

KI-gestützte Clusterung von studentischen Programmierlösungen zur Verbesserung automatisierter Feedbackprozesse

AI-supported clustering of student programming solutions to improve automated feedback processes

Gregor Germerodt

Bachelor-Abschlussarbeit

Betreuer: Prof. Dr. Michael Striewe

Trier, 07.07.2025

---

## **Vorwort**

Ein Vorwort ist nicht unbedingt nötig. Falls Sie ein Vorwort schreiben, so ist dies der Platz, um z.B. das Unternehmen vorzustellen, in der diese Arbeit entstanden ist, oder um den Personen zu danken, die in irgendeiner Form positiv zur Entstehung dieser Arbeit beigetragen haben.

Auf keinen Fall sollten Sie im Vorwort die Aufgabenstellung näher erläutern oder vertieft auf technische Sachverhalte eingehen.

---

## Kurzfassung

In der Kurzfassung soll in kurzer und prägnanter Weise der wesentliche Inhalt der Arbeit beschrieben werden. Dazu zählen vor allem eine kurze Aufgabenbeschreibung, der Lösungsansatz sowie die wesentlichen Ergebnisse der Arbeit. Ein häufiger Fehler für die Kurzfassung ist, dass lediglich die Aufgabenbeschreibung (d.h. das Problem) in Kurzform vorgelegt wird. Die Kurzfassung soll aber die gesamte Arbeit widerspiegeln. Deshalb sind vor allem die erzielten Ergebnisse darzustellen. Die Kurzfassung soll etwa eine halbe bis ganze DIN-A4-Seite umfassen.

Hinweis: Schreiben Sie die Kurzfassung am Ende der Arbeit, denn eventuell ist Ihnen beim Schreiben erst vollends klar geworden, was das Wesentliche der Arbeit ist bzw. welche Schwerpunkte Sie bei der Arbeit gesetzt haben. Andernfalls laufen Sie Gefahr, dass die Kurzfassung nicht zum Rest der Arbeit passt.

---

## Abstract

The same in English.

---

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung und Problemstellung</b>	<b>1</b>
<b>2</b>	<b>Theoretische Grundlagen</b>	<b>3</b>
2.1	Künstliche Intelligenz	3
2.2	Stand der Technik	4
<b>3</b>	<b>Weitere Kapitel</b>	<b>5</b>
3.1	Bausteine	5
3.2	Abschnitt	5
3.2.1	Unterabschnitt	6
3.3	Abbildungen und Tabellen	6
3.4	Listings	6
3.5	Mathematische Formel	7
3.6	Sätze, Lemmata, Definitionen, Beweise, Beispiele	8
3.7	Fußnoten	9
3.8	Literaturverweise	9
<b>4</b>	<b>Beispiel-Kapitel</b>	<b>10</b>
4.1	Warum existieren unterschiedliche Konsistenzmodelle?	10
4.2	Klassifizierung eines Konsistenzmodells	11
4.3	Linearisierbarkeit (atomic consistency)	11
<b>5</b>	<b>Zusammenfassung und Ausblick</b>	<b>13</b>
	<b>Literaturverzeichnis</b>	<b>14</b>
	<b>Index</b>	<b>15</b>
	<b>Glossar</b>	<b>16</b>
	<b>Eigenständigkeitserklärung</b>	<b>17</b>

---

# Abbildungsverzeichnis

3.1	Bezeichnung der Abbildung .....	6
-----	---------------------------------	---

---

## Tabellenverzeichnis

3.1	Bezeichnung der Tabelle . . . . .	7
4.1	Linearisierbarkeit ist erfüllt . . . . .	12
4.2	Linearisierbarkeit ist verletzt, sequentielle Konsistenz ist erfüllt. . . . .	12
4.3	Linearisierbarkeit und sequentielle Konsistenz sind verletzt. . . . .	12

---

## Listings

3.1	Quicksort-Implementierung in Python . . . . .	7
3.2	Quicksort-Implementierung in JavaScript . . . . .	7



## Einleitung und Problemstellung

Für Lehrkräfte an Hochschulen oder Universitäten kann das Kontrollieren und Bewerten von studentischen Einreichungen je nach Anzahl zu einer großen Herausforderung werden. Gerade bei einer hohen Anzahl an Abgaben steigt der Korrekturaufwand erheblich, was den zeitlichen Rahmen für individuelles Feedback einschränken könnte. Eine Untersuchung zeigte, dass das Kontrollieren und Bewerten von Arbeiten der Hauptfaktor für Arbeitsbelastung und Beeinträchtigung des Wohlbefindens ist (vgl. [JS21]). In der Informatik könnte es sich hier auf Programmieraufgaben beziehen. Dabei müssen Lehrkräfte konsistente Bewertungen abliefern, während jede Abgabe unterschiedliche Syntax und Semantik beinhalten könnte.

