Analyse des Projekts FKT-V4.1-Audit-Release

Forschungsbericht zum Projekt „FKT-V4.1-Audit-Release“

Einleitung und Kontext

Die digitale Transformation in Wissenschaft und Technik setzt zunehmend voraus, dass wissenschaftliche Theorien und deren praktische Implementierungen nicht nur entwickelt, sondern auch transparent, nachvollziehbar und langlebig dokumentiert werden. Diese Erwartung trifft in besonderem Maße auf Projekte zu, die innovative interdisziplinäre Ansätze wie die Fraktale Kausale Theorie (FKT) mit modernen Methoden des Software-Engineerings verbinden. Das Projekt „FKT-V4.1-Audit-Release“ hat es sich zur Aufgabe gemacht, genau auf dieser Schnittstelle zu verankern: Es adressiert die wissenschaftliche Operationalisierung der FKT und legt den Fokus auf Auditierbarkeit, Modularität, Nachvollziehbarkeit und dauerhafte Referenzierbarkeit durch die Nutzung offener Plattformen wie GitHub und Zenodo[1].
Die Prinzipien der Open Science - offene Software, freie Zugänglichkeit zu Forschungsdaten und reproduzierbare Workflows - bilden den Hintergrund des Projektes. Aufbauend auf modernen Open-Source-Lizenzmodellen, Versionierung und DOI-basierten Publikationsstrategien, bietet dieses Projekt ein Beispiel für zukunftsorientierte, interdisziplinäre Forschungs- und Entwicklungsarbeit. Die Zielsetzung dieses Berichts ist es, die Ziele, Inhalte, Methodik sowie die Ergebnisse des Projekts umfassend zu analysieren und kontextualisieren. Darüber hinaus werden die zugrunde liegenden theoretischen und technischen Grundlagen, die rechtlichen Rahmenbedingungen und die Perspektiven für künftige Entwicklungen erläutert.

Projektziele

Das Projekt „FKT-V4.1-Audit-Release“ hat sich mehrere, eng miteinander verbundene Ziele gesetzt. Im Zentrum steht die auditierbare, transparente Implementierung der Fraktalen Kausalen Theorie (FKT) in einer modularen Softwarearchitektur, die sowohl wissenschaftlichen als auch technischen Ansprüchen genügt[2]. Folgende Kernziele wurden identifiziert:

1. **Transparente Operationalisierung der FKT**: Die Fraktale Kausale Theorie, die sich mit der Beschreibung komplexer, nichtlinearer, oft selbstähnlicher Kausalzusammenhänge beschäftigt, soll in ausführbarer Software abgebildet werden. Dies fördert nicht nur das wissenschaftliche Verständnis, sondern ermöglicht auch die Anwendung in vielfältigen Domänen - von Physik über Sozialwissenschaften bis hin zu Technik und Informatik.
2. **Auditierbarkeit durch Open-Source-Methoden**: Auditierbarkeit bedeutet im Kontext des Projekts, dass alle relevanten Artefakte, angefangen beim Quellcode über Dokumentation bis zu Metadaten und Lizenzinformationen, öffentlich zugänglich, überprüfbar und nachvollziehbar sind. Dies ermöglicht unabhängigen Dritten, die Umsetzung und Einhaltung der konzeptionellen Vorgaben zu kontrollieren[2].

3. **Langfristige wissenschaftliche Nachvollziehbarkeit und Referenzierbarkeit**: Durch die DOI-basierte Archivierung auf Zenodo sowie durch die Versionierung und Dokumentation im GitHub-Repository wird sichergestellt, dass der Projektstand sowohl eindeutig zitierbar als auch dauerhaft abrufbar bleibt. Ziel ist es, dem Prinzip der guten wissenschaftlichen Praxis zu genügen - insbesondere in Bezug auf Nachweis, Transparenz und Reproduzierbarkeit[3].
4. **Modularität und Erweiterbarkeit**: Die Entwicklung einer klar strukturierten, modularen Softwarearchitektur dient nicht nur der Wartbarkeit, sondern schafft auch die Möglichkeit zur Integration neuer Funktionen und Validierungskriterien, um das Projekt künftig auszuweiten.
5. **Wissenschaftliche und gesellschaftliche Wirkung**: Eine der zentralen Ambitionen des Projekts ist es, Methoden der FKT auf andere Disziplinen zu übertragen und so komplexe Systeme mithilfe mathematischer Modellierung und Visualisierung besser zu verstehen.

