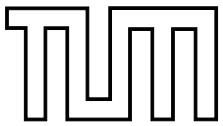
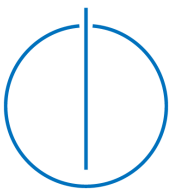


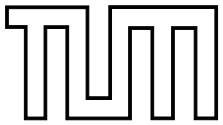
FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

Neukonzeption des Tutortools der Fakultät für Informatik an der TU München

Philipp Reindl-Spanner





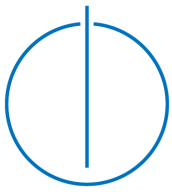
FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

Neukonzeption des Tutortools der Fakultät für Informatik an der TU München

Reconception of the tutor tool of the TUM Department of Informatics

Bearbeiter:	Philipp Reindl-Spanner
Aufgabensteller:	Prof. Dr. Helmut Krcmar
Betreuer:	Dr. Robert Heininger
Abgabedatum:	15.05.2021



Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Garching b. München, den 15.05.2021 _____

Ort, Datum Unterschrift

Zusammenfassung

Begleitend zu den Lehrveranstaltungen werden an der Fakultät für Informatik der TU München Tutorien durch Studierende angeboten. Für das Einstellungsverfahren sowie die Verwaltung der Tutorverträge steht den Mitarbeitern des Tutorbüros das sogenannte „Tutortool“ zur Verfügung. Aufgrund der jahrelangen evolutionären Entwicklung dieses Tutortools und der hiermit einhergegangenen unkontrollierten Softwareevolution ist das „alte“ Tutortool (in seiner zu Beginn dieser Masterarbeit verwendeten Version) strukturell und funktionell in einem schlechten Zustand.

Die vorliegende Masterarbeit hat sich daher zum Ziel gesetzt, das Tutortool neu zu konzeptionieren und in diesem Zuge eine Neuimplementierung zu erarbeiten. Der Arbeitsprozess hat sich hierbei an der Design Science Methodik orientiert. Zunächst ist eine entsprechende Literaturrecherche mit dem Thema „Softwareevolution“ durchgeführt worden, deren Fokus vor allem auf zwei zentralen Schwerpunkten lag. Zum einen sind Veröffentlichungen betreffend Methoden und Regeln, durch deren Anwendung bereits in den frühen Phasen des Software Lifecycles der Arbeitsaufwand in der späteren Phase der Softwarewartung reduziert werden kann, betrachtet worden. Zum anderen sind Veröffentlichungen bezüglich Methoden und Regeln betrachtet worden, durch deren Anwendung das Reengineering von Altsystemen unterstützt werden kann. Im nächsten Schritt sind die relevanten Prozesse des Tutorbetriebs, die in Zusammenhang mit dem Tutortool stehen, erhoben und mit Hilfe von BPMN 2.0 abgebildet worden. Prozesse, die Probleme aufwiesen, sind im Nachgang identifiziert und optimiert worden. Auf Basis dieser optimierten Prozesse sind anschließend Funktionalitäten für den Schritt der Neuimplementierung formuliert worden. Im letzten Schritt sind diese Funktionalitäten in einer MERN-Stack basierten Webapplikation umgesetzt und iterativ mit den intendierten Nutzergruppen getestet worden.

Durch die Neukonzeption und deren Umsetzung als Webapplikation konnte die unkontrollierte Softwareevolution des Tutortools gestoppt werden. Die Ergebnisse der Nutzertests der neu entwickelten Applikation sind mit einem durchschnittlichen System Usability Scale Wert (kurz: SUS-Wert) von 96,5 deutlich überdurchschnittlich ausgefallen und haben den erzielten SUS-Wert der alten Anwendung (55) übertroffen.

Stichworte: Softwareevolution, Software Reengineering, Prozessanalyse, Software Engineering, Design Science

Summary

In addition to regular courses, the Faculty of Computer Science at the TU Munich offers tutorials conducted by students. For the recruitment process as well as the general administration of tutor contracts, the staff of the tutoring office has the so-called "Tutortool" at their disposal. Due to the evolutionary development of the Tutortool over the years as well as the uncontrolled software evolution that went along with it, the "old" Tutortool (in its version used at the beginning of the writing process of the present Master's thesis) is structurally and functionally in a poor state.

The present Master's thesis is therefore focused on reconceptualizing the Tutortool and simultaneously developing a new implementation. The work process was based on the Design Science Methodology. Initially, a respective literature research concerning the topic "software evolution" was carried out while mainly focusing on two key aspects. On the one hand, publications concerning methods and rules by whose application (in the early phases of the software lifecycles) the work expenditure in the later phase of the software maintenance may be reduced were regarded. On the other hand, publications concerning methods and rules by whose application the reengineering of old systems may be supported were considered as well. In the next step, the relevant processes of tutor operation which are directly related to the Tutortool were pointed out and visualized with the help of BPMN 2.0. Processes that displayed shortcomings were subsequently identified and optimized. Based on these optimized processes, functionalities were then formulated for the new implementation. In the final step, these functionalities were implemented in a MERN-stack based web application and iteratively tested with the intended user groups.

Through the reconception and its implementation as a web application, the uncontrolled software evolution of the Tutortool was concluded. The results of the user tests of the newly developed application were clearly above average with an average System Usability Scale value (SUS value) of 96.5 and exceeded the SUS value of the old application (55).

Keywords: software evolution, software reengineering, process analysis, software engineering, design science

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACM	Association for Computing Machinery
API	Appliance Programming Interface
BPMN	Business Process Model and Notation
EV	Einstellungsvorschlag
IEEE	Institute of Electrical and Electronics Engineers Inc.
IS	Informationssystem
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
LRZ	Leibnitz-Rechenzentrum
RBG	Rechnerbetriebsgruppe
SDLC	Software Development Life Cycle
SPA	Single Page Application
SUS	System Usabiliy Scale
TUM	Technische Universität München

1 Einleitung

Die folgenden Abschnitte sollen einen kurzen Überblick über die hier vorliegende Arbeit geben. Dabei wird zunächst die Problemstellung erörtert und das Thema motiviert, gefolgt von der Zielsetzung dieser Arbeit. Dieser Einleitungsteil wird abgeschlossen durch das Aufzeigen der Struktur der Arbeit und der Vorstellung des methodischen Vorgehens.

1.1 Problemstellung

Im Rahmen der Exzellenzinitiative der Technischen Universität München (TUM) stehen der Fakultät für Informatik zusätzliche Gelder zur Verfügung, um Tutorien für Studierende anzubieten. Hierdurch können jedes Semester für über 30 Veranstaltungen, z.B. Übungen in Kleingruppen, Hausaufgabenkorrekturen und Repetitorien, angeboten werden. Da diese Aufgaben überwiegend von studentischen Hilfskräften übernommen werden, muss für jedes Semester eine Vielzahl an Arbeitsverträgen vorbereitet und erstellt werden. Hierdurch entsteht ein hoher organisatorischer Aufwand; gleichzeitig besteht aber auch ein hohes Standardisierungspotenzial. Daher wurde vor einigen Jahren zur Unterstützung eine webbasierte Plattform – das sogenannte „Tutortool“ – geschaffen und ist insbesondere unter Berücksichtigung der Mengengerüste inzwischen unverzichtbar geworden.

Das Tutortool ist eine Webplattform, die von den Mitarbeitern des Tutorbetriebs, den Studierenden, sowie den Übungsleitern der Veranstaltungen genutzt wird, um die Einstellungen von Studierenden als studentische Tutoren zu unterstützen. Die Entwicklung des Tutortools wurde 2009 von der TUM School of Education begonnen und ist in einer im Laufe der Jahre stetig weiterentwickelten Form bis heute im Einsatz. Durch meine Anstellung als studentische Hilfskraft mit der primären Aufgabe der technischen Betreuung des Tutortools konnte ich mich in den letzten Monaten intensiv mit diesem befassen. Hierbei stellte ich fest, dass die nicht gesteuerte, evolutionäre Entwicklung des Tutortools einige Probleme mit sich bringt.

Die alte Version des Tutortools beinhaltet Funktionalitäten, die vom Verwaltungsteam nicht mehr genutzt werden. Ein Beispiel hierfür sind die bis 2018 üblichen Hospitationen, für welche im Tutortool bis heute eine ungenutzte Organisationsmaske enthalten ist. Solche nicht entfernten, überflüssigen Funktionalitäten machen das System nicht nur von einem technischen Standpunkt aus komplexer, sondern vor allem für Benutzer im täglichen Gebrauch unübersichtlich.

Die zu Beginn der Entwicklung verwendeten Technologien (wie beispielsweise Java 8) gelten heute als veraltet, wurden bei der Weiterentwicklung des Tutortools jedoch nicht durch neuere Technologien ersetzt. Hierdurch wird die Plattform nicht nur in ihrem Nutzererlebnis und der Usability deutlich eingeschränkt, sondern es können gleichzeitig signifikante Sicherheitslücken entstehen. Im derzeitigen Betrieb führen diese veralteten Technologien zudem zu einer Instabilität des Systems, wodurch ein deutlich erhöhter Backup- und Wartungsaufwand durch die technische Betreuung des Tutortools erforderlich ist. Hinzu kommt, dass der Code in der alten Version wenig bis gar nicht dokumentiert ist, wodurch neue und notwendige Änderungen nur durch großen Aufwand zu bewerkstelligen sind.

Ein weiterer Kritikpunkt an der alten Version des Tutortools ist die Datenbank. Diese ist durch das stetige Hinzufügen von Funktionalitäten in ihrer Struktur unübersichtlich. Da es die Implementierung des Tutortools erfordert, öfters Daten direkt in der Datenbank zu ändern (beispielsweise das direkte Einfügen neuer Dokumente), beeinträchtigt diese Unübersichtlichkeit die Verwaltung stark. Eine weitere Komplikation, die hierdurch auftritt, ist die äußerst zeitaufwändige Ursachenfindung im Falle von Problemen.

All diese Punkte spiegeln die zunächst in Lehman (1980, S. 1067 f.) definierten Gesetze zur Software Evolution wieder, die dann später in Lehman, Ramil, Wernick, Perry, und Turski (1997) erneut aufgegriffen wurden. Diese Gesetze beschreiben Prozesse in Bezug auf Softwareentwicklung, die zum einen stetige Weiterentwicklung fordern, zum anderen jedoch durch diese Weiterentwicklung die Qualität der Software verschlechtern. Bezugnehmend auf das Tutortool können hier vor allem die Gesetze eins, zwei, sechs und sieben (Lehman et al., 1997, S. 2) hervorgehoben werden[1]. Diese besagen, dass das System ständig weiterentwickelt werden muss, da es sonst nutzlos wird, sich die Komplexität eines Softwaresystems mit steigender Lebensdauer (exponentiell) entwickelt, das System einem ständigen Wachstum unterliegt (steigende Anzahl an funktionellen Inhalten) und die Qualität von Softwaresystemen mit der Zeit rückläufig wird. Lehman beschreibt, dass Software, die einem evolutionären Wachstum unterzogen wird, viel Pflege und Wartung benötigt. Ludewig und Lichter (2013) fassen diesen Prozess so zusammen, dass Software an veränderte Anforderungen und Umgebungsbedingungen angepasst werden muss. Mit dieser Evolution ist aber unvermeidlich ein Qualitätsverlust verbunden. Wenn diese Anpassungen jedoch nicht stattfinden wird die Software mit der Zeit nutzlos (Ludewig & Lichter, 2013, S. 489 ff.). Bei solchen Anpassungen wird in der Fachliteratur von Software-Wartung gesprochen. In der alten Version des Tutortools ist dieser Pflege- und Wartungsaufwand unverhältnismäßig hoch, sodass bei Veränderungen nicht mehr sichergestellt werden kann, dass der Nutzen einer Veränderung die Kosten übersteigt.

Auf Basis dieser Faktoren nennen Ludewig und Lichter (2013) als einen möglichen Schritt die Neuentwicklung der Software (S. 489 ff.). Hierbei gilt jedoch besondere Vorsicht, da das neue System zuverlässig arbeiten und das alte System im gesamten Funktionsumfang ersetzen muss. Bezogen auf das Tutortool kann durch eine Neuentwicklung auf Basis aktueller Webtechnologien sowohl das Benutzererlebnis gesteigert als auch der Wartungsaufwand verringert werden. Ebenso können potenzielle (eventuell unbekannte) Sicherheitslücken durch veraltete Technologien geschlossen werden. Letzteres ist besonders hervorzuheben, da die Privatheit der personenbezogenen Daten der Tutoren und Übungsleiter zu jedem Zeitpunkt gewährleistet werden muss.

[1] Hierbei ist zu erwähnen, dass in der Praxis normalerweise nur die ersten beiden Gesetze Anwendung finden (siehe Kapitel 2.1). Der Vollständigkeit halber werden hier dennoch alle theoretisch anwendbaren Gesetze aufgezählt.

1.2 Zielsetzung

Das primäre Ziel dieser Masterarbeit ist es, zunächst die Prozesse und Funktionalitäten des alten Tutortools zu ermitteln, diese falls möglich zu optimieren und auf dieser Basis eine Neuentwicklung für das Tutortool durch die Verwendung aktueller Technologien zu erstellen. Der Fokus liegt hierbei auf der Umsetzung der erhobenen Funktionalitäten in mehreren Iterationen durch einen Prototyp der Applikation. Am Ende jeder Iteration wird die zu diesem Zeitpunkt aktuelle Version mit Personen aus den Nutzergruppen getestet. Die Ergebnisse dieser Tests werden daraufhin weiterführend verwendet, um die nächste Iteration des Prototyps zu erstellen. Das methodische Vorgehen orientiert sich dabei an der der Design Science Methodik (Hevner et al., 2004). Das methodische Vorgehen wird genauer in Abschnitt 1.4 erläutert.

Die Realisierung dieser Arbeit erfolgt dabei anhand der folgenden vier Forschungsfragen:

- Wie ist der aktuelle Stand der Literatur zum Thema Softwareevolution?

Für die Beantwortung dieser Frage wird im Rahmen eines Rigor Cycles[2] eine Literaturrecherche nach vom Brocke et al. (2009) durchgeführt. Diese Literaturrecherche befasst sich dabei vor allem mit zwei Aspekten der Softwareevolution. Zum einen werden Veröffentlichungen herausgearbeitet, die Methoden und Regeln beinhalten, die in der Phase der Softwareimplementierung angewendet werden können, um die spätere Softwarewartung zu erleichtern. Zum anderen werden Veröffentlichungen betrachtet, die Methoden und Regeln vorstellen, die den Schritt des Reengineerings vereinfachen.

- Wie lassen sich die aktuellen Prozesse innerhalb des Tutorbetriebs modellieren, wo treten Probleme auf und wie lassen sich die Prozesse optimieren?

Für die Beantwortung dieser Forschungsfrage sind mehrere Schritte notwendig. Zunächst werden die Nutzergruppen und Prozesse innerhalb des Tutorbetriebs betrachtet, die in Verbindung mit dem Tutortool stehen. Dabei werden zunächst der Gesamtprozess einer Einstellung und die wichtigsten Subprozesse (im Folgenden auch Kernprozesse genannt) modelliert. Hiernach werden die Prozesse optimiert, um diese in der Neukonzeption des Tutortools effizienter umsetzen zu können. Um ein Verständnis für die bestehenden Prozesse zu entwickeln, wurden hierzu vor allem persönliche Gespräche mit den Mitarbeitern des Tutorbüros geführt und Videoaufzeichnungen der wichtigsten Schritte im Arbeitsalltag der Mitarbeiter angefertigt. Anschließend wurden die Prozesse mit Hilfe von BPMN 2.0 (OMG Group, 2011) visualisiert.

- Welche Funktionalitäten muss die Neukonzeption des Tutortools für den Tutorbetrieb bereitstellen, damit sie alle (intendierten) Nutzergruppen optimal unterstützt?

Auf der Basis von Forschungsfrage zwei werden zusätzlich zu den Kernprozessen weitere wichtige Funktionalitäten definiert, die in einer Neukonzeption des Tutortools vorhanden sein müssen, damit dieses jede Nutzergruppe optimal bei der Arbeit unterstützt. Ebenso wird hierbei

[2] Für eine Erklärung der Cycles siehe Kapitel 1.4 - Methodisches Vorgehen.

darauf eingegangen, welche Funktionalitäten, die in der alten Version des Tutortools vorhanden sind, nicht in die Neukonzeption übernommen werden.

- Wie können die erhobenen Funktionalitäten für das Tutortool implementiert werden?

Für diese Forschungsfrage wird eine Neukonzeption des Tutortools in mehreren Design Cycles, die bis zu acht Iterationen beinhalteten, erschaffen. Diese Neukonzeption basiert auf den im Relevance Cycle erhobenen Prozessen und Funktionalitäten. Während der Iterationen werden auf Basis von Nutzertests noch zusätzliche weitere Funktionalitäten ermittelt und der Neukonzeption hinzugefügt.

1.3 Struktur der Arbeit

Der Aufbau der Arbeit gliedert sich in drei große Kapitel, die der Design-Science-Research Methodologie nach Hevner et al. (2004) beziehungsweise Hevner (2007) folgen und in Abschnitt 1.4 im Detail erklärt werden.

Zu Beginn von Kapitel 2 werden wichtige grundlegende Begrifflichkeiten für diese Arbeit vorgestellt und erklärt. Hierbei wird zunächst der Begriff der Softwareevolution vorgestellt und von dem der Softwarewartung abgetrennt. Hierauf folgen die Begriffe der evolutionären Softwareentwicklung und der Begriff der Softwarewartung selbst. Danach wird auf das Software Reengineering und den Software Lifecycle eingegangen. Das Kapitel wird mit einer Erklärung zum Tutorbetrieb abgeschlossen.

Der Rigor Cycle, der in Kapitel 3 beschrieben wird, stellt den ersten großen Abschnitt dieser Arbeit dar. In diesem Cycle wird eine Literaturrecherche zum Thema „Softwareevolution“ durchgeführt, die sich mit der Beantwortung der ersten Forschungsfrage auseinandersetzt.

Kapitel 4 behandelt den Relevance Cycle dieser Arbeit. In diesem Abschnitt werden die Prozesse des alten Tutortools erhoben und die vorgenommenen Optimierungen an diesen vorgestellt. Zudem werden wichtige neue oder optimierte Funktionalitäten vorgestellt, die auf Basis von Befragungen mit den Nutzergruppen und Feedback zum alten Tutortool erarbeitet wurden. Hierin werden vor allem die Forschungsfragen zwei und drei bearbeitet.

Der letzte der drei Zyklen wird in Kapitel 5 thematisiert und befasst sich mit den Design Cycles dieser Arbeit. Während dieser insgesamt vier Design Cycles wird eine Neuimplementierung des Tutortools auf Basis der erarbeiteten Optimierungen erschaffen. Das Kapitel soll dem Leser die durchgeführten Schritte näher bringen und ein Verständnis für die Designentscheidungen schaffen. Abschließend zu jedem Design Cycle werden Nutzertests durchgeführt, um die Benutzerfreundlichkeit des Systems testen zu können.

Zum Abschluss dieser Arbeit werden in Kapitel 6 kurz die wichtigsten Ergebnisse zusammengefasst und es wird erneut auf die in der Einleitung definierten Forschungsfragen eingegangen. Begleitend hierzu wird ein Forschungsausblick und eine Skizzierung des chronologischen Vorgehens für die Integrierung der hier erarbeiteten Neuimplementierung in den realen Arbeitsbetrieb des Tutorbetriebs gegeben.

1.4 Methodisches Vorgehen

Das methodische Vorgehen dieser Arbeit basiert auf der Design-Science-Research Methodik nach Hevner et al. (2004) und folgt dabei dem Drei-Zyklen-Ansatz nach Hevner (2007). In Hevner et al. (2004) werden zunächst sieben Richtlinien[3] präsentiert (Hevner et al., 2004, S. 82 ff.), um Forschung auf Basis der Design Science Methodik durchzuführen. Im Folgenden werden diese Richtlinien kurz vorgestellt, sowie deren Anwendung in dieser Arbeit genauer erläutert.

- Richtlinie 1: Artefakte als Ergebnisse der Forschung

Das Ergebnis von Forschung nach der Design Science Methodik muss ein IT-Artefakt sein. Diese Artefakte können nicht nur die Form einer Applikation (im Original „Instantiation“) gegeben werden, sondern auch in Form von Konstrukten, Modellen oder Methoden, die in der Entwicklung verwendet werden. In dieser Arbeit wird dieses Artefakt durch die Softwareapplikation gegeben, die die Neukonzeption des Tutortools darstellt.

- Richtlinie 2: Problemrelevanz

Das Ziel von Design-Science-Research ist es, Technologie basierte Lösungen für wichtige und relevante (Geschäfts-)Probleme zu erschaffen. Die Problemrelevanz wird in dieser Arbeit durch die Problemstellung im Bereich des Tutorbetriebs gegeben (Abschnitt 1.1), welche durch die Neuimplementierung des Tutortools gelöst werden soll.

- Richtlinie 3: Evaluation des Artefakts

Die Nützlichkeit, die Qualität und die Effizienz des Artefakts müssen auf Basis von anerkannten Evaluierungsmethoden gezeigt werden. Hierzu kommen vor allem während der Design Cycles verschiedene Evaluierungsmethoden zum Einsatz. Um die Benutzerfreundlichkeiten der getesteten Implementierungen erheben und vergleichen zu können, wird der System Usability Scale (SUS) (Brooke, 1996) Fragebogen verwendet. Darüber hinaus werden mit den Mitarbeitern des Tutorbetriebs Messungen durchgeführt, um die Durchlaufzeiten bei verschiedenen im Tutortool abgebildeten Prozessen zu messen und hierdurch die Versionen vergleichen zu können.

- Richtlinie 4: Beitrag der Forschung

Effektive Design-Science Research erzeugt klare und nachvollziehbare Beiträge im Bereich des Artefakts, der Forschungsgrundlagen und/oder den Forschungsmethoden. Diese Arbeit befolgt diese Richtlinie in mehreren Punkten. Zunächst dient die Neuimplementierung des Tutortools als Beispiel für eine erfolgreiche Neukonzeption auf Basis eines durchgeführten Reengineerings eines Altsystems, um die unkontrollierte Softwareevolution des Software-Systems zu stoppen. Zudem stellt die Literaturrecherche eine Synthese aktueller Methoden im Bereich der

[3] Diese können in Abbildung 19 (Anhang A.1) betrachtet werden.

Softwareevolution dar. Durch die Verwendung unterschiedlicher anerkannter Forschungsmethoden ist die Nachvollziehbarkeit der Ergebnisse gegeben (auf die verwendeten Forschungsmethoden wird in Richtlinie 5 und in Kapitel 1.4 genauer eingegangen).

- Richtlinie 5: Durchführung anhand von Forschungsmethoden

Die Durchführung von Arbeiten, die auf der Design-Science-Research Methodik basieren, müssen anerkannte Methoden in der Durchführung und der Evaluation des Design Artefakts beinhalten. Der Aufbau dieser Arbeit orientiert sich zunächst an Hevner (2007), wodurch die drei verschiedenen Zyklen gegeben werden. Im Rigor Cycle folgt die Arbeit der Methode nach vom Brocke et al. (2009), um eine Literaturrecherche durchzuführen. Im Relevance Cycle werden die Prozesse mit Hilfe von Befragungen der Nutzergruppen unter Verwendung von Freitextfragen in den SUS-Fragebögen und Videos zu Arbeitsprozessen des Tutorbetriebs, die der „Think Aloud" Methodik (Van Someren, Barnard, & Sandberg, 1994) folgen, erhoben. Diese Prozesse werden dann mit Hilfe von BPMN 2.0 abgebildet und anhand dieser Visualisierungen optimiert. Im Design Cycle wird das erschaffene Softwareartefakt mit Hilfe der oben (Richtlinie 3) genannten Methoden evaluiert. Für eine ausführliche Darstellung der verwendeten Forschungsmethoden siehe Kapitel 1.4 – Methodisches Vorgehen.

- Richtlinie 6: Forschung als Suchprozess

Die Erhebung eines geeigneten Artefakts benötigt die Verwendung aller zur Verfügung stehenden Mittel, um die gesetzten Ziele im Anwendungsumfeld zufriedenstellend umsetzen zu können. Für diese Arbeit wird aus diesem Grund eng mit den intendierten Nutzergruppen zusammengearbeitet. Hierdurch kann ständig Feedback aus dem intendierten Nutzungsumfeld gesammelt und in den Arbeitsprozess integriert werden.

- Richtlinie 7: Kommunikation der Forschung

Die Ergebnisse der Design-Science Forschung müssen technisch- und management-orientiertem Publikum präsentiert werden. Dies geschieht durch das Zusammentragen aller Ergebnisse in dieser Arbeit.

Neben der Befolgung dieser Richtlinien folgt diese Arbeit (wie oben bereits erwähnt) dem methodischen Vorgehen nach Hevner (2007). Im Mittelpunkt dieser Methodik steht das zu erschaffene Software Artefakt, welches durch iterative Abfolge von verschiedenen Zyklen erschaffen wird. Hierbei werden die Bedürfnisse und Wünsche der unterschiedlichen Nutzergruppen in den Arbeitsprozess mit einbezogen. Diese Arbeitsweise ist für die vorliegende Arbeit von Vorteil, da mit drei (vier – falls die Integration des Verleih-Tools berücksichtigt wird) verschiedenen Nutzergruppen[4] die Abdeckung der unterschiedlichen Anforderungen schneller

[4] Die Nutzergruppe „Technische Betreuung des Tutorbetriebs" wird hierbei nicht berücksichtigt, da zum Zeitpunkt des Verfassens der vorliegenden Arbeit diese Nutzergruppe lediglich aus dem Autor bestand.

umgesetzt werden kann. Die Design Science Methodik unterscheidet dabei drei verschiedene Zyklen.

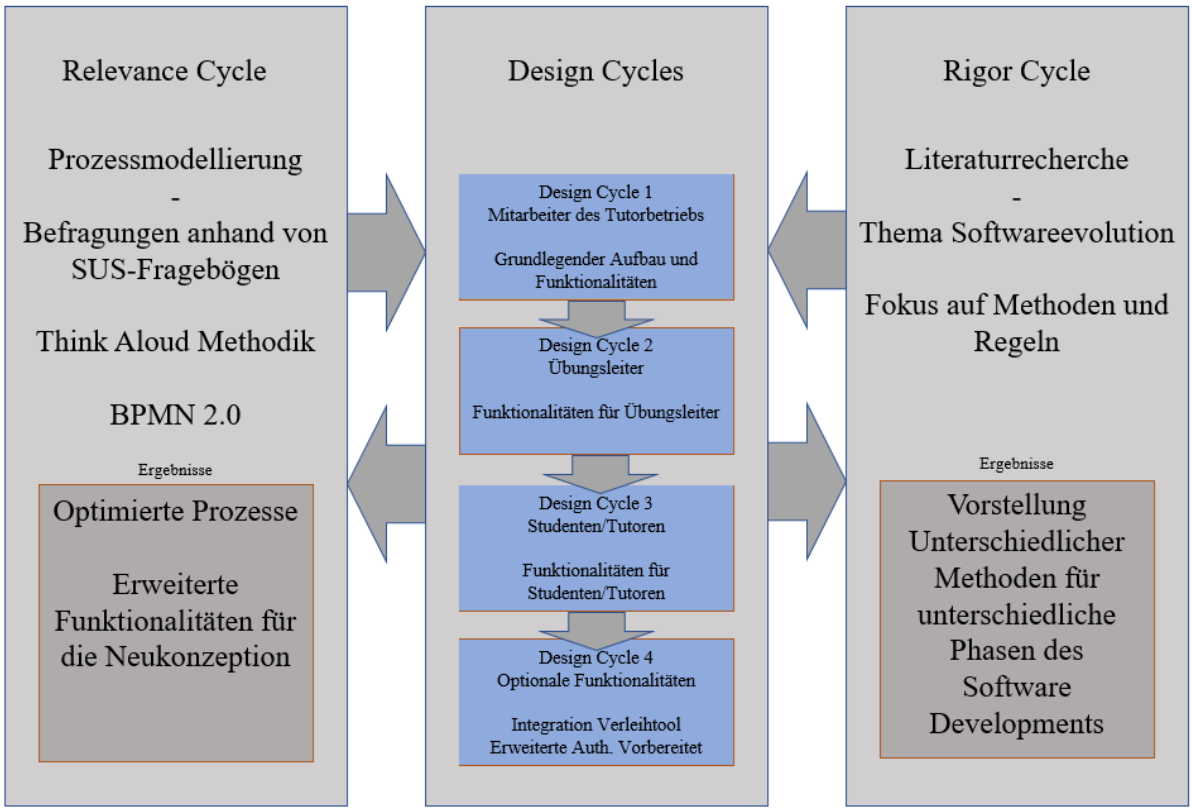


Abbildung 1: Darstellung der Zyklen der Design Science Methode in dieser Arbeit
Quelle: eigene Darstellung nach Hevner (2007)

Im Mittelpunkt dieser Methodik stehen dabei die sogenannten Design Cycles. Während dieser Zyklen wird iterativ jeweils ein Softwareartefakt erschaffen. Dieses wird im Anschluss mit den Nutzergruppen getestet. Für diese Tests wird je eine passende Testmethodik ausgewählt. Basierend auf dem Feedback, das durch die Tests mit den Nutzergruppen gewonnen wird, wird anschließend für die nächste Iteration eine verbesserte Version des Softwareartefakts erschaffen. Zur Unterstützung der Design Cycles wird zunächst ein Relevance Cycle durchgeführt, während dem die Geschäftsprozesse und die damit verbundenen Funktionalitäten ermittelt werden. Hierbei kommen unterschiedliche Methoden der Anforderungserhebung zum Einsatz. Dazu zählen Umfragen bei den intendierten Nutzergruppen, Einzelgespräche mit den Mitarbeitern des Tutorbetriebs und das Beobachten sowie die Dokumentation des Arbeitsalltags. Der Fokus liegt dabei auf den Prozessen, bei denen das alte Tutortool zum Einsatz kommt. Die hierdurch ermittelten Prozesse werden dann mit Hilfe von BPMN 2.0 (OMG Group, 2011) abgebildet. Diese grafische Spezifikation soll dabei helfen, die Prozesse weiter zu verstehen und somit besser optimieren zu können. Eine weitere Aufgabe dieses Zyklus ist es, das erstellte Softwareartefakt in den kontextuellen Arbeitsalltag einzuführen und dort zu testen. Das bedeutet für die vorliegende Arbeit konkret, dass die Mitarbeiter des Tutorbüros einen typischen Vertragsprozess mit Bewerbungen von Studierenden und der Auswahl von Übungsleitern testweise durchführen. Unterstützend zu den beiden anderen Zyklen wird ein sogenannter Rigor Cycle durchgeführt. Die Aufgabe dieses Zyklus ist es, durch die Analyse des aktuellen Stands der

wissenschaftlichen Literatur Methoden und Theorien herauszuarbeiten und in den Entwicklungsprozess einzuführen. Hierfür wird eine Literaturrecherche nach vom Brocke et al. (2009) mit dem Thema Softwareevolution durchgeführt. Eine visualisierte Darstellung dieses Arbeitsprozesses mit Bezug zu dieser Arbeit kann in Abbildung 1 betrachtet werden.

Durch diese Arbeitsweise stehen die zukünftigen Nutzergruppen im Fokus des Entwicklungsprozesses. Aufgrund der zum Zeitpunkt der Anfertigung dieser Arbeit vorherrschenden SARS-CoV-2 Pandemie wurden zur vorschriftsgemäßen Einhaltung der entsprechenden Hygienemaßnahmen alle Treffen mit den Testnutzern durch Online-Videokonferenzen ersetzt.

2 Begriffliche Grundlagen

In diesem Kapitel werden die wichtigsten in dieser Arbeit verwendeten Begriffe vorgestellt und erläutert. Hierbei werden zunächst die Begriffe der Softwareevolution und der evolutionären Software Entwicklung vorgestellt. Darauf folgen die Begriffe der Softwarewartung, des Software Reengineerings und des Software Lifecycles. Abschließend wird der Tutorbetrieb der TU München kurz vorgestellt.

2.1 Softwareevolution

Durch die Verwendung des Begriffs der Evolution wird ein normalerweise in der Biologie angesiedelter Begriff in den Kontext der Informationssysteme eingeführt. Dabei ist jedoch nicht die Veränderung einer Spezies gemeint, sondern eines Software-Systems (Ludewig & Lichter, 2013, S. 171).

