Softwaretechnik 2 (ST2)

Informatik Bachelor, SoSe 2026

Prof. Dr.-Ing. Stefan Bente

Motivation: Wofür Softwaretechnik?

Technology
Arts Sciences
TH Köln

Bild: Jeff Knezovich, https://www.flickr.com/photos/knezovjb/2102314562

Teil 1

Größe & Komplexität

https://ahaslides.com/VR0WC

Das Ziel für diese Veranstaltung

Nach ST2 wissen Sie, wie Sie mit einem größeren (oder richtig großen) Softwaresystem anfangen zu entwickeln.

https://ahaslides.com/VR0WC

Wie übersetzt sich „# Klassen“ in „SLoC“

- SLoC = Source Lines of Code
  - Häufig verwendete Metrik zum Messen der Größe von Software
  - Auch gebräuchlich: KLoC (1000 Zeilen Code)

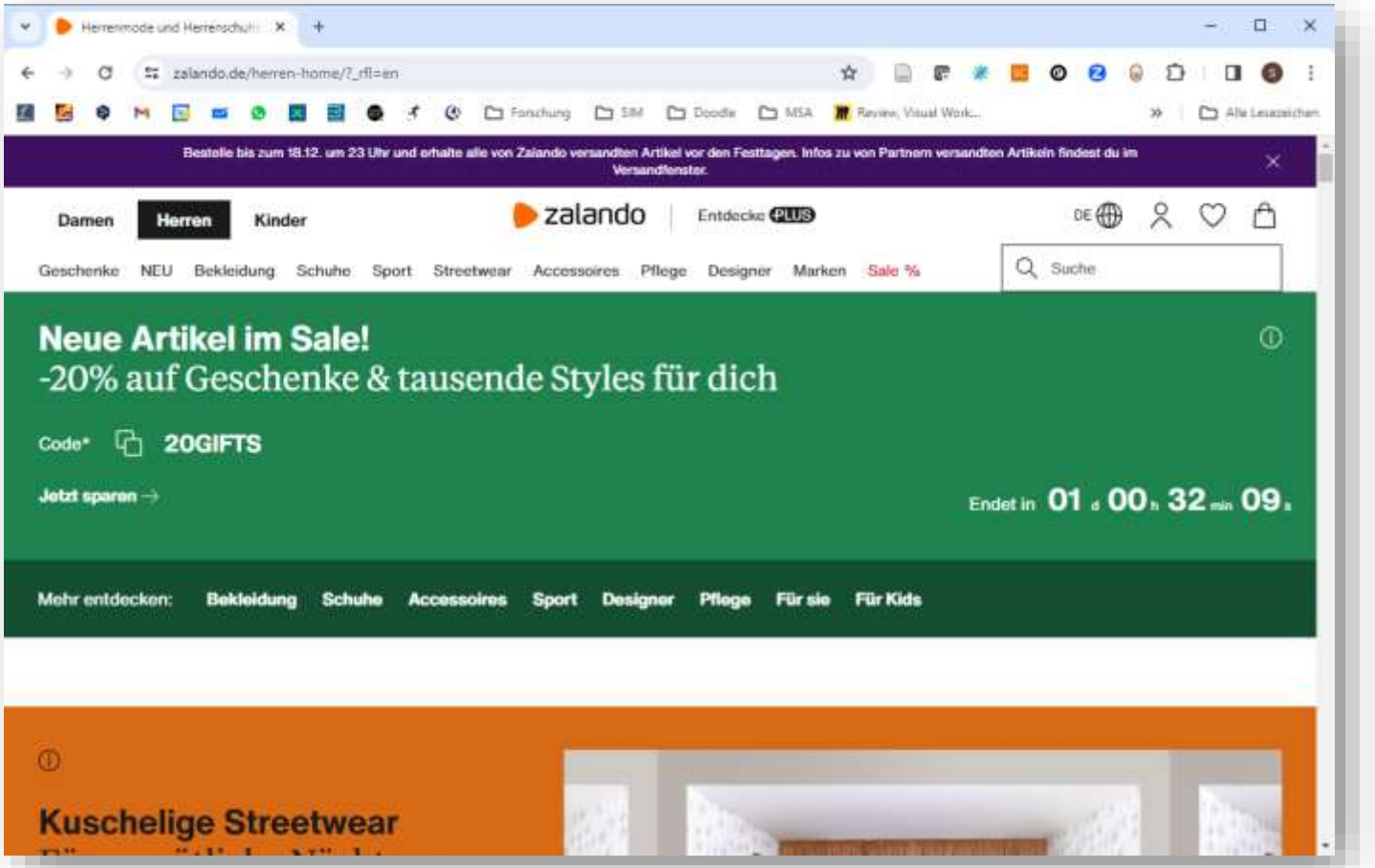
- Zhang & Tan (2007): Mittlere Größe einer (Java-)Klasse =

