BHT Berliner Hochschule für Technik

Vorlage für eine Bachelorarbeit bei Siamak Haschemi an der BHT

Vorlage für eine Bachelorarbeit bei Siamak Haschemi an der BHT

vorgelegt von

Max Mustermann

dem Fachbereich VI - Computer Science and Media
der Berliner Hochschule für Technik Berlin
zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

im Studiengang

Medieninformatik

Tag der Abgabe: 29. Januar 2026

Studiere Zukunft

Reviewer
Prof. Dr. Siamak Haschemi
Prof. Dr. Maria Musterfrau

Zusammenfassung

In einem Abstract sollen die erarbeiteten Ergebnisse präsentiert werden. Dieser Hinweis ist notwendig, da dies manchmal vergessen wird.

Das Deckblatt basiert auf einer angepassten Version von Tschirley [2]. Die Schnittstelle wurde vereinfacht und um eine englische Sprachunterstützung erweitert. Zudem wurden überflüssige, doppelte oder veraltete Bestandteile entfernt, um die Verständlichkeit zu verbessern. Außerdem wurde das Seitenlayout des Deckblatts in zwei separate Dateien, `bht-cover-page.sty` und `layout.sty`, aufgeteilt.

Eigenständigkeitserklärung

Ich, Max Mustermann, versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe.

Es wurden folgende KI-Tools verwendet: ChatGPT 5.2 (OpenAI), Claude Sonnet 4.5 (Anthropic) sowie NotebookLM (Google). Etwaige durch die KI vorgenommenen Änderungen und Empfehlungen wurden von mir kritisch geprüft und gegebenenfalls angepasst, um die wissenschaftliche Integrität und die Originalität der Arbeit sicherzustellen.

Diese Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, den 29. Januar 2026

_____

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

Das Wichtigste zuerst: Gib dem GitHub-Repository[1] einen Stern!!!

Aber im Ernst - die Idee ist, dass jeder ein bisschen zum Repository beiträgt. Wenn du eine Verbesserung hast, einen Fehler entdeckst oder auch nur einen Rechtschreibfehler findest, erstelle bitte einen Pull Request. Ziel ist es, dass alle gemeinsam beitragen, damit die Vorlage mit der Zeit immer besser wird.

Wenn du diese Anleitung zum ersten Mal liest, solltest du folgende Schritte abarbeiten:

1. Erstelle einen Zeitplan mit einigen Meilensteinen und sende ihn Siamak über Discord.[2]
2. Lies dir das FAQ durch.
3. Da dies ein Template Repository ist, kannst du über die GitHub-Oberfläche mit dem Button *Use This Template* deine eigene Version erstellen (siehe fig. 1.1).