Der Begriff der Softwareevolution (auch Software-Evolution oder engl.: software evolution) wird dabei in der Fachliteratur unterschiedlich verwendet. Bei manchen Autoren wird der Begriff der Softwareevolution mit dem der Softwarewartung gleichgesetzt (Paech, Apel, Grunske, & Prehofer, 2016, S. 187 f.). Die in dieser Arbeit verwendete Definition folgt jedoch ebenso wie Paech et al. (2016) der von Godfrey und German (2008). Hierin wird Softwareevolution als Beschreibung für alle verschiedenen Phänomene verwendet, die in Zusammenhang mit Veränderungen von bestehender Software stehen (Godfrey & German, 2008, S. 130). Des Weiteren wird hierin der Begriff der Softwareevolution von dem der Softwarewartung abgegrenzt. So beinhaltet der Begriff der Softwareevolution die Idee von essenziellen Veränderungen an der Software, während der Begriff der Softwarewartung dies nicht impliziert. Ferner erörtern Godfrey und German (2008), dass Softwarewartung typischerweise geplante beziehungsweise planbare Aktivitäten an einem Softwaresystem beschreibt. Im Gegensatz dazu beinhaltet der Begriff der Softwareevolution alle Veränderungen, die das Software-System betreffen. Dies umfasst neben den planbaren beziehungsweise geplanten auch die ungeplanten Veränderungen. Als Beispiele für unerwartete Veränderungen werden hier überladene Interfaces oder ein grundsätzlich neues Verwendungsumfeld genannt.

Beobachtungen zur Langlebigkeit von Software wurden bereits Anfang der 1980er Jahre versucht zusammenzufassen. Ein erster Versuch, diese Beobachtungen in feste Regeln umzuwandeln, wurde durch Lehman (1980) unternommen, worin fünf Gesetze (geläufig auch Lehman's Laws genannt) zur Evolution von Software definiert werden (Lehman, 1980, S. 1067 f.). Von diesen fünf Gesetzen sind in der Praxis aber nur zwei anwendbar und werden durch Beobachtungen gestützt (Ludewig & Lichter, 2013, S. 585 f.). Diese Gesetze sind das „Law of continuous change" und das „Law of increasing entropy" (Vergleiche im Original (Lehman, 1980) "Law of increasing complexity"). Ludewig und Lichter (2013) nehmen an dieser Stelle jedoch Bezug auf die leicht veränderte Variante in Lehman und Belady (1985, S. 158 f.). Das erste Gesetz besagt, dass das (Software-)System andauernder Veränderung unterliegt, bis es ökonomischer wird, es durch ein neues oder umstrukturiertes System zu ersetzen. Das zweite Gesetz besagt, dass die Entropie eines (Software-)Systems mit der Zeit steigt, außer es werden daran Arbeiten vorgenommen, die darauf abzielen, dass die Entropie gleich bleibt oder abnimmt.

Bei Ludewig und Lichter (2013, S. 568) heißt es zur Softwareevolution:

> „Software Evolution ist die wiederholte Anpassung eines bestimmten Software-Systems an veränderte Anforderungen und Umgebungsbedingungen."

Diese Aussage verdeutlicht die Stellung, die die Softwareevolution innehat. Weiter heißt es: „Mit der Evolution ist unvermeidlich ein Qualitätsverlust verbunden; wenn sie aber unterbleibt, wird die Software nutzlos, weil sie den veränderten Anforderungen nicht mehr entspricht". Hierdurch wird klar, dass Softwareevolution positive und negative Effekte mit sich bringt. Die positiven Effekte von Softwareevolution sind somit die Anpassung an die neuen Anforderungen der Umgebung, wodurch das Software-System verwendbar bleibt. Unter dem Gesichtspunkt, dass nach Lehmans erstem Gesetz Veränderung zwingend stattfindet und negative Effekte mit sich bringt (sofern kein zusätzlicher Aufwand betrieben wird, um dies zu verhindern), werden diese negativen Effekte von Softwareevolution in Ludewig und Lichter (2013, S. 586) mit den in Parnas (1994, S. 280 f.) ermittelten Symptomen für alternde Software (unter dem Überbegriff der Softwarealterung, der in dieser Veröffentlichung eingeführt wird) verglichen. Ludewig und Lichter nennen hier die folgenden negativen Effekte (Symptome):

- Die Software wird fehleranfällig
- Es wird schwieriger und damit teurer, sie an neue Anforderungen anzupassen
- Die Leistung (die Effizienz) lässt nach
- Die Architektur degeneriert

Aus all diesen Punkten lässt sich folgern, dass Softwareevolution ein unumgängliches Phänomen bei der Entwicklung und dem Betrieb von Software-Systemen ist, das sowohl positive als auch negative Aspekte mit sich bringt.

2.2 Evolutionäre Software Entwicklung

Der Begriff der evolutionären Software Entwicklung (auch evolutionäre Software-Entwicklung) ist vom Begriff der Softwareevolution klar zu trennen. Während Softwareevolution, wie in Kapitel 2.1 beschrieben, die Anpassung an veränderte Anforderungen und Umgebungsbedingungen beschreibt, beschreibt evolutionäre Software Entwicklung eine Vorgehensweise bei der Softwareentwicklung. Ludewig und Lichter (2013) definieren den Begriff der Evolutionären Software Entwicklung wie folgt: „Vorgehensweise, die eine Evolution der Software unter dem Einfluss ihrer praktischen Erprobung einschließt. Neue und veränderte Anforderungen werden dadurch berücksichtigt, dass die Software in sequenziellen Evolutionsstufen entwickelt wird." (S. 171). In diesem Kontext wird beschrieben, dass Software, die mit diesem Entwicklungsprozess entworfen und umgesetzt wurde, die gewünschten Funktionalitäten aufweist, strukturell jedoch in einem schlechten Zustand ist. Ein typisches Beispiel hierfür ist das alte Tutortool, das evolutionär über mehrere Jahre hinweg entwickelt wurde. Hierbei wurde eine frühe Version der Implementierung in den Praxisbetrieb aufgenommen und an neu auftretende Anforderungen immer wieder angepasst, wodurch immer neue Evolutionsstufen entstanden.

2.3 Softwarewartung

In Kapitel 2.1 wurde bereits der Unterschied zwischen Softwareevolution und Softwarewartung aufgezeigt. Softwarewartung wird bei Ludewig und Lichter (2013, S. 566) so definiert, dass hierunter alle Arten von Veränderungen an der Software fallen, die nach der Auslieferung an den Kunden stattfinden. Der IEEE-Glossar (IEEE, 1990, S. 46) beschreibt den Begriff der Wartung daher wie folgt:

> "The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment."

Bei dieser Definition der Wartung wird nach Ludewig und Lichter (2013) suggeriert, dass es bereits während der Entwicklung Probleme gibt, die behoben werden müssen. Hierbei unterscheidet sich die Wartung lediglich dadurch von der Entwicklung, dass das Software-System bereits produktiv im Einsatz ist. Dies schließt auch inkrementelle oder, wie in Kapitel 2.2 beschrieben, evolutionäre Entwicklungsmethoden mit ein. Aus diesem Grund folgt der in dieser Arbeit verwendete Softwarewartungsbegriff ebenso wie Ludewig und Lichter (2013) der Definition von Opferkuch und Ludewig (2004, S. 35):

> „Software-Wartung ist jede Arbeit an einem bestehenden Software-System, die nicht von Beginn der Entwicklung an geplant war oder hätte geplant werden können und die unmittelbare Auswirkungen auf den Benutzer der Software hat."

Hierbei werden zwei Merkmale der Wartung definiert. Zum einen ist dieser Definition nach die Wartung nicht vorhersehbar und mit einem Überraschungsmoment verbunden und die Wartung ist im unmittelbaren Interesse der Benutzer (Opferkuch & Ludewig, 2004, S. 35). Durch letzteres unterscheidet sich die Softwarewartung vom Software Reengineering (Ludewig & Lichter, 2013, S. 567).

Zudem werden unterschiedliche Arten von Wartung unterschieden. Die wichtigsten sind hierbei die adaptive Wartung, welche Anpassungen an neue Anforderungen (funktional oder nichtfunktional) darstellt und die korrektive Wartung, durch die auftretende Fehler ausgebessert werden. Des Weiteren werden in der Literatur noch die verbessernde Wartung (Verbesserung bestimmter Qualitätsmerkmale) und die präventive Wartung (Ausbessern von Fehlern, die noch nicht aufgetreten sind) genannt, sind aber in ihrer ursprünglichen Definition nicht eindeutig (Ludewig & Lichter, 2013, S567 f.).

2.4 Software Reengineering

Der Begriff des Software Reengineerings wird bei Ludewig und Lichter (2013, S. 585 ff.) so definiert, dass es der Wartung gleicht, jedoch nicht dem Nutzer des Software-Systems dient. Wie in Kapitel 2.3 beschrieben, unterscheidet sich in diesem Punkt die Softwarewartung von Software Reengineering. Ein Reengineering stellt hierbei einen Schritt dar, um schlecht wartbare Software wieder wartbar zu machen. Dies kann unterschiedliche Gründe haben. Beispielsweise kann Software durch unterschiedliche Wartungsschritte strukturell in einem so schlechten Zustand sein, dass weitere Wartung nur unter erheblichem Aufwand zu bewältigen ist. Im Falle des alten Tutortools spielen hier zusätzlich zum schlechten strukturellen Zustand

weitere Faktoren eine Rolle. Beispiele für solche Faktoren im alten Tutortool sind schlechte Lesbarkeit des Quellcodes oder uneinheitliche Benennungen für Variablen. Zudem wurden diverse Kommentare und Tabellen in der Datenbank in unterschiedlichen Sprachen verfasst.

Ausgehend von diesen Kriterien lässt sich feststellen, dass in dieser Arbeit kein reines Reengineering des Tutortools geschieht, sondern eine Kombination aus Reengineering und konstruktiver Softwarewartung, die neue Funktionalitäten für die Nutzer einführt, um aktuellen Anforderungen zu entsprechen.

2.5 Software Lifecycle

Der Software Lifecycle beschreibt die gesamte Lebensspanne von Software-Systemen. Der Begriff wurde im IEEE-Glossar (IEEE, 1990, S. 68) definiert und wird im Folgenden paraphrasiert wiedergegeben. Hierbei wird der Software Lifecycle als die Zeitspanne, die mit der Konzipierung eines Software-Systems beginnt und endet, sobald die Software nicht mehr zur Verwendung zur Verfügung steht, definiert. Hierbei umfasst der Software Lifecycle mehrere Phasen, die zum Teil überlappen können oder iterativ ausgeführt werden. Diese Phasen sind so in der gängigen Fachliteratur allgemein gültig und werden bei Ludewig und Lichter (2013, S. 156) wie folgt ins Deutsche übersetzt:

- Analyse
- Spezifikation der Anforderungen
- Grobentwurf, Spezifikation der Module
- Feinentwurf
- Codierung und Modultest
- Integration, Test, Abnahme
- Betrieb, Wartung
- Stilllegung[5]

Neben dieser Übersetzung der Phasen können die originalen Phasen aus IEEE (1990, S. 68) in Abbildung 20 (Anhang A.2) betrachtet werden.

2.6 Tutorbetrieb

Wie zu Beginn der Arbeit bereits beschrieben, stehen der Fakultät für Informatik der TU München im Rahmen der Exzellenzinitiative zusätzliche Gelder zur Verfügung, um Tutorien für Studierende anzubieten. Diese Tutorien werden dabei für alle Pflichtveranstaltungen, einige Wahlveranstaltungen, die Vorprojekte für Informatik, Nachprojekte und die Repetitorien angeboten.

Hierbei stellt das Tutorbüro die Schnittstelle zwischen allen auftretenden Aktoren im Tutorbetrieb (Studierende, Übungsleiter, Verwaltung, etc.) dar. Die Mitarbeiter des Tutorbüros

[5] Diese Phase wird bei Ludewig und Lichter (2013) nicht genannt, ist aber im Original (Retirement) als optionale, teilweise vorkommende Phase genannt.

übernehmen dabei eine Vielzahl an Aufgaben. Zu den Kernaufgaben des Tutorbüros zählen dabei vor allem die Verwaltung der Einstellungen sowie die Erstellung von Verträgen für Tutoren. Des Weiteren geschieht durch die Mitarbeiter des Tutorbüros die Akquise der Tutoren und die Betreuung der Bewerber. Während des laufenden Tutorbetriebs bietet das Tutorbüro eine Sprechstunde an und ist der erste Ansprechpartner für Tutoren.

3 Rigor Cycle

Aufgrund des evolutionären Wachstums des alten Tutortools ist es für den Rigor Cycle dieser Arbeit erstrebenswert, zusätzliche Informationen zum Thema Softwareevolution zu sammeln. Hierzu ist es sinnvoll, eine Literaturrecherche mit dem Leitthema „Softwareevolution“ durchzuführen. Das Ziel dieses Rigor Cycles ist es daher, die erste Forschungsfrage („Wie ist der aktuelle Stand der Literatur zum Thema Softwareevolution?“) zu beantworten und hierzu Methoden aus aktuellen wissenschaftlichen Veröffentlichungen zusammenzutragen.

Das methodische Vorgehen dieser Literaturrecherche folgt vom Brocke et al. (2009). Dieses Paper stellt anhand von relevanten Veröffentlichungen eine Vorgehensweise für eine Literaturrecherche im Kontext der Informationssysteme vor. Hierbei folgt das Paper selbst den entwickelten Vorgaben. Der dabei vorgestellte methodische Rahmen für eine Literaturrecherche bringt diverse Vorteile mit sich. Zum einen bietet er dem Autor der Literaturrecherche einen Rahmen für seine Arbeit und zum anderen dem Leser eine klar nachvollziehbare Linie, durch die die Ergebnisse rekonstruiert werden können. Um ein Gefühl für die praktische Anwendung dieser Methodik zu bekommen, wurde zu Beginn des Rigor Cycles die Literaturrecherche aus Hewing (2014, S. 49 ff.) eingehend studiert. Aufgrund der konsequenten Umsetzung der Methodik nach vom Brocke et al. (2009) in der Literaturrecherche von Hewing (2014, S. 49 ff.), die zusätzlich noch Ergänzungen (z.B.: Information zu den Suchanfragen) einführt, dient diese dabei auch als strukturelle Vorlage für die folgende Ausarbeitung dieses Rigor Cycles.

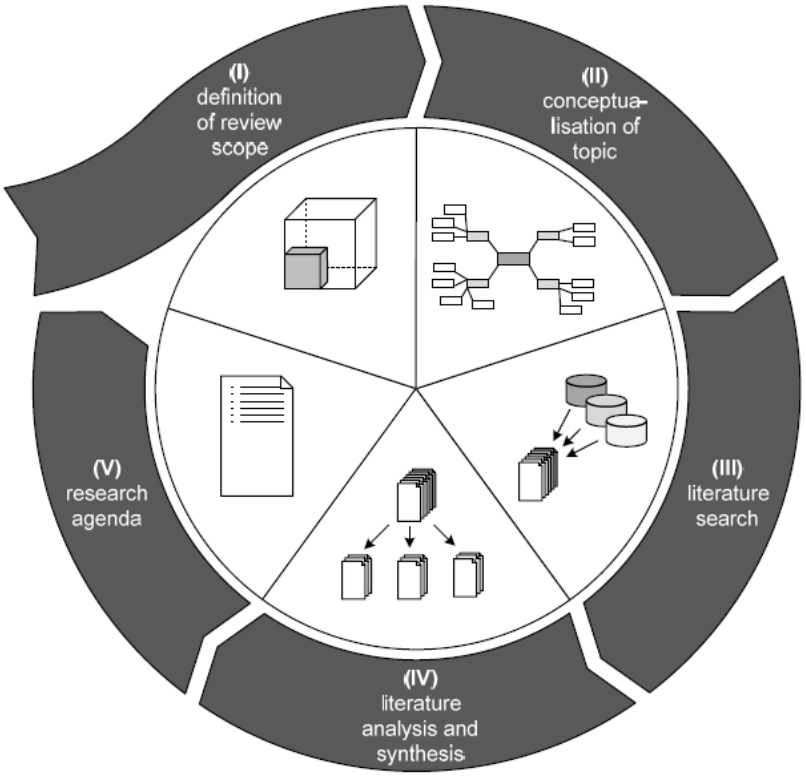


Abbildung 2: Phasen der Literaturrecherche
Quelle: vom Brocke et al. (2009, S. 7)

Die Literaturrecherche besteht aus mehreren Phasen (vom Brocke et al., 2009, S.6 ff.). In der ersten Phase wird der Umfang der Literaturrecherche bestimmt. Hiernach folgt in der zweiten Phase die Konzeptionalisierung des Themas. Die nächste Phase besteht aus der eigentlichen Suche der Literatur auf Basis der vorher definierten Konzepte und dem bestimmten Umfang der Arbeit. In der vierten Phase werden Relevanzkriterien definiert, um die für diese Arbeit relevanten Artikel aus der Masse an Suchergebnissen herauszufiltern. Die hierbei ermittelten relevanten Veröffentlichungen werden nachfolgend qualitativ und quantitativ analysiert. Abschließend wird durch die Ergebnisse der Literaturrecherche ein Forschungsausblick formuliert. Diese Phasen können in Abbildung 2 betrachtet werden.

Um den Prozess dieser Literaturrecherche noch anschaulicher zu gestalten, wird in den folgenden Textabschnitten neben der Literaturrecherche und den dazugehörigen Ergebnissen das methodische Vorgehen nach vom Brocke et al. (2009) in seinen einzelnen Schritten beschrieben.

3.1 Umfang der Literaturrecherche

Zunächst wird der Umfang der Literaturrecherche bestimmt. Hierzu kann die Klassifizierung von Literaturrecherchen nach Cooper (1988, S. 109) zu Hilfe gezogen werden. In dieser werden Literaturarbeiten durch sechs verschiedene Kategorien klassifiziert. Diese Kategorien setzen sich aus „Fokus“, „Ziel“, „Perspektive“, „Abdeckungsgrad“, „Organisation“ und „Zielgruppe“ zusammen. Diese Einteilung soll dem Leser einen Überblick über die Art der Arbeit geben und die Nachvollziehbarkeit der Literaturrecherche verbessern.

Der Fokus der meisten wissenschaftlichen Literaturrecherchen lässt sich in vier verschiedene Kategorien einteilen. Zu unterscheiden sind somit Veröffentlichungen, bei denen die Ergebnisse und Schlussfolgerungen anderer Veröffentlichungen im Mittelpunkt stehen, Veröffentlichungen, die den Fokus auf Methoden legen, Veröffentlichungen, die Theorien zum Thema behandeln und Veröffentlichungen, bei denen Anwendungen bzw. das Anwenden von Methodiken im Vordergrund steht. Dabei ist zu beachten, dass auch mehrere Kategorien auf eine einzelne Veröffentlichung zutreffen können. Diese Literaturrecherche soll sich vor allem mit Methoden und Regeln zur Vermeidung eines schlechten Alterungsprozesses von Software auseinandersetzen. Der Fokus dabei liegt vor allem auf Methoden, die in den ersten Phasen des Software Lifecycles, oder in der Phase eines potenziellen Reengineerings angewendet werden können.

Das Ziel vieler Literaturrecherchen ist das Darstellen von Forschungsergebnissen vorheriger Arbeiten, oder auch das Erklären dieser. Das Ziel der vorliegenden Arbeit hingegen ist vor allem das Synthetisieren unterschiedlicher Methoden, um diese gebündelt in einer Literaturrecherche zusammenzufassen und darzustellen. Hierzu werden relevante Veröffentlichungen aus der Gesamtheit der Ergebnisse durch Konzeptfragen herausgefiltert und in der qualitativen Analyse vorgestellt.

Die Organisation dieser Literaturrecherche erfolgt anhand von Konzepten, durch die das Thema der Softwareevolution sinnvoll unterteilt werden kann. Die Abbildung der Konzepte wird dabei in zwei Schritte unterteilt. Zunächst werden die Ergebnisse der Literatursuche durch Konzeptfragen gefiltert. Die hierbei erarbeiteten Ergebnisse werden dann in eine Konzeptmatrix (Webster & Watson, 2002, S.5 ff.) eingeordnet. Obwohl die Ergebnisse zum Teil auch im

chronologischen/historischen Kontext betrachtet werden, ist die gesamte Literaturrecherche aus den genannten Gründen dennoch als konzeptionell zu beurteilen. Die ermittelten Informationen werden dabei aus einer neutralen Sicht wiedergegeben.

Die Zielgruppe dieser Literaturrecherche sind zum einen Programmierer, die mit den hier ermittelten Informationen vermeiden wollen, dass ihre eigene Software schlecht altert, zum anderen aber auch Personen aus der Wissenschaft, die diese Arbeit als Sammlung von relevanten Veröffentlichungen für weitere Forschung zum Thema verwenden können. Daher ist die vorliegende Literaturrecherche in erster Linie an spezialisierte Fachleute gerichtet.

Der Umfang oder auch Abdeckungsgrad einer Literaturrecherche kann sich von „selektiv" über „repräsentativ" und „vollständig selektiv" bis hin zu „vollständig" erstrecken. Da es das Ziel dieser Literaturrecherche ist, einen Überblick über einen Bereich aus dem Themengebiet der Softwareevolution zu geben, ist ein selektiver Ansatz, der sich auf die Kernliteratur konzentriert, sinnvoll. Dieser selektive Ansatz wird durch das bereits in der Einleitung zu Kapitel 3 beschriebene Vorgehen, das auf dem Sichten und Filtern der Literatur in mehreren Schritten basiert, unterstützt. Abschließend lassen sich die hier ermittelten Parameter für die Literaturrecherche in der folgenden Tabelle 1 zusammenfassen. In dieser werden alle beschriebenen Möglichkeiten der Charakteristiken aufgezählt und die auf diese Arbeit zutreffenden fett markiert.

Charakteristik	Kategorien
Fokus	Ergebnisse / **Methoden** / Theorien / Anwendungen
Ziel	Beschreiben / Erklären / **Synthetisieren**
Organisation	Historische / **Konzeptionelle** / Methodische Darstellung
Perspektive	**Neutrale Wiedergabe** / Kritische Position
Zielgruppe	**Spezialisierte Fachleute** / PraktikerInnen / Allgemeine Öffentlichkeit
Abdeckungsgrad	Vollständig / vollständig selektiv / Repräsentativ / **Selektiv**

Tabelle 1: Taxonomie der Literaturrecherche
Quelle: vom Brocke et al. (2009, S. 7) nach Cooper (1988, S. 109)

3.2 Konzeptualisierung des Themas

Der nächste Schritt der Literaturrecherche besteht darin, das Thema zu konzeptualisieren. Bei diesem Schritt werden durch das Studium bestehender Kernliteratur Konzepte für das Thema herausgearbeitet, auf deren Basis die Schlagworte für die Literatursuche ausgewählt werden. Aus der Recherchearbeit für die Anfertigung des Proposals für die vorliegende Arbeit war Ludewig und Lichter (2013) bereits bekannt und dient als zentraler Punkt für diesen Schritt in der Literaturrecherche. Zudem war es für diesen Schritt hilfreich, Schlagworte bekannter themenbezogener Veröffentlichungen zu analysieren und zusammenzufassen.

Das Ziel dieser Literaturrecherche ist es, den aktuellen Stand der Fachliteratur im Bezug zu Softwareevolution zu identifizieren und Methoden zur Vermeidung von negativen Effekten durch diese wiederzugeben. Das Thema der Softwareevolution lässt sich dabei in unterschiedliche Kontexte setzen, welche sich thematisch voneinander abgrenzen lassen.

Auf der einen Seite steht hierbei der Kontext der Softwarealterung. Anhand dieses Konzepts lassen sich mehrere Unterkonzepte definieren. In diesem Zusammenhang sind solche Unterkonzepte erwähnenswert, die den Lebenszyklus eines Softwareartefakts abbilden. Hierzu zählen Softwareverschleiß, Softwarewartung, Softwarestilllegung und der Begriff des Software Reengineerings. Zusätzlich sind hier die Begriffe der Legacy Software (zu Deutsch etwa: Altsoftware) und der Altsysteme zu betrachten.

Auf der anderen Seite lässt sich der Begriff der Softwareevolution, trotz klarer begrifflicher Trennung, in den Kontext der evolutionären Softwareentwicklung setzen, da solche methodischen Vorgehensweisen das Phänomen der Softwareevolution als Begleiterscheinung des Entwicklungsprozesses mit einschließen. Dieses Konzept scheint sich zunächst stark von dem der Softwarealterung zu unterscheiden, jedoch gibt es bei genauerer Betrachtung teils große Überschneidungen. Folgende Textstelle fasst die thematischen Überschneidungen durch die Folgen der Anwendung der evolutionären Softwareentwicklung auf das Endprodukt zusammen:

> „Das Resultat zeigt typischerweise die gewünschte Funktionalität, ist aber strukturell in schlechtem Zustand. Denn die Architektur wurde für den ersten Versuch entworfen, sie ist für das Endprodukt weniger gut geeignet; zudem führen Änderungen zu einer Korrosion der Strukturen.“(Ludewig & Lichter, 2013, S. 171)

Dieses Zitat gibt Anlass, das Konzept der evolutionären Softwarearchitektur im Kontext der Folgen für die Alterung des Systems, zu dem die Unterkonzepte der evolutionären Softwarearchitektur gehören, in die Konzeptualisierung des Themas aufzunehmen. Als letzter Schritt der Konzeptualisierung wurde aus den gesammelten Konzepten und Begriffen zur besseren Visualisierung des Themas die folgende „Concept Map“ aus Abbildung 3 angefertigt.

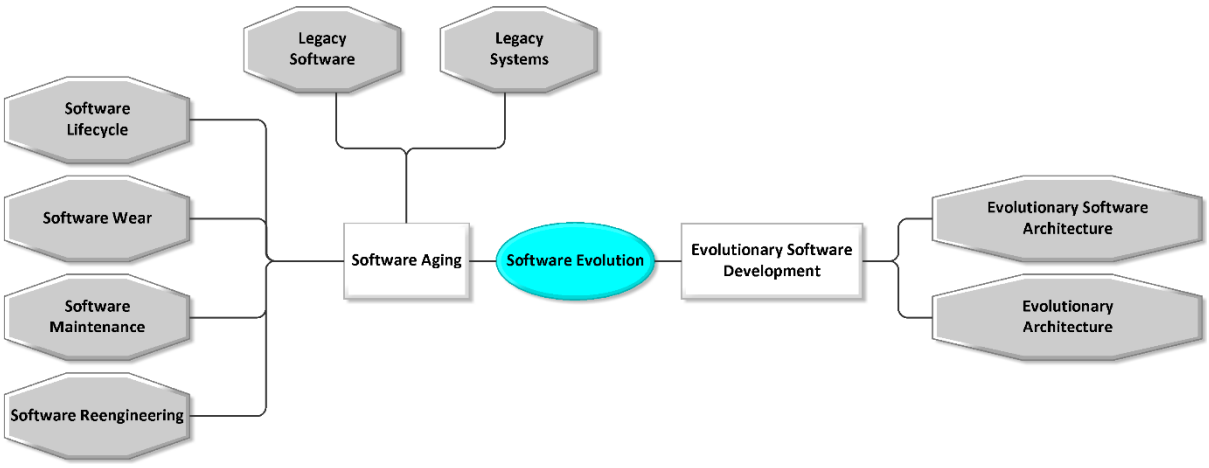


Abbildung 3: Concept Map Literaturrecherche
Quelle: eigene Darstellung

3.3 Literatursuche

Als nächster Schritt nach der Konzeptualisierung des Themas und der Definition des Umfangs folgt nun die eigentliche Suche nach geeigneter Fachliteratur für den vorliegenden Rigor Cycle. Dieser Schritt wird in vom Brocke et al. (2009) in vier weitere Phasen unterteilt. Diese Phasen sind wie folgt benannt:

- Journal Suche
- Datenbank Suche
- Schlagwort Suche
- Vorwärts- und Rückwärtssuche

Zunächst liegt somit der Fokus auf Artikeln, die in bekannten Journalen oder Konferenzberichten des Fachgebiets veröffentlicht wurden, da diese meistens peer-reviewed wurden und daher von hoher Qualität sind. Unterstützend bei der Auswahl wurden dabei Ranglisten wie die „Journal Quality List“ (Harzing, 2020) oder „VHB-JOURQUAL3“ (VHB e.V, 2019) verwendet, um die Qualität der ausgewählten Journale sicherzustellen.

Zu Beginn wurde hierzu der sogenannte „Basket of Eight“(AIS, 2016) als Sammlung von qualitativ äußerst hochwertigen Journalen für die Suche ausgewählt, um eine Abdeckung des Fachbereichs der Wirtschaftsinformatik zu gewährleisten.

Auf Basis der Konzeptualisierung lässt sich erkennen, dass viele Begriffe thematisch dem „Software Engineering“ zuzuordnen sind, wodurch es erstrebenswert ist, für diese Literaturrecherche Journale aus diesem Fachgebiet der Auswahl hinzuzufügen. Zu diesen Journalen zählt die als führende Fachzeitschrift dieses Fachgebiets geltende „IEEE Transactions on Software Engineering“ (IEEE Computer Society, 2021). Weitere Journale aus dem Fachbereich, die der Auswahl hinzugefügt wurden, sind das „Journal of Systems and Software“, die Fachzeitschrift „Empirical Software Engineering“ und „ACM Transactions on Software Engineering and Methodology“.

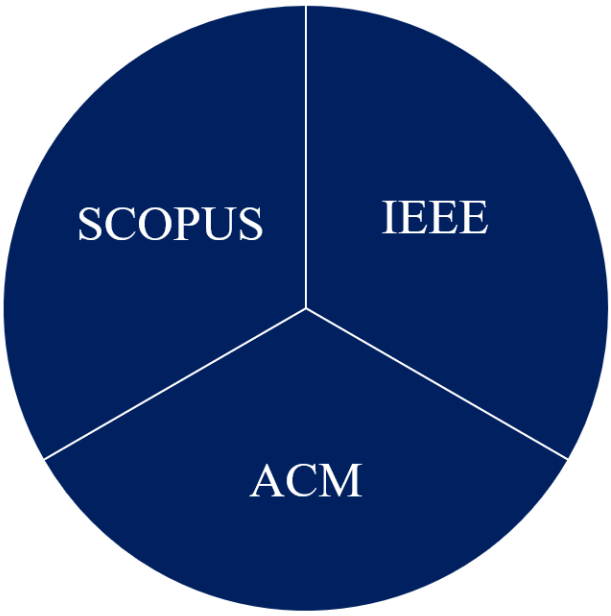
Nachdem die für die Literatursuche wichtigen Fachzeitschriften bestimmt wurden, können basierend auf dieser Auswahl im nächsten Schritt die für die Suche relevanten Datenbanken definiert werden. Da der oben angesprochene „Basket of Eight“ in der SCOPUS Datenbank enthalten ist, empfiehlt es sich, diese auch für die Suche zu verwenden. Dadurch, dass die SCOPUS Datenbank einen breiten Abdeckungsgrad über die Fachrichtungen der Wirtschaftsinformatik und des Software Engineerings hinaus besitzt, werden durch die Auswahl dieser Datenbank auch Veröffentlichungen berücksichtigt, die zwar einer anderen Fachrichtung zugeordnet werden, jedoch für diese Literaturrecherche trotzdem relevant sein können.

Aufgrund der Relevanz der Fachzeitschrift „IEEE Transactions on Software Engineering“ für die vorliegende Literaturrecherche wurde die Datenbank der IEEE ausgewählt (spezifischer die Datenbank „IEEE Xplore“). Diese Datenbank bietet Volltextzugriff auf ungefähr 5,2 Mio. Publikationen des IEEE (IEEE Xplore, 2021).

Zudem wurde, basierend auf der zuvor erwähnten Auswahl an wissenschaftlichen Veröffentlichungen, wie in vom Brocke et al. (2009, S. 9) beschrieben, die Datenbank der ACM der Liste der Datenbanken für diese Suche hinzugefügt.

Um den Schritt der Schlagwortsuche weiter zu verdeutlichen und eine bessere Nachvollziehbarkeit zu ermöglichen, wird zunächst auf die Formulierung der Suchanfragen eingegangen. Die einzelnen Schlagworte für diesen Schritt können mit den in Schritt zwei erarbeiteten Konzepten und den dazugehörigen Stichworten gleichgesetzt werden. Hier ist es wichtig zu

unterscheiden, dass die Unterkonzepte der Softwareevolution, nämlich Softwarealterung und evolutionäre Software Entwicklung, zu trennen sind.



