Copilot im Einsatz

Copilot produktiv in der Softwareentwicklung einsetzen

01 INTRO

Copilot hat sich vom Werkzeug für Code Completion zu einer umfassenden AI-Unterstützung für die Softwareentwicklung entwickelt. Neben Vorschlägen direkt im Editor können Entwickler mit Copilot Code erklären, verändern und testen, Entwicklungsaufgaben agentisch bearbeiten und Reviews unterstützen.

Der Kurs vermittelt den praktischen Einsatz von Copilot im Entwicklungsalltag. Die Teilnehmer arbeiten mit einer bestehenden Codebasis und lernen, Copilot gezielt für Implementierung, Analyse, Refactoring, Tests, Debugging und Code Review einzusetzen. Dabei werden auch Projektinstruktionen, Agent Mode, Copilot CLI, MCP sowie die kontrollierte Nutzung agentischer Funktionen behandelt.

DAUER

3 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Praktische Erfahrung in der Softwareentwicklung, Git und einer unterstützten Entwicklungsumgebung

02 TRAININGSPROGRAMM

Inhalte

Copilot einordnen

- Entwicklung von Code Completion zu AI-assisted Development
- Copilot im Editor
- Copilot Chat
- Agent Mode
- Copilot Coding Agent
- Copilot CLI
- Copilot Code Review
- unterschiedliche Einsatzformen
- Zusammenspiel der Komponenten

Einrichtung und Entwicklungsumgebung

- GitHub- und Copilot-Zugang
- unterstützte Entwicklungsumgebungen
- Copilot konfigurieren
- verfügbare Modelle
- Modellwahl
- Einstellungen
- organisatorische Vorgaben
- erste Arbeit mit einem vorhandenen Repository

Code Completion produktiv einsetzen

- Inline Suggestions
- Vorschläge im Kontext des vorhandenen Codes
- Funktionen und Methoden ergänzen
- wiederkehrenden Code erzeugen
- Kommentare und Code
- geeignete Aufgaben für Completion
- Vorschläge beurteilen
- Vorschläge nicht unkritisch übernehmen

Copilot Chat

- Fragen zum Code stellen
- Code erklären lassen
- Änderungen vorbereiten
- Fehler analysieren
- Tests erzeugen
- Dokumentation erstellen
- Kontext gezielt bereitstellen
- gute Aufgabenstellungen für Entwicklungsaufgaben

Mit einer bestehenden Codebasis arbeiten

- Repository erschließen
- Projektstruktur untersuchen
- relevante Dateien finden
- Abhängigkeiten nachvollziehen
- bestehende Patterns erkennen
- Implementierungen verfolgen
- unbekannten Code erklären lassen
- Aussagen von Copilot gegen den tatsächlichen Code prüfen

Kontext richtig einsetzen

- aktueller Editor-Kontext
- Dateien und Codebereiche
- Repository-Kontext
- relevanten Kontext auswählen
- unnötigen Kontext vermeiden
- Anforderungen und Code kombinieren
- Grenzen des Kontextes
- Kontextqualität als Voraussetzung guter Ergebnisse

Entwicklungsaufgaben formulieren

- Ziel der Änderung
- Ausgangszustand
- gewünschtes Verhalten
- Randbedingungen
- Akzeptanzkriterien
- Nicht-Ziele
- vorhandene Konventionen
- Definition of Done

Code mit Copilot entwickeln

- neue Funktionen implementieren
- bestehenden Code erweitern
- Schnittstellen implementieren
- Fehlerbehandlung
- Datenverarbeitung
- vorhandene Patterns übernehmen
- größere Aufgaben zerlegen
- Änderungen schrittweise kontrollieren

Refactoring mit Copilot

- Refactoring-Kandidaten erkennen
- Funktionen zerlegen
- Namen verbessern
- Duplikate reduzieren
- Strukturen vereinfachen
- bestehendes Verhalten erhalten
- Tests als Sicherheitsnetz
- Änderungen über Diffs kontrollieren

Tests mit Copilot

- Unit Tests erzeugen
- vorhandene Tests verstehen
- Testfälle ergänzen
- Grenzfälle identifizieren
- Fehlerfälle testen
- Testdaten erzeugen
- Testcode überprüfen
- Implementierung und Tests unabhängig betrachten

Debugging und Fehleranalyse

- Fehlermeldungen analysieren
- Stacktraces untersuchen
- mögliche Ursachen formulieren
- relevante Codebereiche finden
- Hypothesen überprüfen
- Fixes entwickeln
- Tests zur Reproduktion einsetzen
- vorgeschlagene Lösungen verifizieren

Copilot Agent Mode

- Unterschied zwischen Chat und Agent Mode
- mehrstufige Entwicklungsaufgaben
- Projektdateien untersuchen
- Dateien verändern
- Kommandos und Entwicklungswerkzeuge verwenden
- Tests ausführen
- Fehler erkennen und nacharbeiten
- Agentenarbeit kontrollieren