Eine Lösung dieses Problems bieten etablierte Systeme zur automatischen Auswertung von Programmieraufgaben. Systematische Übersichtsarbeiten zeigen, dass viele Werkzeuge vorwiegend auf Unit-Tests oder statische Analysen setzen, was meist zu eher generischem Feedback führt (vgl. [MBKS]). Die Hochschule Trier benutzt beispielsweise ASB - Automatische Software-Bewertung<sup>1</sup>. Nachdem Studierende die von der Lehrkraft gestellte Aufgabe bearbeitet haben, können sie online ihre Lösungen hochladen. Das Programm prüft danach nach statischen Kriterien, ob z. B. alle benötigten Dateien hochgeladen wurden, ob sie der Namenskonvention entsprechen, etc. Daraufhin wird das hochgeladene Programm mit Testdaten ausgeführt und geprüft, ob die zu erwarteten Ergebnisse ausgegeben werden. Sollte das nicht der Fall sein, wird eine Fehlermeldung ausgegeben, dass das Programm oder bestimmte Module nicht erwartungsgemäß funktionieren.

Das erzeugte Feedback solcher statischen Systeme dient zur Orientierung, jedoch weniger zur Fehlersuche, da es recht allgemein gehalten ist. Um die Feedbackgenerierung zu verbessern könnten KI-gestützte Verfahren eingesetzt werden. Damit befassten sich beispielsweise die Autoren der wissenschaftlichen Arbeiten ... (Hier Quellen einfügen und erläutern).

Dazu wurde in dieser Arbeit versucht, einen Schritt vor der Feedbackgenerierung zu entwickeln. Er befasst sich mit der KI-gestützten Clusterung studentischer Programmierlösungen. Dieser Ansatz ermöglicht es für mehrere Einreichungen ein gemeinsames Feedback zu generieren, indem sie nach Ähnlichkeit in Bezug auf Syn-

---

<sup>1</sup> <https://www.hochschule-trier.de/informatik/forschung/projekte/asb>

---

tax und Semantik geclustert bzw. gruppiert werden. Weiterführende Progamme oder auch Lehrkräfte könnten dann einen Kandidat pro Cluster wählen, Feedback erzeugen und dieses an alle anderen Teilnehmer des Clusters weiterleiten. Dies könnte eine erhebliche Zeitersparnis zur Folge haben und weiterhin mehr Spielraum für präziseres individuelles Feedback ermöglichen.

## Theoretische Grundlagen

### 2.1 Künstliche Intelligenz

Künstliche Intelligenz (KI) bezeichnet die Wissenschaft und Technik der Entwicklung von Maschinen, die Aufgaben ausführen können, für die normalerweise menschliche Intelligenz erforderlich ist (Russell & Norvig, 2021, S. 1). Der Mensch hat sich damit Systeme geschaffen, um die Kapazitäten menschlicher Intelligenz für bestimmte Aufgaben zu schonen oder zu erweitern. Dabei übernimmt die KI den Teil der Automatisierung monotoner Arbeit, also jenen Part, der durch menschliches Handeln erwiesenermaßen fehleranfällig ist, um konsistente Ergebnisse bereitzustellen. Automatisierung bedeutet in diesem Zusammenhang, dass Maschinen bestimmte Entscheidungs- und Handlungsprozesse eigenständig durchführen, ohne dass eine Person direkt in jeden einzelnen Schritt eingreifen muss. Während sich solche Systeme ursprünglich vor allem in der industriellen Fertigung etablierten, stellt sich zunehmend die Frage, wie sich vergleichbare Konzepte auf andere Bereiche übertragen lassen, wie etwa auf das Bildungswesen und hier speziell auf die Bewertung und Rückmeldung (Feedback) zu studentischen Programmierlösungen. Wie kann eine intelligente Automatisierung Lehrkräfte dabei unterstützen, qualitativ hochwertiges und individualisiertes Feedback zu generieren, ohne jede Lösung einzeln manuell prüfen zu müssen?

In dieser Arbeit wird KI anhand verschiedener Open-Source-Bibliotheken genutzt. Das System kombiniert mehrere KI-Techniken wie

- Deep Learning - ein Teilbereich des Machine Learnings (ML), der künstliche neuronale Netze mit vielen Schichten verwendet, um komplexe Muster in Daten zu erkennen,
- Unsupervised Machine Learning - ein Verfahren, bei denen Modelle ohne beschriftete Trainingsdaten Muster oder Strukturen in den Daten erkennen, z.B. durch Clustering, und
- verschiedene Machine Learning Methoden, bei denen Computer aus Beispieldaten eigenständig Muster und Zusammenhänge erkennen, um daraus Vorhersagen oder Entscheidungen abzuleiten.

Das Zusammenspiel dieser Methoden führt letztlich zur Unterstützung der Lehrkräfte zur Feedbackgenerierung, was in den Ergebnissen zu sehen sein wird.