„software maintenance“ | „software lifecycle“ | „legacy systems“ | „software reengineering“ | „software aging“ in **TITLE/ABSTRACT AND** „software evolution“ in **ABSTRACT**
OR
„evolutionary software architecture“ | „evolutionary software development“ in **TITLE/ABSTRACT**

Abbildung 4: Datenbanken und Suchalgorithmus
Quelle: eigene Darstellung

Durch die Konzeptualisierung in Abschnitt 3.2 können die Schlagworte für die eigentliche Literatursuche festgelegt werden. Hierfür werden die Konzepte und Unterkonzepte einzeln in die Datenbanken eingegeben und die Ergebnisse nach Relevanz für diese Literaturrecherche geprüft. Ein Beispiel hierfür ist die Suche nach dem Schlagwort „software evolution“. Eine Suche in SCOPUS nach diesem Begriff ohne Anführungszeichen führt dazu, dass nach software AND evolution in allen Metadatenbereichen gesucht wird, was ungefähr 650.000 Suchergebnisse ergibt[6]. Eine Einschränkung der Suche durch die Verwendung von Anführungszeichen, wodurch sich der Suchbegriff „ALL(„software evolution“)“ ergibt, produziert nur noch ungefähr 18.000 Ergebnisse. Diese Suche lässt sich weiter einschränken, indem vorausgesetzt wird, dass die Relevanz für das Thema der Suche der Veröffentlichung dann gegeben ist, wenn der gesuchte Begriff in Titel oder Abstract Erwähnung findet. Hierdurch kann die Anzahl der Ergebnisse der Suche für den Begriff der „software evolution“ auf ungefähr 2200 Ergebnisse reduziert werden. Auf diese Weise wird auch mit den restlichen erarbeiteten Schlagworten aus der Konzeptualisierungsphase verfahren. Da die Begriffe aus der Konzeptgruppe der

[6] Stand: 03. März 2021

Softwarealterung in allen Datenbanken eine hohe Zahl an Ergebnissen erzeugen, wird für diese Begriffe zusätzlich die Bedingung eingefügt, dass der Begriff „software evolution“ in der Zusammenfassung der Veröffentlichung enthalten sein muss. Die Relevanz der Suchergebnisse wird durch das stichprobenartige Studium von Titeln und Abstracts der Suchergebnisse festgestellt. Nach der Implementierung der oben erwähnten Verbesserungen in die Suche kann eine deutliche Verbesserung der Qualität der Suchergebnisse in Bezug auf die Themenrelevanz festgestellt werden.

Auf diese Art und Weise können Schlagworte und Konzepte aus den Suchzeichenketten entfernt werden, da diese zwar konzeptuell zum Thema passen, jedoch in den Datenbanken (nach dem zusätzlichen Einfügen der Beschränkungen) keine Suchergebnisse liefern. Ein Beispiel hierfür ist der Begriff des „Softwareverschleißes“. Der Begriff wird im Kontext von Softwareevolution vor allem bei Ludewig und Lichter (2013) verwendet. Bei Suchanfragen in SCOPUS zu diesem Begriff (englisch: „software wear“) werden zwar einige Ergebnisse gefunden, diese können aber eher im Bereich des sogenannten „Wear Levelings“ angesiedelt werden. „Wear Leveling“ beschreibt hierbei einen Mechanismus, der angewandt wird, um die Lebensdauer von Solid-State-Festplatten zu erhöhen (Qureshi et al., 2009) und ist daher für die Thematik dieser Recherche nicht relevant. Ähnlich wird für die weiteren Konzepte verfahren, die am Ende nicht in der finalen Suche verwendet werden. In Abbildung 4 kann der ausformulierte verwendete Suchalgorithmus betrachtet werden. Zudem wurden der Grafik die verwendeten Datenbanken hinzugefügt.

Datenbank	Treffer	Algorithmus
ACM	33	(Abstract:("software evolution") AND (Abstract:("software maintenance" OR "software lifecycle" OR "legacy software" OR "legacy systems" OR "software reengineering" OR "software re-engineering" OR "software aging") OR Title:("software maintenance" OR "software lifecycle" OR "legacy software" OR "legacy systems" OR "software reengineering" OR "software re-engineering" OR "software aging"))) OR (Abstract:("evolutionary software architecture" OR "evolutionary software development") OR Title:("evolutionary software architecture" OR "evolutionary software development")
IEEE	124	((("Document Title": "software maintenance" OR "Abstract": "software maintenance") OR ("Document Title": "software lifecycle" OR "Abstract": "software lifecycle") OR ("Document Title": "software reengineering" OR "Abstract": "software reengineering") OR ("Document Title": "legacy software" OR "Abstract": "legacy software") OR ("Document Title": "software re-engineering" OR "Abstract": "software re-engineering") OR ("Document Title": "software aging" OR "Abstract": "software aging") OR ("Document Title": "legacy systems" OR "Abstract": "legacy systems")) AND "Abstract": "software evolution") OR (("Document Title": "evolutionary software architecture" OR "Abstract": "evolutionary software architecture") OR ("Document Title": "evolutionary software development" OR "Abstract": "evolutionary software development"))
SCOPUS	229	(TITLE-ABS ("software maintenance" OR "software lifecycle" OR "legacy software" OR "legacy systems" OR "software reengineering" OR "software re-engineering" OR "software aging") AND ABS ("software evolution")) OR (TITLE-ABS("evolutionary software architecture") OR TITLE-ABS("evolutionary software development"))
Gesamt:	**386**	

Tabelle 2: Übersicht der Suchanfragen für die einzelnen Datenbanken
Quelle: eigene Darstellung

In Tabelle 2 können die in den jeweiligen Datenbanken verwendeten Suchstrings sowie die jeweilige Anzahl an Suchergebnissen betrachtet werden. Ein Nachteil der bei der Literatursuche verwendeten Datenbanken ist, dass jede Datenbank die Suchanfragen unterschiedlich interpretiert, wodurch die verwendeten Suchstrings an jede Datenbank angepasst werden müssen.

3.4 Analyse der Ergebnisse

Die folgenden Abschnitte sollen einen Überblick über die Ergebnisse der Suche in den Datenbanken geben. Die Suche für die im Folgenden beschriebene Auswertung wurde am 5. März 2021 durchgeführt. Zunächst wird hierfür eine quantitative Analyse, die sich auf die Ergebnisse und die darin enthaltenen Keywords konzentriert, durchgeführt. Darauf folgend wird eine qualitative Analyse, die sich mit dem Inhalt der Ergebnisse und deren Auswertung auseinandersetzt, durchgeführt.

3.4.1 Quantitative Analyse

Die Suche in den drei Datenbanken wurde am 5. März 2021 durchgeführt und ergab insgesamt 386 Ergebnisse (siehe Tabelle 2). Hierbei traten insgesamt 101 Duplikate auf, wodurch sich nach deren Entfernung die Anzahl der eindeutigen Resultate auf 270 reduzierte. Die älteste hierbei gefundene Veröffentlichung stammt aus dem Jahre 1983, die aktuellste aus dem Jahre 2021. In Abbildung 5 kann eine Aufschlüsselung der gefundenen Veröffentlichungen nach ihrem Erscheinungsjahr betrachtet werden. Zunächst wurde auf Basis dieser Ergebnisse eine quantitative Analyse durchgeführt. Hierzu wurden die Keywords der Veröffentlichungen gesammelt und nach der Häufigkeit ihres Vorkommens sortiert. Insgesamt wurden in den 270 eindeutigen Veröffentlichungen 623 Keywords verwendet, von denen 448 eindeutige Keywords sind. Die niedrige Zahl an Keywords lässt sich damit begründen, dass viele Veröffentlichungen vor dem Jahre 2001 keine Keywords enthielten, oder die Datenbanken für diese

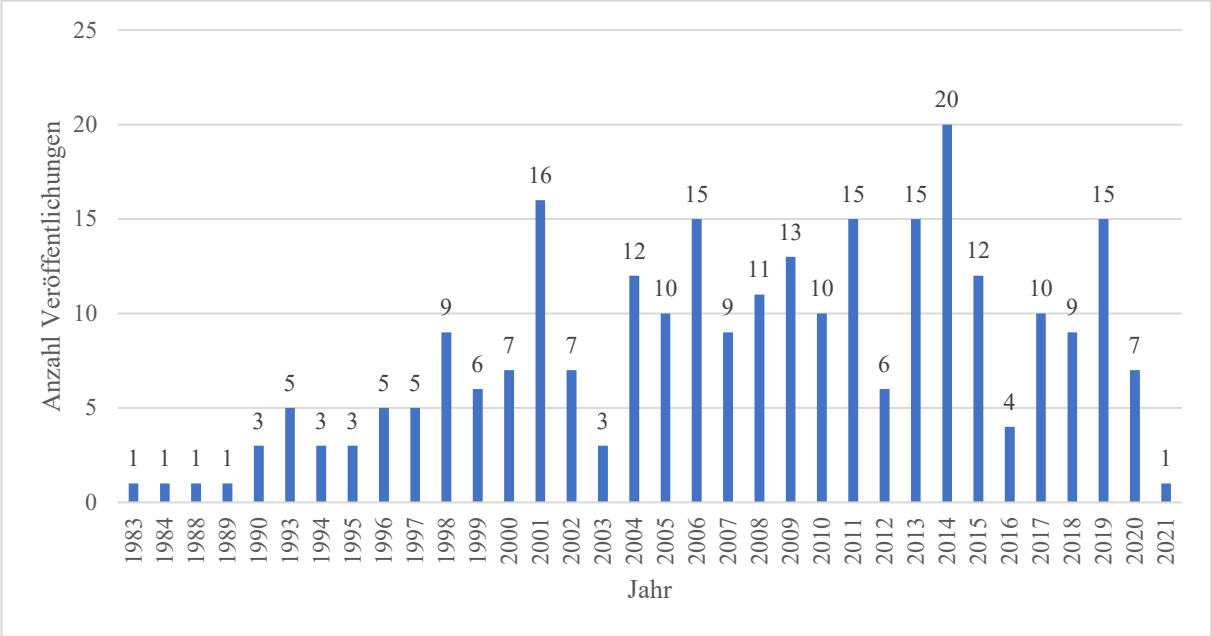


Abbildung 5: Aufschlüsselung der gefundenen Veröffentlichungen nach Erscheinungsjahr
Quelle: eigene Erhebung

Veröffentlichungen keine Keywords bereitstellen. Ein weiterer Punkt hierzu ist die Häufung an Konferenzberichten, die oftmals keine Autoren-Keywords beinhalten. Die zehn häufigsten Keywords sind in Abbildung 6 dargestellt. Hierdurch konnten bereits erste Schlüsse aus der Literatursuche gezogen werden. Die häufigsten Keywords sind „Software Evolution“, „Software Maintenance“ und „Software Engineering“. Hieraus kann gefolgert werden, dass die Ergebnisse der Suche die in der Konzeptualisierung angestrebten Themen umfassten und der erstellte Suchalgorithmus die erwünschten Ergebnisse lieferte.

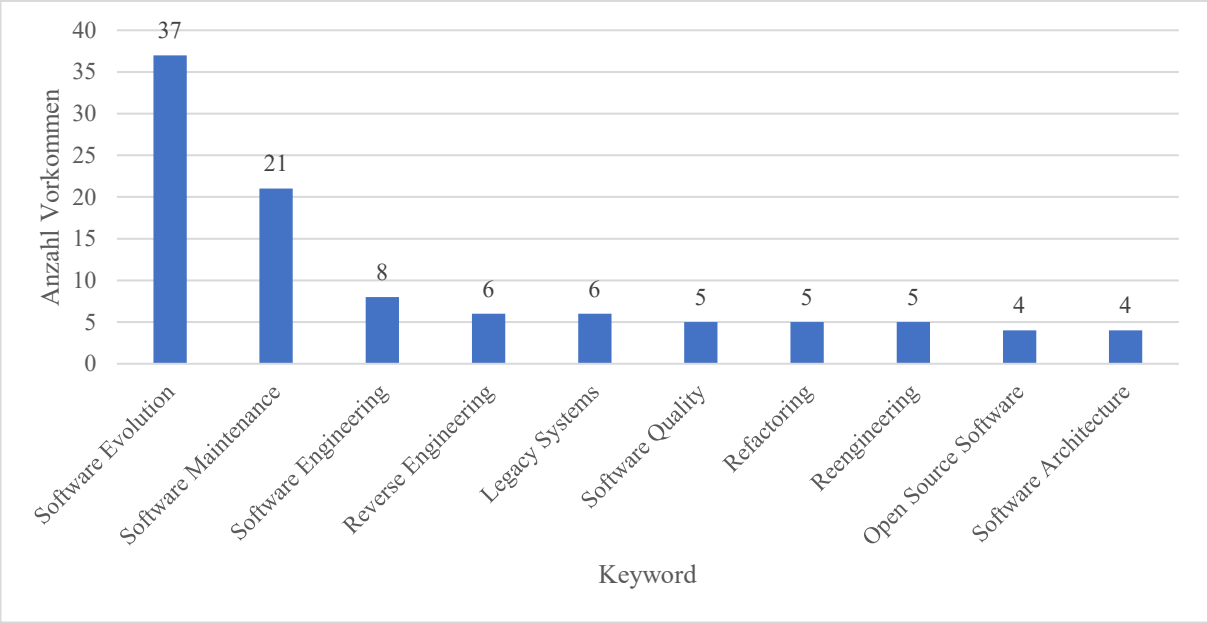


Abbildung 6: Top 10 verwendete Keywords aus den gefundenen Veröffentlichungen
Quelle: eigene Erhebung

3.4.2 Qualitative Analyse

Für die qualitative Analyse der Suchergebnisse wurden zusätzliche Regeln aufgestellt, um die Kernliteratur beziehungsweise relevanten Artikel für diese Literaturrecherche aus der Masse der Suchergebnisse herauszufiltern. Hierzu wurden die Titel und Abstracts aller Suchergebnisse drei Mal gelesen. Hierbei wurden die folgenden Fragen gestellt: „Enthält diese Veröffentlichung Methoden oder Regeln, durch deren Anwendung bereits in einer frühen Phase des Software Lifecycles der Arbeitsaufwand in der Phase der Softwarewartung beziehungsweise der evolutionären Weiterentwicklung des Programms reduziert werden kann?“ (Konzeptfrage 1) und „Enthält diese Veröffentlichung Methoden oder Regeln, durch deren Anwendung der Vorgang des Reengineerings von Altsystemen unterstützt werden kann?“ (Konzeptrage 2). Diese Fragen wurden basierend auf mehreren Kriterien ausgewählt. Wie in der Einleitung bereits beschrieben, stellt die Softwarewartung einen großen prozentualen Anteil an den Gesamtkosten (sowohl monetär als auch aus zeitlicher Sicht) eines Softwareprojekts dar. Deswegen ist es zunächst für jede Softwareapplikation wichtig, dass die Grundstrukturen des Systems so aufgebaut werden, dass evolutionäres Wachstum unterstützt und der Wartungsaufwand reduziert werden kann. Des Weiteren ist das Reengineering ein wichtiger Schritt im Software Lifecycle und kann durch die potenziell hier erarbeiteten Methoden unterstützt werden. Zudem kann der Vorgang des Reengineerings auch eine Neuimplementierung der Softwareapplikation

bedeuten, wodurch Methoden aus den Veröffentlichungen beider Kontextfragen kombiniert werden können. Konnte eine dieser Frage nach der Sichtung des Titels und des Abstracts positiv beantwortet werden, so wurde die Veröffentlichung in die Sammlung der relevanten Artikel aufgenommen. Wurde die Frage negativ beantwortet, wurde die Veröffentlichung nicht in die Sammlung der relevanten Artikel aufgenommen. Für Konzeptfrage 1 konnten durch diese Vorgehensweise 28 relevante Artikel identifiziert werden. Für Konzeptfrage 2 wurden auf diese Weise neun relevante Artikel identifiziert.

Nach diesem Schritt wurden die Volltexte aller im vorherigen Schritt identifizierten Veröffentlichung gelesen. Hierbei wurde abermals überprüft, ob der Artikel für die vorliegende Literaturrecherche von Bedeutung ist. Bei diesem Schritt wurden beispielsweise Легалов, Легалов, und Матковский (2018) und Gonçalves, Lima, und Costa (2015) aussortiert, da diese beiden Veröffentlichungen zwar über ein englischsprachiges Abstract verfügen, die Volltexte jedoch ausschließlich auf Russisch beziehungsweise Spanisch verfasst sind.

Nach diesem Schritt wurde die Liste der relevanten Artikel für Konzeptfrage 1 auf 14 reduziert und die Liste der relevanten Artikel für Konzeptfrage 2 auf sechs. Für diese Artikel wurde dann eine Konzeptmatrix nach Webster und Watson (2002, S. 5 ff.) für beide Konzeptfragen erstellt. Die Ergebnisse dieses Schrittes sind in Tabelle 3 und Tabelle 5 dargestellt und nach ihrem Erscheinungsjahr in absteigender Reihenfolge geordnet.

Für die erwähnten (reduzierten) Ergebnisse wurde zusätzlich eine Vorwärts- und Rückwärtssuche durchgeführt, um diese in der qualitativen Analyse zu inkludieren (dieser Schritt ist mit „Sub Phase 4“ aus vom Brocke et al. (2009) S. 9 gleichzusetzen). Die Vorwärts- und Rückwärtssuche wurde hierzu in zwei Schritte aufgeteilt. Zunächst wurden die Titel aller angegebenen Quellen gelesen und nach ihrer Relevanz für die Konzeptfrage, der die Veröffentlichung zuzuordnen ist, bewertet. Kann eine Veröffentlichung als relevant betrachtet werden, wurde hierzu noch das Abstract gesichtet und geprüft, ob die Erfüllung der Konzeptfrage bestätigt werden kann. Gleiches wurde für die Vorwärtssuche praktiziert, die mit Google Scholar durchgeführt wurde. Zusätzlich wurde zu jeder Veröffentlichung aus der Literatursuche ein Graph mit Hilfe des Webtools „Connected Papers“ (Eitan, Smolyansky, Harpaz, & Perets, 2021b) erstellt. Das Webtool „Connected Papers“ erstellt dabei einen Graphen bezüglich der gesuchten Veröffentlichung. Dieser Graph beinhaltet Veröffentlichungen, die der Algorithmus durch Verwendung von Ähnlichkeitskonzepten als ähnlich zur gesuchten Veröffentlichung identifiziert hat (Eitan, Smolyansky, Harpaz, & Perets, 2021a). Auf diese Weise können vereinfacht Relationen zwischen den Veröffentlichungen erkannt werden. Der Vorteil von „Connected Papers“ gegenüber einer herkömmlichen Vorwärts- und Rückwärtssuche ist, dass die dahinterliegende Datenbank auch Veröffentlichungen berücksichtigt, die zwar nicht als Quellen in dem gesuchten Artikel enthalten sind, jedoch thematische Ähnlichkeiten aufweisen. Die bei der Suche mit „Connected Papers“ gefundenen Ergebnisse wurden daraufhin nach dem Kriterium „Similarity to Origin“ in absteigender Reihenfolge sortiert und anschließend die Zusammenfassungen der Veröffentlichungen gelesen. Hierdurch wurden für Konzeptfrage 1 sieben und für Konzeptfrage 2 drei weitere relevante Veröffentlichungen für diese Literaturrecherche identifiziert. Diese Ergebnisse wurden ebenso in eine Konzeptmatrix (mit den gleichen Konzepten wie die entsprechende Konzeptfrage) nach Webster und Watson (2002) eingeordnet und können in Tabelle 4 und Tabelle 6 betrachtet werden.

Quelle								
	Archi-tektur	Coding Rules	For-male Regeln	KM[7]	Develop-ment Methode	Require-ments Enginee-ring	First Ap-proach	Other
Tsoukalas, Kehagias, Siavvas, und Chatzigeorgiou (2020)		**X**			**X**			**X**
Piantadosi, Fierro, Scalabrino, Serebrenik, und Oliveto (2020)		**X**						
Ajouli und Henchiri (2019)	**X**		**X**		**X**	**X**		
de Vasconcelos, Kimble, Carreteiro, und Rocha (2017)				**X**			**X**	
Johanssen, Kleebaum, Bruegge, und Paech (2017)				**X**			**X**	
Haitzer, Navarro, und Zdun (2017)	**X**	**X**			**X**			
Bajaj, Patel, und Patel (2015)					**X**			
Mokni, Huchard, Urtado, Vauttier, und Zhang (2014a)	**X**		**X**					
Chang, Lu, und Chu (2008)						**X**		
Kimura, Hotta, Higo, Igaki, und Kusumoto (2014)		**X**						
Breivold, Crnkovic, und Larsson (2012)	**X**							**X**
Awang, Wan-Kadir, und Shahibuddin (2011)	**X**							
Mohan, Gold, und Layzell (2004)		**X**						
Berzins, Shing, Riehle, und Nogueira (2000)					**X**			

Tabelle 3: Konzept Matrix für Konzeptfrage 1
Quelle: eigene Darstellung in Anlehnung an Webster und Watson (2002)

Für Konzeptfrage 1 können die Veröffentlichungen in acht verschiedene Kategorien eingeordnet werden. Die ersten drei Konzepte behandeln Regeln und Methoden, die direkt auf den Softwarecode oder die Architektur angewendet werden können. Diesen Kategorien können insgesamt neun Artikel zugeordnet werden. Zwei Veröffentlichungen behandeln das Einführen von Knowledge Management Methoden direkt in den Entwicklungsprozess (de Vasconcelos et al. (2017), Johanssen et al. (2017)). Vier Veröffentlichungen befassen sich mit Entwicklungsmethoden und eine Veröffentlichung versucht, die behandelten Probleme durch Methodiken im Requirements Engineering zu lösen. Die Kategorie „Other“ beinhaltet ein Paper. Diese

[7] Knowledge Management

Veröffentlichung ist ein Review Paper (Breivold et al., 2012) zum Thema Software Architektur im Kontext der evolutionären Entwicklung.

Das erste Konzept, das mehrere der identifizierten Artikel aufgreifen, beruht auf der Evolvierbarkeit der Softwarearchitektur. Aufgrund des oft gewählten evolutionären Entwicklungsansatzes können während der Softwarewartung und -evolution diverse architekturelle Probleme auftreten. Die hier vorgestellten Veröffentlichungen greifen diese Probleme auf und versuchen diese durch unterschiedliche Ansätze zu lösen. So behandelt Haitzer et al. (2017) den Ansatz, dass eine potenzielle Fehlausrichtung zwischen Softwarearchitektur und -code zu weitreichenden Fehlern führen kann. Hierbei wird auf der Basis von Design Science zunächst ein Ansatz erstellt und überprüft. Dieser Ansatz soll durch eine stärkere Kombination von Softwarearchitektur und dem Quellcode durch das Befolgen von Codingregeln das Entstehen von „architectural drift“[8] und „architectural erosion“ verhindern. Im Verlauf des Artikels werden die aufgestellten Regeln dann noch durch eine Fallstudie mit Open-Source Projekten überprüft. Hierdurch ist die Veröffentlichung zusätzlich der „Codingrules“ Kategorie zuzuschreiben. Einen anderen Ansatz verwenden Mokni et al. (2014a). Hier wird ein formaler Ansatz durch Regeln entwickelt, um die Evolution von Softwarearchitekturen zu unterstützen. Ein direkterer Ansatz wird in Awang et al. (2011) gewählt. In diesem Artikel wird ein Framework vorgestellt, durch dessen Anwendung die negativen Effekte von Softwareevolution durch die Erhöhung der Anpassungsfähigkeit der Software verringert werden sollen. Dies soll durch das Anpassen der nicht-funktionalen Anforderungen an die Software geschehen. In Ajouli und Henchiri (2019) wird versucht, bereits in der Phase des Softwaredesigns und der Architektur durch Berücksichtigung der vorhersehbaren Wartungen die Wartbarkeit bereits vor der eigentlichen Implementierung zu verbessern. Obwohl in dieser Veröffentlichung keine konkreten Architekturvorschläge gemacht werden, kann der Artikel dennoch diesem Konzept zugeordnet werden, da auf Basis der Befolgung der vorgestellten Regeln von den Entwicklern konkrete Implikationen für eine Zielarchitektur des Softwaresystems getroffen werden können. Eine Veröffentlichung, die in dieser Kategorie eine Sonderstellung einnimmt, ist Breivold et al. (2012), da diese Veröffentlichung selbst eine systematische Literaturrecherche zum Thema „Software Architecture Evolution Research“ ist. Hierin wird versucht, eine ausführliche Übersicht über Methoden zu geben, durch deren Verwendung die Evolvierbarkeit von Softwarearchitekturen gesteigert werden kann. Hierzu wurden von den Autoren insgesamt 82 Veröffentlichungen identifiziert und in fünf verschiedene Kategorien (S. 6 ff.) eingeteilt, die dann zum Teil in weitere Subkategorien eingeteilt wurden. Diese fünf Kategorien werden wie folgt definiert[9]:

- *Qualitätsüberlegungen während des Softwarearchitektur-Designs*: Diese Kategorie beinhaltet Veröffentlichungen, die sich damit auseinandersetzen, wie Software-

[8] Hierbei werden die englischen Begriffe verwendet, da direkte deutsche Übersetzungen den Sinn nicht adäquat wiedergeben.
[9] Für diesen Text wurden die Beschreibungen der Kategorien direkt aus Breivold et al. (2012, S. 6) übernommen und ins Deutsche übersetzt.

Qualität in der Phase des Software-Architektur-Designs eingeführt und explizit berücksichtigt werden kann.
- *Bewertung der architektonischen Qualität*: Diese Kategorie konzentriert sich auf die nachfolgende Iteration, wenn die Architektur beginnt Form anzunehmen. Der Schwerpunkt liegt hierbei auf Methoden zur Bewertung der Architekturqualität, die dabei helfen, zusätzliche Qualitätsattribute und Szenarien zu ermitteln und zu verfeinern.
- *Ökonomische Bewertung*: Diese Kategorie konzentriert sich auf die Berücksichtigung der Kosten, des Aufwands, des Werts und der Ausrichtung auf die Geschäftsziele, bei der Bestimmung eines angemessenen Grads an architektonischer Flexibilität.
- *Architektonisches Wissensmanagement*: Diese Kategorie konzentriert sich darauf, wie die Architekturdokumentation durch die Nutzung verschiedener Informationsquellen angereichert werden kann, um Architekturwissen für Qualitätsattribute und deren Begründung zu erfassen.
- *Modellierungstechniken*: Diese Kategorie konzentriert sich auf die Modellierung der Nachvollziehbarkeit und die Visualisierung der entsprechenden Auswirkungen der Evolution von Softwarearchitektur-Artefakten.

Die Autoren geben basierend auf den Ergebnissen der Literaturanalyse mehrere Implikationen. So sehen es die Autoren als notwendig an, weitere theoretische Grundlagen zur Softwareevolution zu schaffen. Ebenso müssen geeignete Methoden und Techniken im Bereich der Softwareevolution kombiniert werden, um die spezifischen Ansätze zu ergänzen und zu erweitern. Schließlich implizieren die Autoren, dass neue Techniken entwickelt werden müssen, um sehr große Plattformen (die aus vielen unterschiedlichen kleineren Plattformen mit unterschiedlichen Evolutionsmerkmalen bestehen) zu unterstützen.

Das nächste Konzept, das in den relevanten Veröffentlichungen behandelt wird, lässt sich unter dem Überbegriff „Coding Rules“ zusammenfassen. Die hier aufgeführten Artikel beinhalten Regeln, durch die die Qualität des Softwarecodes verbessert wird, um eine bessere Wartbarkeit oder Evolvierbarkeit zu bewirken. Einen ersten Ansatz hierzu behandeln Piantadosi et al. (2020), die untersuchen, wie sich die Lesbarkeit des Codes während der Softwareevolution ändert. Basierend auf den Ergebnissen definieren die Autoren abschließend mehrere Regeln (S. 31 ff.), die helfen sollen, die Menge an schlechtem Code gering zu halten. Einen ähnlichen Ansatz verfolgen Mohan et al. (2004). Hier wird die Verständlichkeit des Quellcodes im Verlauf der evolutionären Entwicklung betrachtet. Aus den Ergebnissen können Regeln abgeleitet werden, durch die die Softwarewartung vereinfacht werden kann. Das nächste Paper, das in diese Kategorie fällt, ist Kimura et al. (2014). Diese Veröffentlichung untersucht, inwieweit das unter Entwicklern übliche Verhalten, den Rückgabewert „null“ für eine Funktion zu verwenden, die Wartbarkeit des Programms negativ beeinflusst. Tsoukalas et al. (2020) greift die Thematik der sogenannten „Technical Debt“ auf. „Technical Debt“ entsteht beim Einfügen von Änderungen in den Quellcode, wodurch zunächst ein kurzzeitiger Vorteil entsteht, diese Änderung sich jedoch langfristig (vor allem in Bezug auf die Wartbarkeit des Programms) negativ auf die Qualität des Quellcodes auswirkt. Zuletzt lässt sich das bereits beschriebene Haitzer et al. (2017) diesem Konzept zuordnen.

Das dritte Konzept behandelt formale Regeln, die während des Entwicklungsprozesses befolgt werden können. Zunächst kann dieser Kategorie die bereits beschriebene Veröffentlichung Mokni et al. (2014a) zugeordnet werden. Die zweite Veröffentlichung in dieser Kategorie verwendet Methoden aus dem Requirements Engineering, um damit sogenannte Design Patterns zu erschaffen, die den Arbeitsaufwand in der Phase der Wartung reduzieren sollen (Ajouli & Henchiri, 2019).

Zwei Paper aus der Literaturrecherche führen Methoden des Knowledge Managements in den Softwareentwicklungsprozess ein. Beide Veröffentlichungen beschreiben hierbei erste Schritte auf dem Gebiet und sind daher auch dem „First Approach" Konzept zuzuordnen. Johanssen et al. (2017) beschreibt zwar lediglich die Forschungsvision der Autoren, wurde aber dennoch aufgrund der thematischen Wichtigkeit in die Auswahl der relevanten Artikel für diese Konzeptfrage aufgenommen. Von de Vasconcelos et al. (2017) wird die Thematik deutlich konkreter behandelt. Das Ziel dieser Veröffentlichung ist es, den Entwicklungsprozess der Software zu vereinfachen und den Arbeitsaufwand in der Softwarewartung zu senken, indem Methoden des Knowledge Managements mit in die Prozesse eingebunden werden. Hierzu stellen die Autoren ein Framework vor, durch das relevante Informationen früherer Phasen des Software Lifecycles in den späteren Phasen (konkret der Phase der Softwarewartung) bereitgestellt werden.

Das nächste Konzept, das auf mehrere der relevanten Veröffentlichungen dieser Literatursuche zutrifft, ist das Beschreiben von Entwicklungsmethoden[10]. Zunächst lässt sich der bereits oben beschriebene Artikel Tsoukalas et al. (2020) dieser Kategorie zuordnen. Basierend auf dieser Veröffentlichung lassen sich Vorgehensweisen in den Entwicklungsprozess implementieren, durch die das Entstehen von „Technical Debt" vermieden werden kann, beziehungsweise die Menge an schlechtem Code reduziert werden kann. Bei Haitzer et al. (2017) steht die enge Verknüpfung von Architektur und Quellcode im Vordergrund. Jedoch bestätigen die Ergebnisse, dass durch geringen zusätzlichen Aufwand die erarbeiteten Konzepte in den Entwicklungs- und Evolutionsprozess eingebunden werden können. In Bajaj et al. (2015) wird ein Arbeitsprozess im Kontext der evolutionären Softwareentwicklung vorgestellt, bei dem nicht wie üblich die Softwaretests nach der Implementierung erstellt werden, sondern davor. Aufgrund der hier verwendeten Vorgehensweise wird immer nur so viel Quellcode geschrieben, dass die vorher festgelegten Tests bestanden werden und danach der Quellcode refaktorisiert wird. Die Ergebnisse zeigen, dass dieses Vorgehen den Code verbessert und wartbarer macht. Berzins et al. (2000) beschreibt eine computerunterstützte Prototyping Methode, die vor allem in den frühen Stadien der Softwareentwicklung hilfreich bei der Fehlerbekämpfung sein kann.

Die letzte Kategorie behandelt eine Veröffentlichung, die Methoden des Requirements Engineering verwendet. In Chang et al. (2008) wird durch die Verwendung von standardisierten Methoden im Requirements Engineering eine konsistente und wartbare Grundlage für die im Requirements Engineering erhobenen Dokumente geschaffen. Diese Grundlage dient als Basis für den Quellcode, auf den sich die Wartbarkeit übertragen soll.

[10] Development Methods

Quelle					
	Architektur	Coding Rules	Formale Regeln	Development Methode	Requirements Engineering
Ajouli (2021)	**X**		**X**	**X**	**X**
Ajouli (2015)	**X**		**X**		
Konersmann (2018)		**X**		**X**	**X**
Konersmann und Goedicke (2012)	**X**				
Tamzalit und Mens (2016)	**X**		**X**		
Mokni, Huchard, Urtado, Vauttier, und Zhang (2014b)	**X**		**X**		
Chu, Chang, und Lu (2008)					**X**

Tabelle 4: Ergebnisse Vorwärts- und Rückwärtssuche für Konzeptfrage 1
Quelle: eigene Darstellung

Im Schritt der erweiterten Literatursuche (Vorwärts- und Rückwärtssuche) wurden sieben weitere Veröffentlichungen als relevant definiert, welche in Tabelle 4 eingesehen werden können. Dabei wurden die nicht verwendeten Konzepte aus der Matrix entfernt.