Die Ausarbeitung und Erreichung dieser Ziele trägt darüber hinaus zur Diskussion über Methodenstandards, offene Wissenschaft und Interdisziplinarität in der modernen Forschung bei.

Methodik

Die Methodik des „FKT-V4.1-Audit-Release“-Projekts zeichnet sich durch eine Synthese aus systematischer Dokumentation, versionierter Softwareentwicklung und DOI-gestützter Veröffentlichung aus. Dies ermöglicht es, technische, wissenschaftliche und organisatorische Anforderungen an ein modernes Forschungssoftwareprojekt zu verbinden[1].

1. Systematische Sammlung, Strukturierung und Veröffentlichung

Die Erarbeitung der Inhalte folgt einem methodischen Ansatz, bei dem Theorie, mathematische Grundlagen, Anwendungsfelder und philosophische Reflexionen explizit als einzelne, miteinander in Beziehung stehende Module behandelt werden. Die Strukturierung in Markdown- und PDF-Dateien, wie sie im GitHub-Repository und im Zenodo-Datensatz zu finden sind, bildet die Grundlage für eine nachvollziehbare, wiederverwendbare Veröffentlichung.

2. Modulare Organisation und Versionierung über GitHub

Das zentrale Entwicklungstool ist das GitHub-Repository. Die darin enthaltenen Ordner und Dateien sind eindeutig versioniert. Release-Management und Versionierung (z.B. V4.1) sorgen dafür, dass jede Änderung dokumentiert, rückverfolgbar und detailgenau abgelegt ist. Die Projektdokumentation umfasst eine README mit Einführung und Installationshinweisen, die Hauptmodule der Software (z.B. main.py), unterstützende Module (z.B. utils/), Testfälle (z.B. tests/) sowie eine ausführliche Erläuterung der Architektur[2].

3. DOI-Archivierung auf Zenodo

Der Zenodo-Datensatz[1] fungiert als langfristiger Speicher und Zitiermechanismus durch die Zuweisung einer DOI. Diese Funktionalität sichert die dauerhafte Referenzierbarkeit und die Rechtssicherheit der veröffentlichten Daten und Software. Der DOI verweist stets auf eine

bestimmte, versionierte Ausprägung der Veröffentlichung, wobei alle vorangegangenen Versionen weiterhin einsehbar und nachvollziehbar bleiben.

4. Auditierung und Validierung

Die Auditierung erfolgt in mehreren Schritten:

- **Bereitstellung aller relevanten Dateien**: Hierzu zählen Quellcode, Lizenzinformationen (LICENSE), README, Dokumentation und ggf. Skripte zur automatisierten Prüfung.
- **Modularisierung und klare Schnittstellen**: Die transparente Definition von Modulen und APIs erleichtert die unabhängige Überprüfung einzelner Komponenten.
- **Nachvollziehbare Testfälle**: Das Vorhandensein von Tests (Unit-Tests, Funktionstests) trägt zur Validierung der Funktionalität bei.
- **Veröffentlichung der Auditversion**: Auditierbare Releases (insbesondere V4.1) werden als stabile, geprüfte Baseline bezeichnet.

Die angewandte Auditmethodik orientiert sich dabei an gängigen Standardverfahren wie Dokumentenprüfung, Interview mit den Verantwortlichen (bei größeren Teams), Checklistenprüfung und Datenauswertung, wobei im vorliegenden Projekt vorrangig Dokumentenprüfung und Datenauswertung im Mittelpunkt stehen[5].

5. Metadaten und Dokumentation

Sämtliche Veröffentlichungen enthalten umfangreiche Metadaten gemäß DataCite-Schema (Titel, Autoren, Keywords, Kurzbeschreibung, Lizenz, Version etc.). Dies erleichtert die Wiederauffindbarkeit und unterstützt die Anbindung an externe Datenbanken und Communities [6].

6. Veröffentlichung und Kommunikationsstrategie

Die Kombination von GitHub (Commander der Entwicklung und Kollaboration) sowie Zenodo (Langzeitarchivierung und DOI-Vergabe) stellt eine Best-Practice-Strategie für moderne Forschungssoftwareprojekte dar. Die Veröffentlichung - inklusive offener Lizenzen - ermöglicht die Einbindung der Community, Feedback-Mechanismen und die Erweiterung durch Drittparteien[3].

Inhalte und Komponenten des Projekts

Die Inhalte sind sowohl wissenschaftlich als auch technisch orientiert und gliedern sich in mehrere Kernbereiche. Die folgende Übersicht und die begleitende Tabelle geben einen systematischen Einblick.

