Künstliche Intelligenz und wissenschaftliches Arbeiten

Bucher / Holzweißig / Schwarzer

2024
ISBN 978-3-8006-7322-3
Vahlen

umfassendes Spezialsortiment im Bereich Recht, Steuern und Wirtschaft mit rund 700.000 lieferbaren Fachbuchtiteln.

6 Nutzung von Softwarequellen und generativer KI in der Softwareentwicklung

Insbesondere in den Fachgebieten der Wirtschaftsinformatik und Informatik, aber auch in anderen Disziplinen wie der Elektrotechnik, der technisch orientierten Betriebswirtschaft oder im Wirtschaftsingenieurwesen, wird Software als Teil von Prüfungsleistungen oder im Rahmen von wissenschaftlichen Arbeiten entwickelt. Daneben hat die Softwareentwicklung für (Wirtschafts-)Informatikerinnen und -Informatikern in der beruflichen Praxis einen bedeutenden Stellenwert und stellt für Teile dieser Berufsgruppe ein Haupttätigkeitsfeld dar. Bei der Softwareentwicklung kommen im Regelfall Softwarequellen Dritter zum Einsatz, die in den eigenen Quellcode eingebunden oder deren Quellcode kopiert und angepasst wird. Daneben kommen heutzutage auch vermehrt generative KI-Werkzeuge wie bspw. GitHub-Copilot zum Einsatz, um den Softwareentwicklungsprozess zu unterstützen. Daher ist es wichtig, den korrekten Umgang mit Softwarequellen einerseits und die Einsatzpotenziale und -risiken von generativer KI andererseits im Kontext der Softwareentwicklung näher zu beleuchten. So soll die Basis für einen verantwortungsvollen, rechtskonformen und ethisch vertretbaren Umgang mit Softwarequellen Dritter sowie generativer KI ermöglicht werden.

Um die Einsatzgebiete von KI in der Softwareentwicklung besser verstehen zu können, ist es zweckmäßig, zunächst auf die zentralen Phasen des Softwareentwicklungsprozesses einzugehen. Dies erfolgt im ersten Unterkapitel dieses Abschnitts. Aufbauend darauf werden im zweiten Unterkapitel der korrekte Umgang mit Softwarequellen diskutiert und entsprechende Handlungsempfehlungen, auch für die berufliche Praxis, gegeben. Um dritte Softwarequellen im eigenen Projekt rechtskonform nutzen zu können, ist es auch wichtig, die Grundzüge von Lizenzierungsfragen von Softwarequellen zu kennen und verstanden zu haben. Ein entsprechender Überblick hierzu erfolgt im dritten Unterkapitel. Zum Abschluss dieses Kapitels wird im vierten Unterkapitel auf den Einsatz von generativen KI-Werkzeugen eingegangen. Anhand von GitHub-Copilot wird der Aufbau und die Funktionsweise entsprechender Werkzeuge erklärt. Dabei wird allen voran auf die Herausforderungen beim Einsatz dieser Werkzeuge eingegangen.

Konkret werden Sie die folgenden Dinge in diesem Kapitel lernen, die von hohem Nutzen für das Studium sowie die betriebliche Praxis sind:

- Phasen des Softwareentwicklungsprozesses und potenzielle Einsatzgebiete von generativer KI kennen
- Verstehen, warum und wie Softwarequellen korrekt kenntlich gemacht werden, um damit rechtkonforn und ethisch korrekt zu handeln
- Grundfragestellungen der Softwarelizenzierung und der Nutzung von Free- and Open-Source-Softwarelizenzen verstehen
- Die Nutzung von generativer KI im Softwareentwicklungsprozess kennenlernen und kritisch-reflektiert anwenden können

6.1 Konzeption des Softwareentwicklungsprozesses

Der Softwareentwicklungsprozess wird gewöhnlicherweise als mehrstufiger Prozess konzeptionalisiert, wobei die Benennung und die Anzahl der Aktivitäten sich je nach Modell, Methodik und Autoren unterscheiden. Der Softwareentwicklungsprozess kann wasserfalltypisch als lineare Abfolge von Aktivitäten (klassische Methodik), als Modell sich überlappender und mit Rücksprüngen versehender Phasen oder als dynamisch-iterativer Kreislauf mit parallelisierbaren Aktivitäten (agile Methodik) konzeptionalisiert werden.[49]