Zunächst können hierbei die Veröffentlichungen betrachtet werden, die sich mit der Architektur von Software befassen. Zunächst ist hierbei die Veröffentlichung Ajouli (2021) zu nennen, die an Ajouli und Henchiri (2019) anknüpft. Hier wird eine weitere Möglichkeit dargestellt, wie Softwareentwickler Wartungsaufgaben bereits in einem frühen Stadium der Softwareplanung beziehungsweise des Softwaredesigns miteinbeziehen können. Es ist wiederum zu erwähnen, dass keine direkten Architekturimplikationen getroffen werden, diese Methodik jedoch darauf abzielt, dass Softwareentwickler eine Architektur verwenden, die die planbaren Wartungsaufgaben unterstützt. Der Artikel Ajouli (2015) stellt im Gegensatz dazu eine konkrete Vorgehensweise vor, um durch die Verwendung einer vorgestellten Programmstruktur die Softwareevolution und Wartung von Java Programmen zu begünstigen. Bei der Entwicklung von Software Architekturen können inkonsistente Architekturdarstellungen und fehlende Spezifizierungen oder Dokumente zu großen Problemen führen. Konersmann und Goedicke (2012) stellen hierzu ein konzeptuelles Framework vor, das drei Hauptziele verfolgt. Zunächst soll hierdurch sichergestellt werden, dass die Implementierung der Architektur konsistent mit den Spezifikationen ist. Des Weiteren soll die Architektur unter Verwendung einer beliebigen ADL[11] betrachtet oder bearbeitet und unter Verwendung von beliebigen CM[12] deployed werden können. Die letzte Anforderung ist, dass die Architektur analysiert und evaluiert werden kann. Um das Thema der ADLs aufzugreifen, wird in Mokni et al. (2014b) ein formeller Ansatz vorgestellt, der im Gegensatz zu sonst gängigen ADLs den gesamten Bereich des Software Developments (Spezifikation, Implementierung und Deployment) abdeckt. In Tamzalit und Mens (2016) wird ein Konzept vorgestellt, das die Wiederverwendung von Wissen über evolutionäre Veränderungen von Architekturen in den Entwicklungsprozess von neuen Systemen einführt.

[11] Architecture Description Languages
[12] Component Models

Dieses Konzept basiert auf der Verwendung von „Design Patterns“, speziell „Evolution Patterns“.

Eine Kombination aus Coding Rules, Development Methode und dem Einbringen von Requirements Engineering Methoden stellt Konersmann (2018) vor. Hierbei wird auf das Problem eingegangen, dass die „Design Models“, die den (Software-)Entwicklungsprozess oftmals begleiten, zwar während der Entwicklungsphase gepflegt werden, jedoch, sobald die Implementierung geändert wird, nicht angepasst werden. Die vorgestellte Methodik integriert diese Modelle in den Quellcode, wodurch Inkonsistenzen vermieden werden sollen.

Das Problem, dass während des Softwareentwicklungsprozesses keine Uniformität zwischen dem Requirements Engineering und Artefakten aus anderen Phasen des Software Lifecycles besteht, wird in Chu et al. (2008) behandelt. Dies liegt oft vor allem daran, dass die Dokumente der Anforderungserhebung in natürlichen Sprachen verfasst werden, wodurch Schwierigkeiten bei der Entwicklung und späteren Wartung entstehen können. Die Autoren stellen hierzu ein Model-basiertes Anforderungsentwicklungsframework[13] vor, mit dem die Qualität der Dokumente aus der Anforderungserhebung gesteigert werden kann. Begleitend hierzu führen die Autoren Mechanismen ein, durch die System Paradigmen aus der Anforderungserhebung in die Implementierung übergeführt werden können.

Quelle					
	Data Reengineering	Process Reengineering	Legacy Modernization	Toolbasiert	Programier-paradigma Migration
Ruiz, Molina, und García (2017)	X				
Khan, Azam, Maqbool, und Anwar (2020)		X			
Cho, Cha, und Yang (2004)			X	X	X
Liu, Yang, Zedan, und Cau (2000)			X	X	
Kontogiannis und Patil (1999)	X				X
Cobo, Mauco, Romero, und Rodriguez (1999)			X	X	X

Tabelle 5: Konzept Matrix für Konzeptfrage 2
Quelle: eigene Darstellung in Anlehnung an Webster und Watson (2002)

Für die zweite Konzeptfrage wurden die relevanten Veröffentlichungen in fünf verschiedene Konzepte eingeteilt, welche in Tabelle 5 zusammengefasst werden. Zudem ist anzumerken, dass alle hier betrachteten Veröffentlichungen methodische Vorgehensweisen beinhalten und daher dieses Konzept nicht in die Konzeptmatrix mit aufgenommen wurde.

Das erste und zweite Konzept befasst sich jeweils mit dem Reengineering von Prozessen beziehungsweise Daten(banken). Ruiz et al. (2017) stellt dabei eine Methodik vor, die Daten und

[13] Model-Based Requirement Development Framework

Datenbanken im Prozess des Reengineerings unter Verwendung von „Model-Driven Engineering“ (MDE) erneuert. „Data Reenginering“ umfasst hierbei die Transformation von Legacy Artefakten (z.B.: Schema, Data und Code) in Artefakte des Zielsystems (Ruiz et al., 2017, S. 2 ff.). Khan et al. (2020) beschreibt im Gegensatz dazu ein Framework, mit dem das Reengineering von Geschäftsprozessen von Altsystemen automatisiert werden kann. Kontogiannis und Patil (1999) stellen eine Methodik zur Identifizierung von objektorientierten Strukturen in prozedural implementierten Applikationen vor. Diese Veröffentlichung kann dem Data Reengineering zugeschrieben werden, da die hierbei aufgezeigten Strukturen in Ruiz et al. (2017) als Zieldaten des Data Reengineerings aufgezählt werden (S. 2ff.).

Dem Konzept der „Legacy Modernization“ können prinzipiell alle hier aufgeführten Veröffentlichungen zugeordnet werden. In der Konzeptmatrix bezieht sich dieses Konzept auf Veröffentlichungen, die Methoden vorstellen, mit denen konkret die Implementierung von Altsystemen auf neuer Technologie unterstützt wird. Alle in diesem Abschnitt aufgeführten Veröffentlichungen folgen auch dem Konzept der toolbasierten Veröffentlichungen aus der Konzeptmatrix. Die ersten beiden Veröffentlichungen sind ausgenommen, da diese Methodiken für das Reengineering vor der Implementierungsphase beschreiben. In Cho et al. (2004) präsentieren die Autoren einen für das Veröffentlichungsjahr neuen, ganzheitlichen Ansatz im Bereich der Legacy Systems Migration. Dabei werden nicht nur die Methodik selbst, sondern auch die dabei verwendeten Tools beschrieben. Die Veröffentlichung Liu et al. (2000) beschreibt Software Reengineering als wichtigen Teil von Softwareevolution. Der Schritt des Software Reengineerings brachte zum damaligen Zeitpunkt zwei Probleme mit sich, nämlich die langsame Geschwindigkeit und schlechte Skalierbarkeit des Reengineeringprozesses. In der Veröffentlichung wird daher ein Ansatz aufgezeigt, der durch Tool unterstützte Abstraktion des Legacy Systems beide Probleme behebt. Die letzte Veröffentlichung aus diesem Konzept, Cobo et al. (1999), beschreibt die Umsetzung von prozeduralen Systemen in eine objektorientierte Architektur auf der Basis von Smalltalk.

Das letzte Konzept der Konzeptmatrix umfasst Veröffentlichungen, die eine Migration des Programmierparadigmas beschreiben. Alle diesem Konzept zugeordneten Artikel stammen aus der Zeit um die Jahrtausendwende (Cobo et al. (1999), Kontogiannis und Patil (1999), Cho et al. (2004)). Eine logische Erklärung hierfür ist, dass zu diesem Zeitpunkt viele Altsysteme von einer prozeduralen Architektur auf eine objektorientierte Architektur migriert wurden.

Quelle				
	Legacy Modernization	Toolbasiert	Programierparadigma Migration	Development Methode
Sahoo, Kung, und Gupta (2016)				**X**
Zou und Kontogiannis (2001)	**X**		**X**	
Newcomb und Kotik (1995)		**X**	**X**	

Tabelle 6: Ergebnisse Vorwärts- und Rückwärtssuche für Konzeptfrage 2
Quelle: eigene Darstellung

Für die Vorwärts- und Rückwärtssuche für Konzeptfrage 2 wurde diese Konzeptmatrix (Tabelle 6) noch um das Konzept der Development Methode ergänzt. Dieses Konzept wurde bereits für

Konzeptfrage 1 verwendet, fand in der zunächst gefundenen relevanten Literatur zu Konzeptfrage 2 jedoch keine Anwendung.

Ähnlich wie diverse zuvor genannte Veröffentlichungen stellen Zou und Kontogiannis (2001) ein Framework vor, um prozeduralen Code in objektorientierte Konzeptionen umzuwandeln. Speziell kommen hierbei sogenannte „XML based Annotated Abstract Syntax Trees“ zum Einsatz, mit denen der Legacy Quellcode repräsentiert wird. Analog hierzu wird in Newcomb und Kotik (1995) ein Reengineering Tool beschrieben, das automatisch prozedurale Programme in objektorientierte Systeme umwandelt.

Dem zusätzlich eingeführten Konzept der „Development Methode“ kann das Paper Sahoo et al. (2016) zugeordnet werden. Hierin beschreiben die Autoren eine agile Vorgehensweise für das Reengineering von objektorientierter Software. Die Vorgehensweise wird dabei in drei Schritte unterteilt: Eine Release Planungsphase, in der die neuen Anforderungen an die Software identifiziert und priorisiert werden. Eine iterative Reengineeringphase, während der in mehreren Iterationen die Anforderungen umgesetzt werden. Abschließend wird eine Validierungsphase durchgeführt, während der die Neuimplementierung gegen das intendierte Design und die neuen Anforderungen getestet wird.

3.5 Limitationen und Diskussion

Der letzte Schritt in der Literaturrecherche nach vom Brocke et al. (2009) ist es, die Ergebnisse der Literaturrecherche zu diskutieren, Limitationen aufzuzeigen und eine Forschungsagenda für die Zukunft zu skizzieren. In diesem Abschnitt wird daher zunächst auf die Limitationen eingegangen. Daraufhin werden die Ergebnisse für Konzeptfrage 1 diskutiert und die zugehörige Forschungsagenda aufgestellt. Dies wird im Anschluss für Konzeptfrage 2 wiederholt. Abschließend wird Bezug auf die erste Forschungsfrage genommen.

Bei der vorliegenden Literaturrecherche gab es diverse limitierende Faktoren. Hierbei ist zunächst die Suche zu betrachten. Durch die verwendeten Konzepte konnte eine breite und allgemeine Abdeckung des Themas erreicht werden, was ferner durch die Auswahl an verwendeten Datenbanken unterstützt wurde. Trotzdem ist es möglich, dass für diese Literaturrecherche relevante Veröffentlichungen nicht erkannt wurden. Hierfür kann es verschiedene Gründe geben. Für den Fall, dass Veröffentlichungen kein englisches Abstract oder keinen englischen Titel besitzen, wurden diese von der Suche nicht erfasst. Ebenso ist es möglich, dass Veröffentlichungen in Journalen oder durch andere Kanäle veröffentlicht wurden, die von den Datenbanken nicht abgedeckt werden.

Basierend auf den Ergebnissen der Suche lässt sich dennoch erkennen, dass Softwareevolution und Softwarewartung stark diskutierte Themen in der Fachwelt sind. Die Wichtigkeit dieses Themas greifen viele der relevanten Veröffentlichungen dabei bereits in der Einleitung auf. So motivieren eine große Anzahl an Autoren ihre Forschung damit, dass ungefähr 50-90% der Zeit und Kosten im Software Lifecycle durch die Wartung der Software entstehen (Um einige aus der Literaturrecherche zu nennen: Ajouli und Henchiri (2019), Sahoo et al. (2016), Mohan et al. (2004)). Die Mehrheit der Veröffentlichungen aller Ergebnisse konzentrieren sich dabei jedoch mehr auf spätere Zeitpunkte im Software Lifecycle.

Im Hinblick auf die Konzepte wurden für Konzeptfrage 1 vor allem Methoden und Regeln aufgestellt, die die Architektur von Softwaresystemen betreffen. Der Grund hierfür kann sein, dass durch die Wahl einer gut evolvier- und wartbaren Architektur die Kosten für spätere Wartungen beeinflusst werden können (Ajouli & Henchiri, 2019). Hierbei konnte vor allem festgestellt werden, dass Methoden zur Vermeidung schlechter Softwareevolution beziehungsweise Methoden zur Garantie von einfacherer Softwarewartung erst in den letzten Jahren zunehmend in Betracht gezogen wurden. Für den Implementierungsteil dieser Arbeit waren vor allem Veröffentlichungen interessant, die sich mit Regeln zur Verbesserung beziehungsweise zur Vermeidung von schlechtem Code befassen, da die Lesbarkeit des Quellcodes ein weiteres großes Problem des alten Tutortools darstellt. Obwohl dieser Rigor Cycle zu einem späteren Zeitpunkt der Implementierungsphase durchgeführt wurde, konnten hierbei vor allem Regeln und Vorgehensweisen aus Kimura et al. (2014) oder Piantadosi et al. (2020) angewendet werden.

Vielversprechend für die Kontrolle von Softwareevolution sind die Bestrebungen, Mechaniken aus dem Knowledge Management in die frühen Phasen des Software Lifecycles einzuführen (de Vasconcelos et al. (2017), Johanssen et al. (2017)). Bei einer Betrachtung der quantitativen Auswertung fällt auf, dass „Knowledge Management“ als Keyword jedoch nur drei Mal in allen Suchergebnissen vorkommt. Im Gegensatz dazu fiel während der Vorwärts- und Rückwärtssuche auf, dass das Konzept des Einführens von Knowledge Management Methoden im Bereich der Softwareevolution sehr wohl betrachtet wurde, zumeist jedoch erst zu einem späteren Zeitpunkt im Software Lifecycle.

Für Konzeptfrage 2 ist zunächst eine Häufung an Veröffentlichungen zu beobachten, die sich mit Methoden auseinandersetzen, die einen Programmierparadigmenwechsel unterstützen. Diese Veröffentlichungen setzen den Fokus darauf, das Reengineering von Applikationen, die auf einer prozeduralen Programmiersprache basieren, so zu unterstützen, dass die Zielarchitektur auf Basis einer objektorientierten Sprache entsteht. Die in Sahoo et al. (2016) vorgestellte agile Vorgehensweise für das Reengineering wurde bei der Umsetzung der vorliegenden Masterarbeit nicht berücksichtigt, da die Veröffentlichung von Sahoo et al. (2016) sich lediglich auf das Reengineering konzentriert. Die vorliegende Masterarbeit beinhaltet neben dem Schritt des Reengineerings auch eine funktionale Überarbeitung des Systems, sodass die Verwendung der zuvor genannten agilen Vorgehensweise nicht gänzlich ausreichend gewesen wäre. Für zukünftige Reengineering Aufgaben stellt die Vorgehensweise dieser Veröffentlichung jedoch eine gute Methodik für den praktischen Einsatz dar.

Für weitere Forschungen sind in dem Bereich des Reengineerings vor allem Methoden sinnvoll, die Prozesse und Daten abbilden und auf deren Basis dann Neuimplementierungen gestaltet werden können. Aus Sicht der Wirtschaftsinformatik ist hier vor allem Khan et al. (2020) hervorzuheben. In dieser Veröffentlichung wird zunächst eine wichtige Forschungslücke aufgezeigt (das automatische Reengineering von Geschäftsprozessen) und eine Methodik zum Schließen der gleichen vorgestellt. Hierdurch können aufwendige Prozessabbildungs- und Optimierungsvorgänge (wie auch im Relevance Cycle dieser Arbeit) deutlich vereinfacht und beschleunigt werden. Zukünftige Forschungen in diesem Bereich könnten die vorgestellte Methode verbessern und erweitern.

Konzeptfragenübergreifend erscheinen vor allem die Forschungsergebnisse und Implikationen von Breivold et al. (2012) sinnvoll. Wie durch die unterschiedlichen Konzepte und Konzeptfragen (die sich auf unterschiedliche Stellen im Software Lifecycle beziehen) gezeigt, behandeln viele aktuelle Veröffentlichungen nur einzelne wenige Aspekte des Phänomens der Softwareevolution. So scheint es für zukünftige Forschungen sinnvoll zu sein, konkrete Konzepte für die Kombination von Methoden, Techniken und Regeln zu finden, um die vorliegenden Ansätze zu vervollständigen und zu erweitern. Hierdurch können Methoden geschaffen werden, die sich nicht mehr nur auf spezifische Evolutionsarten beziehen, sondern allumfassend auf unterschiedliche Evolutionsarten angewandt werden können.

Die erste Forschungsfrage „Wie ist der aktuelle Stand der Literatur zum Thema Softwareevolution?“ konnte mit dieser Literaturrecherche im Hinblick auf Methoden und Regeln für die Implementierungsphase und das Reengineering beantwortet werden. Das Thema der Softwareevolution ist im Bereich des Software Engineering stets präsent und für jeden Softwareentwickler unumgänglich. Bei der Beantwortung der allgemein formulierten Forschungsfrage wurden durch die Konzeptualisierung des Themas und dem Filtern der Suchergebnisse der ausgearbeiteten Konzeptfragen Einschränkungen vorgenommen. Aus methodischer Sicht sind die vorgenommenen Einschränkungen äußerst sinnvoll, sodass einzelne Ergebnisse des Rigor Cycles deutlich zielgerichteter für die Design Cycles der Arbeit verwendet werden konnten. Zudem kann die vorgestellte Literatur in zukünftigen Softwareprojekten Anwendung finden, um die entwickelten Software-Systeme besser für einen langfristigen Einsatz konzipieren zu können.

3.6 Zusammenfassung der Literaturrecherche

Das Ziel dieser Literaturrecherche war es, einen Einblick in die Literatur zum Thema „Softwareevolution“ im Hinblick auf Methoden und Regeln, die sich positiv auf die Softwarewartung beziehungsweise das Reengineering von Softwaresystemen auswirken, zu geben. Hierbei wurden drei Datenbanken durchsucht, in denen 270 eindeutige Ergebnisse zum Thema der Literaturrecherche gefunden wurden. Die quantitative Analyse der Ergebnisse ergab hierbei, dass die Konzeptualisierung des Themas die angestrebten Ergebnisse erzielte. Von den Ergebnissen der Literatursuche wurden in der qualitativen Analyse der Ergebnisse auf der Basis von zwei Konzeptfragen 14 beziehungsweise sechs Veröffentlichungen als relevant für die Literaturrecherche befunden. Die Liste der relevanten Veröffentlichungen wurde dann durch eine Vorwärts- und Rückwärtssuche um sieben beziehungsweise drei Veröffentlichungen ergänzt. Das Vorgehen in der Vorwärts- und Rückwärtssuche wurde zudem durch eine Suche mit dem Webtool „Connected Papers“ erweitert.

Darüber hinaus wurde die Methodik von vom Brocke et al. (2009) um die in Hewing (2014) beschriebenen Punkte erweitert. Dies umfasst vor allem die detaillierte Darstellung des Schritts der Literatursuche. Hierbei wurden zusätzliche Tabellen und Abbildungen verwendet, um der Nachvollziehbarkeit dieser Literaturrecherche eine zusätzliche Schicht hinzuzufügen.

4 Relevance Cycle

Wie bereits in Kapitel 1.4 -Methodisches Vorgehen- erläutert, beschäftigt sich der Relevance Cycle dieser Arbeit mit den Prozessen des Tutorbetriebs, die in Zusammenhang mit der Verwendung des Tutortools stehen. Hierzu werden zunächst die Prozesse betrachtet, die in Zusammenhang mit dem alten Tutortool stehen, um diese anschließend (falls nötig) optimieren zu können. Auf Basis dieser optimierten Prozesse werden dann Funktionalitäten formuliert, die anschließend in den Design Cycles umgesetzt wurden. Dieses Vorgehen dient der Beantwortung der zweiten und der dritten Forschungsfrage.

Erste Erfahrungen mit den Prozessen des alten Tutortools konnte ich durch meine Anstellung als studentische Hilfskraft am Tutorbetrieb der Fakultät für Informatik sammeln. Hierbei hatte ich als technischer Betreuer des Tutortools die Möglichkeit, den Quellcode der alten Implementierung genauer zu studieren. Auf Basis der hierbei gesammelten Erkenntnisse konnten die ersten (weniger komplexen) Prozesse, wie das Anlegen von Veranstaltungen oder das Anlegen eines Nutzeraccounts, skizziert werden.

Die Mehrzahl der Prozesse wurde in Zusammenarbeit mit den Mitarbeitern des Tutorbüros bestimmt, da diese (restlichen) Prozesse zum Teil weitaus komplexer sind. Hierfür wurden die Mitarbeiterinnen des Tutorbüros zunächst gebeten, typische Arbeitsabläufe zu beschreiben und einen typischen Arbeitstag zu dokumentieren. Um die hierbei grob formulierten Abläufe präzise durch Prozesse abbilden zu können, wurde anschließend die „Think Aloud“-Methodik (Van Someren et al., 1994) für die täglichen Aufgaben angewandt. Um (wie in Abschnitt 1.4 erwähnt) die Hygienevorschriften einzuhalten, fertigte eine Mitarbeiterin des Tutorbüros hierzu jeweils ein Video an und beschrieb währenddessen ihr Vorgehen. Bei der „Think Aloud“-Methodik sprechen die Testpersonen ihre Gedanken laut aus, während sie ihre täglichen Aufgaben bewältigen. Auf Basis der hierbei gesammelten Daten wurden dann die Prozesse beschrieben und die wichtigsten Prozesse mit der BPMN 2.0 (OMG Group, 2011) modelliert, um vorhandenes Optimierungspotenzial besser aufzeigen zu können. Im Folgenden werden die ermittelten Prozesse zunächst in ihrer Gesamtheit betrachtet. Später werden einzelne Abläufe, die größeres Optimierungspotenzial beinhalten, zusätzlich genauer betrachtet. Die für diesen Relevance Cycle angefertigten Diagramme wurden mit dem Online Modellierungstool von Signavio (Signavio, 2021) angefertigt. Alle erstellten Diagramme können auch dem digitalen Anhang entnommen werden[14]. Eine Übersicht des Inhalts des digitalen Anhangs kann in Tabelle 12 (Anhang C) betrachtet werden.

Um einen allgemeinen Überblick über die Usability der alten Version des Tutortools zu bekommen, wurde Personen der System Usability Scale Fragebogen nach Brooke (1996) zugeschickt. Die befragten Personen wurden so ausgewählt, dass diese das Tutortool bereits kannten und entweder in der Vergangenheit oder zum Zeitpunkt der Befragung aktiv verwendeten. Zusätzlich wurden dem Fragebogen noch Fragen mit Freitextfeldern zum Antworten hinzugefügt. Der

[14] Der Inhalt des digitalen Anhangs kann beim Autor angefragt werden.

Nutzen hiervon war, einen besseren Einblick zu bekommen, was den Nutzern an der Anwendung gefällt oder nicht gefällt. Die Antworten auf diese Fragen wurden zusätzlich verwendet, um Prozesse zu identifizieren, die einer Überarbeitung bedürfen. Anzumerken ist, dass diese Umfrage aufgrund der niedrigen Teilnehmerzahlen nicht repräsentativ ist und lediglich dazu verwendet wurde, ein Gefühl für die Akzeptanz des alten Tutortools bei den Nutzern zu bekommen.

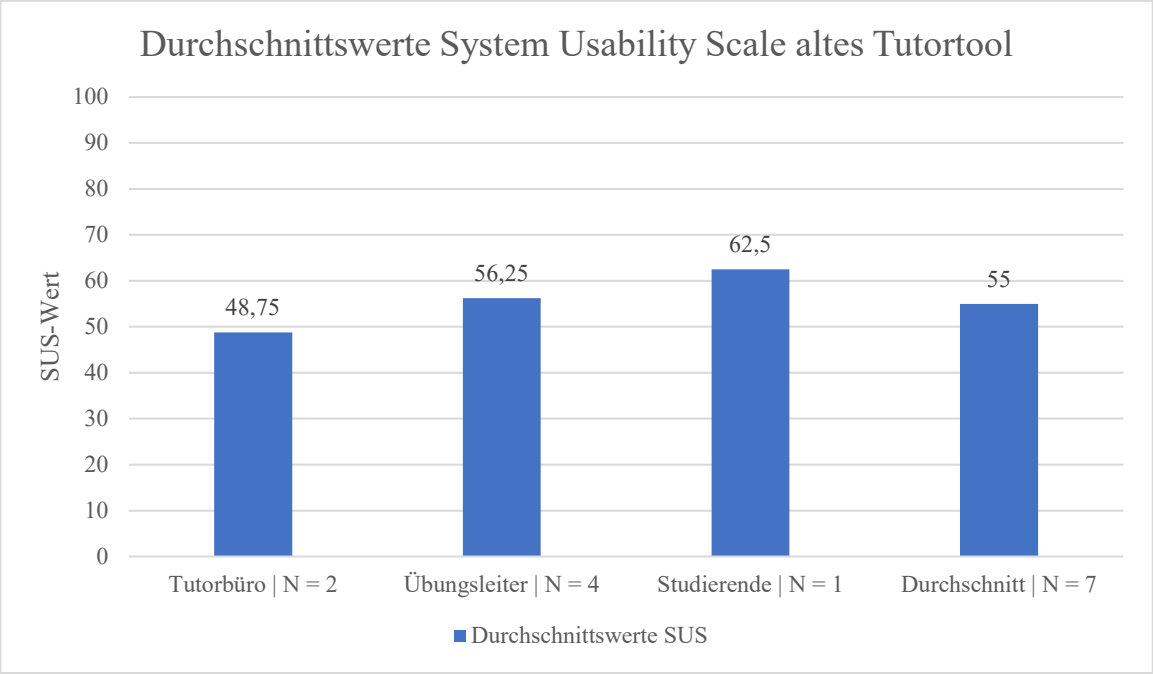


Abbildung 7: SUS Ergebnisse zur alten Version des Tutortools
Quelle: eigene Erhebung

In Abbildung 7 sind die Durchschnittsergebnisse der Befragung dargestellt. Um diese Ergebnisse einordnen zu können, lässt sich die in „Determining What Individual SUSScores Mean: Adding an Adjective Rating Scale“ (Bangor, Kortum, & Miller, 2009) erstellte Vergleichstabelle (Seite 8) zu Rate ziehen. Hierauf bewegen sich die Ergebnisse alle im nicht akzeptablen Bereich und können den Adjektiven „Poor“ und „OK“ zugeordnet werden. Zudem wird traditionell bei der Auswertung ab einem Wert von 70 auf der System Usability Scale von einer zufriedenstellenden Benutzbarkeit des getesteten Artefakts gesprochen, welche von keinem der ausgewerteten Fragebögen erreicht wurde. Diese Ergebnisse bestätigen die Vermutung, dass die alte Version des Tutortools in keinerlei Hinsicht die Ansprüche der verschiedenen Nutzergruppen in Bezug auf Benutzbarkeit erfüllt und eine Neukonzeption einen sinnvollen Schritt zur Verbesserung der aktuellen Situation darstellt.

4.1 Prozesse und Funktionalitäten im alten Tutortool

Dieser Abschnitt soll einen Überblick über die Prozesse bei der Verwendung des alten Tutortools geben. Zunächst werden hierzu die jeweiligen Nutzergruppen vorgestellt und im Anschluss der Gesamtprozess einer Einstellung beschrieben. Im nächsten Schritt werden dann die weiteren Prozesse bei der Verwendung des Tutortools beschrieben und Optimierungen an diesen für die Neukonzeption vorgenommen.

4.1.1 Nutzergruppen in Verbindung mit dem alten Tutortool

Bei der Verwendung des alten Tutortools treten vier verschiedene Nutzergruppen auf, die unterschiedliche Aufgaben haben und hierdurch auch unterschiedliche Anforderungen an das Tutortool haben. Diese Nutzergruppen treten nicht in jedem Prozess auf, stellen aber den Kern der Anwender dar. Ebenso kann es Nutzer aus anderen Gruppierungen geben, die in Sonderfällen das Tutortool nutzen, auf welche aber aufgrund der geringen Relevanz in dieser Arbeit nicht näher eingegangen wird.

Mitarbeiter des Tutorbüros

Die Mitarbeiter des Tutorbetriebs sind für die Organisation des Einstellungsprozesses der Tutoren zuständig. Hierzu zählen unterschiedliche Aufgaben, wie das anfängliche Kontaktieren der Übungsleiter, das Anlegen der Veranstaltungen oder das Zusammentragen, Sichten und Weiterreichen der erforderlichen Vertragsunterlagen bei einer Einstellung. Diese Aufgaben werden zum aktuellen Zeitpunkt bereits zum Großteil durch die Verwendung des Tutortools unterstützt.

Übungsleiter

Übungsleiter sind wissenschaftliches Personal, das mit der Durchführung der Tutorien begleitend zu den Lehrveranstaltungen betraut ist. Die Kernaufgabe dieser Benutzer des Tutortools ist es, die Bewerber auf die offenen Tutorstellen zu sichten und nach unterschiedlichen Kriterien auszuwählen.

Studierende/Tutoren

Die von den Übungsleitern ausgewählten Studierenden werden von der Universität als Tutoren angestellt, um begleitend zu den Lehrveranstaltung den Studierenden, die die Lehrveranstaltung besuchen, im Rahmen von gesonderten Übungsstunden den Inhalt der Vorlesungen weiter zu vermitteln. Zu den weiteren Aufgaben gehören dabei Hausaufgabenkorrekturen und das Beaufsichtigen und Korrigieren von Klausuren. Beim Einstellungsprozess für eine Tutorstelle müssen die Studierenden sich auf die entsprechende offene Stelle bewerben und bei einer Einstellung alle erforderlichen Dokumente dem Tutorbüro zusenden.

Technische Betreuung des Tutorbetriebs

Das alte Tutortool wird zudem von einer studentischen Hilfskraft technisch betreut. Die Kernaufgaben sind hierbei sowohl die Wartung und Bereitstellung der Server und des Tutortools als auch die Pflege der Datenbanken.

4.1.2 Gesamtprozess der Einstellung von Studierenden als Tutor

Um einen ersten Überblick über den gesamten Ablauf der Einstellung eines Tutors zu geben, wird in diesem Abschnitt zunächst der Gesamtprozess einer Einstellung beschrieben und

anhand einer BPMN 2.0 Darstellung zum besseren Verständnis für den Leser visualisiert.[15] Da der Prozess einer Einstellung den Kernprozess bei der Verwendung des Tutortools darstellt, wird dieser in diesem Abschnitt besonders hervorgehoben.