Zentrale inhaltliche Bausteine

1. **Einführung in die Fraktale Kausale Theorie (FKT)**
   Die Theorie beschäftigt sich mit fraktalen, selbstähnlichen Strukturen, nichtlinearen

dynamischen Systemen und ihren kausalen Beziehungen. Interdisziplinäre Anwendungen reichen von Physik bis Sozialwissenschaften.

2. **Mathematische Grundlagen**
   Kernkonzepte wie Iteration, Rückkopplung, Selbstähnlichkeit und Hausdorff-Dimension werden erläutert. Es werden Algorithmen zur Berechnung und Simulation dieser Strukturen vorgestellt und implementiert.
3. **Visualisierungen**
   Die Software beinhaltet Werkzeuge zur Erstellung und Analyse von Fraktalen, einschließlich bekannter Beispiele wie Mandelbrot-Menge, Julia-Mengen, Zeltabbildung und Lorenz-Attraktor[7][8].
4. **Anwendungen in verschiedenen Disziplinen**
   Die Implementierung der FKT wird in verschiedenen Kontexten demonstriert: Physik (z.B. chaotische Systeme), Chemie (Musterbildung), Biologie (zelluläre Muster) und Sozialwissenschaften (Netzwerkdynamik, Rückkopplung in Systemen).
5. **Philosophische Implikationen**
   Es werden Fragestellungen zu Kausalität, Determinismus, Realismus und Erkenntnistheorie diskutiert. Die Verbindungen philosophischer und mathematischer Reflexionen werden insbesondere in der Dokumentationsstruktur sichtbar gemacht[1].
6. **Softwarearchitektur: Modularität und Testbarkeit**
   Die Implementierung verfolgt ein modulares Architekturprinzip: Hauptmodul(e), Hilfsfunktionen (utils), Testskripte und Konfigurationsdateien. Jede Komponente ist dokumentiert, modular erweiterbar und einzeln testbar[9].
7. **Lizenz (rechtlicher Rahmen)**
   Das Projekt steht unter der offenen MIT-Lizenz[10], was die freie Nutzung, Modifikation und Weiterverbreitung der Software unter Einhaltung weniger restriktiver Bedingungen erlaubt.
8. **Dokumentation und Metadaten**
   Jede modulare Einheit ist ausführlich dokumentiert (README.md, PDF-Berichte), mit Hinweisen zur Installation, Nutzung und Nachvollziehbarkeit für Entwickler und Nutzer.

Tabelle: Komponentenübersicht des Projekts

Komponente / Modul	Beschreibung
README.md	Projektbeschreibung, Installationshinweise
LICENSE	Rechtliche Nutzung (MIT-Lizenz)
main.py	Hauptmodul, zentrale Logik der Softwareimplementierung
utils/	Sammlung von Hilfsfunktionen und unterstützenden Skripten
tests/	Testfälle zur (automatischen) Validierung der Softwarefunktionalität
.gitignore	Spezifiziert Dateien/Ordner, die von Git ignoriert werden
Dokumentationsdateien	PDF-Berichte, Markdown-Dateien zu Theorie, Anwendung und Architektur

Zenodo DOI	https://doi.org/10.5281/zenodo.17298463 - Versionsarchiv, dauerhafte Referenz
GitHub Repository	https://github.com/denniskurzer89-cyber/FKT-V4.1-Audit-Release/tree/main
Version V4.1	Aktueller auditierbarer Release mit vollständiger Dokumentation

Die oben stehende Tabelle verdeutlicht, wie die einzelnen Teile des Projekts zusammenwirken. Besonders hervorzuheben ist, dass alle Komponenten auf offene Standards setzen - von der Lizenz (MIT) bis zur versionierten Langzeitarchivierung (Zenodo mit DOI). Dies steht im Einklang mit internationalen Empfehlungen zur Entwicklung von Forschungssoftware und der nachhaltigen Sicherung digitaler Forschungsobjekte[6][11].

Ergebnisse und Resultate

Das Projekt „FKT-V4.1-Audit-Release“ hat mehrere relevante Ergebnisse hervorgebracht, die sowohl der wissenschaftlichen Gemeinschaft als auch Softwareentwicklern und FKT-interessierten Kreisen zugute kommen.