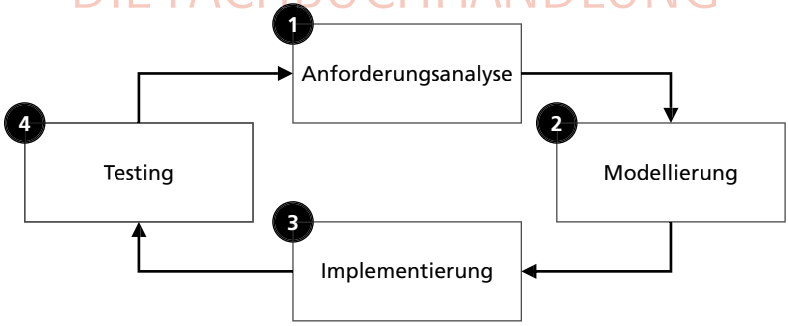

Abbildung 22: Konzeptionalisierung des Softwareentwicklungsprozesses in vier Phasen (Quelle: Eigene Darstellung)

Gewöhnlicherweise umfasst die Softwareentwicklung zunächst die zentrale Aktivität der Anforderungsanalyse. Hierbei werden die funk-

[49] Für einen guten Überblick der Thematik und verschiedener Vorgehensmodell vgl. Laudon et al. (2010), S. 904 ff.

tionalen- und nicht-funktionalen Anforderungen, und damit auch die Akzeptanzkriterien, an die zu entwickelnde Software erhoben und dokumentiert (1). Aus den Anforderungen kann im nächsten Schritt im Rahmen der Softwaremodellierung die Softwarespezifikation bzw. der Systementwurf abgeleitet werden. Hierbei entsteht u. a. ein technisches Design der Software inklusive der angestrebten Lösungsarchitektur. Dabei können auch Prototypen sowie sog. Proof-of-Concepts in unterschiedlicher Breite und Tiefe zum Einsatz kommen (2). Auf Basis der Modellierung erfolgt in der Phase der Implementierung die eigentliche softwaretechnische Umsetzung im Sinne der Programmierung, d. h. es erfolgt die Erstellung von Software-Quellcode (3). Schon während der Programmierung oder gar schon in der Phase der Modellierung können auf Basis der Softwarespezifikation entsprechende Testfälle abgeleitet und als automatisierbare Softwaretests hinterlegt werden (Stichwort: „Testgetriebene Entwicklung"). Mindestens im Anschluss an die Implementierung sollte das Produkt im Testing auf seine korrekte Funktionsweise und die Übereinstimmung mit den gestellten Anforderungen (Akzeptanztests) geprüft werden (4). Zu jeder Zeit können sich weitere Anforderungen ergeben, die dann in der Phase der Anforderungsanalyse untersucht und der weiteren Entwicklung zugeführt werden.

Die Ausführungen zeigen, dass der Softwareentwicklungsprozess aus weit mehr als der eigentlichen Programmierung im Sinne der Entwicklung von Software-Quellcode besteht. Entsprechend breiter als die Unterstützung der Phase der eigentlichen Implementierung kann damit auch der potenzielle Einsatz von generativer KI sein, so zum Beispiel bei der Unterstützung der Phasen der Modellierung oder des Testens. Hierauf soll, insbesondere mit Hinblick auf die testgetriebene Entwicklung, später konkreter eingegangen werden. Damit aber der Einsatz von generativer KI korrekt und nachhaltig erfolgt, ist eine Reihe von Aspekten zu beachten. Ein zentraler Punkt dabei ist die Frage der Kenntlichmachung von genutzten Software-Quellen, um die es im folgenden Unterkapitel gehen soll.

6.2 Korrektes Zitieren von Softwarequellen

In der Softwareentwicklung werden oft Quellen Dritter in Form von Quellcode, Programmierbibliotheken, Software-Dokumentationen, Software-Tutorials, Stack-Overflow-Beiträgen usw. verwendet. Daneben kommen auch zunehmend generative KI-Werkzeuge (bspw. GitHub-Copilot) zum Einsatz. Es stellt sich die Frage, wie sich ein korrektes Zitieren dieser Quellen gestaltet. Im Folgenden wird hierüber Auskunft gegeben und das Vorgehen anhand mehrerer Beispiele ver-

deutlicht. Es werden die rechtlichen Grundlagen und weitere Gründe, die eine Kenntlichmachung der Entnahme verlangen, diskutiert.

