

# Herzlich Willkommen.

MEHR WISSEN, BESSER ENTSCHEIDEN.

IZRESEARCH 

# KI in der Softwareentwicklung

Ingo Graser, Leiter Softwareentwicklung / CTO  
Immobilien Zeitung

Seite 2

KI in der Softwareentwicklung, *Frankfurt*, 15.05.2024



IZRESEARCH 

# Inhaltsverzeichnis

1. ChatGPT im Vorstellungsgespräch
2. LLMs vs. Programmiersprachen
3. Was ist Softwareentwicklung
4. Praktischer Einsatz
5. Fazit



# ChatGPT im Vorstellungsgespräch

# ChatGPT im Vorstellungsgespräch

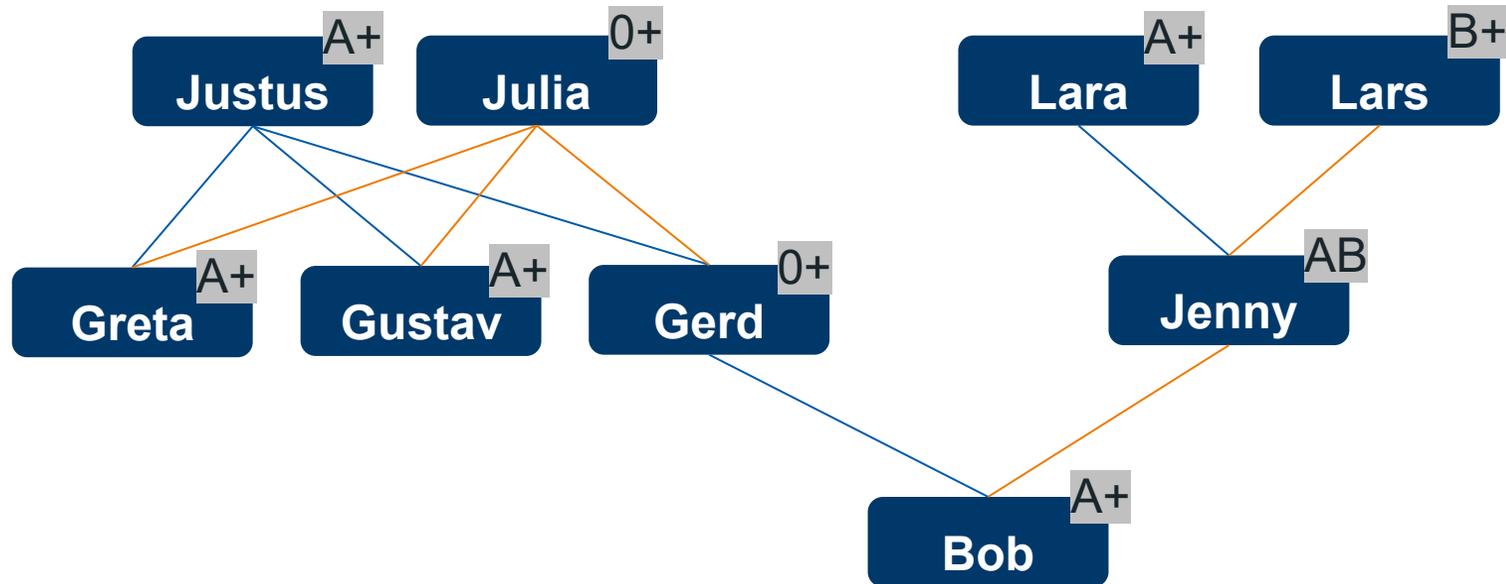
ChatGPT 4 ▾



**You**

Schreibe Java-Code um eine Struktur von Personen mit den Informationen Namen, Blutgruppe und deren Kind-Eltern-Verwandschaft ausgehend von einer Person zu durchsuchen nach dem nächsten Verwandten mit der gleichen Blutgruppe.