## 2.2 Stand der Technik

Inspiration für diese Arbeit wurde aus aktuellen Werken entnommen, wie Orvalho et al. (2022), die mit InvAASTCluster ein Verfahren zur Clusterung von Programmierlösungen mittels dynamischer Invarianten-Analyse vorstellen (vgl. [OJM]); aus Paiva et al. (2024) die AsanasCluster, ein inkrementelles k-means-basiertes Verfahren, zur Clusterung von Programmierlösungen für automatisiertes Feedback entwickelt haben (vgl. [PLF24]); und Tang et al. (2024) die Large Language Models<sup>1</sup> (LLMs) und Clustering kombinieren, um personalisiertes Feedback in Programmierkursen zu skalieren (vgl. [TWH<sup>+</sup>]).

Diese Quellen und weitere Recherche dienten zum Kennenlernen und zum späteren Einbinden von Algorithmen, die aufgrund ihrer besonderen Eignung zur Clusterung von Programmieraufgaben besonders herausstachen, wie

- k-means - eine Methode zur Aufteilung von N Beobachtungen in k Cluster, wobei jede Beobachtung zu dem Cluster mit dem nächstgelegenen Mittelwert gehört, der als Prototyp des Clusters dient (vgl. [Mac67]), und
- hdbscan - eine Erweiterung des Density-Based Spatial Clustering of Applications with Noise (DBSCAN) Algorithmus, indem es eine Hierarchie von Clustern aufbaut und die stabilen Cluster über unterschiedliche Dichteebenen hinweg extrahiert (vgl. [CMS13]),

Weiterhin wurden Verfahren eingebunden die hochdimensionale Vektoren in ihren Dimensionen reduzieren, um sie für weiterführende Prozesse wie z. B. zur Clusterung und Visualisierung der Daten nutzbar zu machen. In dieser Arbeit wurden

- Principal component analysis (PCA) - ein Verfahren zur Projektion hochdimensionaler Daten in einen lineareren Unterraum mit geringerer Dimension, indem neue Achsen, entlang derer die Daten am stärksten streuen, berechnet werden und stellt die Daten entlang dieser Achsen dar (vgl. [Kar01]),
- t-distributed stochastic neighbor embedding (t-SNE) - ein Verfahren zur Visualisierung hochdimensionaler Daten, indem es berechnet wie ähnlich Punkte zu ihren Originaldaten sind, um sie entsprechend weit auseinander oder nahe zusammen zu platzieren (vgl. [Lau08]), und
- Uniform Manifold Approximation and Projection (UMAP) - ein Verfahren welches Topologie und Geometrie nutzt, um skalierbare, strukturtreue Einbettungen in niedrigere Dimensionen zu erreichen (vgl. [MHM]).

---

<sup>1</sup> auf Textdaten trainierte KI-Modelle, die natürliche Sprache verarbeiten und generieren

## Weitere Kapitel

Die Gliederung hängt natürlich vom Thema und von der Lösungsstrategie ab. Als nützliche Anhaltspunkte können die Entwicklungsstufen oder -schritte z.B. der Software-Entwicklung betrachtet werden. Nützliche Gesichtspunkte erhält und erkennt man, wenn man sich

- in die Rolle des Lesers oder
- in die Rolle des Entwicklers, der die Arbeit z.B. fortsetzen, ergänzen oder pflegen soll,

versetzt. In der Regel wird vorausgesetzt, dass die Leser einen fachlichen Hintergrund haben - z.B. Informatik studiert haben. Nur in besonderen Fällen schreibt man in populärer Sprache, so dass auch Nicht-Fachleute die Ausarbeitung prinzipiell lesen und verstehen können.

Die äußere Gestaltung der Ausarbeitung hinsichtlich Abschnittformate, Abbildungen, mathematische Formeln usw. wird im Folgenden kurz dargestellt.

### 3.1 Bausteine

Der Text wird in bis zu drei Ebenen gegliedert:

1. Kapitel (`\chapter{Kapitel}`)
2. Abschnitte (`\section{Abschnitt}`)
3. Unterabschnitte (`\subsection{Unterabschnitt}`)

### 3.2 Abschnitt

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua [?]. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

### 3.2.1 Unterabschnitt

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

## 3.3 Abbildungen und Tabellen

Abbildung und Tabellen werden zentriert eingefügt. Grundsätzlich sollen sie erst dann erscheinen, nachdem sie im Text angesprochen wurden (siehe Abbildung 3.1). Abbildungen und Tabellen (siehe Tabelle 3.1) können im Fließtext (**h=here**), am Seitenanfang (**t=top**), am Seitenende (**b=bottom**) oder auch gesammelt auf einer nachfolgenden Seite (**p=page**) oder auch ganz am Ende der Ausarbeitung erscheinen. Letzteres sollte man nur dann wählen, wenn die Bilder günstig zusammen zu betrachten sind und die Ausarbeitung nicht zu lang (< 20 Seiten) ist.