1. Vollständiges, auditierbares Softwarepaket

Der veröffentlichte Release (V4.1) auf GitHub und Zenodo stellt ein vollständig geprüftes Softwarepaket dar, das:

- Die wesentlichen Prinzipien der Fraktalen Kausalen Theorie in ausführbarer Form umsetzt.
- Modular aufgebaut ist, was Wartbarkeit und Erweiterbarkeit sicherstellt.
- Alle Quelltexte, ausführlichen Dokumentationen und Testskripte enthält.
- Einen offenen Review, Feedback und künftige Weiterentwicklungen ermöglicht.

Die klare Gliederung und die in den Releases enthaltenen Testfälle sorgen zudem für eine Verifikationsbasis, sodass die Richtigkeit und Funktionalität der Implementierung nachvollziehbar geprüft werden kann.

2. Versionierte Archivierung mit DOI

Die Veröffentlichung auf Zenodo[3], versehen mit einem einzigartigen DOI, garantiert Langzeitverfügbarkeit und rechtssichere Zitierbarkeit der dargestellten Ergebnisse. Die Auditversion erhält damit eine klare, unveränderliche Identität im wissenschaftlichen Diskurs.

3. Wissenschaftliche und praktische Anschlussfähigkeit

Die Projektdokumentationen und Beispiele belegen Anwendungen der FKT in unterschiedlichen Bereichen. Es werden nicht nur mathematische und philosophische Grundlagen dokumentiert, sondern auch Visualisierungen und Simulationen bereitgestellt. Dies erleichtert anderen Forschenden den Zugang zur Thematik und schafft eine wertvolle Basis für weiterführende Arbeiten.

4. Offene Lizenz und nachhaltige Nutzbarkeit

Durch die Wahl der MIT-Lizenz ist der Quellcode ohne Einschränkungen für Dritte nutzbar und darf weiterentwickelt oder in andere Projekte integriert werden[10]. Die rechtlichen Rahmenbedingungen sind damit klar und fördern die langfristige Nutzung im Sinne der Open Science.

5. Nachvollziehbare Softwarearchitektur

Die Architektur ist so gestaltet und dokumentiert, dass unabhängige Auditoren oder Entwickler die Funktionsweise, Schnittstellen und Zusammensetzung der Module nachvollziehen können[12]. Dies erleichtert externe Prüfungen, Updates und die Erweiterung um neue Funktionen oder Module.

6. Beitrag zur Methodendiskussion in der Informatik

Das Projekt dient als vorbildhaftes Beispiel für die Umsetzung moderner Forschungspraktiken: Kombination von iterativer Entwicklung, offener Auditierung, maschinenlesbarer Dokumentation und Langzeitarchivierung - dies sind Aspekte, die zunehmend auch in anderen Bereichen wissenschaftlicher Softwareentwicklung eingefordert werden[13].

Analyse des GitHub-Repositories

Das GitHub-Repository von „FKT-V4.1-Audit-Release“ ist klar strukturiert und erfüllt nicht nur die Anforderungen an kollaborative Softwareentwicklung, sondern bildet auch die Grundlage für die gesamte Projektauditierung.

Strukturelle Merkmale

Das Repository unter https://github.com/denniskurzer89-cyber/FKT-V4.1-Audit-Release/tree/main folgt etablierte Konventionen der Open-Source-Community. Das Verzeichnis enthält:

- README.md: Ausführliche Beschreibung des Projekts, Installationshinweise und Nutzungsszenarien.
- LICENSE: Offenlegung der Lizenzbedingungen (MIT).
- Hauptmodul(e) und Subordner (main.py, utils/, tests/).
- PDF-Berichte zur Theorie und zum Projektfortschritt auf Deutsch und Englisch.
- .gitignore zur Steuerung der in der Projekthistorie nachverfolgten Dateien.
- Keine Releases im klassischen GitHub-Releases-Reiter; die Release-Strategie ist an die Veröffentlichung auf Zenodo gekoppelt.

Dokumentation und Audit-Tauglichkeit

Die Dokumentation ist als fester Bestandteil jeder Version vorgesehen. Die README verschafft einen ersten Einstieg, während die PDF-Berichte fundierte Erklärungen und Hintergründe liefern. Die MIT-Lizenz, die allen Kopien beiliegt, regelt klar die rechtlichen Aspekte der Nutzung. Die Einhaltung von Best Practices (Commit-Historie, nachvollziehbare Ordnerstruktur, umfangreiche Projektdokumentation) und die Integration externer Tools wie Zenodo erleichtern die externe Auditierung und Evaluation der Softwarequalität.