# ChatGPT im Vorstellungsgespräch



# Phase 1:



```
public static Person findRelativeWithSameBloodType(Person start, String bloodType) {
    Queue<Person> queue = new ArrayDeque<>();
    queue.add(start);
    while (!queue.isEmpty()) {
        Person current = queue.poll();
        // Prüfen, ob die aktuelle Person die gesuchte Blutgruppe hat, aber nicht die Startperson ist
        if (current != start && current.getBloodType().equals(bloodType)) {
            return current;
        }
        // Füge den Elternteil hinzu, wenn vorhanden
        if (current.getParent() != null) {
            queue.add(current.getParent());
        }
        // Füge alle Kinder hinzu
        for (Person child : current.getChildren()) {
            queue.add(child);
        }
    }
    return null; // Kein Verwandter mit derselben Blutgruppe gefunden
}
```

## Phase 2:



ChatGPT 4 ▾

**You**

Ich glaube der Code kommt so in eine Endlosschleife. Schau nochmal drüber.

## Phase 3:



### ChatGPT 4 ▾



**You**

Gibt es andere Vorgehensweisen für die Umsetzung des Algorithmus und welche Vor- und Nachteile hätten diese jeweils?



# LLMs vs. Programmiersprachen

# Natürliche Sprache

- **mehrdeutig**

„Bank“, „die Daumen drücken“, „Ich sah den Mann mit dem Fernglas“

- **dynamisch**

kreative Sprachverwendung, ständige Anpassung der Sprache

- **unregelmäßig**

- Basis: *historisch gewachsen*

# Programmiersprachen

- **eindeutig**  
Syntax, Kontrollstrukturen
- **statisch**  
keine kreative Verwendung in diesem Sinne
- **gemeinsame Konzepte**  
prozedural, funktional, Objektorientierung, event-driven
- Basis: *logische Konzepte*

# LLMs beim Coden

**Entwickler haben früh das Potenzial von LLMs als effektive Werkzeuge erkannt.**

- überraschend hohe Qualität der Ergebnisse
- demonstrieren ein tiefes semantisches Verständnis
- Interaktion fühlt sich an wie eine echte Zusammenarbeit
- diskutiert sachlich – der ideale Programmierpartner

# LLMs beim Coden

Erste Studien belegen das:

“[...] the computational efficiency of the code generated by GPT-4 is comparable to that of human programmers. These results suggest that GPT-4 has the potential to serve as a reliable assistant in programming code generation and software development.”