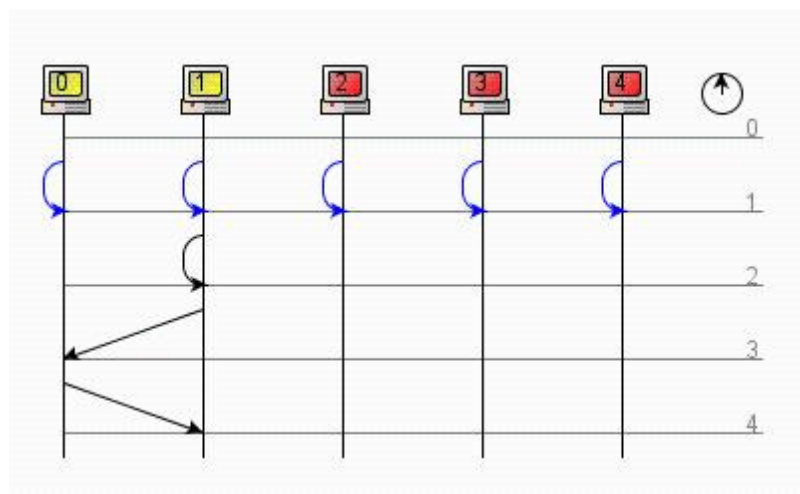


Abbildung 3.1: Bezeichnung der Abbildung

## 3.4 Listings

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua (siehe Listing 21, Zeile 8).

Prozesse	Zeit $\rightarrow$
$P_1$	$W(x)1$
$P_2$	$W(x)2$
$P_3$	$R(x)2 \quad R(x)1$
$P_4$	$R(x)2 \quad R(x)1$

Tabelle 3.1: Bezeichnung der Tabelle

```

1  def quicksort(arr):
2  less = []
3  pivotList = []
4  more = []
5  if len(arr) <= 1:
6      return arr
7  else:
8      pivot = arr[0] # the pivot element
9      for i in arr:
10         if i < pivot:
11             less.append(i)
12         elif i > pivot:
13             more.append(i)
14         else:
15             pivotList.append(i)
16     less = quicksort(less)
17     more = quicksort(more)
18     return less + pivotList + more
19
20 print(quicksort([4, 65, 2, -31, 0, 99, 83, 782, 1]))

```

Listing 3.1: Quicksort-Implementierung in Python

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirm-  
od tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua  
(siehe Listing 10, Zeilen 3 und 5).

```

1  function quicksort([pivot, ...others]) {
2      return pivot === undefined ? [] : [
3          ...quicksort(others.filter(n => n < pivot)),
4          pivot,
5          ...quicksort(others.filter(n => n >= pivot))
6      ];
7  }
8
9  console.log(quicksort([11.8, 14.1, 21.3, 8.5, 16.7, 5.7]));

```

Listing 3.2: Quicksort-Implementierung in JavaScript

Größere Code-Fragmente sollten im Anhang eingefügt werden. [?]

### 3.5 Mathematische Formel

Mathematische Formeln bzw. Formulierungen können sowohl im Fließtext (z.B.  $y = x^2$ ) oder abgesetzt und zentriert im Text erscheinen. Gleichungen sollten für Referenzierungen nummeriert werden (siehe Formel 3.1).

$$e_i = \sum_{i=1}^n w_i x_i \quad (3.1)$$

Entscheidungsformel:

$$\psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ -1 & \frac{1}{2} \leq t < 1 \\ 0 & \text{sonst} \end{cases} \quad (3.2)$$

Matrix:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \quad (3.3)$$

Vektor:

$$\bar{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \quad (3.4)$$

### 3.6 Sätze, Lemmata, Definitionen, Beweise, Beispiele

Sätze, Lemmata, Definitionen, Beweise und Beispiele können in speziell dafür vorgesehenen Umgebungen erstellt werden.

**Definition 3.1.** (*Optimierungsproblem*) Ein Optimierungsproblem  $\mathcal{P}$  ist festgelegt durch ein Tupel  $(I_{\mathcal{P}}, \text{sol}_{\mathcal{P}}, m_{\mathcal{P}}, \text{goal})$  wobei gilt

1.  $I_{\mathcal{P}}$  ist die Menge der Instanzen,
2.  $\text{sol}_{\mathcal{P}} : I_{\mathcal{P}} \mapsto \mathbb{P}(S_{\mathcal{P}})$  ist eine Funktion, die jeder Instanz  $x \in I_{\mathcal{P}}$  eine Menge zulässiger Lösungen zuweist,
3.  $m_{\mathcal{P}} : I_{\mathcal{P}} \times S_{\mathcal{P}} \mapsto \mathbb{N}$  ist eine Funktion, die jedem Paar  $(x, y(x))$  mit  $x \in I_{\mathcal{P}}$  und  $y(x) \in \text{sol}_{\mathcal{P}}(x)$  eine Zahl  $m_{\mathcal{P}}(x, y(x)) \in \mathbb{N}$  zuordnet (= Maß für die Lösung  $y(x)$  der Instanz  $x$ ), und
4.  $\text{goal} \in \{\min, \max\}$ .