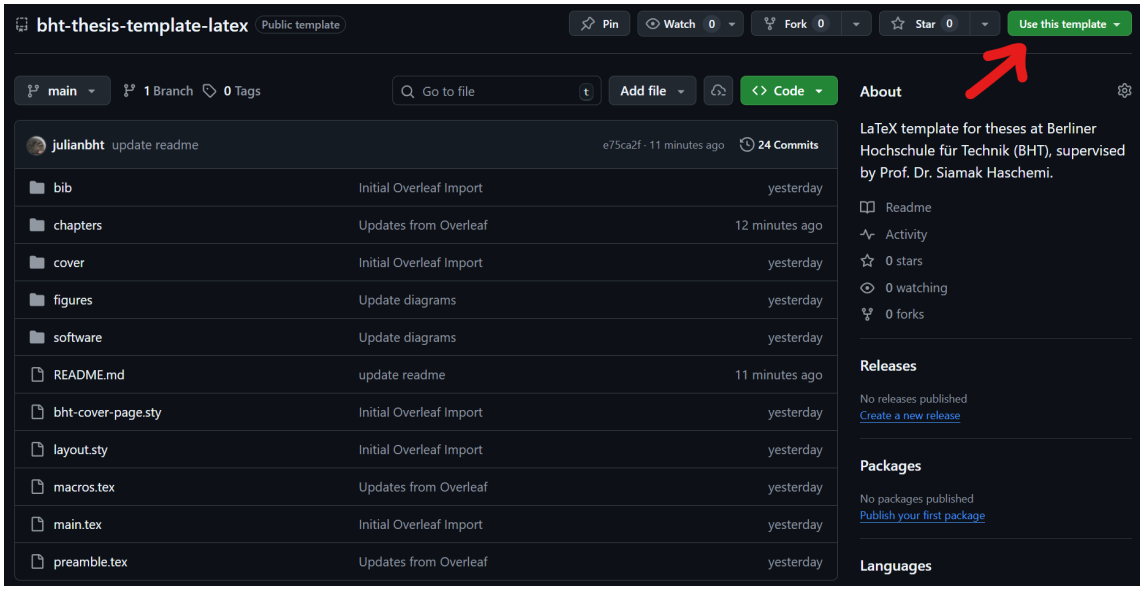

Abbildung 1.1: Schritt 3: Template Repository verwenden

[1] `https://github.com/julianbht/bht-thesis-template-latex`

[2] `TODO:DiscordLink`

4. Füge Siamak zu dem Repository hinzu. Dieser importiert anschließend dein GitHub-Projekt nach Overleaf und fügt dich bei dem Overleaf Projekt hinzu (siehe fig. 1.2).

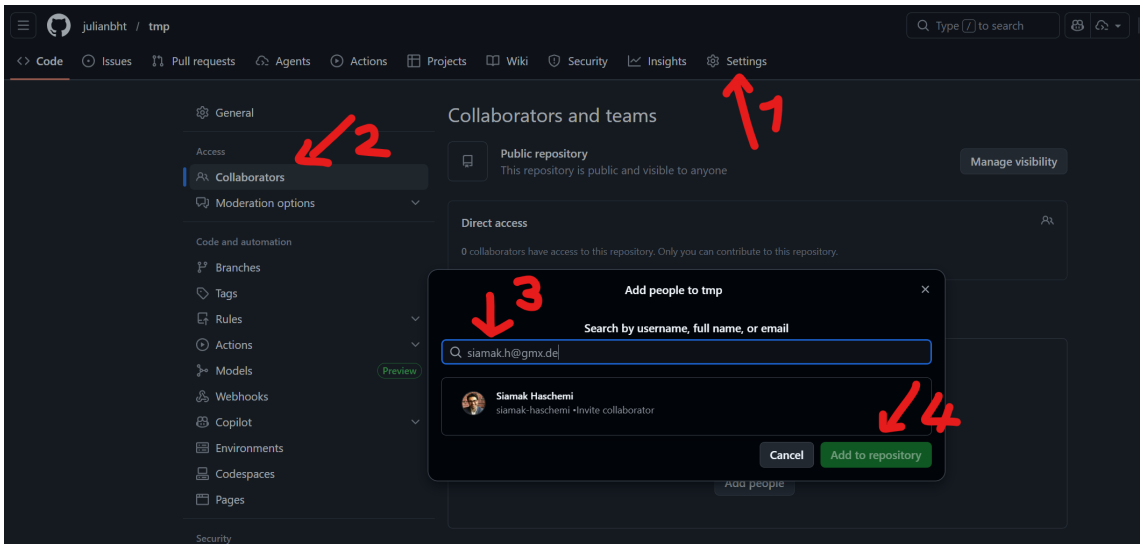

Abbildung 1.2: Schritt 4: Siamak hinzufügen

5. Lösche den Erklärungskram und lege los!

Kapitel 2

Hinweise

2.1 Häufige Fehler

Dieses Kapitel fasst wichtige Hinweise und Empfehlungen zur Erstellung der Bachelorarbeit zusammen und weist auf typische Fehler sowie bewährte Vorgehensweisen hin.

2.1.1 Keine Trennung zwischen Konzeption und Implementation

Ein typischer Fehler beim Erstellen des schriftlichen Teils ist eine zu geringe Trennung zwischen Konzeption und Implementierung. In der Regel benötigt es in der Konzeption UML Diagramme. Die Konzeption soll grundsätzlich framework- und sprachunabhängig formuliert werden. Das heißt: Die beschriebenen Strukturen, Abläufe und Überlegungen müssen so allgemein sein, dass eine Umsetzung in unterschiedlichen Programmiersprachen möglich wäre.

Nur wenn das Thema klar auf ein bestimmtes Framework oder eine bestimmte Sprache ausgelegt ist, ist eine technologisch spezifische Konzeption angemessen.