**Quelle:**

**A systematic evaluation of large language models for generating programming code**

<https://arxiv.org/html/2403.00894v1>

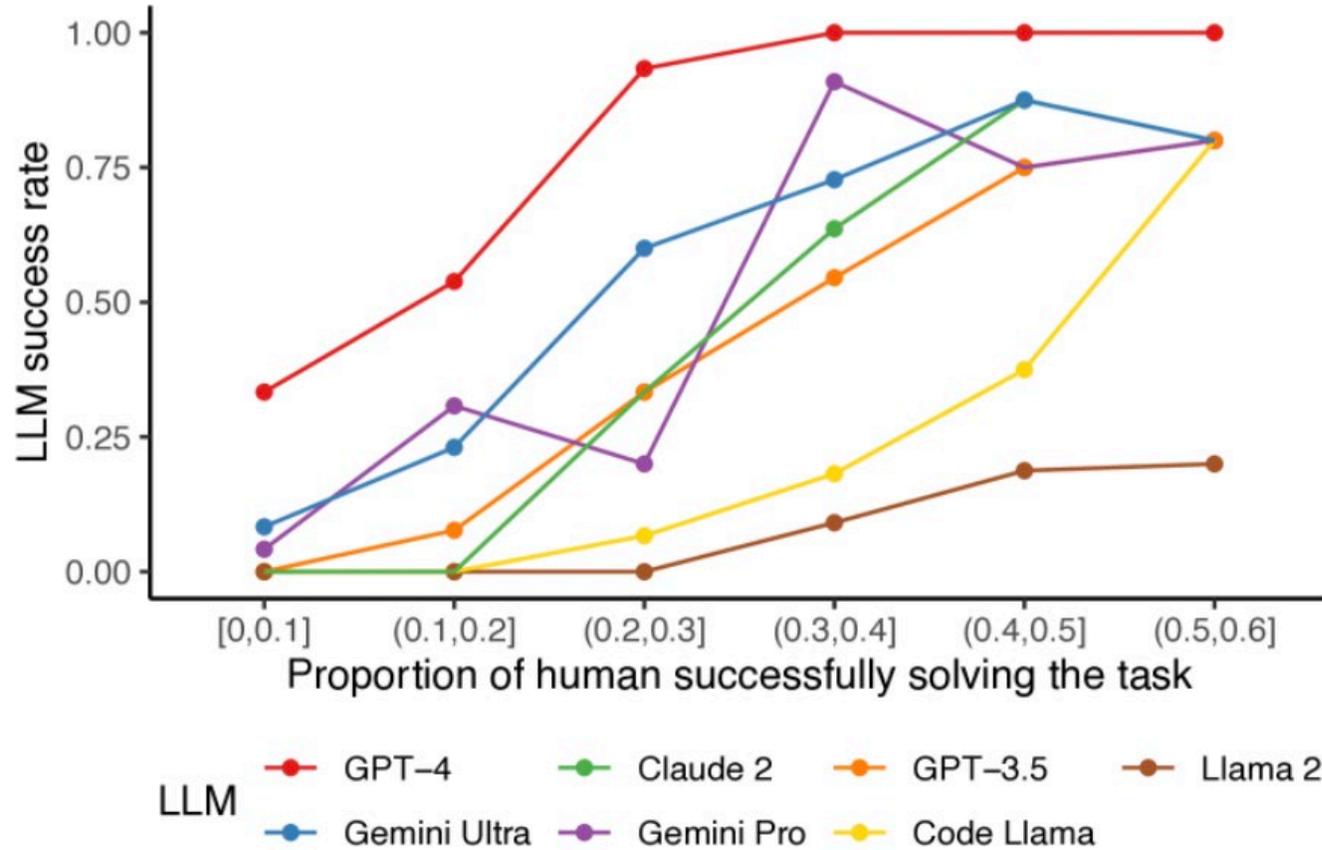
# Ergebnisse der Studie

## Wann funktionieren LLMs als Werkzeug gut?

- in iterativen Kommunikationsprozessen
- mit hochwertigem, möglichst vollständigem Input
- Entwickler müssen den Output verstehen und weiterentwickeln können

„GPT-4 may empower individuals with little or no programming expertise to solve programming tasks of easy or medium difficulty. For those who have advanced programming expertise, GPT-4 may share the workload, allowing human programmers to focus on more challenging tasks.“

# Ergebnisse der Studie





# Was ist Softwareentwicklung

# Aufgaben im Softwareentwicklungsprozess

- fachliche Konzeption
- technische Konzeption
- Code schreiben
- testen
- deployen

Alle Schritte sind kommunikative Prozesse, die aufeinander rückwirken (iterativ)