*Beispiel 3.2.* MINIMUM TRAVELING SALESMAN (MIN-TSP)

- $I_{\text{MIN-TSP}} =_{\text{def}} \text{s.o.}$ , ebenso  $S_{\text{MIN-TSP}}$
- $\text{sol}_{\text{MIN-TSP}}(m, D) =_{\text{def}} S_{\text{MIN-TSP}} \cap \mathbb{N}^m$
- $m_{\text{MIN-TSP}}((m, D), (c_1, \dots, c_m)) =_{\text{def}} \sum_{i=1}^{m-1} D(c_i, c_{i+1}) + D(c_m, c_1)$
- $\text{goal}_{\text{MIN-TSP}} =_{\text{def}} \min$

□



**Satz 3.3.** *Sei  $\mathcal{P}$  ein  $\mathbf{NP}$ -hartes Optimierungsproblem. Wenn  $\mathcal{P} \in \mathbf{PO}$ , dann ist  $\mathbf{P} = \mathbf{NP}$ .*

*Beweis.* Um zu zeigen, dass  $\mathbf{P} = \mathbf{NP}$  gilt, genügt es wegen Satz A.30 zu zeigen, dass ein einziges  $\mathbf{NP}$ -vollständiges Problem in  $\mathbf{P}$  liegt. Sei also  $\mathcal{P}'$  ein beliebiges  $\mathbf{NP}$ -vollständiges Problem.

Weil  $\mathcal{P}$  nach Voraussetzung  $\mathbf{NP}$ -hart ist, gilt insbesondere  $\mathcal{P}' \leq_T \mathcal{P}$ . Sei  $R$  der zugehörige Polynomialzeit-Algorithmus dieser Turing-Reduktion. Weiter ist  $\mathcal{P} \in \mathbf{PO}$  vorausgesetzt, etwa vermöge eines Polynomialzeit-Algorithmus  $A$ . Aus den beiden Polynomialzeit-Algorithmen  $R$  und  $A$  erhält man nun leicht einen effizienten Algorithmus für  $\mathcal{P}'$ : Ersetzt man in  $R$  das Orakel durch  $A$ , ergibt dies insgesamt eine polynomielle Laufzeit.

**Lemma 3.4.** *Aus  $\mathbf{PO} = \mathbf{NPO}$  folgt  $\mathbf{P} = \mathbf{NP}$ .*

*Beweis.* Es genügt zu zeigen, dass unter der angegebenen Voraussetzung  $\mathbf{KNAPSACK} \in \mathbf{P}$  ist.

Nach Voraussetzung ist  $\mathbf{MAXIMUM KNAPSACK} \in \mathbf{PO}$ , d.h. die Berechnung von  $m^*(x)$  für jede Instanz  $x$  ist in Polynomialzeit möglich. Um  $\mathbf{KNAPSACK}$  bei Eingabe  $(x, k)$  zu entscheiden, müssen wir nur noch  $m^*(x) \geq k$  prüfen. Ist das der Fall, geben wir 1, sonst 0 aus. Dies bleibt insgesamt ein Polynomialzeit-Algorithmus.

□

## 3.7 Fußnoten

In einer Fußnote können ergänzende Informationen<sup>1</sup> angegeben werden. Außerdem kann eine Fußnote auch Links enthalten. Wird in der Arbeit eine Software (zum Beispiel Java<sup>2</sup>) eingesetzt, so kann die Quelle, die diese Software zur Verfügung stellt in der Fußnote angegeben werden.

## 3.8 Literaturverweise

Jede verwendete Literatur wird im Literaturverzeichnis angegeben<sup>3</sup>. Jeder im Verzeichnis vorkommende Eintrag muss mindestens einmal im Text referenziert werden [?].

<sup>1</sup> Informationen die für die Arbeit zweitrangig sind, jedoch für den Leser interessant sein könnten.

<sup>2</sup> <https://www.oracle.com/java/technologies/>

<sup>3</sup> Dazu wird eine sogenannte BibTeX-Datei (literatur.bib) verwendet.

## Beispiel-Kapitel

In diesem Kapitel wird beschrieben, warum es unterschiedliche Konsistenzmodelle gibt. Außerdem werden die Unterschiede zwischen strengen Konsistenzmodellen (Linearisierbarkeit, sequentielle Konsistenz) und schwachen Konsistenzmodellen (schwache Konsistenz, Freigabekonsistenz) erläutert. Es wird geklärt, was Strenge und Kosten (billig, teuer) in Zusammenhang mit Konsistenzmodellen bedeuten.