6.2.1 Urheberrecht und Kennzeichnung der Entnahme

Das deutsche Urheberrecht legt fest, dass die Nutzung fremden geistigen Eigentums stets deutlich kenntlich gemacht werden muss.[50] Zudem hat der Urheber das Recht auf Anerkennung seiner Urheberschaft am Werk, so dass eine Nennungspflicht besteht.[51] Dabei meint „fremdes geistiges Eigentum" alles, was nicht Ergebnis der eigenen geistigen Schöpfung ist und über eine gewisse Schöpfungshöhe[52] verfügt. Die Kenntlichmachung geschieht, wie aus anderen Kontexten bekannt, durch eine Kennzeichnung der Entnahme im Rahmen des Zitierens. Die Nutzung von Software-Quellcode, Programmierbibliotheken, Software-Dokumentationen usw. besitzt hier keine Sonderrolle, „Computerprogramme" werden explizit vom Urheberrecht als geschützte Werke benannt.[53] So handelt es sich bei den vorstehend genannten Quellentypen genauso um Quellenmaterial, wie es bei Fachartikeln, Monographien sowie Ton- und Bildmaterial der Fall ist. Entsprechend gelten auch hier die Regelungen des deutschen Urheberrechts sowie die Standards guter wissenschaftlicher Praxis.

Daneben gibt es noch weitere Gründe, warum eine Kenntlichmachung der Entnahme erfolgen muss. So kann beispielsweise der genutzte Software-Quellcode einer Softwarelizenz unterliegen, die die Angabe und sogar noch weitere Dokumentationspflichten und/oder Bedingungen bei einer Nutzung verlangt.[54] Hierauf wird vertieft im

[50] Vgl. § 63 Abs. 1 UrhG

[51] Vgl. § 13 UrhG. Das Namensnennungsrecht steht grundsätzlich jedem Urheber bzw. jeder Urheberin zu, auch z. B. Programmiererinnen und Programmierern von Software (vgl. Eichelberger/Wirth/Seifert, 4. Auflage 2022, UrhG § 13 Rn. 1 m. w. N.).

[52] Der Begriff der Schöpfungs- oder auch Gestaltungshöhe ist das zentrale Kriterium bei der Beantwortung der Frage, ob es sich im jeweiligen Falle, also bei einer persönlichen geistigen Schöpfung, um ein geschütztes Werk handelt oder nicht (vgl. Wandtke/Bullinger/Bullinger, 6. Auflage 2022, UrhG § 2 Rn. 23–25). Die Schöpfungshöhe wird am Grad der Individualität bestimmt, wobei die Bestimmung dieses Grads nicht trivial sein kann und die Schwelle für verschiedene Werkarten auch unterschiedlich ausfallen kann (vgl. ebenda). Sollte keine Sicherheit über das Erreichen der Schwelle der Schöpfungshöhe bestehen, so empfiehlt es sich im Zweifelsfall, von einem Urheberschutz auszugehen und die Entnahme kenntlich zu machen (vgl. https://www.uni-regensburg.de/assets/rechtsgrundlagen/Urheberrecht/Urheberrecht_in_der_Lehre.pdf, 25.5.23). Ferner sei auf die Ausführungen zur „kleinen Münzen" im folgenden Unterkapitel verwiesen.

[53] Vgl. § 2 Abs. 1 Nr. 1 UrhG

[54] Ein Beispiel hierfür ist die Apache 2.0 License (vgl. https://www.apache.org/licenses/LICENSE-2.0.html, 19.5.23), die bspw. u. a. verlangt, dass Code-

Folgekapitel eingegangen werden. Ferner dient im Rahmen von Prüfungsprozessen die Kenntlichmachung der Transparenzstiftung, um die Eigen- von einer Fremdleistung abzugrenzen. Sie dient ferner auch aus studentischer Sicht zum eigenen Schutz, um den Verdacht einer etwaigen Nutzung generativer KI-Werkzeuge, die ggf. für Prüfungsleistungen als Hilfsmittel ausgeschlossen wurden, von vornherein entkräften zu können. In der betrieblichen Praxis ist die Dokumentation von Entnahmen zudem sehr bedeutsam, damit die Nutzung fremder Software-Quellcodes und fremder Programmierbibliotheken über den Softwarelebenszyklus hinweg nachvollzogen werden kann. Dies ist insbesondere aus Sicherheitsgründen und aus rechtlichen Aspekten hinsichtlich Lizenzierungsfragen, so zum Beispiel bei der Nutzung von FOSS[55], wichtig. Zuletzt kann argumentiert werden, dass die Dokumentation einer Entnahme aus ethischen Gründen erfolgen sollte, um die Leistung der Urheberinnen und Urheber zu würdigen und ihre Arbeit, von der man profitieren darf, anzuerkennen („to give credit“).