Mit agentischen Aufgaben arbeiten

- geeignete Aufgabengröße
- Aufgabe zunächst analysieren lassen
- Plan vor Implementierung
- Arbeitsschritte begrenzen
- Zwischenstände prüfen
- Änderungen nachvollziehen
- unerwünschte Seiteneffekte erkennen
- Kontrolle beim Entwickler behalten

Copilot Coding Agent

- lokale und cloudbasierte Agenten unterscheiden
- Aufgaben an einen Coding Agent übergeben
- Repository-Kontext
- selbstständige Bearbeitung von Entwicklungsaufgaben
- erzeugte Änderungen überprüfen
- Pull-Request-basierte Arbeitsweisen
- geeignete und ungeeignete Aufgaben
- Agent nicht mit Freigabeinstanz verwechseln

Custom Instructions

- dauerhafte Projektinstruktionen
- copilot-instructions.md im Repository
- pfadspezifische Instructions
- AGENTS.md
- Coding Conventions
- Architekturregeln
- Testanforderungen
- Build- und Validierungsregeln
- Instruktionen wartbar strukturieren

Prompt Files und wiederverwendbare Arbeitsweisen

- wiederkehrende Entwicklungsaufgaben
- Aufgaben standardisieren
- Projektkontext wiederverwenden
- Review- und Analyseaufgaben
- Test- und Dokumentationsaufgaben
- Teamkonventionen
- wartbare Prompt-Strukturen
- Abgrenzung zu Custom Instructions

Custom Agents

- spezialisierte Agenten
- Aufgaben und Verantwortlichkeiten
- eigenes Toolset
- projektspezifische Agenten
- Reviewer
- Analyse- und Entwicklungsagenten
- Arbeit auf mehrere Agenten verteilen
- Grenzen und Kontrolle

Copilot CLI

- Copilot im Terminal
- Repository analysieren
- Entwicklungsaufgaben aus dem Terminal bearbeiten
- Dateien und Änderungen untersuchen
- Kommandos kontrolliert ausführen
- Sessions und Kontext
- Custom Instructions in der CLI
- CLI in den Entwicklungsworkflow integrieren

Copilot Code Review

- Änderungen analysieren lassen
- Review vor Commit
- Pull Requests prüfen
- Review-Kontext bereitstellen
- Custom Instructions für Reviews
- echte Fehler von wenig hilfreichen Hinweisen unterscheiden
- AI-Review verifizieren
- Copilot als zusätzliche Review-Instanz

MCP mit Copilot

- Model Context Protocol einordnen
- externe Tools und Datenquellen anbinden
- MCP Server konfigurieren
- Datenbanken und APIs
- Entwicklungswerkzeuge
- eigene MCP Tools
- Tool-Berechtigungen
- Vertrauensgrenzen bei externen Werkzeugen

Copilot an das Projekt anpassen

- Repository-weite Regeln
- pfadspezifische Regeln
- Agent Instructions
- Skills und wiederverwendbare Arbeitsabläufe
- projektspezifische Werkzeuge
- gemeinsame Konfiguration im Team
- Regeln versionieren
- Projektkonfiguration als Bestandteil des Repositories

Qualität von Copilot-generiertem Code

- generierten Code verstehen
- Diffs kontrollieren
- Compiler und Interpreter
- Tests
- Linter
- statische Analyse
- Security-Prüfungen
- bestehende Qualitäts-Gates beibehalten

Halluzinationen und typische Fehler

- erfundene APIs
- falsche Signaturen
- nicht vorhandene Dateien
- falsche Annahmen über die Codebasis
- veraltete Bibliothekskenntnisse
- unnötige Abhängigkeiten
- scheinbar plausible Lösungen
- ausführbare Verifikation statt Vertrauen

Security und Datenschutz

- Quellcode und vertrauliche Informationen
- Secrets
- Logs und Konfigurationen
- externe Inhalte
- Prompt Injection
- Tool-Zugriffe
- agentische Aktionen
- organisatorische Richtlinien

Copilot im Team

- gemeinsame Instructions
- Coding Standards
- einheitliche Qualitätsanforderungen
- unterschiedliche Entwicklungsumgebungen
- Reviews
- Wissen im Repository festhalten
- geeignete Team-Workflows
- individuelle Nutzung und gemeinsame Regeln

Ein professioneller Copilot-Workflow

- Issue oder Anforderung analysieren
- Repository untersuchen
- Kontext bereitstellen
- Änderung planen
- Tests und Akzeptanzkriterien festlegen
- Implementierung mit Copilot
- Tests und Qualitätswerkzeuge ausführen
- Diff reviewen
- Code Review durchführen
- Änderung dokumentieren und committen

Praktische Übungen

- bestehendes Repository mit Copilot analysieren
- Custom Instructions für das Projekt erstellen
- Entwicklungsaufgabe formulieren
- Code mit Chat und Agent Mode verändern
- Tests erzeugen und verbessern
- Fehler analysieren und beheben
- Refactoring durchführen
- Copilot CLI einsetzen
- Code Review mit Copilot durchführen
- agentische Aufgabe kontrolliert bearbeiten
- vollständigen Copilot-Workflow durchführen

03 DURCHFÜHRUNG

Rahmen und Organisation

Dauer	3 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 unterstützter Entwicklungsumgebung, Git und Copilot-Zugang

> **Inhouse-Schulungen**
> Programmiersprachen, Frameworks, Repository-Struktur, Entwicklungsumgebung, Testwerkzeuge und vorhandene GitHub-Prozesse können in die Übungen einbezogen werden.