  114 SLoC

- 5 Klassen ~ 570 SLOC
- 50 Klassen ~ 5700 SLOC
- 500 Klassen ~ 57.000 SLOC

- Source: https://ieeexplore.ieee.org/abstract/document/4425860
  - H. Zhang and H. B. K. Tan, "An Empirical Study of Class Sizes for Large Java Systems," 14th Asia-Pacific Software Engineering Conference (APSEC'07), Nagoya, Japan, 2007, pp. 230-237, doi: 10.1109/ASPEC.2007.64

Beispiel 1: Zalando Logistics System (Zalos)

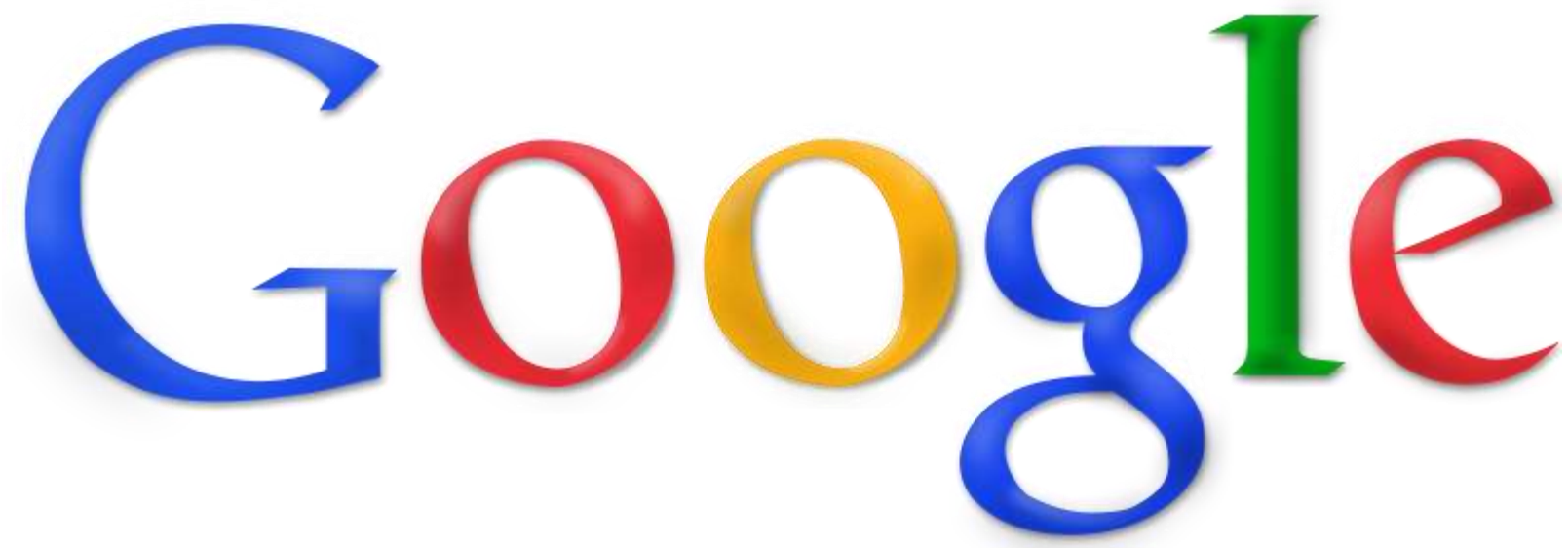


Zalos = Lager- und Logistik-Verwaltung von Zalando

Beispiel 2: Windows 10

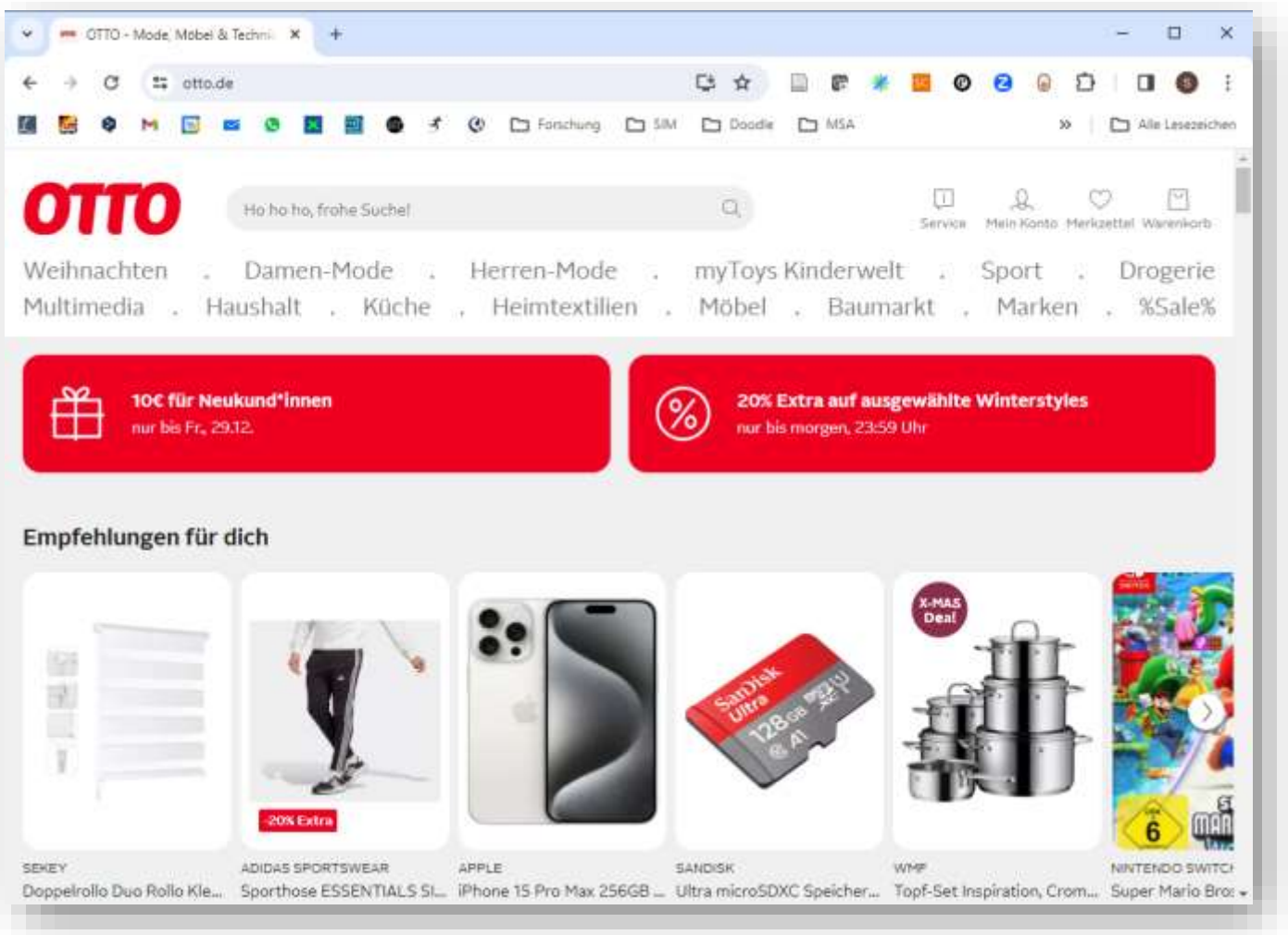
Gesamtheit des Betriebssystem-Codes

Beispiel 3: Google Monorepo



(alle Google Services, Front- & Backend, ohne Google Chrome und Android)

Beispiel 4: Otto Versandplattform – Open Source Kern



Open Source Kernplattform

Beispiel 5: Apollo-Mondlandung 1969

Software auf allen Computern
in Rakete und Landemodulen

https://ahaslides.com/VR0WC

Wie viele Zeilen Code …

- 145.000 - **Apollo 11**
  - https://www.synopsys.com/blogs/software-security/apollo-11-software-development.html
- 710.000 - **Otto.de Plattform**