2.1.2 Grafiken

Beim Layout gibt es viele Details, die nicht kritisch sind. Zwei Punkte sind jedoch wichtig. Die in Grafiken verwendete Schriftart sollte identisch oder ähnlich zur Textschrift sein. Außerdem sollte die Schriftgröße sich am Fließtext orientieren. In der Praxis lässt sich die passende Größe meist nur durch Ausprobieren finden. In nahezu allen professionellen Veröffentlichungen ist dies Standard – daran sollte man sich frühzeitig gewöhnen, da es die Lesbarkeit und den Gesamteindruck der Arbeit deutlich verbessert. Ein Beispiel kann man finden in Section 4.3.1.

2.2 Genereller Aufbau

Eine klassische Bachelorarbeit ist typischerweise wie folgt aufgebaut und kann als grundlegende Orientierung dienen.

- Einleitung
- Grundlagen
- Konzeption
- Implementierung
- Methodik
- Ergebnisse
- Diskussion
- Zusammenfassung

Nicht in jeder Arbeit werden alle genannten Kapitel benötigt. Ein Methodik-Kapitel ist beispielsweise vor allem dann sinnvoll, wenn Experimente oder empirische Untersuchungen durchgeführt werden. Die dargestellte Struktur dient primär als Orientierungshilfe und muss nicht eins zu eins übernommen werden. Generell gilt es folgende grundlegende Punkte zu beachten.

Klare Trennung der Kapitel Konzeption und Implementierung sind klar voneinander zu trennen. Ebenso müssen Ergebnisse und Diskussion separat behandelt werden: Während die Ergebnisse objektiv dargestellt werden, erfolgt die Interpretation erst in der Diskussion. Die Methodik beschreibt ausschließlich die Vorgehensweise und wird nicht mit anderen Inhalten vermischt.

Architektur- und Technologieentscheidungen begründen Die Auswahl von Frameworks und Technologien (z. B. Next.js oder gRPC vs. REST) sollte stets anhand sachlicher Kriterien erfolgen, etwa Community-Größe, Stabilität, Hosting-Möglichkeiten, Performance oder dem konkreten Use Case. Persönliche Präferenzen („damit kenne ich mich gut aus“) sollten dabei nicht im Vordergrund stehen. Es ist wichtig aufzuzeigen, welche Alternativen es gegeben hätte und warum letztlich eine bestimmte Lösung gewählt wurde. So wird nachvollziehbar, dass die Entscheidungen bewusst getroffen wurden und nicht zufällig entstanden sind.

Anforderungsanalyse Die Anforderungsanalyse ist ein Teil der Konzeption und sollte in jeder Abschlussarbeit vorhanden sein. Sie beschreibt systematisch, was das System leisten soll und unter welchen Rahmenbedingungen es betrieben wird. Dabei wird häufig zwischen Akteuren, Use Cases, User Stories sowie funktionalen Anforderungen (FA) und nicht-funktionalen Anforderungen (NFA) unterschieden.

Reflexion Die Diskussion sollte einen Reflexionsteil enthalten. In diesem werden die Limitationen der eigenen Arbeit sowie mögliche Bias offen benannt und kritisch eingeordnet. Ziel ist es zu zeigen, dass die Grenzen der Ergebnisse verstanden werden und realistisch eingeschätzt werden können.

2.3 Abgabe

Der erarbeitete Source Code muss zusammen mit der Arbeit abgegeben werden. Dieser ist allerdings meist zu groß, um ihn per E-Mail zu verschicken. Die empfohlene Variante ist, die Dateien in der BHT-Cloud[1] hochzuladen und den entsprechenden Link in die Abgabe-E-Mail einzufügen.

Bei komplexen Softwaresystemen kann eine kurze Demo oder ein Screencast hilfreich sein. Für Gutachter ist es oft schwierig, allein anhand der schriftlichen Beschreibung einen vollständigen Eindruck des entwickelten Systems zu gewinnen. Dies ist jedoch keine Pflicht. Falls vor der Abgabe nicht ausreichend Zeit vorhanden ist, kann ein Screencast auch noch etwa eine Woche nach der Abgabe nachgereicht werden. In der Praxis erfolgt die Begutachtung häufig erst nach mehreren Wochen, sodass hierfür meist ein zeitliches Fenster besteht.

2.4 Erfolgreiche Bachelorarbeiten

Hier sind Links von erfolgreichen Bachelorarbeiten zu finden, welche als Inspiration dienen können.