### 4.1 Warum existieren unterschiedliche Konsistenzmodelle?

Laut [?] sind mit der Replikation von Daten immer zwei gegensätzliche Ziele verbunden: die Erhöhung der Verfügbarkeit und die Sicherung der Konsistenz der Daten. Die Form der Konsistenzsicherung bestimmt dabei, inwiefern das eine Kriterium erfüllt und das andere dementsprechend nicht erfüllt ist (Trade-off zwischen Verfügbarkeit und der Konsistenz der Daten). Stark konsistente Daten sind stabil, das heißt, falls mehrere Kopien der Daten existieren, dürfen keine Abweichungen auftreten. Die Verfügbarkeit der Daten ist hier jedoch stark eingeschränkt. Je schwächer die Konsistenz wird, desto mehr Abweichungen können zwischen verschiedenen Kopien einer Datei auftreten, wobei die Konsistenz nur an bestimmten Synchronisationspunkten gewährleistet wird. Dafür steigt aber die Verfügbarkeit der Daten, weil sie sich leichter replizieren lassen.

Nach [?] kann die Performanzsteigerung der schwächeren Konsistenzmodelle wegen der Optimierung (Pufferung, Code-Scheduling, Pipelines) 10-40 Prozent betragen. Wenn man bedenkt, dass mit der Nutzung der vorhandenen Synchronisierungsmechanismen schwächere Konsistenzmodelle den Anforderungen der strengen Konsistenz genügen, stellt sich der höhere programmiertechnische Aufwand bei der Implementierung der schwächeren Konsistenzmodelle als ihr einziges Manko dar.

In [?] ist beschrieben, wie man sich Formen von DSM vorstellen könnte, für die ein beachtliches Maß an Inkonsistenz akzeptabel wäre. Beispielsweise könnte DSM verwendet werden, um die Auslastung von Computern in einem Netzwerk zu speichern, so dass Clients für die Ausführung ihrer Applikationen die am wenigsten ausgelasteten Computer auswählen können. Weil die Informationen dieser Art innerhalb kürzester Zeit ungenau werden können (und durch die Verwendung

der veralteten Daten keine großen Nachteile entstehen können), wäre es vergebliche Mühe, sie ständig für alle Computer im System konsistent zu halten [?]. Die meisten Applikationen stellen jedoch strengere Konsistenzanforderungen.

## 4.2 Klassifizierung eines Konsistenzmodells

Die zentrale Frage, die für die Klassifizierung (streng oder schwach) eines Konsistenzmodells von Bedeutung ist [?]: wenn ein Lesezugriff auf eine Speicherposition erfolgt, welche Werte von Schreibzugriffen auf diese Position sollen dann dem Lesevorgang bereitgestellt werden? Die Antwort für das schwächste Konsistenzmodell lautet: von jedem Schreibvorgang, der vor dem Lesen erfolgt ist, oder in der „nahen“ Zukunft, innerhalb des definierten Betrachtungsraums, erfolgt wird. Also irgendein Wert, der vor oder nach dem Lesen geschrieben wurde.

Für das strengste Konsistenzmodell, Linearisierbarkeit (atomic consistency), stehen alle geschriebenen Werte allen Prozessoren sofort zur Verfügung: eine Lese-Operation gibt den aktuellsten Wert zurück, der geschrieben wurde, bevor das Lesen stattfand. Diese Definition ist aber in zweierlei Hinsicht problematisch. Erstens treten weder Schreib- noch Lese-Operationen zu genau einem Zeitpunkt auf, deshalb ist die Bedeutung von „aktuellsten“ nicht immer klar. Zweitens ist es nicht immer möglich, genau festzustellen, ob ein Ereignis vor einem anderen stattgefunden hat, da es Begrenzungen dafür gibt, wie genau Uhren in einem verteilten System synchronisiert werden können.

Nachfolgend werden einige Konsistenzmodelle absteigend nach ihrer Strenge vorgestellt. Zuvor müssen wir allerdings klären, wie die Lese- und Schreib-Operationen in dieser Ausarbeitung dargestellt werden.

Sei  $x$  eine Speicherposition, dann können Instanzen dieser Operationen wie folgt ausgedrückt werden:

- $R(x)a$  - eine Lese-Operation, die den Wert  $a$  von der Position  $x$  liest.
- $W(x)b$  - eine Schreib-Operation, die den Wert  $b$  an der Position  $x$  speichert.

## 4.3 Linearisierbarkeit (atomic consistency)

Die Linearisierbarkeit im Zusammenhang mit DSM kann wie folgt definiert werden:

- Die verzahnte Operationsabfolge findet so statt: wenn  $R(x)a$  in der Folge vorkommt, dann ist die letzte Schreib-Operation, die vor ihr in der verzahnten Abfolge auftritt,  $W(x)a$ , oder es tritt keine Schreib-Operation vor ihr auf und  $a$  ist der Anfangswert von  $x$ . Das bedeutet, dass eine Variable nur durch eine Schreib-Operation geändert werden kann.
- Die Reihenfolge der Operationen in der Verzahnung ist konsistent zu den Echtzeiten, zu denen die Operationen bei der tatsächlichen Ausführung aufgetreten sind.