Untersuchung des Zenodo-Datensatzes

Der Zenodo-Datensatz (https://doi.org/10.5281/zenodo.17298463) stellt die DOI-registrierte, langfristig archivierte Version des Projektes dar[11].

Aufbau und Funktionen

- **DOI und Metadaten**: Der Datensatz verfügt über einen citable DOI und ist mit ausführlichen Metadaten ausgestattet (Titel, Autor(en), Beschreibung, Lizenz, Schlagworte, Version, Community-Zuordnung). Dies gewährleistet Auffindbarkeit und wissenschaftliche Nachverfolgbarkeit.
- **Dateien**: Der Datensatz enthält die strukturierten Markdown- und PDF-Dateien, Lizenz, Quellcode und sonstige Softwarekomponenten.
- **Dateiformate**: Zenodo akzeptiert eine Vielzahl von Dateiformaten (z.B. .md, .pdf, .csv, .zip) und stellt eine Vorschaufunktion für viele Typen bereit; dies zahlt direkt auf die Open-Science-Prinzipien ein[14].
- **Versionierung**: Jede Änderung des aktuellen Datensatzes wird als neue Version veröffentlicht und mit eigenem DOI versehen; ältere Versionen bleiben zugänglich und zitierbar.
- **Langzeithaltbarkeit**: Zenodo als Plattform, vom CERN betrieben, garantiert die Langzeitarchivierung unabhängig von der fortlaufenden Entwicklung auf GitHub.

Relevanz und Best Practices

Mit der Nutzung von Zenodo folgt das Projekt aktuellen Best Practices zur Archivierung und Veröffentlichung von Forschungsdaten und -software[11]. Durch den DOI können nicht nur die Software und Metadaten selbst, sondern auch weiterführende Publikationen und Anwendungen dauerhaft und eindeutig referenziert werden.
Die Referenzierung via DOI hebt das Projekt auf eine dem wissenschaftlichen Standard entsprechende Ebene; es wird zitierfähig und kann in Datenbanken, Verzeichnisse und Publikationen aufgenommen werden.

Hintergrund: Fraktale Kausale Theorie (FKT)

Um die Tragweite des Projekts zu verstehen, ist ein Einblick in die Fraktale Kausale Theorie notwendig.

Theoretische Grundlagen

Die FKT beschäftigt sich mit der Beschreibung und Analyse von Systemen, deren Kausalstrukturen eine fraktale, selbstähnliche (oft rekursive) Organisation aufweisen. Zentrale mathematische Konzepte sind:

- **Iteration**: Wiederholte Anwendung einer Funktion auf ihr eigenes Ergebnis; Grundlage für viele fraktale Prozesse.
- **Selbstähnlichkeit**: Strukturen, die auf verschiedenen Skalen gleiche oder ähnliche Muster aufweisen.
- **Rückkopplung**: Dynamische Systeme, deren Ausgang wieder als Eingabe dient - typisch für chaotische, fraktale oder komplexe Systeme.
- **Hausdorff-Dimension**: Verallgemeinerter Dimensionsbegriff zur mathematischen Charakterisierung fraktaler Objekte.

Bekannte Beispiele umfassen die Mandelbrot-Menge, Julia-Mengen, das Sierpinski-Dreieck, Zeltabbildung und verschiedene seltsame Attraktoren wie den Lorenz- oder Hénon-Attraktor, die in der Software visualisiert und analysiert werden können[8][7].

Interdisziplinäre Bedeutung

Die Relevanz der FKT liegt in ihrer Anwendbarkeit auf verschiedenste Bereiche: von der Vorhersage komplexer Naturphänomene und Wirtschaftsdynamiken über biologische Wachstumsprozesse bis hin zur Künstlichen Intelligenz und Netzwerkanalyse. Sie verbindet mathematische, philosophische, physikalische und informationstheoretische Perspektiven. Das Projekt stellt somit eine Brücke zwischen theoretischer Forschung und praktischer, softwaregestützter Nutzung her.

Software-Architektur und Modulübersicht

Eine zentrale Stärke des Projekts liegt in der modular aufgebauten Softwarearchitektur, die nach modernen Gesichtspunkten (u. a. nach den Empfehlungen von IEC 62304 und aktuellen Best Practices im Forschungssoftwarebereich) gestaltet wurde[12].

Bausteine und Struktur

- **Hauptmodul(e)**: Implementiert die Kernalgorithmen zur Analyse, Simulation und Visualisierung fraktaler, kausaler Strukturen.
- **Hilfsfunktionen/Utils**: Bieten wiederverwendbare Funktionen (z.B. mathematische Tools, Datenkonvertierung, Visualisierungswerkzeuge).