Der erste Schritt für eine Einstellung besteht darin, dass die betroffene Veranstaltung (Kapitel 4.1.3 - Anlegen einer Veranstaltung) vom Tutorbüro angelegt wird und der zugeteilte Übungsleiter identifiziert wird. Hiernach wird der Übungsleiter angelegt (Kapitel 4.1.3 - Anlegen von Übungsleitern) und der Veranstaltung zugeordnet. Nachdem die Veranstaltung fertig angelegt ist, können sich die Studierenden für das Fach, in dem sie sich als Tutor anstellen lassen wollen, bewerben. Hierzu muss der Studierende unter dem Menüpunkt „Tutor" das Feld „Bewerben" auswählen. In der Bewerbungsmaske hat der Studierende dann die Möglichkeit, über das Feld „Bewerbung ändern" sich durch Angabe von Note und Priorität (in diesem Kontext gibt die Priorität den Übungsleitern[16] beziehungsweise Mitarbeitern des Tutorbetriebs einen Anhaltspunkt darüber, welche Bewerbung bei Mehrfachbewerbungen aus Sicht des Studierenden zu priorisieren ist) für eine Tutorenstelle zu bewerben. Im nächsten Schritt muss ein Mitarbeiter des Tutorbüros der Bewerbung des Studierenden zusagen. Das heißt in diesem Kontext, dass die Bewerbung für die Veranstaltung akzeptiert wird und im Folgenden dem Übungsleiter angezeigt wird. In der Theorie ist es möglich, dass einer Bewerbung nicht zugesagt wird, dies kommt in der Praxis aber nur äußerst selten vor. Üblicher ist es hier, bei Massenbewerbungen eines Studierenden die Bewerbung zurückzuhalten und die höher priorisierten Bewerbungen zuerst zuzulassen. Im nächsten Schritt sichtet der Übungsleiter der Veranstaltung dann die ihm vorliegenden Bewerbungen und entscheidet, welche davon akzeptiert und welche abgelehnt werden. Nachdem der Studierende für die Veranstaltung als Tutor akzeptiert wurde, muss der Vertrag im nächsten Schritt von einem Mitarbeiter des Tutorbüros angelegt werden. Hierbei muss zunächst festgestellt werden, ob es sich um einen geteilten (gesplitteten) Vertrag[17] handelt. Für den Fall, dass der Tutor einen gesplitteten Vertrag benötigt, muss hier zunächst im Unterpunkt „Veranstaltung – Überblick" die betroffene Person noch einmal eingestellt werden. Hierzu muss die Veranstaltung ausgewählt werden und unter „Einstellen" der Studierende ausgewählt werden. Für den Fall, dass kein gesplitteter Vertrag benötigt wird, entfällt dieser Schritt.[18] Hiernach müssen die Vertragsdaten unter „Einstellung – Verträge" eingefügt werden. Nachdem dies erfolgt und abgespeichert ist, öffnet sich automatisch eine E-Mail Maske, mit der ein Hinweis an den betroffenen Studierenden geschickt werden kann, um diesem mitzuteilen, dass die Bewerbung akzeptiert wurde und die nötigen Dokumente eingereicht werden müssen. Für die Vertragsunterlagen ist es zudem wichtig, ob es sich bei der Anstellung um eine Weiterbeschäftigung handelt. Dies ist der Fall, falls der letzte Arbeitstag einer vorherigen Beschäftigung nicht länger als ein Jahr zurückliegt. Sobald alle benötigten Vertragsunterlagen

[15] Aufgrund des Umfangs des Prozesses und der entsprechenden Größe des daraus resultierenden BPMN 2.0 Diagramms wurde die Darstellung in den digitalen Anhang verschoben. Sämtliche hierin befindlichen Diagramme können beim Autor angefragt werden.
[16] Hier wird die Intention der Priorität beschrieben. In der alten Version des Tutortools haben die Übungsleiter jedoch nicht die Möglichkeit, die Note und gesetzte Priorität der Studierenden einzusehen.
[17] Für eine Erklärung zu den geteilten Verträgen siehe Kapitel 4.1.3 – Geteilte Verträge.
[18] Im beiliegenden BPMN 2.0 Diagramm ist der Schritt der Vertragserstellung als Subprozess „Vertrag erstellen" ausgelagert und kann in einem eigenständigen BPMN 2.0 Diagramm betrachtet werden.

vom Studierenden gesammelt und eingereicht wurden, müssen diese von den Mitarbeitern des Tutorbüros gesichtet und überprüft werden. Sofern die Unterlagen vollständig und korrekt vorliegen, kann der Vertrag, der die Einstellung betrifft, auf „Vollständig“ gesetzt werden und der dazugehörige Einstellungsvorschlag erstellt werden. Zusammen mit dem Einstellungsvorschlag werden die Dokumente im Anschluss an das Verwaltungsbüro weitergeleitet und der Status des Vertrags im Tutortool auf „In Bearbeitung“ gesetzt. Hier werden die Dokumente erneut geprüft und der Vertrag für die Anstellung erstellt. Sobald der Vertrag erstellt und an das Tutorbüro geschickt wurde, wird der Vertragsstatus im Tutortool auf „Unterschriftsbereit“ gesetzt und der Tutor benachrichtigt, dass der Vertrag nun zur Unterschrift bereitliegt. An dieser Stelle muss zwischen einem „normalen“ Semester und den Semestern während der zum Zeitpunkt der Arbeit vorherrschenden SARS-CoV-2 Pandemie unterschieden werden. Während eines normalen Semesters werden die Studierenden aufgefordert, während der Sprechstunde das Tutorbüro aufzusuchen und dort den Vertrag zu unterzeichnen. Um direkten Kontakt mit Personen zu vermeiden, werden während der Pandemie daher die Vertragsunterlagen zur Unterzeichnung an die Studierenden postalisch versandt, dort unterschrieben und anschließend wieder an das Tutorbüro zurückgeschickt. Abschließend wird der Vertrag von den Mitarbeitern des Tutorbüros an das Servicebüro Hilfskräfte (SB-Z) weitergeleitet und der Status des Vertrags im Tutortool auf „Abgeschlossen“ gesetzt. Bei Bedarf muss zusätzlich ein MiLoG[19]-Schreiben an die Übungsleitung weitergeleitet werden. Hiernach ist der Vorgang einer Einstellung für die Mitarbeiter des Tutorbüros beendet.

4.1.3 Kernprozesse bei der Verwendung des Tutortools

Durch die evolutionäre Entwicklung des alten Tutortools wurden Funktionalitäten implementiert, die zwar zum damaligen Zeitpunkt nötig beziehungsweise sinnvoll waren, jedoch nach zehn Jahren nicht mehr verwendet werden, oder optimiert werden können. In diesem Abschnitt werden diverse Prozesse bei der Verwendung des Tutortools genauer vorgestellt. Zum einen Prozesse, die für die Neukonzeption überarbeitet wurden und zum anderen Prozesse, die aufgrund ihrer Sinnhaftigkeit in die Neuimplementierung übernommen wurden.

Anlegen von neuen Übungsleitern

Das Anlegen eines neuen Übungsleiters erfolgt in der alten Version des Tutortools durch einen unnötig komplizierten Prozess, der zudem eine Sicherheitslücke darstellt. Für die Erstellung eines neuen Übungsleiters muss ein Mitarbeiter des Tutorbüros zunächst den für die Veranstaltung zuständigen Übungsleiter durch eine Anfrage am zuständigen Lehrstuhl identifizieren. Diese Person wird dann der technischen Betreuung des Tutorbetriebs mitgeteilt. Zunächst muss dann in der Datenbank des Tutortools überprüft werden, ob die identifizierte Person bereits einen Account mit der gleichen E-Mail-Adresse im Tutortool besitzt. Falls dies zutrifft, besagt die offizielle Dokumentation, dass der Account deaktiviert werden muss und um Komplikationen zu vermeiden, die (alte) E-Mail-Adresse mit einem vorangestellten Unterstrich versehen werden muss. Hierauf folgend muss der Übungsleiter nun neu im Tutortool angelegt werden.

[19] Mindestlohngesetz

Hierzu wird unter dem Menüpunkt „Administration- Verwaltung – Benutzer“ ein neuer Übungsleiter erstellt. Das Passwort kann dann entweder (falls ein alter Account besteht) übernommen werden, oder muss in der Datenbank neu gesetzt werden. Diesen Schritt muss eine Person mit Kenntnissen der Datenbank übernehmen. Hier tritt das Problem auf, dass die gesetzten Passwörter über eine unsichere Kommunikationsschnittstelle (E-Mail) versandt werden. Das BPMN 2.0 Diagramm für diesen Prozess kann dem digitalen Anhang entnommen werden.

Die Lösung dieses Problems besteht darin, dass von den Mitarbeitern des Tutorbüros ein eindeutiger Einladungslink für den entsprechenden Übungsleiter erstellt werden kann, der dann dem Übungsleiter per E-Mail zugeschickt wird. Dieser Einladungslink ist für zwei Tage gültig und verfällt, falls dieser nicht genutzt wird. Für den Fall, dass dieser Einladungslink von einer nicht intendierten Person übernommen wird, kann entweder die Rolle des hierdurch erstellten Accounts von den Mitarbeitern des Tutorbüros geändert werden oder der Account ganz gelöscht werden. Auch tritt hierdurch keine Sicherheitslücke auf, da das Erstellen eines Übungsleiteraccounts noch keinerlei Zugriff auf Datensätze gewährt. In Abbildung 8 ist dieser optimierte Prozess durch ein BPMN 2.0 Diagramm abgebildet.

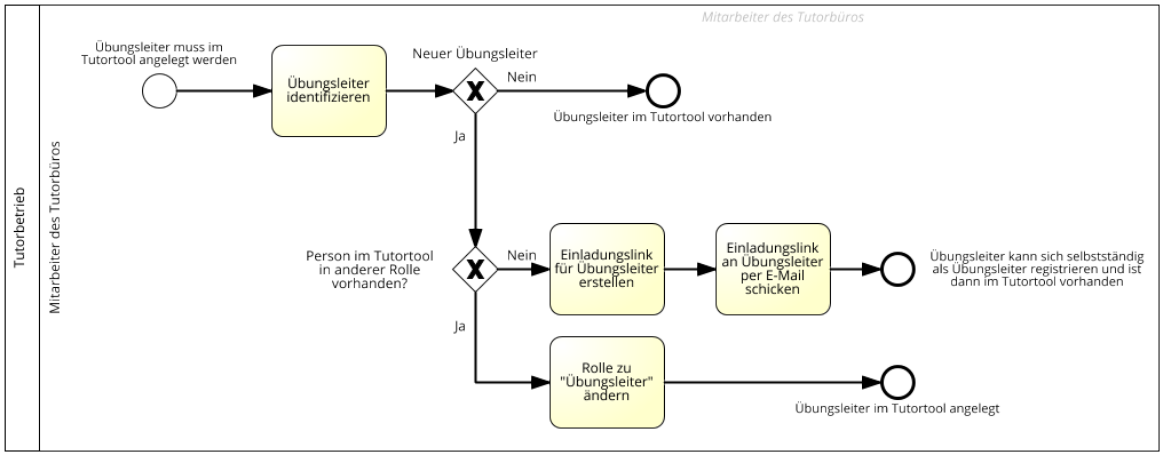


Abbildung 8: BPMN 2.0 Diagramm – Übungsleiter anlegen in der Neukonzeption
Quelle: eigene Darstellung

Anlegen einer Veranstaltung

Für das Anlegen einer Veranstaltung sind in der alten Version des Tutortools zwei separate Schritte erforderlich. Handelt es sich um eine Veranstaltung, für die bisher noch keine Tutorien abgehalten wurden, so muss zunächst eine sogenannte „Metaveranstaltung“ angelegt werden. Eine Metaveranstaltung stellt ein Modul dar, das von der TU München als Lehrveranstaltung angeboten wird. Hierzu werden von einem Mitarbeiter des Tutorbüros unter dem Menüpunkt „Metaveranstaltung“ alle für die Metaveranstaltung nötigen Informationen eingetragen. Hierzu zählen das Kürzel der Veranstaltung, die Bezeichnung, die Modulnummer, die Maßnahme (hier besteht eine Auswahlmöglichkeit aus Tutorien, Repetitorien, Programmierprojekte und Vorkurse), die Fondsnummer und die Kostenstelle.

Für den Fall, dass es bereits eine Metaveranstaltung für die anzulegende Veranstaltung gibt, kann der oben beschriebene erste Schritt weggelassen werden. Nun kann von einem Mitarbeiter

des Tutorbüros direkt eine Veranstaltung für das betroffene Semester angelegt werden. Der Prozess kann in Abbildung 9 betrachtet werden.

Dieses Konzept wurde anhand von zwei unterschiedlichen Kriterien in die Neukonzeption übernommen. Zum einen stellt das beschriebene System eine übersichtliche Vorgehensweise dar, um Veranstaltungen, die ein Semester betreffen, von den allgemeinen Veranstaltungen (hier Metaveranstaltungen genannt), wie sie in TUMonline dargestellt sind, zu unterscheiden. Hierdurch müssen Daten, die sich nicht häufig ändern (LV-Nr., LV-Titel, etc.) nicht jedes Semester erneut eingegeben werden. Zum anderen sind die Mitarbeiter des Tutorbüros bereits an diesen Prozess gewöhnt, wodurch beim Umstieg auf das neue Tutortool kein neues Vorgehen erlernt werden muss. Somit unterscheidet sich der Prozess in der Neukonzeption nicht wesentlich von dem Prozess, der im alten Tutortool verwendet wird. Die einzige Neuerung hierbei ist, dass in der Neukonzeption einer Veranstaltung deutlich mehr Informationen (Beispielsweise: Daten aus der Budgetplanung) beigefügt werden können, wodurch den Mitarbeitern des Tutorbüros die spätere Arbeit erleichtert wird.

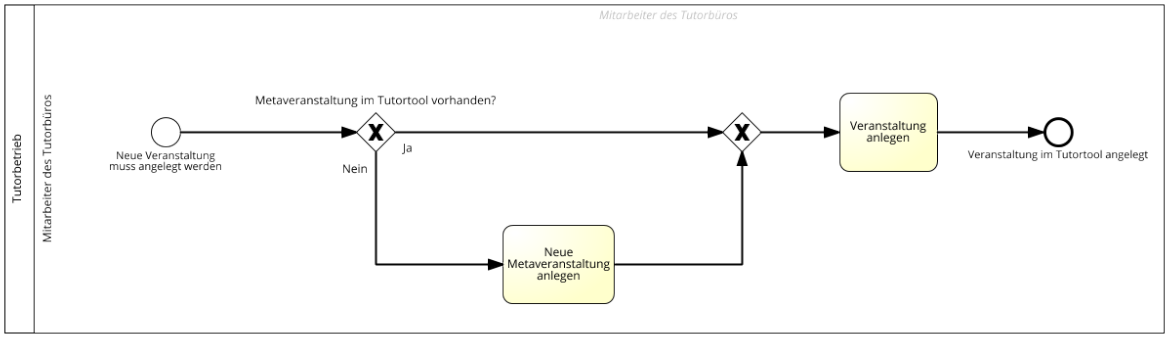


Abbildung 9: BPMN 2.0 Diagramm – Veranstaltung anlegen
Quelle: eigene Darstellung

Vertrag erstellen

Für das Erstellen eines Vertrags müssen in beiden Versionen des Tutortools die bestätigte Bewerbung des Studierenden ausgewählt und anschließend in der folgenden Ansicht alle erforderlichen Informationen bezüglich des Vertrags eingetragen werden. Abschließend muss der Studierende dann noch per E-Mail über den erstellten Vertrag informiert werden.

Zudem kann es bei der Erstellung von Verträgen erforderlich sein, dass der Vertragszeitraum in mehrere Abschnitte mit unterschiedlicher Wochenstundenzahl aufgeteilt wird. Dieser Vorgang wird im Umfeld des Tutorbetriebs als „Vertragssplitting“ bezeichnet. Solche geteilten Verträge werden üblicherweise verwendet, wenn die Wochenstundenzahl zwischen Vorlesungszeit und vorlesungsfreier Zeit variiert. Ein Grund hierfür kann zum Beispiel die Korrektur von Klausuren sein. In der alten Implementierung des Tutortools ist diese Praktik jedoch nicht darstellbar. Um dennoch einen geteilten Vertrag abzubilden, müssen zwei separate Verträge von den Mitarbeitern des Tutorbüros angelegt werden. Dies hat mehrere Folgen: Den Übungsleitern werden durch die doppelten Verträge die betroffenen Tutoren mehrfach in ihrer Liste angezeigt, was die Übersichtlichkeit der Benutzeroberfläche einschränkt und zu Verwirrung führen kann. Die Mitarbeiter des Tutorbüros müssen ohne Unterstützung des Tutortools selbstständig die Übersicht behalten, welche Dokumente bereits für beide Vertragszeiträume gültig

eingereicht wurden beziehungsweise welche Dokumente separat für jeden Vertragszeitraum eingereicht werden müssen. Hinzu kommt, dass beim Erstellen des Einstellungsvorschlags die Daten des zweiten Vertragszeitraums manuell hineineditiert werden müssen. Dem digitalen Anhang kann der nicht optimierte Prozess des alten Tutortools als BPMN 2.0 Diagramm entnommen werden.

Ein konkreter Lösungsansatz für dieses Problem ist, dass es für jeden Vertrag die Möglichkeit gibt, einen zweiten Vertragszeitraum zusammen mit einer eigenen Wochenstundenzahl direkt in das Vertragserstellungsformular einzufügen. Dieser zweite Vertragszeitraum knüpft automatisch direkt an den ersten an. Das bedeutet, sobald ein Startdatum für den zweiten Vertragszeitraum ausgewählt wird, wird dieses auf den Folgetag des Enddatums des ersten Zeitraums gesetzt. Die optimierte Version des Prozesses kann in Abbildung 10 betrachtet werden.

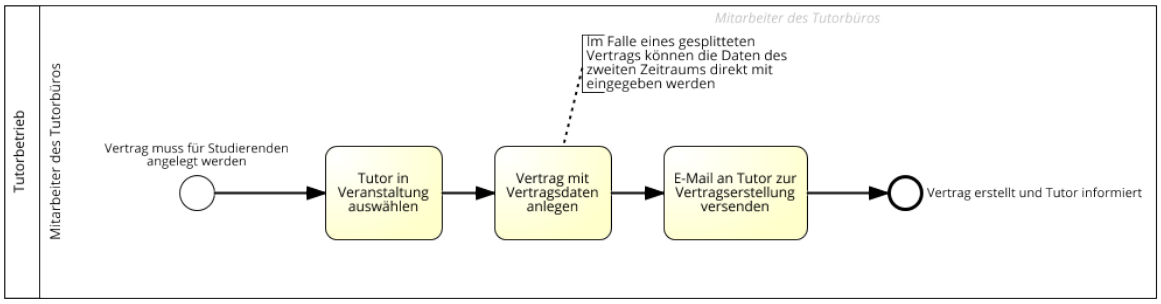


Abbildung 10: BPMN 2.0 Diagramm – Vertrag anlegen in der Neukonzeption
Quelle: eigene Darstellung

Hinzufügen und Ändern von Formularen

Ein wichtiger Prozess in der Verwendung des Tutortools ist das Hinzufügen und Ändern von Formularen, damit diese immer auf dem aktuellen Stand für die Tutoren verfügbar sind. Bei der

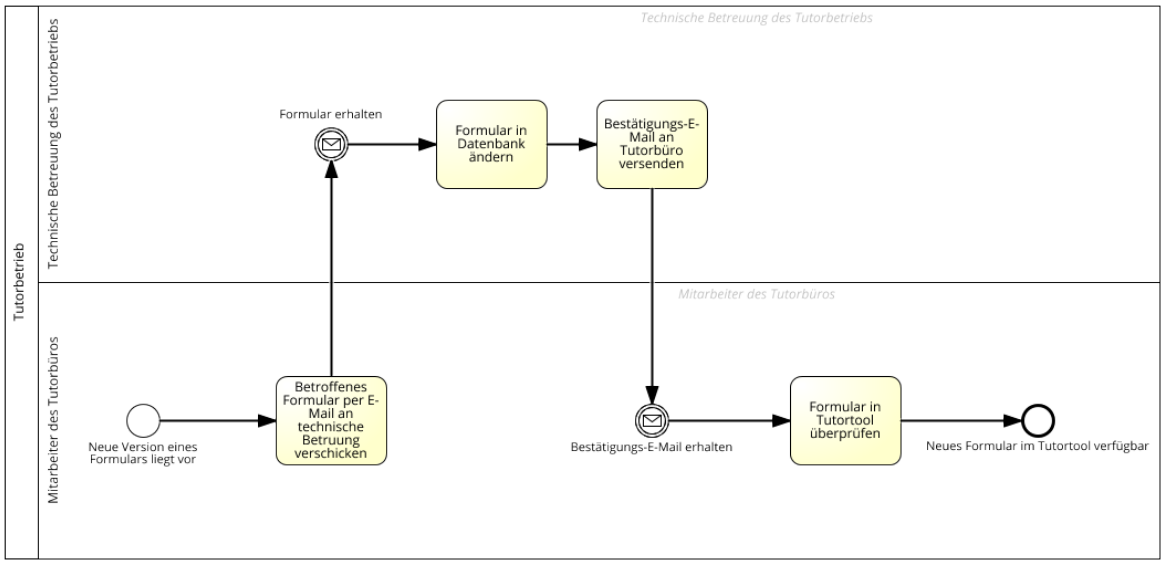


Abbildung 11: BPMN 2.0 Diagramm – Formular ändern im alten Tutortool
Quelle: eigene Darstellung

Einstellung von Studierenden als Tutoren müssen diverse Formulare ausgefüllt und vorgelegt werden. Hinzu kommt das Merkblatt für Tutoren, das jedes Semester erneuert wird.

Um ein Formular zu ändern, muss dieses zunächst per E-Mail der technischen Betreuung des Tutortools zugeschickt werden. Als nächstes muss die technische Betreuung dieses Formular dann in der Datenbank in der Tabelle „Formulare“ mit einer fortlaufenden ID eingefügt werden. Eine Ausnahme hierzu bildet das Formular „Einstellungsvorschlag“. Dieser muss zwingend mit ID 1 eingefügt werden und immer das alte Formular ersetzen. Im nächsten Schritt muss dann in der Tabelle „Document“ die neue ID des Formulars in die entsprechende Zeile der Spalte „Formulare“ eingetragen werden. Dieser alte Prozess wird in Abbildung 11 abgebildet.

Für den optimierten Prozess zur Änderung von Dokumenten soll es für die Mitarbeiter des Tutorbetriebs möglich sein, die benötigten Formulare über eine Upload-Ansicht direkt im Tutortool hochzuladen. Der im Gegensatz zum alten Tutortool stark vereinfachte Prozess, wie er in der Neuimplementierung eingefügt werden soll, kann in Abbildung 12 betrachtet werden.

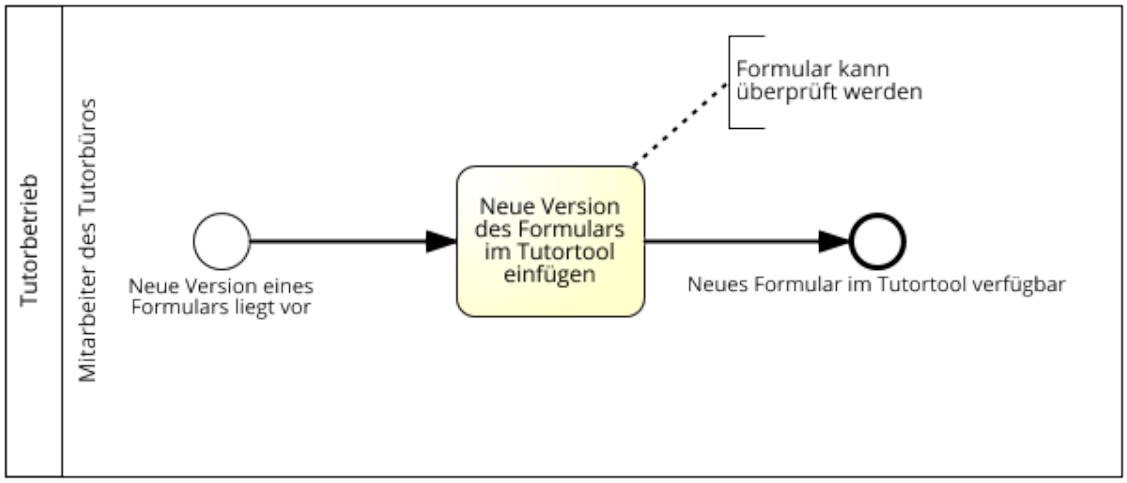


Abbildung 12: BPMN 2.0 Diagramm – Formular ändern in der Neukonzeption
Quelle: eigene Darstellung

Bewerben für Tutorstelle

Damit sich Studierende auf eine offene Tutorstelle bewerben können, sind auch in der alten Version des Tutortools nur wenige Schritte nötig. Zunächst muss sich der Studierende in seinen Account einloggen und dann unter „Tutor“ und „Bewerben“ die Priorität[20] und die Note für die offene Stelle eintragen. Dieses Konzept wurde für die Neukonzeption übernommen und lediglich um die Möglichkeit erweitert, mit dem Feld „Details“ der Bewerbung zusätzliche Informationen (wie besondere Qualifikationen) hinzuzufügen.

[20] Unter „Priorität“ ist in diesem Kontext die vom Studierenden gesetzte Priorität für das Belegen einer Tutorstelle zu verstehen, gesetzt dem Fall, dass sich der Studierende auf mehrere Stellen bewirbt.

4.1.4 Probleme im alten Tutortool

Für diesen Relevance Cycle wurden, wie bereits in der Einleitung zu Kapitel 4 erwähnt, verschiedenen Personen aus den Nutzergruppen, die das alte Tutortool bereits öfter verwendet haben, ein Fragebogen zur Ermittlung des SUS-Werts (Brooke, 1996) des Systems zugeschickt. Das in den Freitextfeldern dieser Befragungen ermittelte Feedback zum alten Tutortool gab Rückschlüsse auf Probleme die aktuell bei der Verwendung des Tutortools auftreten. In diesem Abschnitt werden die wichtigsten Probleme, die bei der Verwendung des alten Tutortools auftreten, genau dargelegt und die erfolgten Optimierungen an den Funktionalitäten beschrieben.

Noten werden den Übungsleitern nicht angezeigt

In der alten Version des Tutortools ist es den Studierenden möglich, die Note anzugeben, mit der sie die Prüfungsleistung für die Lehrveranstaltung abgelegt haben, für die sie sich als Tutor bewerben. Die angegebenen Noten sind jedoch bislang nur für die Mitarbeiter des Tutortools beziehungsweise die Administratoren sichtbar. Für die Neukonzeption wurde dies so angepasst, dass auch die Übungsleiter die Noten der Studierenden in den Bewerbungen einsehen können.

Daten Export

Für die Übungsleiter stellt der Export der personenbezogenen Tutordaten im alten Tutortool ein Hindernis dar. Dies hat mehrere Gründe. Bislang ist es lediglich möglich, die Gesamtheit aller Daten der Studierenden, die sich für eine Stelle als Tutor im betreffenden Fach beworben haben, gesammelt zu exportieren. Des Weiteren ist der Export nur im Format einer Excel Tabelle möglich. Dies hat den Nachteil, dass Lehrstühle, die die Daten nicht in Binärformaten abspeichern (zum Beispiel bei der Verwendung einer Git Versionsverwaltung), den Export nicht für ihre Zwecke verwenden können. Zudem enthalten die aktuell verfügbaren Exporte lediglich die Namen der Studierenden und die E-Mail-Adressen.

Für die Neuimplementierung wird der Datenexport für die Übungsleiter überarbeitet. Diese Überarbeitung umfasst mehrere Punkte. Zunächst werden die Informationen, die exportiert werden können, erweitert. So werden zusätzlich zu Vor- und Nachnamen sowie E-Mail-Adressen der Tutoren auch die Daten der Vertragszeiträume mit exportiert. Diese umfassen Vertragsstartdatum, Vertragsenddatum und die wöchentliche Stundenzahl für den Vertragszeitraum. Des Weiteren können Übungsleiter nun die Daten sowohl als Excel Datei als auch als „.csv“ Datei exportieren. Um zu vermeiden, dass die Gesamtheit aller Bewerber auf eine Tutorenstelle kombiniert in den Exporten inkludiert werden, wurde der Export so angepasst, dass nur die Daten der Tutoren exportiert werden, für die ein Vertrag für die Veranstaltung angelegt wurde.

Mehrere Übungsleiter

In der alten Version des Tutortools ist es nicht möglich, einer Veranstaltung mehrere Übungsleiter zuzuordnen. Somit besteht kein Zugriff für andere Übungsleiter auf die Veranstaltung im Tutortool, was zu Problemen führen kann falls der eingetragene Übungsleiter verhindert ist. Dies kann ebenfalls Verzögerungen in der Einstellung zur Folge haben. In der neuen Implementierung wird dieses Problem umgangen, indem bis zu drei Übungsleiter einer Veranstaltung zugeordnet werden können.

Kommunikation mit dem Tutorbüro

In der alten Version des Tutortools gibt es keine Möglichkeit der direkten Kommunikation zwischen Übungsleitern und den Mitarbeitern des Tutorbüros. Stattdessen müssen alternative Kommunikationswege wie beispielsweise E-Mails oder Telefonate gewählt werden. Gesetzt dem Fall, dass ein Übungsleiter lediglich eine kurze Nachricht an das Tutorbüro übermitteln möchte, entsteht hierbei viel Overhead, der potenziell vermieden werden kann. Hierzu wurden in der neuen Version mehrere Mechanismen eingeführt, um die Kommunikation zwischen Übungsleitern und dem Tutorbüro zu vereinfachen.

Zunächst wurde eine allgemeine Kommentarfunktion für die Veranstaltung eingeführt. Hier haben Übungsleiter die Möglichkeit, Vorschläge und Wünsche zu äußern. Zusätzlich wurde eine Kommentarfunktion für einzelne Verträge der Implementierung hinzugefügt, damit Informationen direkt den entsprechenden Verträgen beigefügt werden können. Für die Mitarbeiter des Tutorbüros (falls zusätzliche Informationen zu einer Veranstaltung oder einer Bewerbung/Vertrag vorliegen) wird dies über ein rotes Ausrufezeichen im „Details“ Button der Veranstaltung beziehungsweise dem „Kommentar“ Button der Bewerbung kenntlich gemacht.

4.2 Prozesse und Funktionalitäten in der Neukonzeption

Dieser Abschnitt gibt einen Überblick über die wichtigsten Veränderungen in der Neukonzeption des Tutortools im Vergleich zur alten Version. Dabei wird speziell auf die Prozesse eingegangen, die in dieser Form nicht in der alten Version des Tutortools existieren. Nicht behandelt werden Prozesse wie das Erstellen eines Vertrags, die bereits im alten Tutortool existieren und an denen, bezogen auf die Prozessstruktur, nichts verändert wurde.

4.2.1 Nutzergruppen in Verbindung mit der Neukonzeption

Bei der Benutzung der Neukonzeption des Tutortools treten die gleichen Nutzergruppen auf wie bei der Verwendung des alten Tutortools. Da in Design Cycle 4 das Verleihtool in die Neukonzeption integriert wurde, tritt in Zukunft bei der Verwendung eine weitere Nutzergruppe auf. Diese Nutzergruppe besteht aus den Mitarbeitern der RBG. Die Hauptaufgabe der Nutzer dieser Rolle besteht darin, im in Design Cycle 4 integrierten Verleihtool die Leihe von Hardware zu bestätigen und die zurückgegebene Hardware wieder abzuzeichnen. Eine Besonderheit dieser Nutzergruppe ist, dass die Mitarbeiter des RBG das Dashboard nicht nutzen und direkt an das integrierte Verleihtool weitergeleitet werden.

4.2.2 Gesamtprozess der Einstellung von Studierenden als Tutor

Der Gesamtprozess einer Einstellung in der Neuimplementierung umfasst ähnliche Schritte wie der Gesamtprozess einer Einstellung im alten Tutortool. Hierbei tritt im Ablauf des Prozesses nur eine große Veränderung auf, die aber in diesem Abschnitt kurz beschrieben werden. Die meisten Veränderungen, die vorgenommen wurden, treten hierbei in den Subprozessen auf, die bereits in den Kapiteln 4.1.3 und 4.1.4 beschrieben wurden.

Die größte Änderung, die im Gesamtprozess auftritt und auch im zugehörigen BPMN 2.0 Diagramm[21] ersichtlich ist, ist, dass die Bewerbungen von Studierenden nicht mehr von den Mitarbeitern des Tutorbüros akzeptiert werden müssen (siehe für Details hierzu Kapitel 4.2.5 – Akzeptieren von Bewerbungen). Die Bewerbungen sind für die Übungsleiter direkt sichtbar und bedürfen keiner Bestätigung mehr.

4.2.3 Neue Prozesse in der Neukonzeption

Durch die Neukonzeption des Tutortools wurden auch diverse neue Funktionalitäten eingefügt, die bisher nicht in Kontext mit dem Tutortool standen. So wurde im vierten Design Cycle das Verleihtool (inklusive der darin abgebildeten erweiterten Prozesse) mit in das Tutortool aufgenommen. Der Verleihprozess stellt somit den einzigen gänzlich neuen Prozess im Tutortool dar. Im Folgenden wird der Prozess so vorgestellt, wie er bis zum Zeitpunkt der Neukonzeption unter Verwendung des separaten Verleihtools üblich war. Anschließend wird der neue Prozess (nach Implementierung in die Neukonzeption) erneut beschrieben.

Für die Bewerbung von Leihgeräten war es bis zur Integrierung des Prozesses üblich, dass die Tutoren eine Anfrage um Leihmaterial per E-Mail an das Tutorbüro schickten. Im besten Fall enthielt diese Anfrage alle benötigten Informationen, um eine Leihe anzulegen. Bei den benötigten Informationen handelt es sich um die gesamte Anschrift des Tutors, die LRZ-Kennung des Tutors und die benötigten Leihgeräte. Falls die Anfrage nicht vollständig war, war zusätzliche E-Mail-Korrespondenz zwischen dem Tutor und dem Tutorbüro erforderlich. Anschließend musste jede Leihe im „Verleihtool“ einzeln angelegt werden und alle Daten aus der E-Mail-Korrespondenz eingefügt werden. Waren die Daten vollständig, wurde der Leihschein erstellt und an den Tutor versandt. Sobald der Schein unterschrieben zurückgeschickt wurde, konnte das Leihmaterial bei der RBG abgeholt werden. Dort wurde dann im Verleihtool der Status der Leihe auf „HW[22] ausgegeben“ gesetzt. Nach Ende einer Leihe wurde, sobald die Hardware bei der RBG zurückgegeben wurde, der Status der Leihe auf „Abgeschlossen“ gesetzt, wodurch der Leihvorgang beendet wurde.