  - Fürstenau, D., Rothe, H., Baiyere, A., Schulte-Althoff, M., Masak, D., Schewina, K., & Anisimova, D. (2019). Growth, Complexity, and Generativity of Digital Platforms: The Case of Otto.de. *Fortieth International Conference on Information Systems*. https://aisel.aisnet.org/wi2019/track07/papers/10
- 1,3 Mio - **Zalando Logistics System (Zalos)**
  - https://engineering.zalando.com/posts/2018/04/migrating-java-8.html
- 50 Mio. - **Windows 10**
  - 3 Mio. - Kernel: 3 Mio
  - 25 Mio. - Applications (Office, Outlook, Store, …)
  - 22 Mio. – 3rd Party (Treiber, …)
  - https://softkeys.uk/blogs/blog/how-many-lines-of-code-in-windows-10
- 2 Mrd. - **Google Monorepo**
  - https://qeunit.com/blog/how-google-does-monorepo

Produktivität von Entwicklern – LoC / day

- Gesamtzahl aggregiert über längeren Zeitraum
  - Inklusive Tests, Doku, Bugfixing, Team-Events, Smalltalk, Training, sinnlose Meetings, …

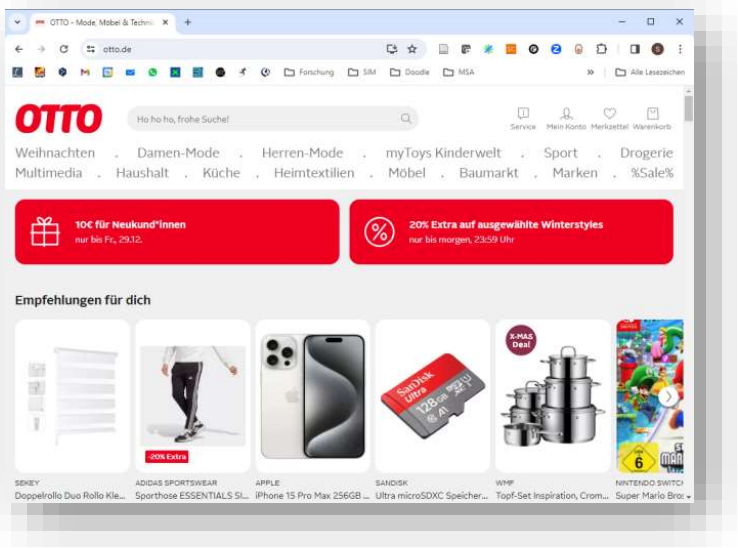
LoC / day	Quelle
10	Fred Brooks, „The Mythical Man Month“
16 … 38	Capers Jones
20 … 125	McConnel, small projects (<= 10.000 LoC)
1,5 … 25	McConnel, large projects (>= 10 Mio LoC)
50	Andy Brice, own data (90.000 – 150.000 LoC systems)