TODO: Link einfügen

[1] `https://cloud.bht-berlin.de/`

Kapitel 3

FAQ

In diesem Kapitel werden häufige Fragen beantwortet. Bitte lies dir alle Fragen einmal durch, damit wir häufige Punkte nicht mehrfach erklären müssen und wertvolle Zeit sparen.

Wie viel technisches Vorwissen kann beim Leser vorausgesetzt werden? Der Bericht richtet sich an Informatiker, daher kann grundlegendes technisches Wissen wie Git vorausgesetzt werden. Spezifisches Fachwissen, etwa der technische Aufbau einer Browser-Extension, sollte jedoch nicht erwartet werden.

Welche Schriftart und Formatierung soll ich verwenden? Die Wahl der Schriftart ist nicht entscheidend. Für gedruckte Berichte eignen sich Serifenschriften, da sie besser lesbar sind; du kannst den Standardfont oder den BHT-Font verwenden. Abstände und genaue Formatierung spielen keine Rolle für die Bewertung – entscheidend ist allein der Inhalt. Eine zu große oder zu kleine Zeilenhöhe führt nicht zu einer schlechteren Note.

Sollte man den zweiten Gutachter kontaktieren? Es ist nicht verpflichtend, den zweiten Gutachter zu kontaktieren, und häufig erhält man keine Antwort. Dennoch ist es empfehlenswert, eine kurze Probe oder einen Auszug der Arbeit zu schicken und um Feedback zu bitten. So kann man später nachvollziehbar machen, dass man Rücksprache gesucht hat – besonders hilfreich, falls es größere Meinungsunterschiede geben sollte.

Wo sollte ich für die Bachelorarbeit recherchieren? Geeignete Recherchequellen sind Google Scholar, arXiv, Fachartikel, Journals, Webseiten sowie Bücher; auch ChatGPT Deep Research kann zur Orientierung genutzt werden. Der Anteil reiner Webseitenquellen sollte jedoch maximal ein Drittel der gesamten Literatur ausmachen. Überwiegen Webseiten deutlich, wirkt die Arbeit nicht ausreichend wissenschaftlich fundiert – darauf wird bei der Bewertung geachtet.

Ist der Einsatz von KI-Werkzeugen erlaubt? KI-Werkzeuge dürfen unterstützend eingesetzt werden (z. B. für Code-Generierung oder Textüberarbeitung), solange die Inhalte verstanden und verantwortet werden. Unreflektiertes Übernehmen ist nicht zulässig.

Welche formalen Anforderungen gelten für den Schreibstil? Die Arbeit wird nicht in Ich-Form verfasst. Aussagen müssen mit Quellen belegt werden. Typischerweise werden 20–30 Referenzen erwartet, wobei Webquellen nur einen begrenzten Anteil ausmachen sollten. Zudem empfiehlt es sich, das Abstract erst am Ende zu schreiben, da dort die Ergebnisse zusammengefasst werden und oft erst gegen Abschluss der Arbeit klar ist, was tatsächlich erreicht wurde.

Kapitel 4

Tips & Tricks

Dieses Kapitel sammelt praktische Tipps, Werkzeuge und Workflows, die den Schreib- und Entwicklungsprozess unterstützen.

4.1 Overleaf Verknüpfung mit Git und GitHub

Overleaf bietet sowohl eine direkte Git-Integration als auch eine GitHub-Anbindung an.[1] Beide Varianten sind sehr hilfreich im Arbeitsalltag. Leider ist zu beachten, dass diese Integrationen nur mit einer kostenpflichtigen Overleaf-Subscription verfügbar sind.

Mit der Git-Integration kann das Overleaf-Projekt wie ein ganz normales Git-Repository verwendet werden: Änderungen lassen sich lokal bearbeiten und anschließend per `git pull` und `git push` synchronisieren. Overleaf übernimmt die Aktualisierung automatisch.

Alternativ kann das Projekt über die Overleaf-Oberfläche mit einem GitHub-Repository verknüpft werden. In diesem Fall erfolgt die Synchronisation über die UI von Overleaf.

Persönlich finde ich die direkte Git-Integration etwas angenehmer, da man sich den zusätzlichen Synchronisationsschritt spart. Man kann einfach lokal committen, pushen, in Overleaf neu laden (F5) und die Änderungen sind sofort sichtbar.