In der Neukonzeption des Tutortools, können die Tutoren eine Leihanfrage direkt aus dem Tutortool heraus stellen. Hierbei ist das Erstellen einer Anfrage nur dann möglich, wenn auch alle benötigten Informationen der Leihanfrage beiliegen. Nach einer Bewerbung durch einen Tutor muss dann die Leihe ebenfalls von einem Mitarbeiter des Tutorbüros angelegt werden. Hierzu muss der Mitarbeiter des Tutorbetriebs in der Verleihübersicht den Unterpunkt „Leihanfragen“ auswählen. Hierin werden dann alle noch nicht bearbeiteten Leihanfragen angezeigt. Wird eine Leihe ausgewählt, werden alle bereits zur Verfügung stehenden Informationen in das Leihformular eingetragen. Sobald die Informationen vervollständigt und geprüft sind, kann die

[21] Aufgrund des Umfangs des Prozesses und der entsprechenden Größe des daraus resultierenden BPMN 2.0 Diagramms wurde die Darstellung in den digitalen Anhang verschoben. Die hierin befindlichen Diagramme können beim Autor angefragt werden.
[22] Hardware

Leihe angelegt werden. Nun kann der Leihschein erstellt und an den Tutor verschickt werden. Ab diesem Zeitpunkt sind der alte und der neue Verleihprozess exakt gleich.

Für die Abbildungen dieser Prozesse mit BPMN 2.0 wurden für den alten Prozess und den Prozess unter Verwendung der Neukonzeption jeweils getrennte Diagramme angefertigt. Die angefertigten Diagramme können dem digitalen Anhang entnommen werden[23].

4.2.4 Wichtige ermittelte Funktionalitäten für die Neukonzeption

Dieser Abschnitt behandelt die Funktionalitäten, die auf Basis von Prozessen aus der alten Version des Tutortools und der hierzu durchgeführten Befragungen zu Beginn dieses Relevance Cycles ermittelt wurden.

DSGVO Auskunft

Die alte Version des Tutortools bietet den Mitarbeitern des Tutorbüros keinerlei Möglichkeit, ohne großen Aufwand selbstständig eine DSGVO Auskunft bei entsprechender Anfrage abzuwickeln. Hierzu muss derzeit ein technischer Mitarbeiter konsultiert werden, der daraufhin über SQL-Anfragen die benötigten Informationen direkt aus der Datenbank extrahieren muss. Um diesen Prozess zu optimieren, wurde eine „One-Click“ Lösung entwickelt, bei der ein Dokument mit allen benötigten Daten erstellt wird.

Budgetübersicht

Aus der zu Beginn dieses Cycles durchgeführten Befragung der Mitarbeiter des Tutorbüros ging hervor, dass es sich oftmals schwierig gestaltet, die Zahlen der Tutoren in Relation zu den vorgesehenen Wochenstunden beziehungsweise Gesamtwochenstunden der Veranstaltung zu setzen. Hierzu muss aktuell die Excel Tabelle der Budgetplanung[24] zu Rate gezogen werden und die Wochenstunden der bereits eingestellten Tutoren im betroffenen Zeitraum berechnet werden. In dieser Tabelle sind vor allem die wöchentlichen Gesamtstunden für die Mitarbeiter des Tutorbetriebs relevant, da anhand dieser die noch offenen Stellen ermittelt werden. Zusätzlich hierzu legen die Mitarbeiter des Tutorbetriebs aktuell für jede Veranstaltung im betroffenen Semester eine Excel Tabelle an, in der die Vertragsdaten der Tutoren zusammengefasst werden, um diese anschließend mit den Zahlen aus der Budgetplanung zu vergleichen. Um hier zusätzlichen Arbeitsaufwand zu reduzieren, können die Daten dieser Tabelle in der Neukonzeption bei der Erstellung einer Veranstaltung miteingefügt werden. Eine weitere sinnvolle Funktionalität in diesem Kontext ist eine Budgetübersicht, über die die noch offenen und bereits besetzten Stellen eingesehen werden können.

[23] Der Inhalt des digitalen Anhangs kann beim Autor angefragt werden.
[24] Diese Excel Tabelle wird vor jedem Semester durch den Budgetausschuss erstellt und dient als Grundlage für die Anzahl der Einstellungen für ein Semester. Kleinere Abweichungen können dennoch bei Bedarf vorgenommen werden.

Semesterstatistik/Vertragsstatistik

Eine Übersicht über den aktuellen Stand der Verträge für das gegenwärtige oder vergangene Semester lässt sich über die alte Version des Tutortools nicht abrufen. Hierfür muss über eine SQL-Abfrage die Datenbank ausgelesen werden, was die Mitarbeiter des Tutorbüros vor große Probleme stellt. Derzeit muss hierzu von einem technischen Mitarbeiter die gewünschte SQL-Abfrage für das Semester erstellt werden, die dann von den Mitarbeitern des Tutorbüros weiterverwendet wird.

Um diesem Problem Abhilfe zu schaffen, ist für die Neukonzeption die Möglichkeit einer direkten Semesterstatistik-Abfrage sinnvoll. Hierbei sollen die Mitarbeiter sofort Auskunft über den Stand aller Verträge in den ausgewählten Semestern bekommen. Für diese Abfrage wurden die folgenden Anforderungen erhoben:

- Auswahl des betroffenen Semesters
- Datum der Abfrage
- Anzahl Verträge mit Status „Erstellt“
- Anzahl Verträge mit Status „Unvollständig“
- Anzahl Verträge mit Status „In Bearbeitung“
- Anzahl Verträge mit Status „Unterschriftsbereit“
- Anzahl Verträge mit Status „Abgeschlossen“
- Gesamtzahl Verträge im ausgewählten Semester
- Export aller Vertragsdaten für das ausgewählte Semester als „.xlsx“ Datei

Automatische Erkennung von Weiterbeschäftigungen

Bei der Erstellung von Verträgen ist es für die Mitarbeiter des Tutorbetriebs für das Anfordern der benötigten Unterlagen erforderlich zu wissen, ob es sich bei der Einstellung um eine Neueinstellung oder eine Weiterbeschäftigung handelt. Um eine Weiterbeschäftigung handelt es sich in diesem Kontext, wenn das Datum des Vertragsbeginns (neuer Vertrag) nicht länger als ein Jahr nach dem Enddatum der letzten vom betroffenen Tutor erfüllten Beschäftigung liegt. Im Falle einer Neueinstellung müssen vom Bewerber alle für die Vertragserstellung nötigen Dokumente eingereicht werden. Für den Fall, dass es sich um eine Weiterbeschäftigung handelt, müssen die folgenden Dokumente nicht erneut eingereicht werden:

- Fragebogen zu Scientology
- Fragebogen zur Verfassungstreue
- Krankenkassenbescheinigung
- Personalbogen Bezügestelle
- Personalbogen für Studierende
- Bescheinigung des Finanzamts über die SteuerID Nummer

In der Neukonzeption wird eine mögliche Weiterbeschäftigung durch das Programm automatisch erkannt und der Mitarbeiter des Tutorbüros in der Vertragerstellungs- beziehungsweise Vertragsbearbeitungsansicht darauf hingewiesen. Nachdem eine Weiterbeschäftigung erkannt wurde und der zuständige Mitarbeiter des Tutorbüros dies bestätigt, werden die Felder für die oben aufgeführten Dokumente automatisch auf „Liegt vor“ gesetzt.

Aufenthaltstitel Warnung

Bei der Verwendung der alten Version des Tutortools kann die Einstellung von Studierenden aus Drittländern[25] zu Problemen führen. Diese entstehen dadurch, dass die Mitarbeiter des Tutorbüros zum Zeitpunkt der Vertragserstellung zum Teil noch nicht den aktuellen Aufenthaltstitel des Studierenden beziehungsweise das Datum, bis zu welchem dieser gültig ist, vorliegen haben. In der Neuimplementierung wird dieses Problem dadurch gelöst, dass anhand der Nationalität des Studierenden automatisch erkannt wird, ob ein Aufenthaltstitel nötig ist und falls dies der Fall ist, der Studierende darauf hingewiesen wird, das entsprechende aktuell gültige Datum einzutragen. Die Mitarbeiter des Tutorbüros erhalten parallel dieselbe Meldung, sodass die Studierenden eventuell erneut darauf hingewiesen werden können. Die umgesetzte Implementierung dieser Funktionalität kann in Abbildung 21 (Anhang B.1) betrachtet werden.

Hinweis Prüfung durch Verfassungsschutz

Für den Fall, dass ein Bewerber eine Staatsbürgerschaft aus einem Land besitzt beziehungsweise in einem Land geboren ist, welches in Teil 2 Punkt 4.2 der „Bekanntmachung der Bayerischen Staatsregierung über die Pflicht zur Verfassungstreue im öffentlichen Dienst (Verfassungstreue-Bekanntmachung – VerftöDBek) vom 3. Dezember 1991 (AllMBl. S. 895, StAnz. Nr. 49), die zuletzt durch Bekanntmachung vom 27. September 2016 (AllMBl. S. 2138) geändert worden ist“ definiert wurde, werden die Mitarbeiter des Tutorbüros automatisch sowohl beim Anklicken des Profils des Studierenden als auch bei Vertragserstellung und -bearbeitung darüber informiert, dass eine Prüfung des Verfassungsschutzes vor der Einstellung des Studierenden nötig ist. Dies ist sinnvoll, da eine solche Prüfung nicht selten mehrere Wochen in Anspruch nimmt und sich so eine Anstellung des Studierenden als Tutor deutlich hinauszögern kann. Durch diese Funktion können frühzeitig Schritte eingeleitet werden, um diesen Prozess durch eine priorisierte Bearbeitung des Einstellungsvorschlags zu beschleunigen, damit ein fristgerechter Arbeitsantritt gewährleistet werden kann.

Filtermöglichkeiten für Übungsleiter

Bereits in den letzten Jahren bekamen die Mitarbeiter des Tutorbüros Feedback zur alten Version des Tutortools. Ein wichtiger Gesichtspunkt war dabei das Einfügen von Filtermöglichkeiten für die Bewerbungen auf die Tutorstellen. Die Hauptfiltermöglichkeit sollte visualisieren, welche Bewerbungen seit dem letzten Login neu hinzugekommen sind. Diese Änderungen sollen eine benutzerfreundlichere Arbeitsumgebung kreieren, mit der die Übungsleiter einfacher ihre Aufgaben mit dem Tutortool erledigen können.

4.2.5 Entfernte Funktionalitäten und Prozesse aus dem Tutortool

Aufgrund des Alters und evolutionären Wachstums der alten Version des Tutortools sind viele integrierte Funktionalitäten überholt und werden nicht mehr aktiv genutzt. Hierdurch treten

[25] Von einer Herkunft aus einem Drittland wird gesprochen, wenn der Studierende keine deutsche Staatsangehörigkeit, Staatsangehörigkeit eines Mitgliedsstaates der EU, der Schweiz und der EWR-Länder (Island, Liechtenstein und Norwegen) besitzt.

mehrere Probleme auf. Dadurch wird die Datenbank unnötig komplexer, da die ungenutzten Datensätze nach wie vor vorhanden sind. Ein Löschen dieser ungenutzten Datensätze hat jedoch zur Folge, dass nicht nur die zugehörigen SQL-Abfragen nicht mehr ausgeführt werden können, sondern auch Abfragen, die die Tabellen über JOIN-Abfragen miteinbeziehen. Ein weiteres Problem, das durch diese ungenutzten Funktionalitäten auftritt, ist, dass die Benutzeroberfläche unnötig verkompliziert wird. In diesem Abschnitt werden daher die überholten Funktionalitäten beschrieben und Gründe aufgeführt, warum diese in der Neukonzeption nicht mehr umgesetzt werden.

Seminare

Unter diesem Menüpunkt wurden bis zum Wintersemester 2018 die Teilnehmer der Veranstaltung „Didaktisches und pädagogisches Training für Tutoren[26] (IN9028)“ organisiert. Bei dieser Veranstaltung handelt es sich um ein nicht verpflichtendes Seminar, durch das Studierende, die im gleichen Semester eine Tutorstelle belegen, nebenbei zusätzliche ECTS Punkte für den Bereich der Schlüsselqualifikationen in ihrem Studiengang erlangen können. Diese Veranstaltung wird seit 2018 über das Campus Management System der TU München - TUMonline – organisiert, weswegen dieser Unterpunkt im Tutortool nicht mehr benötigt wird. Zum jetzigen Zeitpunkt ist nur noch eine kurze Prüfung erforderlich, ob die Teilnehmer im Semester ihrer Teilnahme am Seminar eine Tutorstelle belegen.

Hospitation

Bis zum Wintersemester 2018/2019 war es für neue Tutoren erforderlich, eine Hospitation bei einem erfahrenen Tutor zu absolvieren, bevor sie selbstständig Übungsgruppen betreuen. Um dies zu organisieren, verfügt die alte Version des Tutortools über die Option „Hospitation“ in der Menüauswahl. Dadurch, dass die Hospitationen nun nicht mehr zwingend erforderlich sind, wird diese Option nicht mehr genutzt. Somit wird auch in der Neuimplementierung keine Funktionalität für die Organisation von Hospitationen umgesetzt. Hierzu gehören auch alle diesem Punkt zugehörigen Unteransichten, wie der Hospitationskalender oder die Übungsgruppeneinteilung.

Akzeptieren von Bewerbungen

Derzeit müssen die Mitarbeiter des Tutorbüros, nachdem sich Studierende auf eine Stelle beworben haben, diese dem Übungsleiter zur Durchsicht freigeben. In der Vergangenheit hatte dies den Hintergrund, dass die Mitarbeiter des Tutorbüros eine Vorauswahl bei den Bewerbungen treffen sollten, um nicht in Frage kommende Kandidaten auszusortieren. Hierdurch sollte die Arbeitslast der Übungsleiter reduziert werden. Faktisch wird dies jedoch nur bei „Massenbewerbungen“ oder offensichtlich nicht den Ansprüchen entsprechenden Bewerbungen durchgeführt. Massenbewerbungen stellen ein Problem dar, da Tutoren nur für maximal 20 Wochenstunden angestellt werden dürfen. Üblicherweise können aufgrund dieser Obergrenze nur bis zu zwei Tutorstellen von einem Studierenden gleichzeitig bedient werden. Für den Fall,

[26] Geläufig auch als „Tutorseminar“ bekannt.

dass sich ein Studierender für mehr als zwei Stellen bewirbt und bei mehreren Veranstaltungen als Tutor akzeptiert wird, führt dies zu Komplikationen bei der Bearbeitung durch die Mitarbeiter des Tutorbüros. Zunächst muss beim Studierenden angefragt werden, für welche zwei Stellen ein konkreter Vertrag erstellt werden soll. Daraufhin werden die restlichen Bewerbungen zurückgezogen und die Übungsleiter informiert, dass eine erneute Auswahl getroffen werden muss. In der Neukonzeption des Tutortools wird dies dadurch umgangen, dass sich ein Studierender nur für maximal drei Stellen gleichzeitig bewerben kann, welche nach Priorität sortiert werden können. Daher kann der Schritt der Vorauswahl aus der Neukonzeption gestrichen werden.

Vorstellungsgespräche

Das alte Tutortool umfasst die Möglichkeit, Tutoren zu Vorstellungsgesprächen einzuladen. Diese Option wurde hauptsächlich bis 2018 genutzt, danach kaum noch. Für die neue Implementierung des Tutortools wurde in Rücksprache mit den Mitarbeitern des Tutorbüros beschlossen, diese Option aus dem Tutortool zu entfernen. „Entfernt“ bedeutet in diesem Zusammenhang lediglich eine Abkopplung aus dem Tutortool, damit keine ungenutzten Funktionen die Neukonzeption unnötig verkomplizieren. Den Übungsleitern ist es trotzdem noch freigestellt, in Frage kommende Bewerber zu einem Vorstellungsgespräch einzuladen.

4.2.6 Gesamtübersicht Funktionalitäten für die Neukonzeption

In den bisherigen Abschnitten des Relevance Cycles wurden die Prozesse des alten Tutortools beschrieben und optimiert, Veränderungen im Gesamtprozess beschrieben und erweiterte Funktionalitäten für eine Neukonzeption ermittelt. Auf dieser Basis wird im folgenden Kapitel eine Gesamtübersicht über alle benötigten Funktionalitäten (in tabellarischer Form) für jede Nutzergruppe gegeben, die die Neukonzeption beinhalten muss. Dabei wurde die technische Betreuung des Tutorbetriebs nicht berücksichtigt, da diese Nutzergruppe keine direkten Funktionalitäten in der Neukonzeption benötigt.

Die zuvor genannten, benötigten grundlegenden Funktionalitäten bilden die Grundlage für die Umsetzung der Neukonzeption in den Design Cycles (Kapitel 5). Eine Ausnahme bilden hierbei die Funktionalitäten, die das Verleihtool umfassen. Diese wurden während dieses Relevance Cycles erhoben, stellen aber keine Kernfunktionalität des Tutortools dar. Die Funktionalitäten des zu integrierenden Verleihtools werden im Folgenden trotzdem der Vollständigkeit halber aufgeführt. Erweiterte Funktionalitäten wie besondere Eingabefelder (z.B.: mehrere Übungsleiter in der Veranstaltungserstellung) werden nur dann angegeben, wenn diese auch in den bisherigen Abschnitten von Kapitel 4 Erwähnung fanden. Die Sortierung der Funktionalitäten in den Tabellen erfolgt anhand ihrer kategorischen Zugehörigkeit (Bspw.: Benutzer, Vertrag).

Allgemeine Funktionalitäten

Die in diesem Abschnitt vorgestellten Funktionalitäten treffen auf alle Nutzer des Tutortools zu und sind daher nutzergruppenübergreifend.

Bezeichnung Funktionalität	Kurzbeschreibung
Benutzerregistrierung	Benutzer können sich im System registrieren
Benutzeranmeldung	Benutzer können sich in ihren erstellten Account einloggen
Erstellen Benutzerprofil	Benutzer können ein Benutzerprofil erstellen
Bearbeiten Benutzerprofil	Benutzer können ihr Benutzerprofil bearbeiten
Passwort Ändern	Das Benutzerpasswort muss vom Benutzer selbst geändert werden können
Passwort Wiederherstellen	Im Falle, dass das Passwort eines Accounts verloren geht, muss ein neues angefordert werden können

Tabelle 7: Allgemeine Funktionalitäten der Neukonzeption
Quelle: eigene Erhebung

Mitarbeiter des Tutorbüros

Die in diesem Abschnitt aufgelisteten Funktionalitäten sind für die Nutzergruppe der Mitarbeiter des Tutorbüros intendiert.

Bezeichnung Funktionalität	Kurzbeschreibung
Übersicht aller Benutzer im System	Übersicht über alle registrierten Benutzer (filterbar, durchsuchbar)
Einzelansicht der Benutzer	Ansicht von Benutzerprofilen
Rolle Ändern von Benutzern	Das Ändern der Benutzerrolle aus dem Tutortool heraus
Bearbeiten von Benutzerprofilen	Daten in Benutzerprofilen sind von den Mitarbeitern bearbeitbar
DSGVO Export	Die Daten der Benutzerprofile sind für DSGVO-Auskunftsanfragen exportierbar
Hinweis Prüfung durch Verfassungsschutz	Die Mitarbeiter bekommen wie in Kapitel 4.2.4 beschrieben einen Hinweis im Profil, falls eine Verfassungsprüfung notwendig ist
Anlegen/Bearbeiten von Semestern	Anlegen/Bearbeiten von Semestern im System
Übersicht von Semestern	Übersicht von angelegten Semestern im System
Anlegen/Bearbeiten von Metaveranstaltungen	Anlegen/Bearbeiten von Metaveranstaltungen im System
Übersicht von Metaveranstaltungen	Übersicht von angelegten Metaveranstaltungen im System (durchsuchbar)
Anlegen/Bearbeiten von Veranstaltungen	Anlegen/Bearbeiten von Veranstaltungen im System
Einfügen mehrerer Übungsleiter in Veranstaltung	Möglichkeit mehrere Übungsleiter für eine Veranstaltung einzufügen
Übersicht von Veranstaltungen	Übersicht von angelegten Veranstaltungen im System (filterbar, durchsuchbar)
Budgetübersicht für alle Veranstaltungen in Semester	Auflistung aller Veranstaltung mit dazugehörigem Budget
Budgetübersicht für einzelne Veranstaltungen	Detaillierte Budgetübersicht für eine ausgewählte Veranstaltung (mit Überschreitungswarnung)
Bewerberübersicht für einzelne Veranstaltungen	Übersicht über alle Bewerbungen auf eine Veranstaltung
Exportmöglichkeit aller Verträge für Veranstaltung	Export aller Vertragsdaten für eine Veranstaltung
Akzeptieren/Ablehnen von Bewerbungen	Akzeptieren/Ablehnen von Bewerbungen von Studierenden auf offene Tutorstellen
Kommentarfeld Bewerbungen	Kommentarfeld durch das Übungsleiter und Mitarbeiter des Tutorbüros kommunizieren können
Einzelansicht von Bewerbungen	Detaillierte Einzelansicht von Bewerbungen

Bezeichnung Funktionalität	Kurzbeschreibung
Erstellen von zusätzlichen Verträgen ohne Bewerbung	Erstellen eines Vertrags, nachdem die Bewerbung akzeptiert wurde
Vertrag bearbeiten	Bearbeiten von erstellten Verträgen
Vertragsübersicht	Übersicht über alle erstellen Verträge (filterbar, durchsuchbar)
Vertrag löschen	Löschen eines erstellten Vertrags
Erstellen Einstellungsvorschlag	Erstellen des Einstellungsvorschlags basierend auf den Informationen im Vertrag
Vertragsstatistik Übersicht	Statistik zum Status der Verträge in ausgewähltem Semester
Versenden von E-Mails	Versenden von E-Mails aus dem Tutortool heraus an einzelne Personen oder voreingestellte Gruppen
E-Mail-Vorlage erstellen/ bearbeiten	Erstellen/Bearbeiten von E-Mail-Vorlagen für die mehrmalige Verwendung
Einladen von Übungsleitern aus dem Tutortool heraus	Erstellen von Einladungslinks für Übungsleiter, damit diese direkt die Rolle Übungsleiter erhalten
Übersicht Formulare	Übersicht aller herunterladbaren Formulare
Hinzufügen/Ändern von Formularen	Hinzufügen/Ändern von herunterladbaren Formularen aus dem Tutortool heraus
Übersicht Leihen	Übersicht über alle Leihen
Anlegen/Bearbeiten von Leihen	Anlegen/Bearbeiten von Leihen mit/ohne Leihanfrage
Leihanfragenübersicht	Übersicht über alle gestellten Leihanfragen
Leihstatistik	Übersicht über die Anzahl der entliehenen/noch verfügbaren Leihgeräte

Tabelle 8: Funktionalitäten der Neukonzeption für die Mitarbeiter des Tutorbetriebs
Quelle: eigene Erhebung

Übungsleiter

Die in diesem Abschnitt aufgelisteten Funktionalitäten sind für die Nutzergruppe der Übungsleiter intendiert.

Bezeichnung Funktionalität	Kurzbeschreibung
Übersicht der betreuten Veranstaltungen	Übersicht über alle Veranstaltungen, denen der Übungsleiter zugeteilt ist
Übersicht der Bewerbungen für eine Veranstaltung	Übersicht über alle Bewerbungen für eine Veranstaltung mit allen nötigen Informationen (z.B.: Note und Priorität, Anzahl offene Stellen, etc.) (filterbar, durchsuchbar)
Einzelansicht Bewerbung	Detaillierte Einzelansicht für ausgewählte Bewerbung
Akzeptieren von Bewerbungen	Akzeptieren von Bewerbungen von Studierenden auf offene Tutorstellen
Ablehnen von Bewerbungen	Ablehnen von Bewerbungen von Studierenden auf offene Tutorstellen
Kommentarfeld Bewerbungen	Kommentarfeld durch das Übungsleiter und Mitarbeiter des Tutorbüros kommunizieren können
Kommentieren von Bewerbungen	Beifügen von Kommentaren zu Bewerbungen, die für die Mitarbeiter des Tutorbüros sichtbar sind
Exportieren von Vertragsdaten	Der Export von den Vertragsdaten aller Tutoren für die Veranstaltung in unterschiedlichen Formaten

Tabelle 9: Funktionalitäten der Neukonzeption für die Übungsleiter
Quelle: eigene Erhebung

Studierende/Tutoren

Die in diesem Abschnitt aufgelisteten Funktionalitäten sind für die Nutzergruppe der Studierenden beziehungsweise Tutoren intendiert.

Bezeichnung Funktionalität	Kurzbeschreibung
Hinzufügen von Ausbildungsdaten	Hinzufügen von Ausbildungsdaten für die Bewerbungen
Hinzufügen von beruflichen Erfahrungen	Hinzufügen von beruflichen Erfahrungen für die Bewerbungen
Übersicht über Veranstaltungen mit offenen Stellen	Übersicht über alle Veranstaltungen, für die die Studierenden sich bewerben können
Abgeben/Ändern von Bewerbungen für offene Tutorstellen	Erstellen von Bewerbungen auf offene Tutorstellen und das Ändern dieser
Übersicht über alle Bewerbungen	Detaillierte Übersicht über alle Bewerbungen des Studierenden
Übersicht über alle Verträge	Detaillierte Übersicht über alle Verträge des Studierenden
Herunterladen von benötigten Vertragsformularen	Bereitstellen der benötigten Formulare für die Studierenden zum Herunterladen aus dem Tutortool heraus
Erstellen von Leihanfragen	Erstellen von Anfragen für Leihhardware direkt im Tutortool

Tabelle 10: Funktionalitäten der Neukonzeption für die Studierenden/Tutoren
Quelle: eigene Erhebung

Mitarbeiter der RBG

Die in diesem Abschnitt aufgelisteten Funktionalitäten sind für die Nutzergruppe der RBG Mitarbeiter intendiert.

Bezeichnung Funktionalität	Kurzbeschreibung
Übersicht Leihen	Übersicht über alle Leihen
Bearbeiten von Leihen	Bearbeiten von Leihen

Tabelle 11: Funktionalitäten der Neukonzeption für die Mitarbeiter der RBG
Quelle: eigene Erhebung

4.3 Zusammenfassung Relevance Cycle

Das Ziel dieses Relevance Cycles war es, die Forschungsfragen zwei („Wie lassen sich die aktuellen Prozesse innerhalb des Tutorbetriebs modellieren, wo treten Probleme auf und wie lassen sich die Prozesse optimieren?“) und drei („Welche Funktionalitäten muss die Neukonzeption des Tutortools für den Tutorbetrieb bereitstellen, damit sie alle (intendierten) Nutzergruppen optimal unterstützt?“) zu beantworten. Hierzu wurden die Prozesse des Tutorbetriebs, die in Zusammenhang mit dem Tutortool stehen, erhoben, abgebildet und (wo möglich) optimiert. Zudem sollte die Benutzerfreundlichkeit durch die optimierten Prozesse bereits in diesem Schritt verbessert werden. Zunächst wurde hierzu durch Fragebögen die Benutzerfreundlichkeit des alten Tutortools erhoben und zusätzlich Wünsche und Anregungen bei den intendierten Nutzergruppen durch Freitextfragen erfragt. Die Kernprozesse der Mitarbeiter des Tutortools wurden durch persönliche Gespräche und Videoaufzeichnungen erarbeitet. Auf Basis dieser optimierten Prozesse wurden dann weitere Funktionalitäten ermittelt, die eine Neuimplementierung des Tutortools sinnvoll ergänzen. Die größten Veränderungen wurden dabei an

wichtigen Subprozessen vorgenommen, wodurch die Komplexität der meisten Prozesse deutlich reduziert wurde. Zudem wurden Teilprozesse, die bisher nur unter Verwendung weiterer Programme (z.B.: Excel-Tabellen für gesplittete Verträge) oder Anwendungen (z.B.: Verleihtool) in das Konzept des Tutortools integriert, um den Arbeitsaufwand zu minimieren. Abschließend wurde auf Basis der Prozesse und den ermittelten Funktionalitäten noch eine Übersicht über alle Funktionalitäten für die Neukonzeption gegeben. Die Ergebnisse dieses Relevance Cycles dienen als Grundlage für die Design Cycles, in denen die optimierte Version des Tutortools implementiert wird.

5 Design Cycles

Den zentralen Punkt der Design Science Methodik nach Hevner (2007) stellen die Design Cycles dar. Im Verlauf dieser Arbeit wurden vier solcher Design Cycles durchgeführt. Der erste Design Cycle befasste sich mit dem grundlegenden Aufbau der Neukonzeption/Neuimplementierung, der Umsetzung der wichtigsten Komponenten (Funktionalitäten) für alle Nutzergruppen und speziell dem Umsetzen und Testen der Anforderungen der Mitarbeiter des Tutorbetriebs. Die Design Cycles zwei und drei befassten sich vor allem mit der Verfeinerung der Funktionalitäten für Übungsleiter und Studierende. Abschließend wurde ein vierter Design Cycle durchgeführt, während dem zusätzliche Funktionalitäten in die Neukonzeption eingefügt wurden, die aber nicht essenziell für die grundlegende Verwendung der Neuimplementierung sind.

Das vorliegende Kapitel ist wie folgt aufgebaut: Zunächst werden dem Leser die verwendete Architektur und die dabei verwendeten Technologien kurz vorgestellt. Hierbei werden auch die Beweggründe für die Verwendung der gewählten Architektur und den zugehörigen Technologien erläutert. Danach wird ein kurzer Einblick in die technische Umsetzung der Architektur in dieser Arbeit gegeben. Den Hauptteil dieses Kapitels stellen die Beschreibungen der vier Design Cycles dar. Hierbei werden zunächst die umgesetzten Funktionalitäten durch die implementierten Ansichten beschrieben. Im Anschluss werden die Nutzertests der einzelnen Zyklen und die hierbei erzielten Resultate vorgestellt.

5.1 Architektur der Implementierung

Für die Implementierung dieses Projekts wurde der MERN-Stack als Software Stack[27] ausgewählt. Ein Software Stack fasst alle Softwarekomponenten zusammen, aus denen eine spezifische Plattform besteht (Augusten & chrissikraus, 2018). Beim MERN-Stack handelt es sich um eine Kombination der vier Kerntechnologien MongoDB, Express.js, React.js und Node.js. Hieraus ergibt sich eine klassische 3-Schichten-Architektur bestehend aus Frontend, Server- oder Anwendungsschicht und Datenbank, die ausschließlich JSON und JavaScript verwendet (MongoDB, 2021). Somit kann der MERN-Stack auch als Full-Stack bezeichnet werden. Das Frontend besteht bei diesem Stack aus React.js. Node.js und Express.js stellen in diesem Kontext die mittlere Schicht (Serverschicht) der Architektur dar. Bei MongoDB handelt es sich, wie aus dem Namen bereits abgeleitet werden kann, um die Datenbank der Anwendung. Abbildung 13 ist eine schematische Darstellung der verwendeten Architektur. Hierin lassen sich die Kommunikationswege zwischen den Schichten erkennen. Nach diesem Architektur Paradigma kommunizieren ausschließlich die jeweils benachbarten Schichten miteinander. Dieser Aufbau trägt zur Robustheit des Systems bei und hilft, Fehler, die innerhalb einer Schicht auftreten, besser analysieren und beheben zu können. Zudem wurde die Implementierung um weitere Frameworks und Toolkits innerhalb der Schichten erweitert, um die bestmögliche Umsetzung der ermittelten Anforderungen gewährleisten zu können. Des Weiteren gilt der MERN-Stack als

[27] Auch „Solution-Stack“ oder „Lösungsstack“ genannt.

zukunftssichere Architektur und unterstützt durch seinen Aufbau potenzielle weitere Veränderungen, um neue Anforderungen des Benutzerumfelds umsetzen zu können. In diesem Abschnitt werden daher die einzelnen verwendeten Technologien zunächst genauer vorgestellt.