Quelle: Andy Brice (2017), https://successfulsoftware.net/2017/02/10/how-much-code-can-a-coder-code/

- => Wir gehen für die weiteren Überlegungen mal von **50 LoC / day** aus

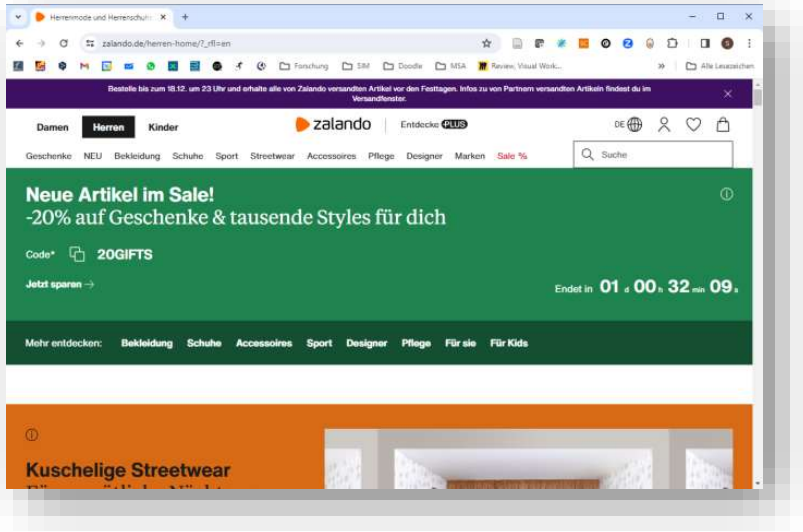
Wie viele Entwickler:innen arbeiten an so etwas?

Otto.de Plattform



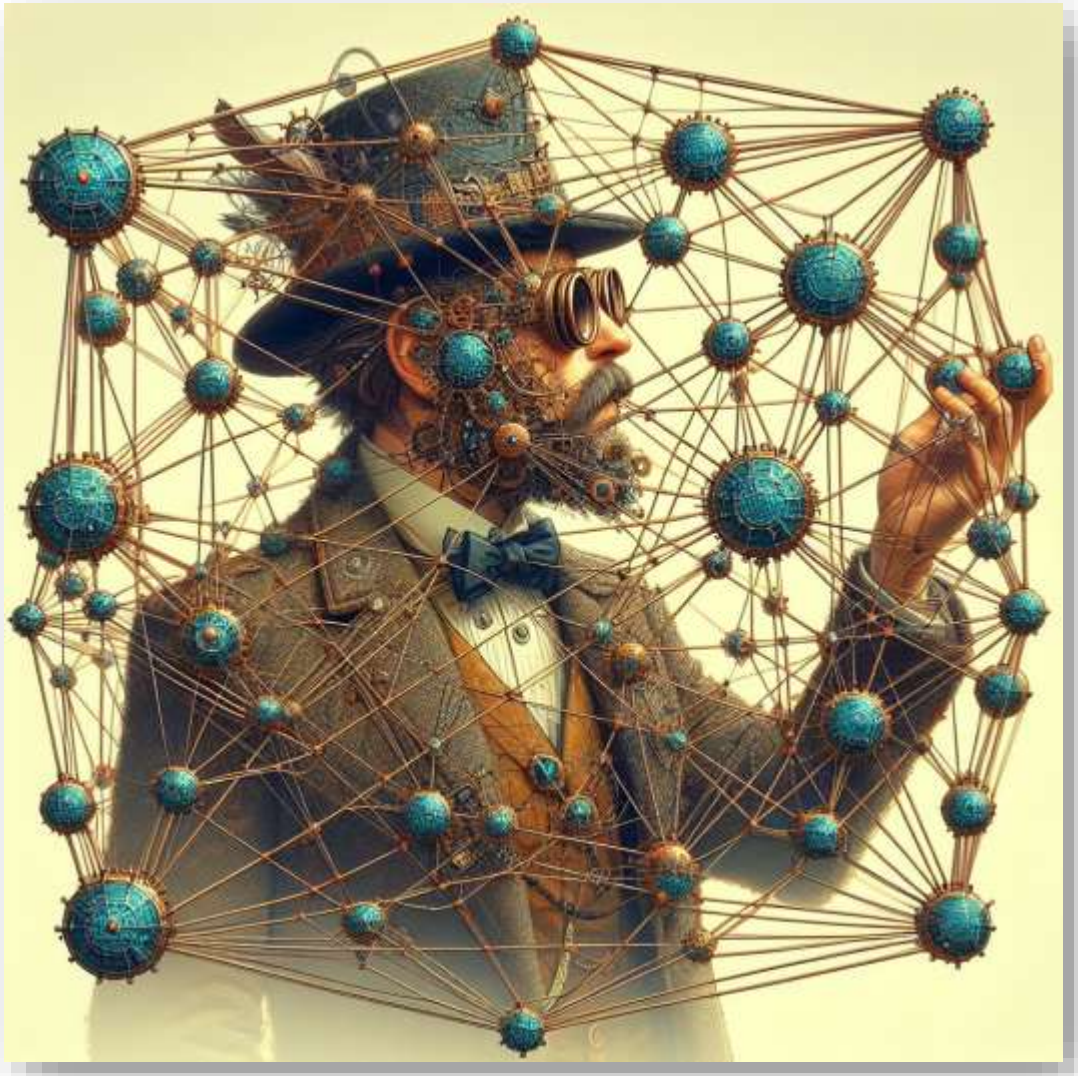
- 506 Contributors in 65 repos
- Growth: ~25.000 LoC / month
  - Feb 18 – Feb 19
- => ca. 30 Developer
- => ca. 4-6 Teams