# KI als Unterstützung beim Code schreiben

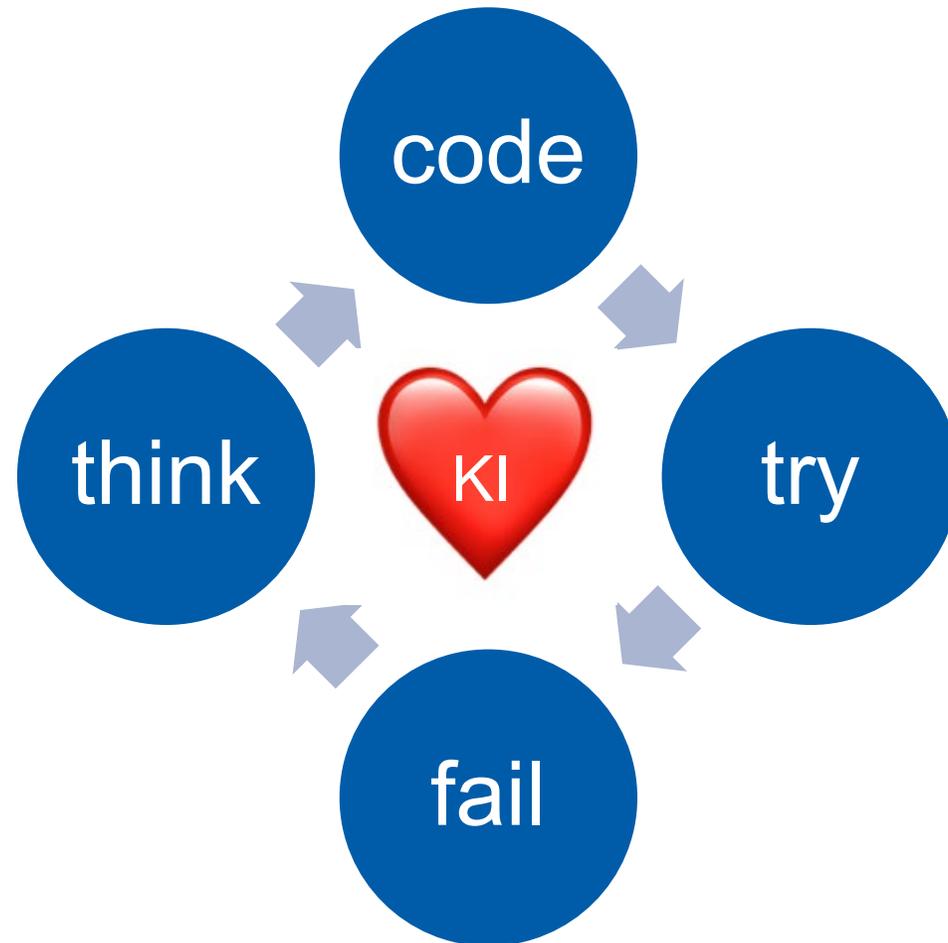
## Was geht

- Übersetzung natürlicher Sprache in Code und umgekehrt
- Fehler im Code finden
- Ideen für Verbesserungen entwickeln
- langweilige Arbeiten automatisieren

## Was nicht geht (zumindest aktuell)

- Überblick behalten

# Code schreiben: Reality Check



# Pair Programming

- einer schreibt, der andere kann freier denken
- einigen auf ein gemeinsames Vorgehen – das Beste aus zwei Welten
- wenn einer nicht weiterweiß, kann der andere den Weg weisen
- Einlassen erforderlich

# Pair Programming mit KI



# Praktischer Einsatz

# GitHub Copilot

# Integration in IDEs

- Zugriff des LLMs auf den Kontext (die Codebase)
  - verbessert den Input für das LLM
  - weniger explizite Kommunikation erforderlich
  - beschleunigt die Zusammenarbeit

CHAT

GitHub Copilot

Hallo @IGraser, wie kann ich Ihnen helfen?

Fragen Sie Copilot, oder geben Sie "/

```

J Willkommen | J Person.java | J FamilyTree.java 2 | J Main.java 1
J FamilyTree.java > FamilyTree
1  import java.util.ArrayDeque;
2  import java.util.Queue;
3  import java.util.List;
4  import java.util.ArrayList;
5
6  public class FamilyTree {
7      public static Person findRelativeWithSameBloodType(Person start, String bloodType) {
8          Queue<Person> queue = new ArrayDeque<>();
9          queue.add(start);
10
11         while (!queue.isEmpty()) {
12             Person current = queue.poll();
13
14             // Prüfen, ob die aktuelle Person die gesuchte Blutgruppe hat, aber nicht die Startperson ist
15             if (current != start && current.getBloodType().equals(bloodType)) {
16                 return current;
17             }
18
19             // Füge den Elternteil hinzu, wenn vorhanden
20             if (current.getParent() != null) {
21                 queue.add(current.getParent());
22             }
23
24             // Füge alle Kinder hinzu
25             for (Person child : current.getChildren()) {
26                 queue.add(child);
27             }
28         }
29
30         return null; // Kein Verwandter mit derselben Blutgruppe gefunden
31     }

```

TERMINAL | PROBLEME 3 | AUSGABE | DEBUGGING-KONSOLE | PORTS

bash

The default interactive shell is now zsh.  
 To update your account to use zsh, please run `chsh -s /bin/zsh`.  
 For more details, please visit <https://support.apple.com/kb/HT208050>.

Ingo-Graser---Mac-19-06-0001:test igraser\$

CHAT

GitHub Copilot

Hallo @IGraser, wie kann ich Ihnen helfen?

