

Der Kurs verbindet drei umfangreiche Themenbereiche: AI-Unterstützung des klassischen Testprozesses, AI-gestützte Testautomatisierung und Qualitätssicherung AI-basierter Systeme. Alle drei werden praktisch behandelt und durch Methoden zur Verifikation AI-generierter Ergebnisse verbunden. Eine kürzere Durchführung ist möglich, erfordert aber eine deutliche Reduktion der praktischen oder vertiefenden Inhalte.

Kann der Kurs auf vier Tage verkürzt werden?

Ja. Eine viertägige Variante kann auf einen Schwerpunkt zugeschnitten werden, beispielsweise AI im klassischen Testprozess und Testautomatisierung oder die Qualitätssicherung AI-basierter Systeme. Für den vollständigen beschriebenen Umfang sind fünf Tage vorgesehen.

Kann der Kurs an unsere Testlandschaft angepasst werden?

Ja. Bei Inhouse-Schulungen können vorhandene Testframeworks, Anwendungen, APIs, CI/CD-Prozesse, Testmanagementsysteme und unternehmensspezifische Qualitätsanforderungen in die Übungen einbezogen werden.

UC IT Service · Ulrich Cuber · kontakt@uc-it.de
https://www.uc-it.de/coaching/ki-und-qualitaetssicherung/