04 ZIELGRUPPE

Für wen ist der Kurs gedacht?

Der Kurs richtet sich an Softwareentwickler, Softwarearchitekten, technische Consultants und Entwicklungsteams, die Copilot professionell und systematisch in der Softwareentwicklung einsetzen möchten.

Er eignet sich sowohl für Entwickler, die Copilot bisher hauptsächlich für Code Completion verwenden, als auch für Teams, die Chat, Agent Mode, Coding Agents, CLI und erweiterte Integrationen produktiv einsetzen möchten.

> **Voraussetzungen**
> Praktische Erfahrung in der Softwareentwicklung, Git und mindestens einer Programmiersprache wird vorausgesetzt. Die Teilnehmer sollten mit einer von Copilot unterstützten Entwicklungsumgebung arbeiten können. Vorherige Copilot-Erfahrung ist hilfreich, aber nicht erforderlich.

05 LERNZIELE

Was Sie nach dem Kurs können

Nach dem Kurs können die Teilnehmer:

- Copilot für unterschiedliche Entwicklungsaufgaben gezielt einsetzen
- Code Completion und Chat effizient nutzen
- bestehende Codebasen mit Copilot analysieren
- Entwicklungsaufgaben mit geeignetem Kontext formulieren
- Code, Tests und Refactorings mit Copilot entwickeln
- Agent Mode für mehrstufige Aufgaben einsetzen
- Coding Agents sinnvoll einordnen und kontrollieren
- Custom Instructions für Projekte erstellen
- wiederverwendbare Arbeitsweisen etablieren
- Copilot CLI im Entwicklungsworkflow einsetzen
- Copilot Code Review sinnvoll verwenden
- MCP und externe Tools anbinden und einordnen
- AI-generierten Code systematisch verifizieren
- typische Fehler und Halluzinationen erkennen
- Security- und Datenschutzaspekte berücksichtigen
- Copilot in bestehende Entwicklungs- und Qualitätsprozesse integrieren

06 FAQ

Häufige Fragen

Ist der Kurs nur für Visual Studio Code geeignet?

Nein. Die grundlegenden Copilot-Arbeitsweisen sind nicht auf einen einzelnen Editor beschränkt. Für praktische Übungen wird jedoch eine unterstützte Entwicklungsumgebung benötigt. Die konkrete Umgebung kann bei Inhouse-Schulungen abgestimmt werden.

Muss ich Copilot bereits verwendet haben?

Nein. Der Kurs beginnt mit den grundlegenden Arbeitsweisen und führt anschließend bis zu agentischen Funktionen und erweiterten Integrationen.

Ist das ein Prompt-Engineering-Kurs?

Nein. Gute Aufgabenstellungen und Kontext sind wichtig, aber der Schwerpunkt liegt auf dem vollständigen Entwicklungsworkflow mit Copilot.

Werden Agent Mode und Coding Agent behandelt?

Ja. Beide werden behandelt und voneinander abgegrenzt. Dabei geht es insbesondere darum, welche Aufgaben sich eignen und wie die resultierenden Änderungen kontrolliert werden.

Wird Copilot CLI behandelt?

Ja. Copilot CLI ist Bestandteil des Kurses und wird praktisch in den Entwicklungsworkflow integriert.

Wird MCP behandelt?

Ja. MCP wird für die Anbindung zusätzlicher Werkzeuge und Datenquellen behandelt. Dabei werden auch Berechtigungen und Vertrauensgrenzen betrachtet.

Werden Custom Instructions behandelt?

Ja. Repository-weite und pfadspezifische Instructions sowie AGENTS.md werden praktisch eingesetzt – sowohl zur Projektanpassung als auch in agentischen Workflows.

Werden auch Custom Agents behandelt?

Ja. Spezialisierte Agenten mit eigenen Instruktionen und Toolsets werden im Kurs eingeordnet und praktisch eingesetzt.

Geht es auch um Code Review?

Ja. Neben dem Review von Änderungen im Entwicklungsprozess wird Copilot Code Review behandelt, einschließlich des agentischen Review-Workflows in der CLI.

Warum dauert der Kurs drei Tage?

Der Kurs behandelt Copilot nicht nur als Code Completion, sondern als Entwicklungswerkzeug einschließlich Chat, Agent Mode, Coding Agent, CLI, Custom Instructions, Code Review und MCP. Drei Tage ermöglichen, diese Funktionen praktisch an einer Codebasis einzusetzen, ohne daraus lediglich eine Funktionsdemonstration zu machen.

Kann der Kurs an unsere Entwicklungsumgebung angepasst werden?

Ja. Bei Inhouse-Schulungen können Programmiersprachen, Frameworks, Repository-Struktur, Entwicklungsumgebung, Testwerkzeuge und vorhandene GitHub-Prozesse in die Übungen einbezogen werden.

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