4.2 Quellen

Lade für alle Quellen, die du findest, immer auch das PDF herunter. So kannst du die Dateien in NotebookLM[2] hochladen. Es spart erstaunlich viel Zeit, mit den Quellen „sprechen“ zu können und zum Beispiel schnell herauszufinden, wo man bestimmte Informationen gefunden hat.

Es empfiehlt sich dringend, ein Quellenverwaltungstool zu verwenden. Im Folgenden werden einige geeignete Tools vorgestellt.

[1] `https://docs.overleaf.com/integrations-and-add-ons/git-integration-and-github-synchronization/git-integration`

[2] `https://notebooklm.google/`

4.2.1 JabRef

JabRef ist ein Java-basiertes Quellenverwaltungsprogramm (optisch etwas rustikal), das direkt mit `.bib`-Dateien arbeitet, zum Beispiel mit `/bib/sources.bib`. Alle Änderungen werden unmittelbar in dieser Datei gespeichert.

Ein kurzes Erklärungsvideo zu JabRef kann man hier finden: `https://cloud.bht-berlin.de/index.php/s/Qsmr3fGs6FnSrcm`. TODO: Auf Siamaks YouTube Kanal hochladen und verlinken.

Da JabRef lokal läuft, müssen Änderungen an der `.bib`-Datei anschließend nach Overleaf übertragen werden. Dafür bietet sich die Kombination mit der Git-Integration an (siehe Section 4.1). Alternativ bleibt der manuelle Upload der Datei über die Overleaf-Oberfläche.

Theoretisch lässt sich sogar die OpenAI-API per API-key anbinden, um mit den eigenen Quellen zu interagieren. In der Praxis ist es jedoch oft einfacher, die PDFs direkt in NotebookLM zu laden und dort zu benutzen.

4.2.2 Zotero

Zotero ist ebenfalls ein beliebtes Tool zur Quellenverwaltung. Es bietet offenbar eine direkte Integration mit Overleaf, was es unter Umständen komfortabler als JabRef machen könnte. Bisher hat das jedoch noch niemand ausprobiert. Falls jemand Erfahrungen damit sammelt, wäre es hilfreich, diese hier zu ergänzen.

4.3 Nützliche Software

In diesem Abschnitt stellen wir Software und kleine Hilfstools bereit, die sich beim Schreiben der Arbeit als nützlich erwiesen haben. Oft geht es dabei darum, mit einfachen Automatisierungen Zeit zu sparen oder repetitive Aufgaben zu vermeiden. Falls du selbst etwas in diese Richtung entwickelst, erstelle gerne einen Pull Request, damit auch andere davon profitieren können.

4.3.1 Matplotlib

Falls du matplotlib benutzen willst, findest du in `/software/matplotlib` ein Skript, das Konstanten definiert, die genau auf die Seitenbreite und Schriftgrößen dieses Dokuments abgestimmt sind. Dieses Dokument hat eine Textbreite von 426.79134pt, und das Skript definiert die entsprechende Breite in Inches. Figure 4.1 zeigt ein Beispiel.