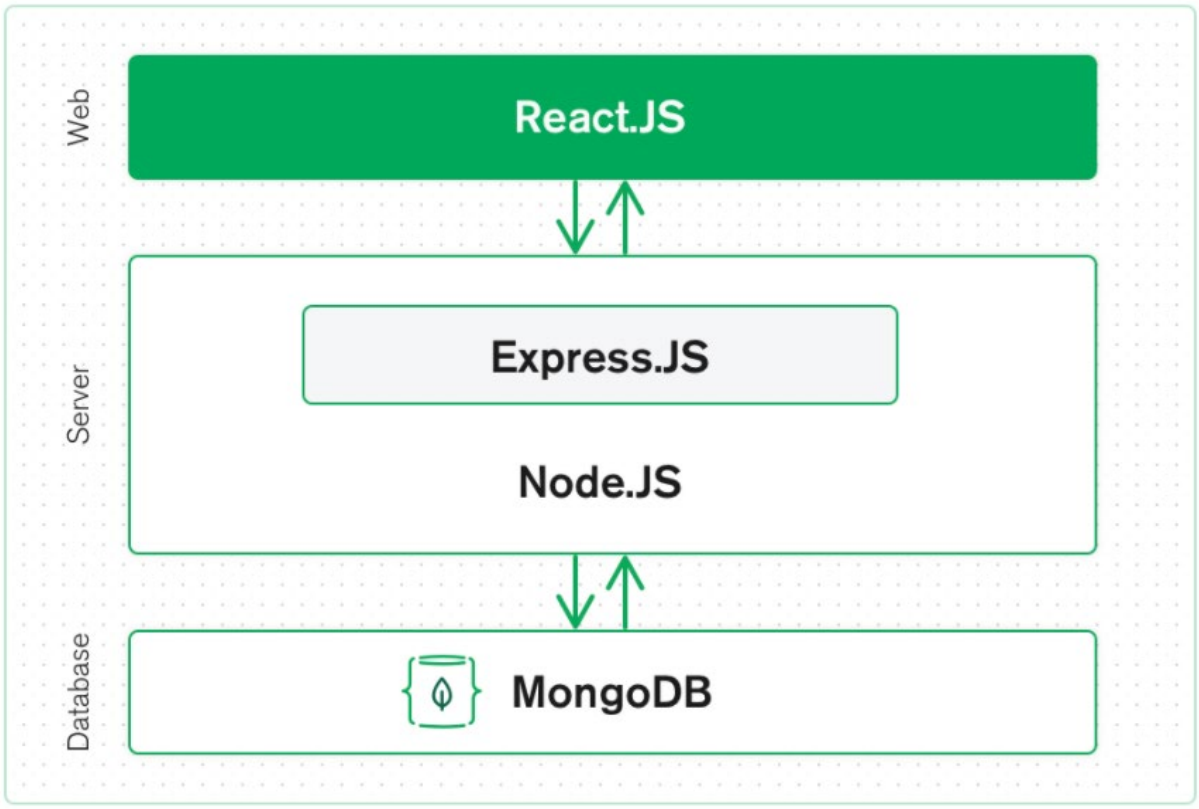


Abbildung 13: MERN-Stack Architektur
Quelle: MongoDB (2021)

5.1.1 Node.js

Bei Node.js handelt es sich um eine asynchrone, eventbasierte JavaScript Laufzeitumgebung[28]. Diese zeichnet sich vor allem dadurch aus, dass sie dazu entwickelt wurde, mit ihrer Hilfe skalierbare Webanwendungen zu entwickeln. Diese Eignung entsteht daraus, dass Node.js entgegen anderen geläufigen Laufzeitumgebungen „non-blocking I/O“ (asynchron) statt „blocking I/O“ (synchron) verwendet (Node.js, 2020). „Blocking“ heißt in diesem Kontext, dass ein JavaScript Prozess in Node.js warten muss, bis eine Nicht-JavaScript Operation ausgeführt wurde (Node.js, 2019).

5.1.2 Express.js

Express.js ist ein Webframework für Node.js. Express.js übernimmt hierbei das URL-Routing innerhalb der Serverschicht der Applikation. Express.js eignet sich durch die Unterstützung

[28] Geläufiger unter dem englischen Begriff „Runtime Environment“ bekannt.

vieler HTTP-Dienstprogrammmethoden und Middlewarefunktionen besonders für das Erstellen leistungsfähiger APIs und stellt hierdurch die serverseitige Schnittstelle zwischen Frontend und Serverschicht dar (OpenJS, 2020).

5.1.3 React.js

React.js (oder auch nur React genannt) ist eine JavaScript Bibliothek, um User Interfaces zu erstellen. Innerhalb des MERN-Stacks stellt React das Frontend dar. Durch die Verwendung von React.js ist es zudem möglich, sogenannte „single page applications“ (SPAs) - wie die vorliegende Implementierung -, zu erstellen. Eine SPA ist eine Webanwendung, bei der auf das Aktualisieren einer Webseite verzichtet werden kann, um neue Inhalte zu laden (Subramanian, 2017, S. 1). Für die Verarbeitung der vom Backend bereitgestellten Daten kommt die Zustandsverwaltungsbibliothek Redux (Abramov, 2021) zum Einsatz.

5.1.4 MongoDB

MongoDB ist eine open-source, dokumentenbasierte Datenbank, die JSON-artige Dokumente speichert. MongoDB Datenbanken zählen durch ihren nicht-relationalen Ansatz zu den sogenannten No-SQL (Not only SQL) Datenbanken. Jegliche Daten, die persistenter Speicherung bedürfen, werden direkt in dieser Datenbank gespeichert (MongoDB, 2021). Eine MongoDB Datenbank besteht dabei aus mehreren logischen Datenräumen, den sogenannten Collections. Diese Collections sind analog zu den Tabellen in relationalen Datenbanken zu verstehen. Innerhalb einer solchen Collection werden die Dokumente verwaltet.

5.2 Umsetzung der Architektur

Zusätzlich zu der Vorstellung der verwendeten Architektur im letzten Abschnitt sollen die folgenden Absätze dem Leser die Umsetzung der verwendeten Technologien in der Neuimplementierung des Tutortools näherbringen.

5.2.1 Datenbank

Für die Definition und das Erstellen von Schemata und die Kommunikation mit der MongoDB Datenbank wurde die freie Object Data Modeling (ODM) Bibliothek „Mongoose“(Mongoose, 2021) für Node.js verwendet. Die hiermit erstellten Schemata in Node.js stellen in der MongoDB Datenbank dann die Collections dar. Um eine logisch sinnvolle Speicherung aller verwendeten Daten gewährleisten zu können, wurden auch Referenzen in andere Collections verwendet, damit gleiche Datensätze nicht an unterschiedlichen Stellen mehrfach gespeichert werden müssen. Hierdurch kann auf die benötigten Daten aus der Collection, auf die verwiesen wird, durch die „populate“ Funktion zugegriffen werden. Die in der Neukonzeption verwendete Datenbank umfasst die folgenden 13 Collections[29]:

[29] Stand: 15. Mai 2021

- Applications
- Contracts
- Courses
- Forms
- Invitations
- Mailtemplates
- Metacourses
- Profiles
- Rentals
- Rentalsapplications
- Rentalstats
- Semesters
- Users

Zusätzlich zu der Erstellung von Collections bietet Mongoose diverse Funktionen für Node.js, um Datenbankabfragen zu erstellen und Datensätze in die Datenbank zu schreiben oder zu verändern.

5.2.2 Backend

Das Backend dieser Implementierung ist in mehrere Teile zu unterteilen, die alle unterschiedliche Aufgaben erfüllen. Die wichtigste Aufgabe des Backends besteht darin, als Server der Applikation zu fungieren. Hierzu stellt das Backend die für die Kommunikation zwischen Frontend und Backend benötigte API (dt. Programmierschnittstelle) bereit. Diese wird kontextabhängig in mehreren Dateien definiert. Bei der bereitgestellten API handelt es sich um eine sogenannte RESTful[30] API. Diese Art von API wurde durch Fielding (2000) eingeführt und stellt seither eine der meistverwendeten API Architekturen dar. Zu Beginn der Implementierungsphase wurde zum Testen der erstellten Routen das Programm Postman verwendet, da es zu diesem Zeitpunkt noch kein Frontend gab, durch das Anfragen an das Backend gestellt werden konnten. Mit Postman können API-Anfragen an einen Server erstellt und abgeschickt werden. Die Antworten des Servers werden dann in der GUI-Oberfläche des Programms dargestellt. Eine weitere Funktion des Backends ist die zuvor bereits beschriebene Kommunikation mit der Datenbank und die Bereitstellung dieser über die API an das Frontend. Die letzte Funktion, die das Backend übernimmt, ist die Validierung der Daten aus dem Frontend, bevor diese weiterverarbeitet werden.

5.2.3 Frontend

Um das React Frontend zu erstellen, wurde zunächst „Create React App“ verwendet. „Create React App“ ist ein Entwicklungswerkzeug, das offiziell von React zur Verfügung gestellt wird, um Single Page React Anwendungen zu erstellen. Dabei übernimmt die Bibliothek das (sonst aufwändige) Erstellen und Konfigurieren aller Grundeinstellungen (Facebook, 2021). Das

[30] Representational State Transfer

React Frontend übernimmt die Aufgabe der Darstellung aller Daten für die Nutzer. Die Kommunikation mit dem Backend erfolgt hierbei durch die Verwendung der HTTP-Client Bibliothek Axios (Axios, 2021). Die hierbei im JSON Format erhaltenen Daten aus dem Backend werden dann in den sogenannten Redux-Store geladen, durch welchen sie dem Frontend für die Weiterverarbeitung zur Ansicht für den Nutzer bereitgestellt werden.

5.3 Design Cycle 1

Der erste Design Cycle erstreckte sich über einen Zeitraum von ungefähr drei Monaten. Das Ziel des ersten Design Cycles war es zunächst, die erhobenen Grundanforderungen und Funktionalitäten so umzusetzen, dass der erstellte Prototyp der Anwendung vollständig funktionell von den Mitarbeitern des Tutorbüros getestet werden konnte. Hierbei sollten Fehler in der Umsetzung ermittelt und behoben werden, sodass die finale Version nach Beendigung des Design Cycles vollumfänglich seitens der Mitarbeiter des Tutorbüros in der realen Arbeitsumgebung für die Verwaltung genutzt werden könnte. Dieser Design Cycle stellt daher den längsten und umfangreichsten aller Design Cycles dar.

5.3.1 Umsetzung der Funktionalitäten

In diesem Abschnitt werden die in Design Cycle 1 umgesetzten Funktionalitäten anhand der implementierten Ansichten beschrieben. Dabei werden aufgrund der großen Zahl an implementierten Ansichten nur die übergeordneten Ansichten beschrieben, von denen die weiteren Funktionalitäten aus genutzt werden können. Die in diesem Design Cycle vorgestellten Funktionalitäten werden aus Sicht der Administratoren beziehungsweise der Mitarbeiter des Tutorbüros, falls nicht anders erwähnt, erklärt.

Landingpage

Diese Seite stellt den Einstiegspunkt der Website dar. Der Nutzer hat hier die Möglichkeit, sich entweder zu registrieren oder in das Tutortool einzuloggen. Beim Aufrufen der Seite wird zudem überprüft, ob ein Token aus dem letzten Authentifizierungsvorgang noch im Browser des Clients vorhanden ist und ob dieser Token noch gültig ist[31]. Falls die Ergebnisse dieser Überprüfungen positiv sind, wird der Nutzer direkt auf das Dashboard weitergeleitet.

Dashboard

Das Dashboard stellt den Mittelpunkt der Anwendung und den Einstiegspunkt nach erfolgreichem Login dar. Sollte es sich um einen neuen Account handeln, wird das eigentliche Dashboard noch nicht angezeigt, sondern zunächst ein Button, der den Benutzer auf die „Create Profile“ Seite weiterleitet (gilt für jede Accountart). Nachdem der Nutzer ein Profil angelegt hat, wird er auf das eigentliche Dashboard weitergeleitet. Das Dashboard enthält Auswahlbuttons für die weiteren Unterseiten. Diese sind für die Mitarbeiter des Tutorbetriebs gruppiert

[31] Bei der Authentifizierung wird ein Bearer Token mit acht Stunden Gültigkeit erzeugt und im Browser des Users gespeichert.

nach „Persönlich“, „Tutor Management“, „Setup“ und „Mail“. Wie das Dashboard für die Mitarbeiter des Tutorbetriebs umgesetzt wurde, kann in Abbildung 22 (Anhang B.2) betrachtet werden.

Profil bearbeiten

Dieser Menüpunkt öffnet die Profilbearbeitungsmaske für den eingeloggten Nutzer. Diese enthält (je nach Rolle des Benutzers) unterschiedliche Datenfelder, die eingetragen werden können/müssen.

Benutzerübersicht

Die Benutzerübersicht enthält eine Auflistung aller registrierten Benutzer für die Mitarbeiter des Tutorbetriebs. Dabei stehen dem Benutzer diverse Filter- und Suchmöglichkeiten zur Verfügung. Nach Rücksprache mit den Mitarbeitern des Tutorbüros wurden zur Verbesserung der Anschaulichkeit die in der Tabelle angezeigten Benutzerdaten auf Nachnamen, Vornamen, E-Mail-Adresse, Matrikelnummer (falls vorhanden) und das letzte Profil-Bearbeitungsdatum reduziert. Unter dem Tabelleneintrag „Betrachten“ kann der eingeloggte User das ausgewählte Benutzerprofil aufrufen und bearbeiten. Hierbei gibt es weitere Auswahlmöglichkeiten. Diese beinhalten das Ändern der Rolle des ausgewählten Nutzers, der DSGVO-Export und das Versenden einer E-Mail direkt an den ausgewählten Nutzer. Hierin wird auch (falls nötig) die in Kapitel 4.2.4 - Hinweis Prüfung durch Verfassungsschutz beschriebene Funktionalität angezeigt. Eine Darstellung dieser Funktionalität kann in Abbildung 23 (Anhang B.3) betrachtet werden.

Veranstaltungen/Veranstaltungsübersicht

Unter diesem Menüpunkt erhält der Nutzer eine detaillierte Übersicht über alle aktuell angelegten Veranstaltungen. Hierbei können verschiedene Statusfilter verwendet werden. Für jeden Status der Verträge (Offen, Vorbereitung, Geschlossen, Archiv) gibt es einen einzelnen Filter, der nur jene Verträge in der Tabelle anzeigt, die den ausgewählten Status besitzen. Zudem gibt es den Default-Status „Alle“, unter dem alle Verträge außer denen, die den Status „Archiv“ besitzen, angezeigt werden. In der Tabelle bekommen die Mitarbeiter des Tutorbüros alle Informationen angezeigt, die sie für ihre täglichen Aufgaben benötigen. Zudem können neue Veranstaltungen angelegt werden, bestehende Veranstaltungen bearbeitet und die Vertragsdaten der einzelnen Veranstaltungen in eine „.xlsx“ Datei exportiert werden. Hinzu kommt die Budgetübersicht für alle Veranstaltungen, die bereits in Abschnitt 4.2.4 (Absatz „Budgetübersicht“) genauer erläutert wurde.

Bewerbungsübersicht für Veranstaltungen

Wählt der Benutzer in der Veranstaltungsübersicht eine gelistete Veranstaltung aus, so wird er zur Bewerbungsübersicht weitergeleitet. Auf dieser Seite findet der User eine Übersicht über alle Studierenden, die sich auf eine Stelle als Tutor für die ausgewählte Veranstaltung beworben haben. Hierbei stehen dem User mehrere Möglichkeiten zur Verfügung. Beispielsweise können die Bewerbungen einzeln gesichtet werden und (ebenso wie von Übungsleitern) durch Mitarbeiter des Tutorbüros akzeptiert oder abgelehnt werden. Über das Kommentarfeld kann die Bewerbung des Studierenden mit einem Kommentar versehen werden, der von Übungsleitern und

Mitarbeitern des Tutorbüros eingesehen werden kann. Sobald ein Studierender für eine offene Tutorenstelle akzeptiert wurde, kann hierfür aus dem System ein Vertrag erstellt werden.

Vertragserstellung

Diese Ansicht dient der Vertragserstellung. Hier werden alle nötigen Informationen zum anzulegenden Vertrag eingegeben. Folgende Felder sind hierbei vorhanden:

- Vertrag Start und Ende für den Hauptvertrag
- Wochenstunden für den Hauptvertrag
- Vertrag Start und Ende für eventuelles Vertragssplitting
- Abschluss des angehenden Tutors
- Auswahlfeld Weiterbeschäftigung
- Status der benötigten Formulare
- Status der Vertragserstellung
- Kommentarfeld zum Vertrag

Für die Vertragszeiträume ist hierbei zusätzlich anzumerken, dass im Falle einer kumulierten Wochenstundenzahl (ab mehr als 20 Wochenstunden in einem Zeitraum) den Mitarbeitern des Tutorbetriebs eine Warnung angezeigt wird, da dies nicht zulässig ist. Die Warnung kann in Abbildung 24 (Anhang B.4) betrachtet werden. Bei dieser Warnung wird dem Nutzer in einer Klammer die Gesamtwochenstundenzahl des Tutors (im Falle der Beispielabbildung: 24) angezeigt.

Nachdem die Vertragserstellung abgeschlossen wurde, kann der jeweilige Mitarbeiter des Tutorbüros entweder einen weiteren Vertrag für dieselbe Person und dieselbe Veranstaltung anlegen oder direkt eine E-Mail an den Studierenden mit dem Hinweis, dass der Vertrag erstellt wurde, versenden. Danach wird der Benutzer wieder auf die Bewerbungsübersicht zurückgeleitet.

Verträge

Diese Ansicht beinhaltet alle Daten zu den angelegten Verträgen und stellt damit für die Mitarbeiter des Tutorbüros eine der wichtigsten Ansichten in der gesamten Applikation dar. Die meisten tabellenbasierten Ansichten der Neukonzeption können die Inhalte der Tabellen (in diesem Fall die Verträge) je nach Status filtern. Für die Vertragsübersicht besteht diese Möglichkeit für den Vertragsstatus „Erstellt“, „Unvollständig“, „In Bearbeitung“, Unterschriftsbereit“ und „Abgeschlossen“[32]. Hinzu kommen die Filter „Offen“ (beinhaltet alle Verträge, die nicht den Status „Abgeschlossen“ besitzen) und „Alle“ (beinhaltet die Gesamtheit aller Verträge). Ebenso besteht in dieser Übersicht die Möglichkeit, einen Vertrag unabhängig von einer Bewerbung zu erstellen und die in Abschnitt 4.2.4 unter der Überschrift „Semesterstatistik/Vertragsstatistik“ beschriebenen Statistiken abzurufen. Eine weitere Möglichkeit, die der Benutzer

[32] In der Datenbank sind die Status unter den englischen Begriffen hinterlegt und werden nur für die Administratorenansicht aufgrund der besseren Benutzerfreundlichkeit ins Deutsche übersetzt.

in der Vertragsübersicht hat, ist die Bearbeitung von Verträgen, welche im nächsten Absatz genauer vorgestellt wird.

Vertrag bearbeiten

Unter dieser Option können die bereits erstellten Verträge betrachtet und bearbeitet werden. Diese Ansicht wird vor allem bei Statusänderungen der Verträge und dem Abzeichnen von eingereichten Unterlagen verwendet. Hierbei werden die Felder für die Unterlagen (je nach Status) zur besseren Kenntlichkeit farbig markiert. Nach einer erfolgten Änderung wird das Kürzel des Bearbeiters im Vertrag zusammen mit dem Datum der Aktualisierung hinterlegt.

Semesterübersicht

Die Semesterübersicht beinhaltet alle Informationen zu den Semestern. Hier können neue Semester angelegt und bereits erstellte Semester bearbeitet werden. Technisch gesehen stellt diese Seite eine eindeutige Abbildung des Semester Schemas aus der Datenbank dar.

Metaveranstaltungen

Dieser Menüpunkt beinhaltet alle Informationen zu den Metaveranstaltungen. Das Konzept der Metaveranstaltungen wurde in Abschnitt 4.1.3 unter der Unterüberschrift „Anlegen von Veranstaltungen“ bereits erläutert und wird hier daher nicht erneut erläutert. In dieser Ansicht können zudem neue Metaveranstaltungen angelegt und bearbeitet werden.

Formulare

In dieser Ansicht können von den Mitarbeitern des Tutorbüros die erforderlichen Formulare zur Erstellung von Verträgen hochgeladen und geändert werden. Zudem bietet diese Ansicht eine Übersicht über alle bereits hochgeladenen Formulare und deren letztes Änderungsdatum.

Übungsleiter

In dieser Ansicht können die in Abschnitt 4.1.3 Überschrift „Anlegen von Übungsleitern“ beschriebenen Einladungslinks für Übungsleiter erstellt werden. In der aktuell implementierten Version der Neukonzeption[33] wird dabei ein zufallsgenerierter String erstellt und in der Datenbank hinterlegt. Der Administrator bekommt den gesamten Link in der Form

https://tutor-tool.herokuapp.com/advisorregistration/m3wq27mnis[34]

angezeigt. Dieser kann dann kopiert und dem Übungsleiter per E-Mail zugeschickt werden.

[33] Stand: 15. Mai 2021
[34] Hier angegeben mit der Domain der in der Cloud gehosteten Testumgebung.

Mail versenden

In dieser Ansicht können aus dem Tutortool heraus E-Mails an Studierende versandt werden. Hierbei stehen dem Benutzer verschiedene Möglichkeiten zur Verfügung, die Adressaten der E-Mail einzutragen. Diese Möglichkeiten wurden in Gesprächen mit den Mitarbeitern des Tutorbetriebs ermittelt. Neben der manuellen Eingabe von Adressaten besteht die Möglichkeit, vorgespeicherte Adressaten im System auszuwählen. Die auswählbaren Adressaten bestehen aus einem einzelnen Tutor/Studierenden, allen Tutoren, allen aktiven Übungsleitern, allen Tutoren mit unvollständigen Verträgen und allen Tutoren mit einem Vertrag im auszuwählenden Semester. Nach Auswahl des/der Adressaten kann dann (bei Bedarf) noch eine Textvorlage gewählt werden.

Mail Vorlagen

Unter diesem Menüpunkt können E-Mail-Vorlagen erstellt werden. Diese können dann für den Versand von E-Mails aus dem System als Template verwendet werden, um Zeit und Aufwand zu sparen. Die Vorlagen können sowohl erstellt als auch geändert werden.

5.3.2 Nutzertests

Die Nutzertests wurden aufgrund des Umfangs des bei diesem Design Cycle erprobten Softwareartefakts in mehreren Iterationen durchgeführt. Dieses Vorgehen hatte mehrere Vorteile für das Testen der Implementierung. Aufgrund der geringen Anzahl (zwei) der zur Verfügung stehenden Testpersonen für diesen Design Cycle wurde die Arbeitslast für die Testpersonen stark reduziert. Ein weiterer Vorteil war, dass nachdem die Mitarbeiter des Tutorbüros mit der neuen Umgebung vertraut waren, bestimmte Bereiche durch zusätzliche Aufgabenstellungen zu den jeweiligen Tests genauer betrachtet werden konnten. Auf diese Art und Weise war es möglich, Problemstellen im Softwareartefakt detailliert herauszuarbeiten. Ebenso wurden durch die längere Dauer dieses Design Cycles und den iterativen Tests weitere Probleme identifiziert, die ansonsten erst im Live-Betrieb aufgefallen wären und somit verspätet hätten behoben werden müssen.

Die Iterationen waren dabei wie folgt aufgebaut: Zu Beginn wurde den Testpersonen eine Version der Implementierung über die Cloudplattform „Heroku“ bereitgestellt. Die Testpersonen hatten dann mehrere Tage Zeit, ihre alltäglichen Arbeitsabläufe in der neuen Implementierung zu erproben. Hierbei sollten die Testpersonen alle Ihrer Schritte dokumentieren und Feedback sammeln. Dieses Feedback wurde am Ende des Testzeitraumes (typischerweise vier Tage) in einer gemeinsamen Telefonkonferenz diskutiert. Das Ende einer Iteration bildete dabei das Umsetzen der gesammelten Verbesserungsvorschläge.

5.3.3 Implementierte Verbesserungen

Während der acht Iterationen wurden zahlreiche Verbesserungen an den Funktionalitäten auf Basis des erarbeiteten Feedbacks vorgenommen. Im folgenden Abschnitt werden diese für jede Iteration separat erläutert. Anzumerken ist, dass hierbei nur größere Veränderungen beschrieben werden, da das Beschreiben kleinerer Änderungen und minimaler Bugfixes den Rahmen dieser Arbeit übersteigen würde.

Iteration 1

Die Testphase dieser Iteration stellt den ersten Berührungspunkt der Mitarbeiter des Tutorbetriebs mit der Neukonzeption dar. Aus diesem Grund besteht das Feedback für diese Iteration vor allem aus Verbesserungen, die die allgemeine Bedienbarkeit und Nutzbarkeit betreffen. So wurde beispielsweise für das Anlegen und Bearbeiten von Verträgen das Erfordernis einer Kopie des Sozialversicherungsausweises entfernt. Diese wird im alten Tutortool noch als notwendiges Dokument angezeigt, wird jedoch im aktuell laufenden Betrieb nicht mehr benötigt. Dahingegen wurde (basierend auf dem Feedback) das Erfordernis einer Kopie des Reisepasses für Studierende aus Drittstaaten der Oberfläche hinzugefügt, da diese bisher immer zusätzlich per E-Mail angefragt werden musste. Zudem wurde in der „Vertrag Bearbeiten" Ansicht eine farbliche Kennzeichnung für die Status der einzureichenden Dokumente hinzugefügt sowie ergänzende Änderungen an den in den Tabellen präsentierten Daten vorgenommen. Der nächste Punkt, der in diesem Zusammenhang angesprochen wurde, war die Übersetzung des Frontends ins Deutsche für die Mitarbeiter des Tutorbetriebs. Zu diesem Zeitpunkt basierte die gesamte Neukonzeption auf englischer Sprache.

Iteration 2

Das Feedback und die darauf basierenden Verbesserungen dieser Iteration umfassen zunächst weitere Änderungen an Funktionalitäten. So wurde das Herunterladen des Einstellungsvorschlages aus der Vertragsübersichtstabelle in die „Vertrag Bearbeiten" Ansicht verschoben und dort die Felder für die benötigten Dokumente neu angeordnet, um die Übersicht effizienter zu gestalten. Weitere Anpassungen umfassten Änderungen in der Navigation (nach Erstellung eines Vertrags: Weiterleitung zurück auf die Bewerberübersicht), das Ändern von Eingabefeldern (Erstellung Metaveranstaltung: Drop Down Menü bei „Art der Veranstaltung") und die Möglichkeit, einen Tutoraccount DSGVO konform zu löschen.

Iteration 3

Die nach dieser Iteration implementierten Verbesserungen beziehen sich größtenteils auf den Einstellungsvorschlag. Hierbei wurden diverse Unstimmigkeiten in der Übertragung der Daten auf die exportierte Datei behoben. Beispiele hierfür sind das Eintragen einer vorliegenden Weiterbeschäftigung auf dem Einstellungsvorschlag sowie die korrekte Erfassung der eingereichten Dokumente. Des Weiteren wurde in vielen Ansichten die Sortierung der angezeigten Daten geändert und an die Wünsche der Mitarbeiter des Tutorbetriebs angepasst. Zusätzlich wurde auf Basis dieser Iteration die „20 Stunden Grenze" Warnung für die „Vertrag Erstellen" und „Vertrag Bearbeiten" Ansicht überarbeitet (Abbildung 24[35]). Eine weitere relevante Veränderung aus dieser Iteration war das Hinzufügen der Immatrikulationsbescheinigung des Folgesemesters zur Liste der benötigten Formulare, falls der angelegte Vertrag die Semestergrenze überschreiten sollte.

[35] Das Kommentarfeld befand sich zu diesem Zeitpunkt noch an einer anderen Stelle.

Iteration 4

Die erste größere Änderung, die in dieser Iteration vorgenommen wurde, war die Reduzierung des Vertragssplittings von drei auf zwei mögliche Vertragszeiträume, da dies für die aktuellen Bedürfnisse des Tutorbüros ausreicht. Für den Fall, dass in der Zukunft ein dritter Vertragszeitraum benötigt wird, kann dies mit geringem Aufwand jederzeit wieder umgestellt werden. Zudem wurde auf Basis des Feedbacks für diese Iteration eine weitere Option in die „Vertrag Bearbeiten“ Ansicht eingefügt, welche ermöglicht, für den betroffenen Tutoren direkt aus der Ansicht heraus einen weiteren Vertrag anzulegen.

Iteration 5

In dieser Iteration wurden hauptsächlich kleinere Bugs behoben. Die einzige größere Änderung, die vorgenommen wurde, ist die Einführung des Felds „Kürzel“ für die Rollen „Administrator“[36] und „Übungsleiter“[37]. Das Kürzel besteht aus einer Zeichenfolge von 3 Buchstaben, die den Namen der jeweiligen Person abkürzen und gewährleistet den Mitarbeitern des Tutorbetriebs eine bessere Nachvollziehbarkeit bei der Verwaltung der Tutorverträge (und der später eingeführten Leihen).

Iteration 6

In dieser Iteration wurden weitere Tabellen für die Mitarbeiter des Tutorbetriebs angepasst. So wurde die Ansicht der „Veranstaltungsübersicht“ gänzlich überarbeitet. Dabei wurde die Budgetübersicht entfernt, sodass auf diese nur noch aus der Veranstaltung selbst zugegriffen werden kann. Des Weiteren wurde die Ansicht so verändert, dass kein „Veranstaltung betrachten“ Button mehr nötig ist, sondern der Name der Veranstaltung den Benutzer direkt auf die Bewerberübersicht weiter verlinkt. Zudem wurden zur besseren Übersicht die Bezeichnungen im Tabellenkopf durch ihre jeweiligen Abkürzungen ersetzt. Damit die vollständigen Bedeutungen der Abkürzungen nicht verloren gehen, werden diese angezeigt, sobald der Nutzer den Mauszeiger über die entsprechende Abkürzung bewegt. Zudem wurden die Sortierungen in vielen Drop Down Menüs überarbeitet.

Iteration 7

Auf Basis des Feedbacks dieser Iteration wurden diverse Veränderungen am E-Mail-System der Neukonzeption vorgenommen. Es wurden weitere voreingestellte Möglichkeiten eingefügt, die E-Mails zu adressieren. Hierzu zählen vorgespeicherte Auswahlmöglichkeiten umfassend alle aktiven Übungsleiter, alle Tutoren, die einen Vertrag im ausgewählten Semester haben, sowie alle Tutoren mit einem unvollständigen Vertrag. Zudem wurde eingefügt, dass der Verfasser der E-Mail (der eingeloggte Nutzer) eine Kopie der verfassten E-Mail per BCC zugeschickt bekommt.

---

[36] „Administrator“ ist der systeminterne Rollenbegriff für die Mitarbeiter des Tutorbetriebs.
[37] Später auch für die Rolle „RBG“.

Iteration 8

In dieser letzten Iteration wurden keine größeren Veränderungen an der Neukonzeption mehr vorgenommen. Es wurden abschließend noch einmal die größten Veränderungen diskutiert und kleinere Fehler in der Applikation angesprochen und behoben.