Fragen Sie Copilot, oder geben Sie "/"

```
Willkommen | Person.java | FamilyTree.java 2 | Main.java 1
FamilyTree.java > FamilyTree > findRelativeWithSameBloodType(Person, String)
6 public class FamilyTree {
7     public static Person findRelativeWithSameBloodType(Person start, String bloodType) {
8         Queue<Person> queue = new ArrayDeque<>();
9         queue.add(start);
10
11         while (!queue.isEmpty()) {
12             Person current = queue.poll();
13
14             // Prüfen, ob die aktuelle Person die gesuchte Blutgruppe hat, aber nicht die Startperson ist
15             if (current != start && current.getBloodType().equals(bloodType)) {
16                 return current;
17             }
18
19             // Füge den Elternteil hinzu, wenn vorhanden
20             if (current.getParent() != null) {
21                 queue.add(current.getParent());
22             }
23
24             // Füge alle Kinder hinzu
25             for (Person child : current.getChildren()) {
26                 queue.add(child);
27             }
28         }
29
30         return null; // Kein Verwandter mit derselben Blutgruppe gefunden
31     }
```

TERMINAL | PROBLEME 3 | AUSGABE | DEBUGGING-KONSOLE | PORTS

```
bash
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Ingo-Graser---Mac-19-06-0001:test igraser$
```

```

J FamilyTree.java > FamilyTree
1  import java.util.ArrayDeque;
2  import java.util.Queue;
3  import java.util.List;
4  import java.util.ArrayList;
5
6  public class FamilyTree {
7
8      public static Person findRelativeWithSameBloodType(Person start, String bloodType) {
9          Queue<Person> queue = new ArrayDeque<>();
10         queue.add(start);
11
12         while (!queue.isEmpty()) {
13             Person current = queue.poll();
14
15             // Prüfen, ob die aktuelle Person die gesuchte Blutgruppe hat, aber nicht die Startperson ist
16             if (current != start && current.getBloodType().equals(bloodType)) {
17                 return current;
18             }
19
20             // Füge den Elternteil hinzu, wenn vorhanden
21             if (current.getParent() != null) {
22                 queue.add(current.getParent());
23             }
24
25             // Füge alle Kinder hinzu
26             for (Person child : current.getChildren()) {
27                 queue.add(child);
28             }
29         }
30
31         return null; // Kein Verwandter mit derselben Blutgruppe gefunden

```

```

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Ingo-Graser---Mac-19-06-0001:test igraser$

```

CHAT

GitHub Copilot

Hallo @IGraser, wie kann ich Ihnen helfen?

1

7

1

Java: Ready

Zeile 7, Spalte 1 Leerzeichen: 4 UTF-8 LF {} Java

# Rückmeldungen zu Github Copilot

Für Massen-Arbeiten super. Wenn man einmal vorgemacht hat, was man tun möchte, erledigt die Maschine zuverlässig den Rest.

Copilot ist ein Must-Have für Entwickler.

Sehr gut zum Dokumentieren von Code.

Entlastung anderer, die sonst helfen müssten, beschleunigt beide.

Voice-Input wird immer stärker kommen.

# Rückmeldungen zu Github Copilot

Code-Completion ist praktisch und ein ständiger Begleiter.

Macht mich schneller.

War frustrierend weil zu viel Arbeit erforderlich, um das System zu etwas zu zwingen – Vorschläge wirken ablenkend.

Wenn man an Kleinigkeiten hängt, bekommt man oft die entscheidende Hilfe.

Ersetzt Recherche per stackoverflow und google. Liefert direkt Code Snippet mit Erklärung.



# Fazit

# Meine Einschätzung

- echte Unterstützung für Entwickler die sich darauf einlassen (Toleranz)
- Effizienzsteigerung spürbar
  - Recherche, Routine-Aufgaben, Unterstützung bei komplexen Aufgaben
- höhere Selbstwirksamkeit - mehr Zeit für die interessanten Sachen
- Gesamtergebnis auf lange Sicht besser
  - Rückgriff auf mehr Wissen, gute Codequalität, Dokumentation
- KI ersetzt keine Entwickler, sondern kann sie beschleunigen

**Fazit: unbedingt nutzen, niemanden zwingen**

# Vielen Dank für Ihre Aufmerksamkeit!

## **Kontakt:**

Ingo Graser

Leiter Softwareentwicklung / CTO

Tel. 0 611 9 73 26-47

[graser@iz-research.com](mailto:graser@iz-research.com)

[www.iz-research.com](http://www.iz-research.com)

**MEHR WISSEN, BESSER ENTSCHIEDEN.**

**IZRESEARCH** 