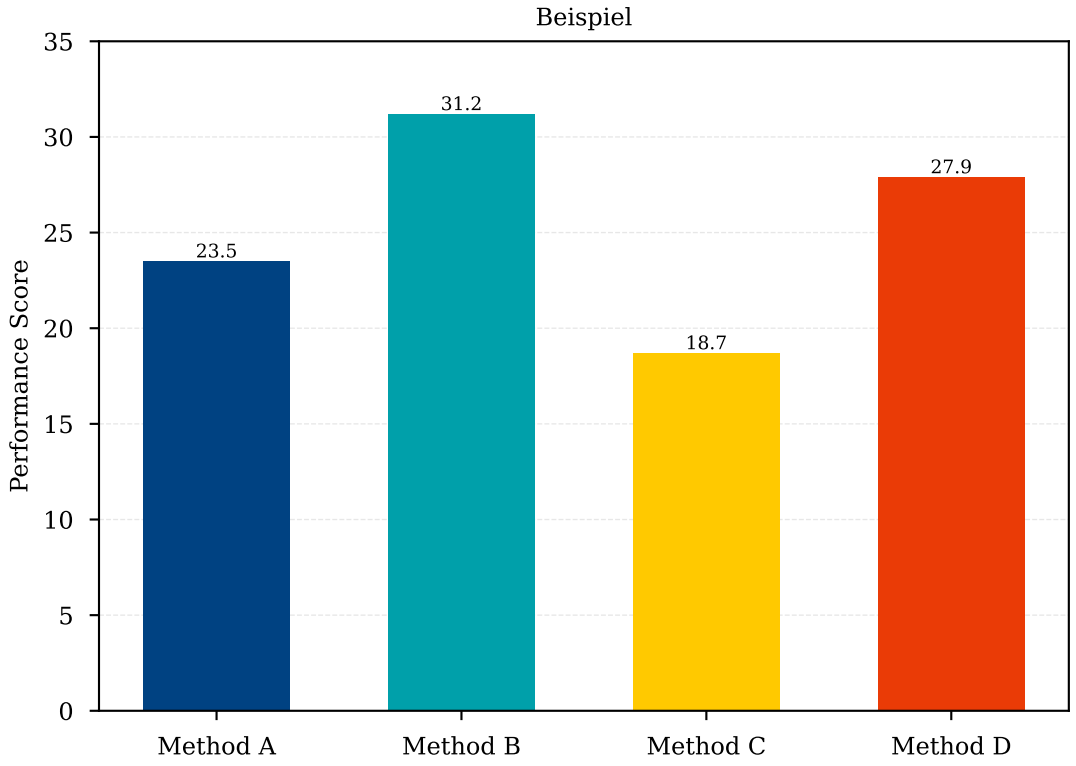

Abbildung 4.1: Beispiel Grafik, mit änlicher Schriftgröße zu dem Text.

4.3.2 Drawio Export Script

Falls du draw.io für Diagramme benutzen möchtest, gibt es in `/software/drawio-svg-export-script` ein Python-Script, das den Export-Workflow automatisiert. Das Script exportiert jede Seite einer `.drawio`-Datei als separate SVG-Datei (benannt nach dem Tab-Titel) in den `/figures` Ordner.

Das Script arbeitet inkrementell: Es exportiert nur Seiten, deren Inhalt sich geändert hat, und verwaltet dazu ein Manifest im Script-Ordner. Nach dem Export werden die Änderungen automatisch via Git committed und gepusht.

Das Script kann wie folgt benutzt werden:

```
cd software/drawio-svg-export-script
python export-drawio-svg.py figures.drawio
```

Oder noch kürzer mit dem Makefile:

```
make svg
```

Die Nutzung des Scripts ist optional. Natürlich können SVG-Dateien auch manuell über die draw.io Oberfläche exportiert werden. Das Script dient lediglich dazu, Zeit zu sparen und repetitive Arbeitsschritte wie das Navigieren durch die UI und das anschließende Hochladen der Dateien zu vermeiden.

Kapitel 5

Erfahrungen von anderen Studenten

Dieses Kapitel dient dazu, allgemeine Erkenntnisse und gesammelte Erfahrungen festzuhalten, die sich im Verlauf verschiedener Abschlussarbeiten ergeben haben und für andere Studierende hilfreich sein können.

Julian

Für Diagramme würde ich beim nächsten Mal direkt Inkscape verwenden, da draw.io teilweise Probleme mit der Vektorisierung von Schriftarten hat. Es gibt zwar Workarounds (Export als PDF, Import in Inkscape, dort vektorisieren und anschließend wieder in draw.io einbinden), aber das hat nicht gut funktioniert. Inkscape ist für SVG-Grafiken besser geeignet und bietet insgesamt professionellere Möglichkeiten.

Siamak hat in seiner Dissertation [1] jedes Kapitel mit einer kurzen Zusammenfassung des vorherigen Kapitels sowie einer kurzen Vorschau auf das aktuelle Kapitel begonnen. Das hilft Lesenden sehr bei der Orientierung und würde ich in zukünftigen Arbeiten auch so machen.

Literatur

[1] Siamak Haschemi. „Model-based testing of dynamic component systems“. Diss. Humboldt-Universität zu Berlin, Juli 2015. URL: https://www.researchgate.net/publication/280889171_Model-based_testing_of_dynamic_component_systems (besucht am 28. 01. 2026) (zitiert auf Seite 11).

[2] Sven Tschirley. *LaTeX – Werkzeuge*. Berliner Hochschule für Technik. 2026. URL: https://prof.bht-berlin.de/tschirley/latex-werkzeuge (besucht am 27. 01. 2026) (zitiert auf Seite i).