Prozesse	Zeit $\rightarrow$
$P_1$	$W(x)1$ $W(y)2$
$P_2$	$R(x)1$ $R(y)2$

Tabelle 4.1: Linearisierbarkeit ist erfüllt

Die Bedeutung dieser Definition kann an folgendem Beispiel (Tabelle 4.1) nachvollzogen werden. Es sei angenommen, dass alle Werte mit 0 vorinitialisiert sind.

Hier sind beide Bedingungen erfüllt, da die Lese-Operationen den zuletzt geschriebenen Wert zurückliefern. Interessanter ist es, zu sehen, wann die Linearisierbarkeit verletzt ist.

Prozesse	Zeit $\rightarrow$
$P_1$	$W(x)1$ $W(x)2$
$P_2$	$R(x)0$ $R(x)2$

Tabelle 4.2: Linearisierbarkeit ist verletzt, sequentielle Konsistenz ist erfüllt.

In diesem Beispiel (Tabelle 4.2) ist die Echtzeit-Anforderung verletzt, da der Prozess  $P_2$  immer noch den alten Wert liest, obwohl er von Prozess  $P_1$  bereits geändert wurde. Diese Ausführung wäre aber sequentiell konsistent (siehe kommander Abschnitt), da es eine Verzahnung der Operationen gibt, die diese Werte liefern könnte ( $R(x)0$ ,  $W(x)1$ ,  $W(x)2$ ,  $R(y)2$ ). Würde man beide Lese-Operationen des 2. Prozesses vertauschen, wie in der Tabelle 4.3 dargestellt, so wäre keine sinnvolle Verzahnung mehr möglich.

Prozesse	Zeit $\rightarrow$
$P_1$	$W(x)1$ $W(x)2$
$P_2$	$R(x)2$ $R(x)0$

Tabelle 4.3: Linearisierbarkeit und sequentielle Konsistenz sind verletzt.

In diesem Beispiel sind beide Bedingungen verletzt. Selbst wenn die Echtzeit, zu der die Operationen stattgefunden haben, ignoriert wird, gibt es keine Verzahnung einzelner Operationen, die der Definition entsprechen würde.

## Zusammenfassung und Ausblick

In diesem Kapitel soll die Arbeit noch einmal kurz zusammengefasst werden. Insbesondere sollen die wesentlichen Ergebnisse Ihrer Arbeit herausgehoben werden. Erfahrungen, die z.B. Benutzer mit der Mensch-Maschine-Schnittstelle gemacht haben oder Ergebnisse von Leistungsmessungen sollen an dieser Stelle präsentiert werden. Sie können in diesem Kapitel auch die Ergebnisse oder das Arbeitsumfeld Ihrer Arbeit kritisch bewerten. Wünschenswerte Erweiterungen sollen als Hinweise auf weiterführende Arbeiten erwähnt werden.

---

## Literaturverzeichnis

- CMS13. CAMPELLO, RICARDO J. G. B., DAVOUD MOULAVI und JOERG SANDER: *Density-Based Clustering Based on Hierarchical Density Estimates*. Seiten 160–172. Springer, Berlin, Heidelberg, 2013.
- JS21. JERRIM, JOHN und SAM SIMS: *When is high workload bad for teacher wellbeing? Accounting for the non-linear contribution of specific teaching tasks*. Teaching and Teacher Education, 105:103395, 2021.
- Kar01. KARL PEARSON: *On lines and planes of closest fit to systems of points in space*, 1901.
- Lau08. LAURENS VAN DER MAATEN und GEOFFREY HINTON: *Visualizing Data using t-SNE*, 2008.
- Mac67. MACQUEEN, J.: *Some methods for classification and analysis of multivariate observations*. In: LE CAM, LUCIEN M. und JERZY NEYMAN (Herausgeber): *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Band 5.1, Seiten 281–298. University of California Press, 1967.
- MBKS. MESSER, MARCUS, NEIL C. C. BROWN, MICHAEL KÖLLING und MIAO-JING SHI: *Automated Grading and Feedback Tools for Programming Education: A Systematic Review*.
- MHM. MCINNES, LELAND, JOHN HEALY und JAMES MELVILLE: *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*.
- OJM. ORVALHO, PEDRO, MIKOLÁŠ JANOTA und VASCO MANQUINHO: *In-vAASTCluster: On Applying Invariant-Based Program Clustering to Introductory Programming Assignments*.
- PLF24. PAIVA, JOSÉ CARLOS, JOSÉ PAULO LEAL und ÁLVARO FIGUEIRA: *Clustering source code from automated assessment of programming assignments*. International Journal of Data Science and Analytics, Seiten 1–12, 2024.
- TWH<sup>+</sup>. TANG, XIAOHANG, SAM WONG, MARCUS HUYNH, ZICHENG HE, YALONG YANG und YAN CHEN: *SPHERE: Scaling Personalized Feedback in Programming Classrooms with Structured Review of LLM Outputs*.