6.2.2 Vorgehen bei der Kenntlichmachung einer Entnahme

Ähnlich wie beim Umgang mit Texten können die bekannten Regeln des direkten und indirekten Zitierens auf die Nutzung von Software-Quellen und entsprechender Dokumentationen übertragen werden. Das Zitieren eines fremden Software-Quellcodes erfolgt hierbei durch eine Kommentierung im eigenen Quellcode und zwar an Ort und Stelle der Übernahme. Die folgenden Dinge sind für jede Entnahme festzuhalten:

1. Anfang und Ende der Übernahme kennzeichnen
2. Art und Höhe der Übernahme ausführen
3. Quellenverweis (URL) auf das Repository / die Homepage / den Ursprungsort setzen
4. Datum des Abrufs dokumentieren
5. ggf., d. h. wenn vorhanden, Softwarelizenz nennen
6. ggf. weitere durch die Softwarelizenz geforderten Angaben vornehmen

Die Dokumentation der Art und des Umfangs der Übernahme ist eine qualitative Beschreibung. Durch diese Beschreibung kann auch eine direkte Entnahme (direktes Zitat – wortwörtlich) von einer sinngemäßen Entnahme (indirektes Zitat – dem Sinn nach) unterschieden

Übernahmen immer nur unter Weitergabe des Lizenztexts erfolgen können und Code-Änderungen im Quellcode dokumentiert werden müssen.

[55] Abkürzung für „Free and Open Source Software“, das Thema wird weiter unten behandelt.

werden. Sollte der genutzte fremde Software-Quellcode einer Lizenz unterliegen, dann ist diese im Regelfall anzugeben und, je nach Bedingungen der Lizenz, sind ggf. auch noch weitere Angaben zu tätigen, so zum Beispiel die Nennung der Autorinnen und Autoren des Werks oder die Wiedergabe des Lizenztexts im Quellcode selbst. Auch ist darauf zu achten, ob der übernommene Quellcode lizenztechnisch gesehen überhaupt kompatibel mit dem eigenen Projekt ist und dort verwendet werden darf bzw. unter welchen Bedingungen.[56]

Als Hilfestellung, sofern die o. g. Punkte eingehalten werden, kann folgende Syntax für die Kenntlichmachung der Entnahme verwendet werden. Zur einfachen Zuordnung kann direkt beim Start der Entnahme eine ganzzahlige Identifikationsnummer [id] vergeben werden, um die Passagen eindeutig zuzuordnen.

```
/* Code citation start [id]
* Description of quality of citation and resource link
*
* Author (if available, else delete): <String>
* Publication date (if available, else delete): <Date>
* Title of resource (if available, else delete): <String>
* Resource link (if available, else delete): <String>
* Accessed on: <Date>
* Software license (if available, else delete): <String>
* Link to software license (if available, else delete):
<String>
* Further license statements (if demanded by license,
else delete): <String>
*/
```



```
Code being cited
```

```
/*
* Code citation end [id]
*/
```

Um die Lesbarkeit des Codes nicht durch zu viele Kommentare zu beeinträchtigen, ist es auch möglich, einen Kurzvermerk [id] zu setzen und die Detailangaben dann in einem zentralen Quellenverzeichnis, bspw. in einer separaten Datei „SOURCES.txt“, zu bündeln. Das hat auch den Vorteil, dass Quellen mehrfach genutzt werden können, ohne die Daten jeweils redundant anzugeben. Eine Angabe würde dann wie folgt aussehen:

```
// Code citation start [1]
// Description of quality of citation

Code being cited

// Code citation end [1]
```