- **Tests**: Sichern die Funktionsfähigkeit und ermöglichen kontinuierliche Validierung.
- **Konfigurations- und Metadaten**: Regeln grundlegende Eigenschaften (z.B. Version, Autorenschaft, Lizenz, verwendete Datentypen).
- **Externe Schnittstellen**: Ermöglichen ggf. die Anbindung an weitere Analyse- und Visualisierungstools oder die Integration in andere Projekte.

Die klare Trennung zwischen Hauptmodul, unterstützenden Modulen, Testskripten und Dokumentation erleichtert die Einarbeitung, externe Auditierung und Weiterentwicklung.

Dokumentation und Rückverfolgbarkeit

Die Erfüllung der Forderungen an eine moderne Softwarearchitektur wird in der Dokumentation sichtbar: Jede Komponente ist beschrieben, Schnittstellen zu anderen Modulen sind erläutert, die Implementierungsdetails nachvollziehbar dokumentiert. Die vollständige Rückverfolgbarkeit ist auch für Audits (z.B. nach ISO-Norm) essenziell[12].

Lizenz und rechtliche Rahmenbedingungen

Das Projekt ist unter der MIT-Lizenz veröffentlicht, einer der populärsten und verständlichsten Open-Source-Lizenzen. Dies hat weitreichende Implikationen für Nutzer, Entwickler und Forschungsgemeinschaft[10].

Wesentliche Lizenzmerkmale

- **Nutzung ohne Einschränkung**: Die Software darf ohne Einschränkung für beliebige Zwecke verwendet, bearbeitet, kopiert, gemerged, veröffentlicht, verteilt, sublicenziert und verkauft werden.
- **Lizenzhinweis**: In allen Kopien oder signifikanten Teilen der Software muss der Urheberrechtshinweis und die Lizenz enthalten sein.
- **Haftungsausschluss und Gewährleistungsausschluss**: Die Software wird „as is“ ohne jegliche Garantie bereitgestellt.
- **Keine Copyleft-Bedingung**: Anders als die GPL verpflichtet die MIT-Lizenz nicht dazu, abgeleitete Produkte ebenfalls unter der MIT-Lizenz zu veröffentlichen.

Bedeutung und Konsequenzen

Die Wahl der MIT-Lizenz entspricht dem Open-Science-Gedanken des Projekts und fördert eine niedrige Einstiegshürde für Weiterentwicklung, Forks und Integration in andere Projekte und Produkte.
Die klaren rechtlichen Rahmenbedingungen gewährleisten, dass keine Verletzung von Urheberrechten droht, sofern die Bedingungen eingehalten werden. Offene Lizenzen wie die MIT-Lizenz sind insbesondere im akademischen Bereich Standard und werden von großen Open-Source-Plattformen wie GitHub, Zenodo und anderen anerkannt und unterstützt[6].

Datenformate und Dokumentation

Die im Projekt verwendeten Datenformate und Dokumentationsstrategien sind konsequent auf Offenheit, Lesbarkeit und Kompatibilität ausgelegt:

- **Markdown- und PDF-Formate**: Für projektspezifische Dokumente, Berichte und Theoriekapitel werden markdown (lesbar im Browser/editierbar in vielen Editoren) und PDF (für den Druck/E-Mail) genutzt.
- **Quellcode in Python** (und ggf. weiteren, dokumentierten Sprachen), die in der Community weit verbreitet sind.
- **Lizenzinformationen als eigene Datei**.
- **Gitignore-Datei**: Regelt, welche Dateien und Ordner von der Versionierung ausgeschlossen werden.
- **Metadaten (z.B. DataCite-Schema)** für die DOI-Archivierung.

Die vollständige Dokumentation (READMEs, Berichtspdfs, Inline-Codekommentare) erleichtert Einarbeitung, Nachvollziehbarkeit und maschinelle Weiterverarbeitung.

Veröffentlichungsstrategie und DOI-Nutzung

Ein wesentlicher Bestandteil der Nachhaltigkeit und wissenschaftlichen Anerkennung des Projekts ist die Veröffentlichungsstrategie:

- **Kombinierte Nutzung von GitHub und Zenodo**: GitHub dient als kollaborative Entwicklungsplattform, Zenodo als Archiv mit DOI-Zuweisung für einzelne Versionen oder Releases[11].
- **Jeder veröffentlichte Release auf Zenodo erhält einen Persistent Identifier (DOI)**: Dies macht jede Version eindeutig referenzierbar und unterstützt die Nachvollziehbarkeit über lange Zeiträume hinweg.
- **Langzeitarchivierung**: Über Zenodo und Plattformen wie re3data.org ist sichergestellt, dass relevante Versionen auch nach Ablauf individueller Wartungszyklen zugänglich bleiben.