---

# Index

- Abbildung, 6
- Abschnitt, 5
- Beispiel, 8
- Beweis, 8
- Definition, 8
- Echtzeiten, 11
- Formel, 7
- Freigabekonsistenz, 10
- Inkonsistenz, 10
- Kapitel, 5
- Konsistenz, 10
  - schwach, 10, 11
  - sequentiell, 10
  - streng, 11
- Konsistenzmodelle, 10
- Lemma, 8
- Linearisierbarkeit, 10, 11
- Listing, 6
- Literatur, 9
- Matrix, 8
- Operation
  - Lesen, 11
  - Schreiben, 11
- Optimierung, 10
- Quellen, 9
- Quelltext, 6
- Replikation, 10
- Satz, 8
- Tabelle, 6
- Unterabschnitt, 6
- Vektor, 8
- Verfügbarkeit, 10

# A

---

## Glossar

DisASter	Distributed Algorithms Simulation Terrain, eine Plattform zur Implementierung verteilter Algorithmen [?]
DSM	Distributed Shared Memory
AC	Atomic Consistency (dt.: Linearisierbarkeit)
RC	Release Consistency (dt.: Freigabekonsistenz)
SC	Sequential Consistency (dt.: Sequentielle Konsistenz)
WC	Weak Consistency (dt.: Schwache Konsistenz)



# B

---

## Eigenständigkeitserklärung

- ☐ Die vorliegende Arbeit wurde als Einzelarbeit angefertigt.
- ☐ Die vorliegende Arbeit wurde als Gruppenarbeit angefertigt. Mein Anteil an der Gruppenarbeit ist im untenstehenden Abschnitt *Verantwortliche* dokumentiert:
- ☐ Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige Hilfe Dritter angefertigt habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche kenntlich gemacht. Darüber hinaus erkläre ich, dass ich die vorliegende Arbeit in dieser oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht habe.
- ☐ Es ist keine Nutzung von KI-basierten text- oder inhaltgenerierenden Hilfsmitteln erfolgt.
- ☐ Die Nutzung von KI-basierten text- oder inhaltgenerierenden Hilfsmitteln wurde von der/dem Prüfenden ausdrücklich gestattet. Die von der/dem Prüfenden mit Ausgabe der Arbeit vorgegebenen Anforderungen zur Dokumentation und Kennzeichnung habe ich erhalten und eingehalten. Sofern gefordert, habe ich in der untenstehenden Tabelle *Nutzung von KI-Tools* die verwendeten KI-basierten text- oder inhaltgenerierenden Hilfsmittel aufgeführt und die Stellen in der Arbeit genannt. Die Richtigkeit übernommener KI-Aussagen und Inhalte habe ich nach bestem Wissen und Gewissen überprüft.

---

Datum

---

Unterschrift der Kandidatin/des Kandidaten

## Verantwortliche

Die Tabellen unten führen auf, wer als Autor für die einzelnen Kapitel der vorliegenden Dokumentation beziehungsweise für einzelne Teile des Quellcodes hauptverantwortlich ist.

Insgesamt beteiligt sind die folgenden Personen:

- Autor 1
- Autor 2
- Autor 3

## Dokumentation

Kapitel	Überschrift	Autor
1	Einleitung	Autor 1, Autor 2, Autor 3
2	Problemstellung	Autor 1, Autor 2, Autor 3
3	Aufgabenstellung und Zielsetzung	Autor 1, Autor 2, Autor 3
4	Übrige Abschnitte (Kapitel und Absätze)	Autor 1
4.1	Abschnitt	Autor 3
4.1.1	Unterabschnitt	Autor 2, Autor 3
4.2	Abbildungen und Tabellen	usw.
4.3	Mathematische Formel	
4.4	Sätze, Lemmas und Definitionen	
4.5	Fußnoten	
4.6	Literaturverweise	
5	Beispiel-Kapitel	
5.1	Warum existieren unterschiedliche Konsistenzmodelle?	
5.2	Klassifizierung eines Konsistenzmodells	
5.3	Linearisierbarkeit (atomic consistency)	

## Quellcode

Paket	Autor
algorithms.search	Autor 1
algorithms.sort	Autor 3

## Nutzung von KI-Tools

KI-Tool	Genutzt für	Warum?	Wann?	Mit welcher Eingabefrage bzw. Aufforderung?	An welcher Stelle der Arbeit übernommen?
ChatGPT	Konzept XY erklären lassen	Erklärung von Verständnisfragen zu...	Bei der Bearbeitung des Theorieteils der Arbeit	Welches sind die zentralen Merkmale des Konzepts XY?	S. 25, 30 ff.
DeepL Write	Neuformulierung meiner Textentwürfe	Bessere Lesbarkeit	Über die gesamte Arbeit hinweg	Formuliere die Kapitel 2 und 3 neu in einfachen und leicht verständlichen Sätzen!	S. 45 ff. , S. 67