[56] Vgl. hierzu die Ausführungen im folgenden Unterkapitel

Im Quellenverzeichnis in der Datei „SOURCES.txt“ werden dann die Details wiedergegeben:

```
Sources

[1] author, publication date, title, resource link, ac-
cessed on dd.mm.yy, software license, link to software
license, further license statements
…
[i] …
…
[n] …
```

6.2.3 Beispiel: Kennzeichnung einer Entnahme aus einer API-Dokumentation mittels Freiform

Abbildung 23 zeigt ein Beispiel einer Übernahme eines Code-Schnipsels aus einer API-Dokumentation der Softwarebibliothek OpenPGP.js (vgl. https://docs.openpgpjs.org). Anhand der Kommentierung im eigenen Quellcode sind der Start und das Ende der Entnahme ersichtlich. Ferner wird dargelegt, wie es sich mit der Art und Höhe der Übernahme verhält. Die URL der Quelle wird angegeben, ebenso das Abrufdatum sowie die Softwarelizenz inkl. Verweis auf den Lizenztext.


```
function encrypt(nachricht) {
  let time_start = Date.now();
  /*
   * Parts of the following code have been taken, adapted and extended from the OpenPGP.js
   * API documentation here: https://docs.openpgpjs.org/ (31.3.23). The code provided by
   * the OpenPGP.js API documentation is licensed under the GNU Lesser General Public
   * License (3.0 or any later version). Please take a look at the LICENSE file for more
   * information: https://github.com/openpgpjs/openpgpjs/blob/main/LICENSE.
   */
  (async () => {
    //reading the public key
    const publicKey = await openpgp.readKey({ armoredKey: pub_key });
    const message_text = nachricht.inhalt;
    //encrypt data using public key provided
    const encrypted = await openpgp.encrypt({
      message: await openpgp.createMessage({ text: message_text }),
      encryptionKeys: publicKey,
    });
    /*
     * End of code taken, adapted and extended from OpenPGP.js API documentation.
     */
    let time_stop = Date.now() - time_start;
    document.getElementById("log-ausgabe").innerHTML =
      time_stop + " Millisekunde(n) zum Verschlüsseln benötigt.";
  })();
}
```

Abbildung 23: Kennzeichnung der Entnahme als Kommentar mit Anfang und Ende und weiteren Angaben

6.2.4 Beispiel: Kennzeichnung einer Entnahme aus Medium-Post mittels Angabesyntax

Abbildung 24 zeigt das Beispiel einer Code-Übernahme von einem Beitrag auf der Plattform „medium.com“ (vgl. https://medium.com/analytics-vidhya/understanding-oscillators-python-2813ec38781d).[57] Zur Kennzeichnung wird die Syntax, die oben vorgestellt wurde, genutzt. Die Qualität der Übernahme wurde kenntlich gemacht, ebenso ist der Quellenverweis (per URL) gegeben. Ferner wurden der Autor der Quelle sowie das Publikationsdatum, der genaue Titel sowie das Abrufdatum angegeben, Anfang und Ende der Übernahme sind ersichtlich. Da keine Lizenzinformation vorliegt, wurde dies so angegeben. Hier ist Vorsicht geboten, denn aufgrund der fehlenden Lizenzinformation ist nicht klar, ob der Quellcode genutzt werden darf. Es empfiehlt sich daher eine schriftliche Anfrage beim Urheber zu stellen.

Start of code citation [1].

The following code is based on and in parts taken from Medium: Understanding Oscillators (Python)

- author: gabijdh
- published on: Apr 20, 2021
- title: Understanding Oscillators (Python)
- resource link: https://medium.com/analytics-vidhya/understanding-oscillators-python-2813ec38781d
- accessed: May 23, 2023
- license: no license information available

Create a damped oscillator with a given damping factor and frequency.

```
# import libraries
import numpy as np
import pandas as pd
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
```

```
t = np.linspace(0,30,10000)
y = [0,1]
# damping constant
gamma = 0.05
# frequency of oscillation
omega_sqr = 5

# function to solve
def sho(t,y):
    solution = (y[1],(-gamma*y[1]-omega_sqr*y[0]))
    return solution

solution = solve_ivp(sho, [0,10000], y0 = y, t_eval = t)
```

End of code citation [1].

Abbildung 24: Kennzeichnung der Entnahme mit Detailvermerk entsprechend der Angabesyntax

[57] Wir danken Herrn Jonas Michel für das Beispiel, das aus einem Praxisprojekt stammt.