Zalando Zalos



- 70 Developer
  - (Angabe aus der Quelle)
- => ca. 10 Teams

Wie schafft man es, dass es nicht zu Spaghetti wird?



Teil 2

Die Disruption durch KI

Ein paar Thesen (nicht mehr ganz neu …)

- Der Fortschritt der künstlichen Intelligenz verändert den Softwareentwicklungsprozess grundlegend.
- Software-Engineering-Aufgaben werden zunehmend durch KI unterstützt oder automatisiert.

- Damit verschiebt sich der Schwerpunkt professioneller Softwareentwicklung hin zu konzeptionellen und architekturellen Kompetenzen.

- Die Fähigkeit, komplexe Systemlandschaften zu strukturieren, technische Leitlinien zu definieren und das Zusammenspiel von Software, Infrastruktur und Entwicklungsprozessen zu gestalten, gewinnt deutlich an Bedeutung.

Die große Frage …

Lernen Sie das im Studium?

Oder verführt das Studium dazu, die KI alles machen zu lassen – und am Ende nichts mehr selbst zu können?

Praktika im Informatik-Studium sind in der Zwickmühle

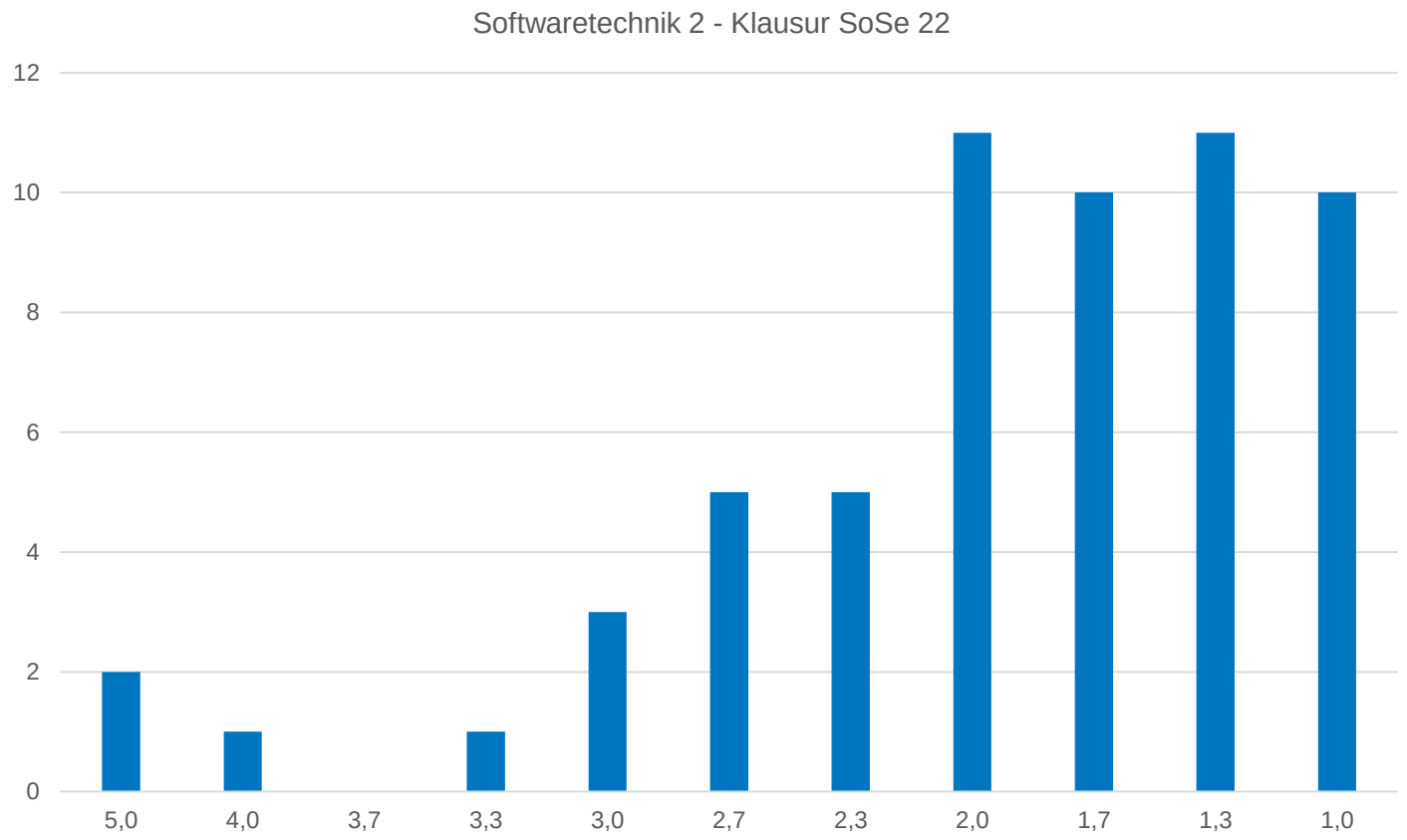
Aufgabenstellungen einfach und klein

- Gut geeignet zum Programmieren lernen
- Aber unrealistisch bezüglich berufspraktischer Aufgaben
- Mit KI mühelos zu umgehen
  - Da genügt dann schon eine nicht-agentische KI wie ChatGPT

Aufgabenstellungen komplex

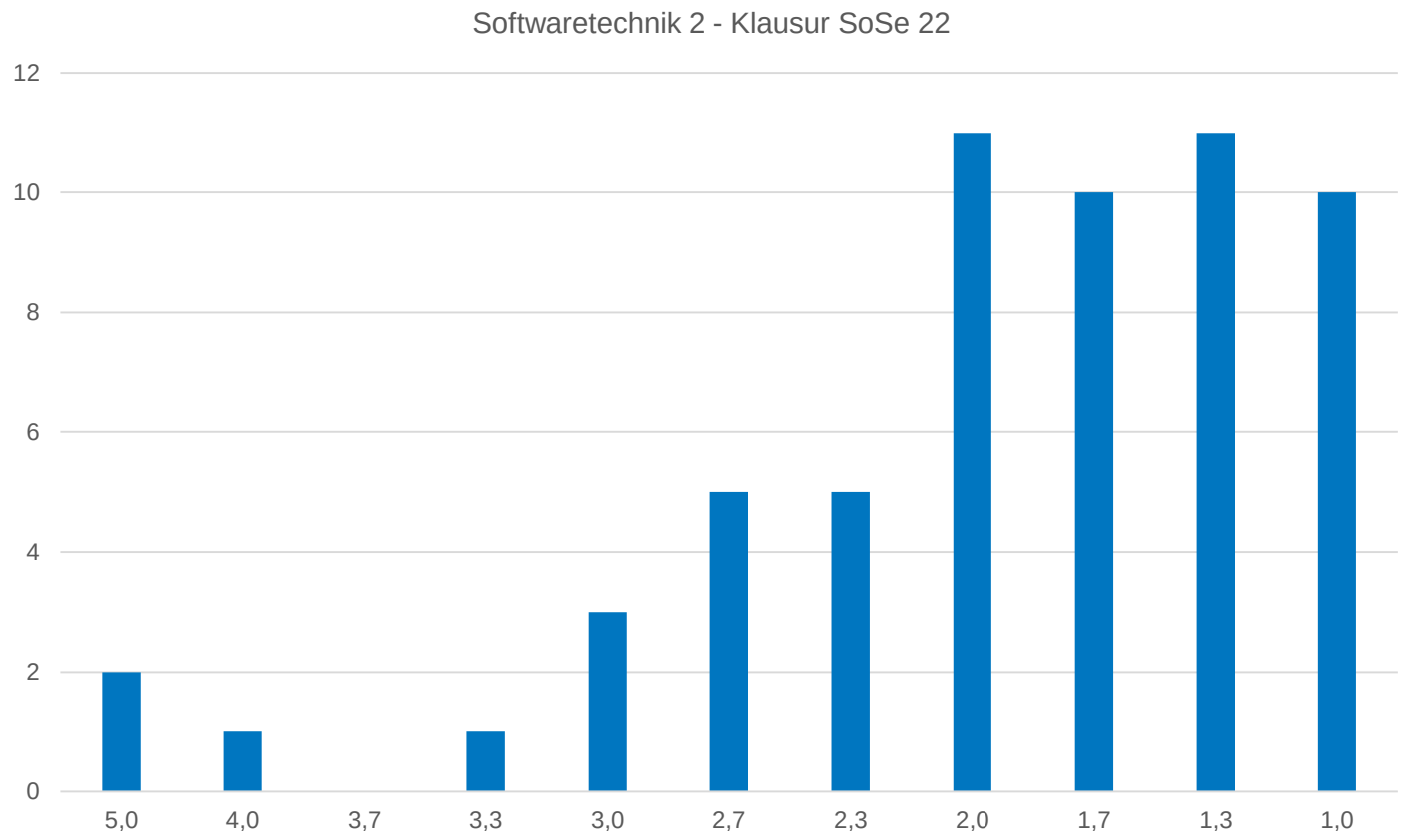
- Kommen berufspraktischer Aufgaben näher
- Lädt dazu ein, mit KI-Unterstützung zu arbeiten
  - „Mensch & KI im Dialog“ wäre hier das Lernziel
- KI „alleine machen lassen“ funktioniert nur noch „so halb“
  - Gerade agentische KI (z.B. Claude Code, Cursor) ist Gamechanger
  - Mit etwas Try & Error ist es aber machbar, ohne Auseinandersetzung mit dem Stoff durchs Praktikum zu kommen