Die DOI-Integration entspricht etablierten Best Practices für den Umgang mit Forschungsdaten und Open-Source-Software[11].

Eingesetzte Werkzeuge und Technologien

Das Projekt bedient sich einer aufeinander abgestimmten Werkzeuge und Technologien, die den aktuellen Stand der Softwareentwicklung reflektieren:

- **Versionskontrolle via Git/GitHub**: Für kollaborative Entwicklung, Rückverfolgbarkeit und verteiltes Arbeiten ist GitHub die zentrale Schaltstelle.
- **Archivierung über Zenodo**: Die Plattform erlaubt es, Releases über DOI zu archivieren und dauerhaft zugänglich zu machen[6].

- **Python als Hauptimplementierungssprache**: Aufgrund der universellen Verbreitung, der Vielzahl an Bibliotheken und der guten Lesbarkeit.
- **Dateistrukturierung nach Common-Practice-Standards**: Trennung von Quellcode, Tests, Konfiguration, Dokumentation.
- **Testautomatisierung**: Über Skripte und Testfälle, die in das Versionierungssystem eingebunden sind.
- **Markdown für Dokumentation**: Ein industriestandardisiertes Textformat für schnelle Bearbeitung und Plattformunabhängigkeit.
- **PDF für strukturierte Berichte**: Für die Präsentation komplexerer Inhalte, Nutzung in Peer Reviews und Langzeitarchivierung.
- **Open-Source-Lizenzierung (MIT)**: Rechtssicherheit und Förderung der offenen Weiterentwicklung.

Audit-Methoden und Validierungskriterien

Bei der Auditierung wissenschaftlicher Software kommt einer Vielzahl von Methoden Bedeutung zu. Zu den wichtigsten gehören:

- **Dokumentenprüfung**: Prüfung der Übereinstimmung von Quellcode, Dokumentation, Lizenzen und Metadaten mit den Projektzielen und Standards.
- **Stichprobenprüfungen und Testautomatisierung**: Verifikation der Implementierung anhand vordefinierter Testfälle.
- **Daten- und Ergebnisanalyse**: Überprüfung der Resultate auf Korrektheit, Plausibilität und Nachvollziehbarkeit.
- **Checklisten und Peer Review**: Nutzung von standardisierten Prüfprotokollen und/oder Peer-Review-Verfahren für unabhängige Beurteilung[5].

Im vorliegenden Projekt spiegeln sich diese Methoden in der Bereitstellung aller relevanten Artefakte (Code, Berichte, Lizenz), der klaren Versionierung und der Integration von Testcases wider.

Versionierung und Release-Management

Die zentrale Bedeutung der Versionierung ist offensichtlich:

- **GitHub als Versionierungsplattform**: Unterstützt Branching, Forking und Rollback auf frühere Versionen.
- **Release-Tagging und DOI für Releases**: Jeder stabile Zustand (Release, z.B. V4.1) wird ausgezeichnet und auf Zenodo veröffentlicht.
- **Change-Logs und Release-Notes**: Sichern die Rückverfolgbarkeit von Änderungen; dies ist unerlässlich für Auditierbarkeit und Qualitätssicherung.

Durch diese Mechanismen wird gewährleistet, dass die Entwicklung zu jedem Zeitpunkt reproduzierbar, dokumentiert und nachvollziehbar bleibt. Künftige Erweiterungen können sauber von den bestehenden Releases abgegrenzt und in ihre Veränderungen eingeordnet werden.

Schlussfolgerungen und Ausblick

Das Projekt „FKT-V4.1-Audit-Release“ stellt ein gelungenes Beispiel für die Synthese von moderner Softwareentwicklung, theoretisch-mathematischer Modellbildung und Open-Science-Prinzipien dar. Die wichtigsten Errungenschaften lassen sich wie folgt zusammenfassen:

- **Implementierung und Operationalisierung der FKT**: Die Theorie wurde nicht nur publiziert, sondern in eine robuste, nachvollziehbare und wartbare Softwarearchitektur überführt, was für den wissenschaftlichen Diskurs essenziell ist.
- **Auditierbarkeit und Nachhaltigkeit**: Die Kombination aus GitHub-Versionierung und Zenodo-DOI-Archivierung macht das Projekt transparent, überprüfbar und dauerhaft referenzierbar, im Sinne bester wissenschaftlicher Praxis.
- **Offene Lizenzierung und Erweiterbarkeit**: Die Verwendung der MIT-Lizenz fördert Kollaboration, Adaptierbarkeit und robustes Community-Wachstum.
- **Interdisziplinäre Anschlussfähigkeit**: Das Projekt leistet einen Beitrag zur Übertragung von FKT-Konzepten auf andere Forschungsbereiche, was insbesondere durch die umfangreiche und fundierte Dokumentation ermöglicht wird.

