Software Engineering - Code Generierung mit LLMs[1]

Eine allererste Einführung

Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de

Version: 0.0.1

Folien: [HTML] https://delors.github.io/se-using-llms/folien.de.rst.html
[PDF] https://delors.github.io/se-using-llms/folien.de.rst.html.pdf

Fehler melden: https://github.com/Delors/delors.github.io/issues

[1] Erstellt unter Zuhilfenahme von ChatGPT

Prompting-Strategien für Code-Generierung mit LLMs

Klare Aufgabenstellung:
Programmiersprache und gewünschte Bibliotheken benennen.

Beispiel

Schreibe eine Funktion in *Python*, die … mit *NumPy* …

Struktur vorgeben: Erwünschte Signatur oder Interfaces spezifizieren

Beispiel

Implementiere `def sort_list(xs: list[int]) -> list[int]:`

Schrittweise Anleitungen:
Erst Algorithmus erklären lassen, dann den Code generieren.
Hilft bei komplexen Problemen (Divide & Conquer im Prompt).

Beispiele nutzen: „Few-Shot Prompting“ mit kleinen Code-Snippets als Vorlage.
Fördert Einhaltung von Stil, Syntax und Konventionen.

Fehlervermeidung: Auf Testfälle im Prompt hinweisen: „Der Code muss diese Tests bestehen …“

Iterative Verfeinerung:
Erst groben Entwurf generieren, dann gezielt Verbesserungen anfordern

Beispiel

„Optimiere die Laufzeit“ oder „Füge Typannotationen hinzu“.

Typische Stolperfallen beim Code-Prompting

Unpräzise Sprache:
- Keine Programmiersprache/Technologien angegeben → Modell wählt zufällig.
- Lösung: Sprache und Version im Prompt festlegen

Fehlende Constraints:
- Ohne Funktionssignatur oder Interfaces generiert LLM „freie“ Lösungen
- Lösung: Signaturen, Klassen oder API-Vorgaben angeben

Vermischte Anforderungen:
- Prompt enthält Code + Prosa → Antwort enthält unnötige Erklärungen
- Lösung: explizit nur *Codeblock* anfordern

Zu komplex in einem Schritt:
- Lange Prompts mit vielen Bedingungen führen zu Fehlern
- Lösung: Problem in mehrere Teilaufgaben aufsplitten

Modernere LLMs können immer besser mit komplexen Prommpts umgehen und hier sind auch weitere Verbesserungen zu erwarten!

Keine Tests spezifiziert:
- Modell weiß nicht, wann Lösung „korrekt“ ist
- Lösung: Testfälle im Prompt beilegen

Übermäßiges Vertrauen:
- ! Generierter Code wird ohne Überprüfung übernommen!
- ! Lösung: Immer Review, Linting und Tests durchführen. Insbesondere Versionen von externen Bibliotheken überprüfen, diese sind häufig nicht aktuell oder veraltet.

Achtung!

Beim Einbiden von Tools oder Bibliotheken ist die Zielbibliothek immer zu prüfen! Am Besten einen Dependency-Manager verwenden.