Entwicklung in ST2-Praktikum und Klausur seit 2022



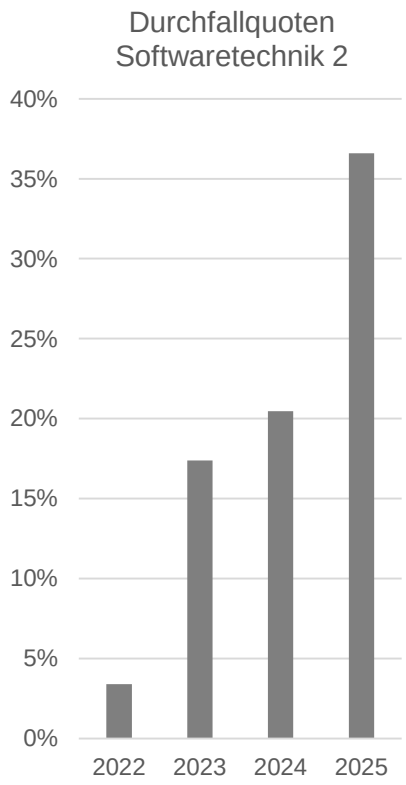
SoSe 2022: Hartes Praktikum – gute Klausur

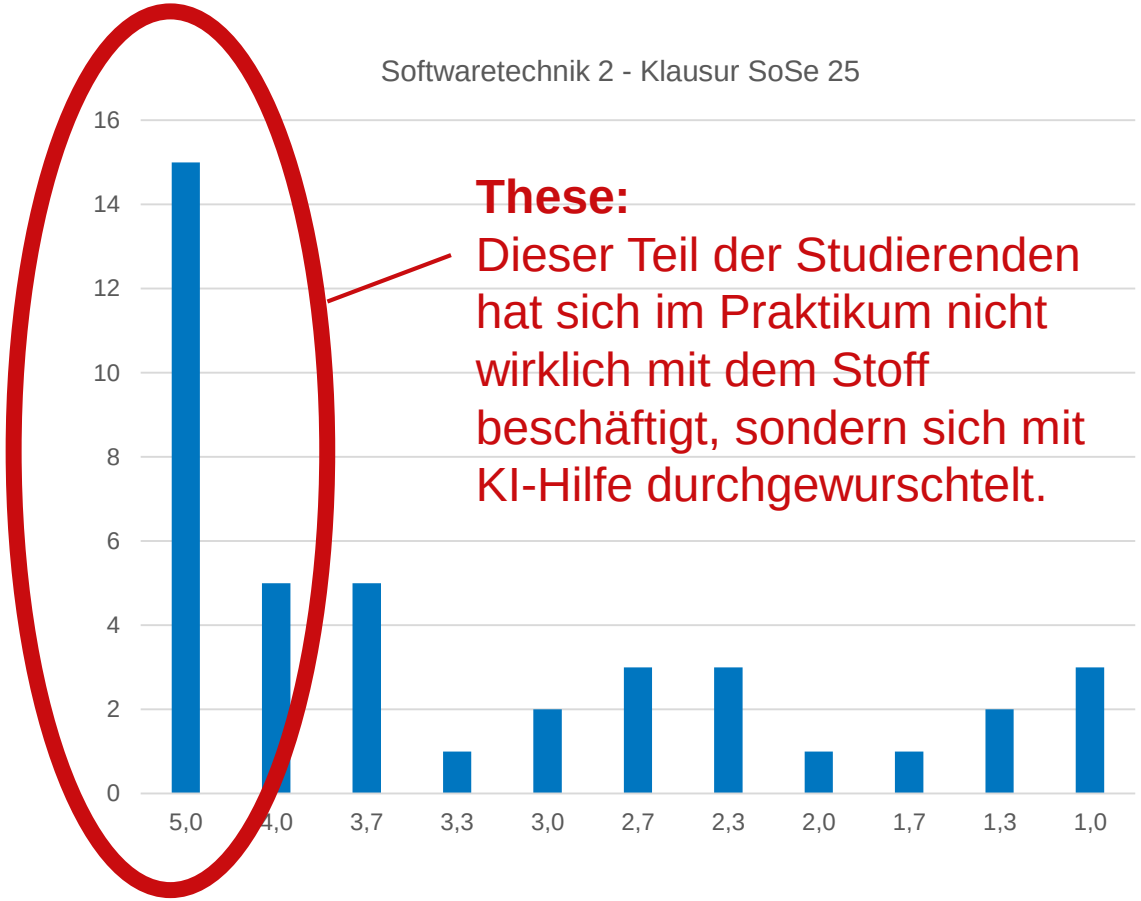
Entwicklung in ST2-Praktikum und Klausur seit 2022 (1)



SoSe 2022: Hartes Praktikum – gute Klausur

Entwicklung in ST2-Praktikum und Klausur seit 2022 (2)





These:
Dieser Teil der Studierenden hat sich im Praktikum nicht wirklich mit dem Stoff beschäftigt, sondern sich mit KI-Hilfe durchgewurschtelt.