Zukünftige Entwicklungsperspektiven:

- **Erweiterbarkeit der Module**: Integration von weiteren mathematischen Modellen, Visualisierungstechniken und Anwendungsfällen.
- **Einbindung weiterer Validierungskriterien**: Umfassendere Testautomatisierung, Peer-Review-Verfahren, Integration externer Datensätze und Benchmarks.
- **Verstärkte Community-Einbindung**: Code-Sprints, Öffnung für Community-Forks, strukturierte Feedback-Kanäle.
- **Verwendung der FKT in domänenspezifischen Anwendungen**: Beispielsweise, Analyse komplexer Netzwerke, Simulation umwelttechnischer Systeme oder Optimierung von Machine-Learning-Algorithmen.

Das Projekt markiert somit einen wichtigen Meilenstein in der Verbindung wissenschaftlicher Theoriebildung, methodischer Softwareentwicklung und nachhaltiger, auditierbarer Forschungspraxis - beispielhaft für ein neues Zeitalter der interdisziplinären Kollaboration und Open Science.

Webquellen (Auswahl zur Vertiefung und zu Hintergrundinformationen):

- GitHub: https://github.com/denniskurzer89-cyber/FKT-V4.1-Audit-Release/tree/main
- Zenodo: https://doi.org/10.5281/zenodo.17298463

- Hintergrundinfos zu Lizenzierung, Softwarearchitektur, Auditmethoden und FKT siehe die mit , [2], [1], [8], [7], [9], [12], [10], [3], [4], , [5] referenzierten Webressourcen.

References (14)

1. *Chaos und Fraktale: Einsichten, Begriffe und Beispiele - Ein Überblick*. https://jlupub.ub.uni-giessen.de/bitstreams/bab2f911-ab09-41c2-9bab-9196575e74df/download
2. *Inhaltsübersich - Heidelberg University*. https://www.psychologie.uni-heidelberg.de/ae/allg/enzykl_denken/Enz_03_Kausal.pdf
3. *Leitfaden: Upload von FD auf Zenodo - Romanistik-Blog*. https://blog.fid-romanistik.de/wp-content/uploads/2020/05/Anleitung_Zenodo_Langfassung.pdf
4. *Audit - Deutsche Gesellschaft für Qualität*. https://www.dgq.de/audit/
5. *DataCite Best Practice Guide - fdm-bayern.org*. https://www.fdm-bayern.org/tool-material/datacite-best-practice-guide/
6. *M.1 - Fraktale in Theorie und Anwendung*. https://studylibde.com/doc/16831601/m.1-%E2%80%93-fraktale-in-theorie-und-anwendung
7. *einf_dyn_sys.dvi - uni-hamburg.de*. https://www.math.uni-hamburg.de/home/lauterbach/scripts/eds07/chap4.pdf
8. *Software-Architektur Dokumentation*. https://www.johner-institut.de/blog/iec-62304-medizinische-software/software-architektur-dokumentation/
9. *Welche Bedingungen gelten für die Nutzung von Open Source … - adjuga*. https://www.adjuga.de/veroeffentlichungen/welche-bedingungen-gelten-fuer-open-source-software/
10. *Zenodo* . https://openscience.cern/zenodo
11. *Die Gestaltung einer unternehmensweiten, flexiblen Software-Architektur …*. https://rd.springer.com/content/pdf/10.1007/3-7908-1612-4_6.pdf
12. *Forschungsberichte - TIB*. https://www.tib.eu/de/publizieren-archivieren/forschungsberichte
13. *FID-Romanistik: Zenodo*. https://www.fid-romanistik.de/forschungsdaten/arbeit-mit-forschungsdaten/sichern-und-publizieren-von-forschungsdaten/zenodo/
14. *Auditmethoden [2025] QUALITY©*. https://www.quality.de/lexikon/auditmethoden/