Technology
Arts Sciences
TH Köln

Einige Prinzipien für das kommende Semester

Wenn Sie die KI nur nutzen, um sich irgendwie durch das Praktikum bringen zu lassen, dann haben Sie

keine Chance, die Klausur zu bestehen.

- In der Klausur lassen wir KI-Tools nicht zu.
- Wir wollen, dass Sie in der Klausur nachweisen, dass Sie Ihren Code selbst verstehen und schreiben können
  - Aufgaben in der Klausur sind kleiner => Umfang ist gut machbar
  - Aber inhaltlich ist es genau dasselbe wie im Praktikum

Einige Prinzipien für das kommende Semester

Lehrmaterialien sind dafür da, dass man sie anschaut. Deshalb gibt es jetzt in jedem Meilenstein ein **kleines Quiz** zu den wesentlichen Videos und Inhalten.

- … nein, ich mache das nicht gern – Multiple Choice ist gestrig.
  - Aber die Erfahrungen in ST1 haben gezeigt, dass die Videos von vielen nicht geschaut werden, sondern dass direkt die KI angeworfen wird.
- Sie können das Quiz beliebig oft absolvieren (in ILU)
  - Schwer ist es nicht, sondern soll Ihnen nur als Feedback dienen
- Außerdem sind ein paar Fragen drin, die man nur beantworten kann, wenn man das Video auch angeguckt hat
  - … anstatt die KI das Quiz machen zu lassen.

Einige Prinzipien für das kommende Semester

Wir werden mit Ihnen thematisieren, wie Sie KI nutzen

- Sie müssen bei der Bearbeitung des Praktikums angeben, ob Sie KI eingesetzt haben.
  - Wenn ja, dann müssen Sie die Chatprotokolle transparent machen, d. h. mit dem Repository zusammen einchecken.
  - Wir schauen uns das stichprobenartig an und geben Ihnen Feedback.
- Wenn Sie im Gespräch Ihren Code nicht erklären können …
  - … dann behalten wir uns vor, Ihnen den Meilenstein abzuerkennen.
  - Damit können Sie das Praktikum nicht weiter bearbeiten.

Einige Prinzipien für das kommende Semester

Wir machen die Struktur des Praktikums so, dass es etwas schwieriger ist, Claude Code zu sagen:

Mach mal!

- Die Acceptance Tests nutzen den BDD-Teststil
  - Siehe nächste Folie - intuitiv und verständlich
- Nicht alle Tests sind für Sie lokal ausführbar
  - Ein größerer Teil der Acceptance-Tests sind als Hidden-Tests angelegt
  - werden erst in der Pipeline ausgeführt werden.
  - Damit wird es für eine agentische KI wie Claude ein bisschen schwieriger, in einem Rutsch die komplette Aufgabe für Sie zu lösen

BDD = Behavior Driven Development

```
Scenario: Guest creates a reservation and queries it
  Given a clean restaurant system
  And a table with number 1 and 4 seats exists
  When a guest creates a reservation for "Anna Mueller"
        on "15.06.2027 19:00" for 3 people at table 1
  Then the reservation should be active
  And the reservation should appear in the list for "15.06.2027"
```

- Test-Szenarien sind menschenlesbar
  - … besser als lange Testmethoden
- Diese Szenarien geben wir als Acceptance Tests vor
  - Einige sind sichtbar, der größere Teil ist hidden
- Zu BDD gehören Interfaces
  - Dafür implementieren Sie Adapterklassen
  - … die dann in Ihren eigenen Code aufrufen

Wie können Sie KI einsetzen und gleichzeitig sicher gehen, dass Sie immer noch selber für die Klausur lernen?

1. Machen Sie eine Klasse selbst ohne KI-Hilfe
   - … und nutzen dann KI, um gleiche Konstrukte woanders auszurollen.
   - Zum Beispiel Basisstruktur für **ein** Aggregate selber machen, die anderen von der KI erstellen lassen.
2. Nutzen Sie KI, um sich Dinge erklären zu lassen.
3. Architektonische Fragestellungen mit der KI diskutieren
   - Fragen Sie die KI, wie sie zwei Möglichkeiten der Implementierung im Vergleich bewertet.
   - Gerade Claude Code zum Beispiel ist wirklich gut in diesem Dialog
4. Machen Sie bei neu erstellten Code immer ein Code Review
   - Korrigieren Sie das, was Sie nicht haben wollen.
   - Bauen Sie ein Developer Memory auf (z.B. Claude.md bei Claude)