5.3.4 Ergebnisse

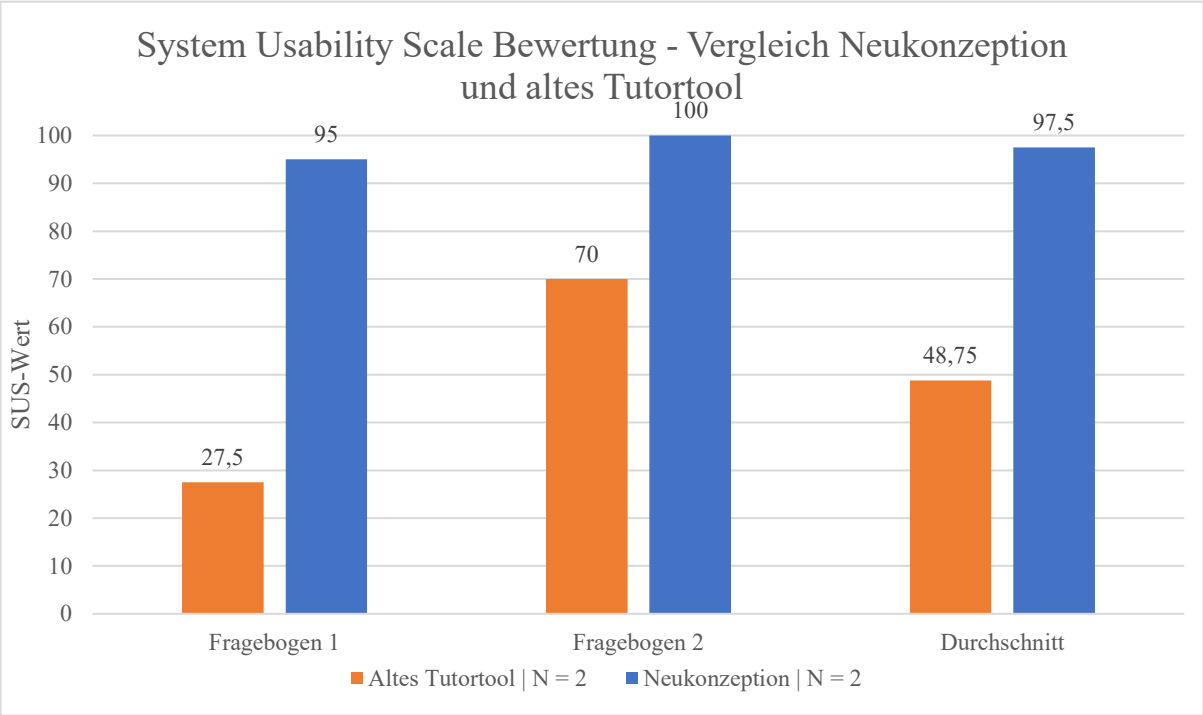


Abbildung 14: SUS Bewertungen des Tutortools durch Mitarbeiter des Tutorbetriebs
Quelle: eigene Erhebung

Abschließend zu diesem Design Cycle wurde den Mitarbeitern des Tutorbüros erneut ein Fragebogen zur Bewertung des System Usability Scale Werts der Anwendung zugeschickt. Die Ergebnisse hierbei können in Abbildung 14 betrachtet werden. Obwohl die Anzahl der Befragten gering war, sind die Ergebnisse durch den Expertenstatus der Befragten dennoch aussagekräftig. Die außerordentlich guten Ergebnisse sowie das Auslassen der Antworten auf die Freitextfrage, was an der Anwendung noch verbessert werden könnte, lassen auf eine hohe Benutzerzufriedenheit schließen. Für diese hohe Benutzerzufriedenheit können mehrere Gründe in Betracht gezogen werden. Zum einen, dass die Mitarbeiter des Tutorbetriebs von Beginn an weitestgehend in den Arbeitsprozess mit einbezogen wurden, wodurch die Anwendung genau an die Anforderungen und Wünsche der Mitarbeiter angepasst werden konnte. Zum anderen dient als Referenz für Benutzerfreundlichkeit zum jetzigen Zeitpunkt lediglich die alte Version des Tutortools, die (wie zu Beginn des Relevance Cycles ermittelt) keine gute, bis schlechte Benutzerfreundlichkeit in allen Aspekten aufweist. Auf die Freitextfrage, was den Benutzern besonders an der Anwendung gefällt, wurden vor allem das klare Design der Anwendung, die vielen Funktionen und Filtermöglichkeiten und das Integrieren essenzieller Funktionalitäten wie die Budgetüberwachung oder das Anlegen von gesplitteten Verträgen genannt.

Zusätzlich zu diesen durchgeführten Tests zur Benutzerfreundlichkeit wurde nach Abschluss der Implementierungsphase die Neukonzeption noch in Hinsicht auf die optimierten Prozesse gegen das alte Tutortool getestet. Hierzu wurden die Mitarbeiter des Tutorbetriebs in einer Videokonferenz gebeten, Vorgänge jeweils im alten Tutortool und in der Neukonzeption zu bearbeiten. Hierbei wurde jeweils die Dauer bis zur Finalisierung der jeweiligen Vorgänge gemessen. Für diesen Test wurden den Mitarbeitern des Tutorbüros drei verschiedene Aufgaben gestellt, die in den folgenden Abschnitten kurz vorgestellt werden. Zusätzlich können die Ergebnisse der Zeitmessungen in Abbildung 15 betrachtet werden.

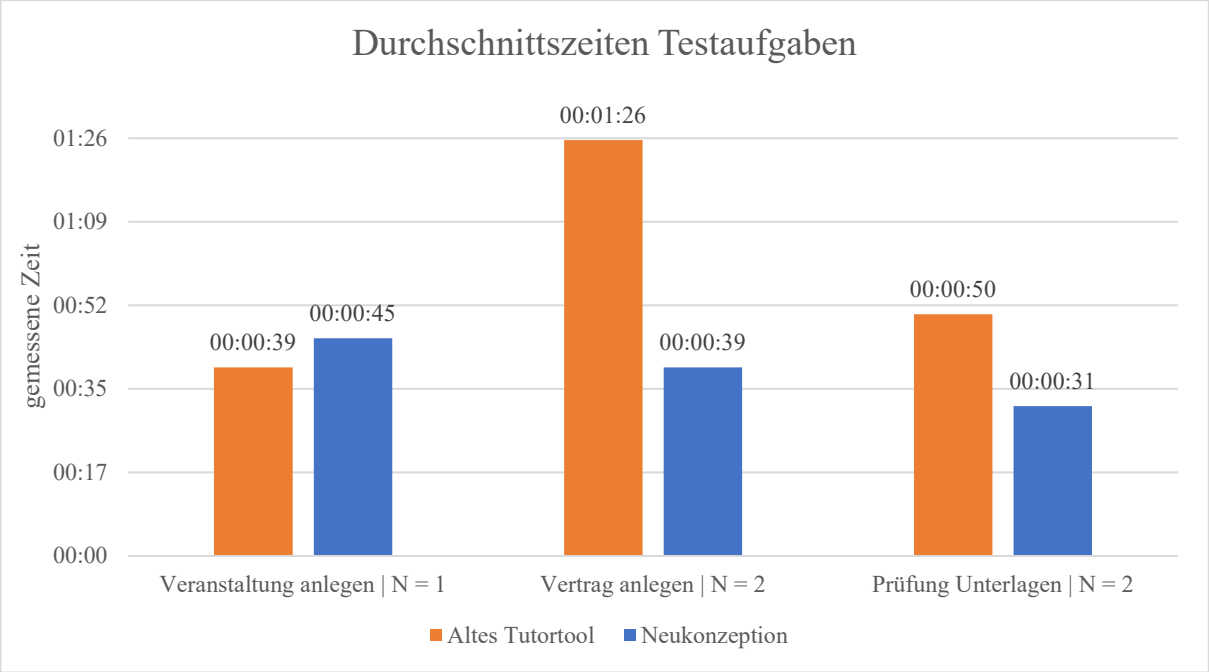


Abbildung 15: Durchschnittszeiten Testaufgaben neues und altes Tutortool
Quelle: eigene Erhebung

Veranstaltung anlegen

Hierbei bestand die Aufgabe für die Mitarbeiter des Tutorbetriebs darin, eine neue Veranstaltung für eine vorliegende Metaveranstaltung anzulegen. Diese Aufgabe wurde nur von einer Person durchgeführt, da die andere verfügbare Person diese Aufgabe im Tutorbetrieb sonst normalerweise nicht durchführt. Für die Finalisierung dieses Vorgangs wurde eine Zeit von 39 Sekunden für die Bearbeitung im alten und 45 Sekunden für die Bearbeitung im neuen Tutortool gemessen. Für die schnellere Zeit bei der Erstellung der Veranstaltung im alten Tutortool lassen sich mehrere Gründe finden. Zum einen wurden die Tests nach Abschluss der Planungsphase für das Sommersemester 2021 durchgeführt. Hierdurch waren die Mitarbeiter des Tutorbetriebs bereits sehr geübt im Umgang mit dem alten Tutortool, hatten jedoch mit der Neukonzeption zu dem Zeitpunkt mehrere Wochen nicht mehr gearbeitet. Zum anderen ist das Anlegen einer Veranstaltung in der Neukonzeption deutlich umfangreicher, da alle Felder bezüglich der Tutorenplanung eingegeben werden müssen, was aber in der späteren Verwendung das Arbeiten deutlich erleichtert. Somit kann der kleine Zeitnachteil der Neukonzeption gegenüber dem alten Tutortool als vernachlässigbar gewertet werden.

Vertrag anlegen

Diese Aufgabe bestand daraus, für einen von einem Übungsleiter akzeptierten Studierenden einen Vertrag bestehend aus zwei Teilverträgen (Vertragssplitting) zu erstellen. Im alten Tutortool benötigten die Mitarbeiter des Tutorbetriebs durchschnittlich 1 Minute und 26 Sekunden. In der Neukonzeption hingegen wurden nur durchschnittlich 39 Sekunden benötigt. Diese große Verbesserung kann darauf zurückgeführt werden, dass gesplittete Verträge in der Neukonzeption direkt in einer Vertragserstellungsmaske angelegt werden können. In der alten Version müssen hierzu zwei einzelne Verträge angelegt werden.

Vertrag Unterschriftsbereit setzen und EV erstellen

Die dritte Aufgabe, die gestellt wurde, bestand daraus, alle einem für einen Vertrag nötigen Dokumente auf „Liegt vor“ beziehungsweise „Liegt bei“ zu setzen, den Vertragsstatus auf „Unterschriftsbereit“ zu setzen und den Einstellungsvorschlag für den Vertrag zu erstellen. Hierbei wurden für die Aufgabe im alten Tutortool durchschnittlich 50 Sekunden benötigt und in der Neukonzeption 31 Sekunden. Durch die Optimierungen in der Neukonzeption müssen die Mitarbeiter des Tutorbetriebs bei der Erstellung eines Einstellungsvorschlages diesen nicht mehr zum Großteil selbst ausfüllen (wie im alten Tutortool), sondern bekommen den Einstellungsvorschlag komplett ausgefüllt als „.pdf“ Export vom Tutortool bereitgestellt. Daraufhin müssen die Einträge lediglich kurz gegengeprüft werden.

5.4 Design Cycle 2

Der zweite Design Cycle hatte zur Aufgabe, die Grundfunktionalitäten, die im ersten Design Cycle für die Übungsleiter implementiert wurden, zu erweitern und Probleme zu beheben. Diese erweiterten Grundfunktionalitäten wurden anschließend mit Übungsleitern getestet, die bereits öfter das alte Tutortool verwendet haben. Hierzu wurde zunächst von den Mitarbeitern des Tutorbüros eine Liste mit Übungsleitern erstellt, die hierfür potenziell in Frage kämen. Im nächsten Schritt wurden diese Übungsleiter dann kontaktiert und ihnen, wie bereits im Relevance Cycle beschrieben, ein System Usability Scale Fragebogen zugeschickt. Mit den im Abschnitt über den Relevance Cycle beschriebenen Vorstellungen und Wünschen wurde dann das Übungsleiter Interface implementiert.

5.4.1 Umsetzung der Funktionalitäten

Das Frontend für Übungsleiter ist in englischer Sprache verfasst, da so auch internationale Übungsleiter das Tutortool problemlos verwenden können. Die Benennungen der einzelnen Ansichten sind daher im Folgenden ebenfalls mit englischen Titeln versehen.

My Courses

Diese Übersicht gibt dem Benutzer (wie der Name widerspiegelt) eine Übersicht über die Veranstaltungen, für die er als Übungsleiter tätig ist. Dem Benutzer werden in der Tabelle der Übersicht wichtige Informationen zu den Veranstaltungen, bestehend aus Veranstaltungstitel, Semester, Anzahl Bewerbungen, akzeptierte Bewerbungen, benötigte Tutoren und dem Status der Veranstaltung, angezeigt und die Möglichkeit gegeben, die Bewerbungen für die Veranstaltung zu überprüfen.

Bewerbungsübersicht

Diese Ansicht enthält eine Tabelle mit allen Studierenden, die sich für die ausgewählte Veranstaltung beworben haben. Über diese Tabelle haben die Übungsleiter die Möglichkeit, die Einträge nach dem Status der Bewerbungen zu sortieren, oder sich die Bewerbungen anzeigen zu lassen, die seit dem letzten Login hinzugekommen sind. In der Tabelle selbst können die Bewerbungen zur ausführlicheren Sichtung ausgewählt, angezeigt, sowie akzeptiert oder abgelehnt werden. Sollte ein Übungsleiter einen Kommentar zu einer einzelnen Bewerbung haben, kann dieser über einen separaten Button angefügt werden und ist von da an auch den Mitarbeitern des Tutorbüros zugänglich. Zusätzlich können Details zu den Veranstaltungen von den Übungsleitern in dieser Übersicht bearbeitet werden. Diese Details werden den Studierenden angezeigt und bieten zum Beispiel die Möglichkeit, weitere Kriterien für eine Anstellung als Tutor für die entsprechende Veranstaltung hinzuzufügen. Über der Tabelle werden den Übungsleitern die akzeptierten Positionen angezeigt, wodurch ein verbesserter Überblick über die noch verbleibenden offenen Stellen geschaffen wird. Zudem besteht die Möglichkeit, die Vertragsdaten aller Tutoren für die Veranstaltung entweder als „.csv“ oder „.xlsx“ zu exportieren.

5.4.2 Nutzertests

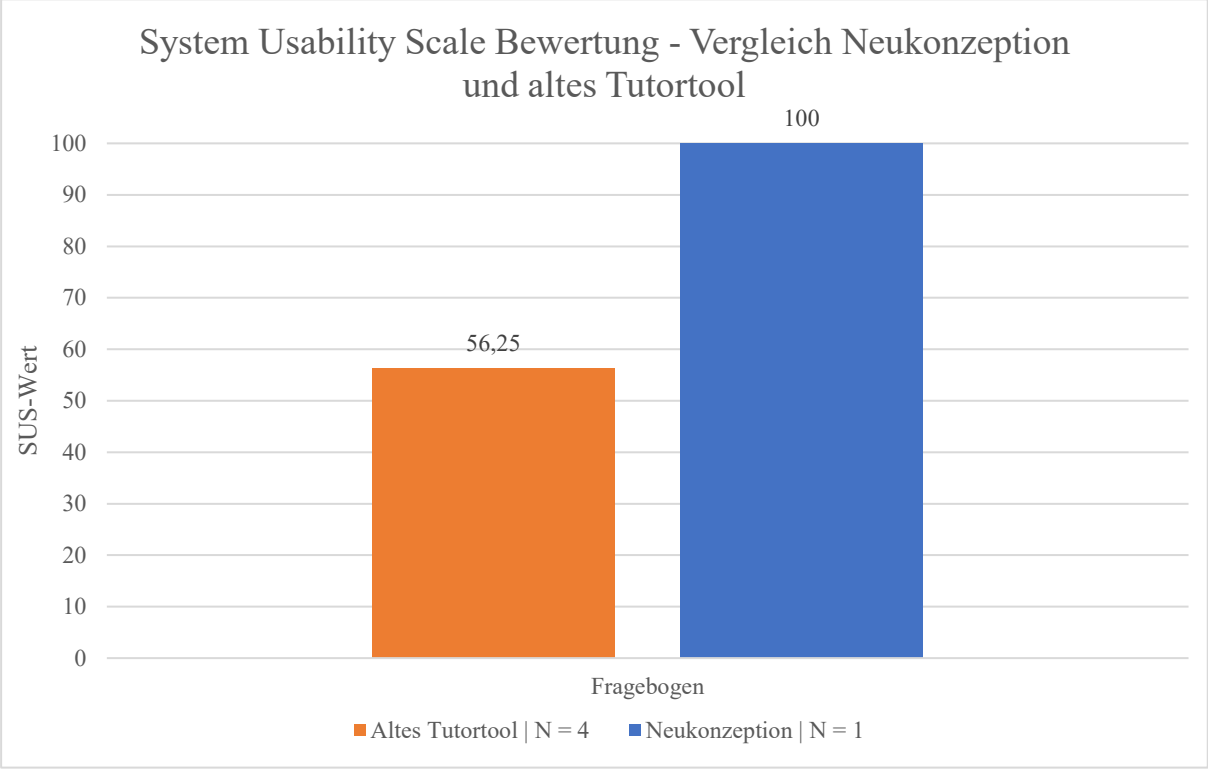


Abbildung 16: SUS Bewertungen des Tutortools durch die Übungsleiter
Quelle: eigene Darstellung

Wie zuvor bereits beschrieben, wurde fünf ausgewählten Übungsleitern, die bereits mehrere Jahre Erfahrung mit dem alten Tutortool hatten, der SUS-Fragebogen zugeschickt. Bedauerlicherweise schied ein Übungsleiter vor Abgabe des ausgefüllten Fragebogens aufgrund gesundheitlicher Probleme aus. Die übrigen vier Übungsleiter füllten, wie im Relevance Cycle

beschrieben, den Fragebogen aus. Nach Implementierung des Frontends wurde für jeden der Übungsleiter ein separater Einladungscode für die direkte Registrierung mit der Rolle „Übungsleiter“ erstellt. Nach der Registrierung wurde den Übungsleitern jeweils eine Veranstaltung zugeteilt. Dieser Veranstaltung wurden dann mehrere Bewerber zugewiesen, sodass die Übungsleiter die Funktionalitäten des Tutortools (unter realistischen Bedingungen) testen konnten. Hierauf wurden erneut SUS-Fragebögen an die Übungsleiter zur Beurteilung übersandt. Leider überschnitt sich die Testphase der Neukonzeption zeitlich mit der ersten Prüfungsphase des Wintersemesters, wodurch die Übungsleiter anderweitig beansprucht wurden. Somit wurde nur ein Fragebogen ausgefüllt zurückgesandt und zwei E-Mails mit separatem Feedback zur Implementierung erhalten. Das Gesamtheit des erhaltenen Feedbacks wurde anschließend verwendet, um die Applikation in den angesprochenen Bereichen zu verbessern.

5.4.3 Ergebnisse

Die Ergebnisse der Nutzertests waren vorwiegend positiv. In Abbildung 16 können die Ergebnisse der Befragungen aus dem Relevance Cycle mit dem Ergebnis der Neukonzeption verglichen werden. Auch bei den Übungsleitern schnitt die Neukonzeption mit einem exzellenten Ergebnis ab. Obwohl nur ein Übungsleiter den Fragebogen für die Neukonzeption zurückgeschickt hatte, konnte aufgrund der positiven Aufnahme des neuen Tutortools und des Feedbacks aus den E-Mails eine positive Veränderung der Benutzerfreundlichkeit auch bei der Nutzergruppe der Übungsleiter beobachtet werden. Wie bereits zuvor bereits erwähnt, soll durch die Fragebögen keine repräsentative Studie der Neukonzeption durchgeführt werden, sondern vor allem Feedback gesammelt und ein Gefühl für die Benutzerfreundlichkeit der Applikation erlangt werden.

5.5 Design Cycle 3

Im Mittelpunkt dieses Design Cycles stand das Umsetzen der Benutzeroberfläche für die Studierenden. Hier wurden zunächst auf Basis des alten Tutortools die Grundfunktionalitäten umgesetzt. Mit den Erkenntnissen des Relevance Cycles wurden im Anschluss weitere Funktionalitäten eingefügt. Abschließend wurde das gesammelte Feedback aus den Nutzertests verwendet, um die Funktionalitäten für die Studierenden zu verbessern.

5.5.1 Umsetzung der Funktionalitäten

Ebenso wie für die Übungsleiter wurde aufgrund der Vielzahl an internationalen Studierenden das Frontend in englischer Sprache verfasst. Das Dashboard ist für die Studierenden in drei Kategorien unterteilt. „Personal Actions“ beinhaltet Links zu Ansichten, in denen die Tutoren persönliche Daten ändern können, „Tutor Actions“ beinhaltet Ansichten bezüglich der Bewerbungen und Verträge zu Tutorstellen, „Rental Actions“ beinhaltet den Unterpunkt zur Bewerbung auf Leihhardware. Zudem wird den Studierenden auf der rechten Seite des Dashboards eine Hilfeseite angeboten. Das implementierte Dashboard für die Studierenden kann in Abbildung 25 (Anhang B.5) betrachtet werden.

Add Experience/Add Education

Diese Ansichten bieten den Studierenden die Möglichkeit, ihre bisherigen Arbeitserfahrungen sowie alle Stationen ihrer fachlichen Ausbildung in das Tutortool einzupflegen. Die hier eingetragenen Daten werden dann auf Dashboard des Benutzers angezeigt und sind für Übungsleiter und Mitarbeiter des Tutorbüros in der Bewerbungsansicht einsehbar. Diese Möglichkeit soll das Hochladen eines Lebenslaufes ersetzen und eine standardisierte Variante darstellen, durch die keine Vor- und Nachteile für Bewerber entstehen.

Tutor Application

Diese Ansicht listet alle Veranstaltungen, die offen für Bewerbungen sind. Dabei bekommen die Studierenden in der Tabelle zusätzlich zu den Namen der Veranstaltungen die „Requirements“ (zusätzliche Bedingungen für die Einstellung als Tutor, die die Übungsleiter oder Mitarbeiter des Tutorbetriebs den Veranstaltungen hinzufügen können), den (Haupt-) Übungsleiter, den Status der Bewerbung angezeigt. Zusätzlich besteht die Möglichkeit, sich für die jeweilige Veranstaltung als Tutor zu bewerben, eine bereits abgegebene Bewerbung zu editieren, oder eine Bewerbung zu löschen. Entscheidet sich ein Studierender, sich für eine Veranstaltung als Tutor zu bewerben, so kann er nach Auswahl des „Apply“ Buttons eine Bewerbung für das entsprechende Fach abgeben. Hierbei können die Note aus der Klausur für die ausgewählte Veranstaltung sowie eine Priorität für die Bewerbung eingetragen werden. Zusätzlich können der Bewerbung noch weitere Details beigefügt werden. Wie bereits in der Prozessanalyse beschrieben, können sich Studierende zur Verhinderung von Massenbewerbungen auf maximal drei Veranstaltungen gleichzeitig bewerben.

My Applications

Unter dieser Ansicht können die Studierenden ihre Bewerbungen einsehen und den Status dieser verfolgen. Zudem können Bewerbungen, die noch nicht akzeptiert oder abgelehnt wurden, bearbeitet werden (wie auch in der „Tutor Application“ Ansicht möglich).

My Contracts

Sobald ein Vertrag für einen Studierenden als Tutor angelegt wurde, erscheint dieser in der Ansicht „My Contracts“. Hier können alle Verträge des Benutzers inklusive der relevanten Informationen (Vertragsstart, Vertragsende und Wochenstunden des Hauptvertrags[38]) eingesehen werden. Sobald der Benutzer einen einzelnen Vertrag auswählt, erhält er hierzu eine detailliertere Ansicht. Diese Ansicht enthält alle Vertragsdaten des ausgewählten Vertrags. Zusätzlich werden in einer Tabelle alle für die Vertragserstellung erforderlichen Dokumente gelistet. Die vom Tutorbetrieb bereitgestellten Formulare sowie das aktuelle Merkblatt können über diese Tabelle heruntergeladen werden. Zudem wird dem Studierenden angezeigt, welche Dokumente noch eingereicht werden müssen und welche dem Tutorbetrieb bereits vorliegen. Zur besseren

[38] Für den Fall, dass ein Vertragssplitting vorliegt, bezeichnet der Hauptvertragszeitraum den ersten Vertragszeitraum.

visuellen Darstellung werden die Einträge, die bereits vorliegen, in der Tabelle grün markiert und die Dokumente, die noch fehlen, rot.

Help

Die Hilfeseite wurde nach den ersten Tests eingerichtet und hilft vor allem Studierenden, die das Tutortool zum ersten Mal verwenden. Auf dieser Seite werden alle Buttons aus dem Dashboard erklärt und können auch aus der Hilfeseite heraus verwendet werden.

5.5.2 Nutzertests

Alle Nutzertests wurden auf die gleiche Art und Weise durchgeführt. Mit den das Tutortool testenden Studierenden wurden Einzeltermine für Videokonferenzen vereinbart. Die Testpersonen sollten darin ihren Bildschirm teilen. Entsprechend der Think Aloud Methodik (Van Someren et al., 1994) wurden die Testpersonen zusätzlich gebeten, ihre Gedanken laut auszusprechen, während sie das Tutortool bedienen. Den Testpersonen wurden dazu folgende Aufgaben gestellt:

1. Erstellen Sie einen Account im Tutortool
2. Bewerben Sie sich für zwei ausgewählte, verschiedene Fächer
3. Überprüfen Sie die Status der Bewerbungen
4. Überprüfen Sie die einzureichenden Dokumente und die Vertragsdaten

Hierbei wurde die nächste Aufgabenstellung immer erst nach Erfüllung der letzten Aufgabe genannt. Abschließend wurde mit den Testpersonen das Nutzererlebnis besprochen und potenzielle Verbesserungen diskutiert. Nach Beendigung der Gespräche wurde den Testpersonen der SUS-Fragebogen zum Ausfüllen zugeschickt. Auf diese Weise wurde das Tutortool mit fünf verschiedenen Studierenden getestet. Dabei hatten zwei Testpersonen Vorkenntnisse mit dem alten Tutortool und drei Testpersonen kannten das alte Tutortool nicht.

5.5.3 Ergebnisse

Ebenso wie die Nutzertests der vorigen Design Cycles fielen die Ergebnisse dieses Tests deutlich besser aus als die Bewertungen der alten Version des Tutortools. Auf die Freitextfragen wurden vor allem Antworten gegeben, die das Design der Benutzeroberfläche lobten. In den persönlichen Gesprächen während der Videokonferenzen wurde zudem mehrmals die gute Integration der Funktionalitäten angesprochen. Auch hatten die Benutzer während der Tests selten Probleme bei der Navigation in der Applikation. Alle Nutzer zeigten ferner bei der Verwendung der Applikation ähnliche Verhaltensmuster. So wurde beispielsweise das Erstellen eines Profils stets problemlos gelöst. Die Testpersonen verwendeten zunächst die Hilfeseite, um einen besseren Überblick für nachfolgenden Aufgaben zu bekommen. Hierauf folgte die Bewerbung für die Veranstaltungen. Diese Aufgabe wurde ebenfalls ohne größere Komplikationen absolviert. Die dritte Aufgabe sorgte bei zwei Testpersonen für kurze Unsicherheit, da diese statt der „My Applications“ Ansicht die „My Contracts“ Ansicht auswählten. Dies klärte sich aber in beiden Fällen nach wenigen Sekunden. Die letzte Aufgabe wurde dann wieder von allen Testpersonen ohne Probleme absolviert.

Zusammengefasst stimmen die Beobachtungen aus den Videokonferenzen mit den Ergebnissen der SUS-Fragebögen, die in Abbildung 17 genauer betrachtet werden können[39], überein. Keine der Testpersonen hatte größere Probleme bei der Verwendung des Tutortools. Die kleineren Unsicherheiten, die auftraten, wurden durch das Aufrufen der Hilfeseite beseitigt. Basierend auf den Antworten auf die Frage „Was könnte an der Anwendung verbessert werden?“ wurden nach den Tests Verbesserungen an der Implementierung vorgenommen. So wurden im Hilfebereich die Beispielbuttons mit Links zu den entsprechenden Ansichten versehen, damit diese voll funktional verwendet werden können. Zudem wurden in mehreren Ansichten kurze Erklärungstexte bezüglich der Funktionalitäten der jeweiligen Ansicht eingefügt.

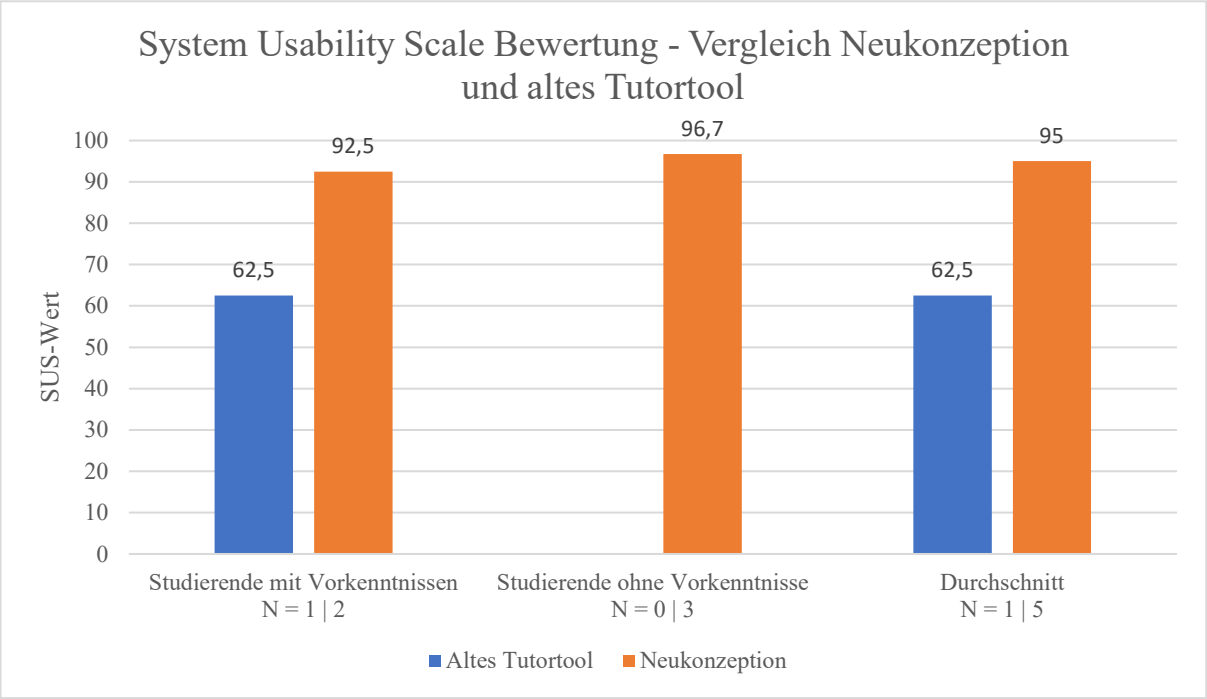


Abbildung 17: SUS Bewertungen für das Tutortool durch die Studierenden/Tutoren
Quelle: eigene Erhebung

5.6 Design Cycle 4

Während dieses abschließenden Design Cycles wurden der Anwendung noch Funktionalitäten hinzugefügt, die nicht zwingend zu den Kernfunktionalitäten gehören. Hierbei handelt es sich um das Integrieren des sogenannten Verleihtools sowie die Vorbereitung auf einen möglichen Anschluss an das LDAP des LRZ durch Shibboleth-Single-Sign-On Authentifizierung.

5.6.1 Implementierung

Die folgenden Absätze sollen die zuvor beschriebenen zusätzlichen Funktionalitäten, die in diesem Design Cycle in das Tutortool eingefügt wurden, vorstellen.

[39] Mit den Studierenden ohne Vorkenntnisse des alten Tutortools wurde zu Beginn des Relevance Cycles keine SUS-Wert-Erhebung durchgeführt.

Verleihtool

Das Verleihtool ist eine MERN-Stack basierte Webanwendung, die Ende 2020 vom Autor dieser Masterarbeit im Rahmen einer Anstellung als studentische Hilfskraft für den Tutorbetrieb entwickelt wurde. Aufgrund der Verschiebung des Lehrbetriebs in den Online Bereich nach Ausbruch der SARS-CoV-2 Pandemie wurden der Fakultät für Informatik zusätzliche Gelder bereitgestellt. Diese wurden zum Kauf von Hardware verwendet, die wiederum den Tutoren bereitgestellt wird, um die Qualität der Tutorien auch im Online Bereich sicherzustellen. Der Grund für die Entwicklung des beschriebenen Verleihtools war die Abschaffung der zwischen dem Tutorbüro und der RBG versandten Exceltabellen zur Dokumentation und Organisation der Ausleihen. Diese Aufgabe übernahm das Verleihtool nach Inbetriebnahme. Hierbei können Leihen organisiert und die zugehörigen Leihscheine erstellt und als „.pdf“ exportiert werden.

Das Verleihtool wurde im vorliegenden Schritt in die bestehende Neukonzeption integriert und erweitert. Zunächst wurde hierfür eine neue Benutzerrolle eingeführt. Diese Rolle mit der Bezeichnung „RBG“ wurde für die Mitarbeiter der RBG erstellt, um diesen direkten Zugriff auf die Funktionalitäten des integrierten Verleihtools zu ermöglichen. Das Verleihtool im vollen Umfang ist lediglich für die Rollen „Administrator“ und „RBG“ verfügbar.

Die Hauptansicht des Verleihtools besteht aus einer Tabelle, in der alle Leihen gelistet werden. Wie in den meisten Ansichten kann der Inhalt der Tabelle je nach Status gefiltert werden. Im Falle der Verleihtabelle gibt es für den Status folgende Auswahlmöglichkeiten: „Unvollständig“, „LS[40] verschickt“, „HW[41] ausgegeben“ und „Abgeschlossen“. Über die Hauptansicht können ferner neue Leihen angelegt oder bestehende Leihen editiert werden. Das Anlegen einer neuen Ausleihe kann auf zwei unterschiedliche Arten erfolgen. Zum einen kann eine Leihe angelegt werden, ohne dass sich ein Tutor auf Hardware beworben hat. Hierzu müssen alle für die Leihe erforderlichen Daten von den Mitarbeitern des Tutorbüros eingetragen werden. Die zweite Möglichkeit besteht darin, eine Leihe über die Leihanfrage anzulegen. Diese Option wurde für die Neukonzeption neu eingeführt. Die Tutoren können sich in ihrem Dashboard über das Feld „Apply for Rental Hardware“ auf Leihhardware bewerben. Die Tutoren müssen hierbei die zusätzlichen für eine Leihe erforderlichen Felder (bestehend aus ihrer kompletten Adresse, ihrer Telefonnummer und der TUM-ID) ausfüllen sowie auswählen, welche Hardware sie für ihre Tutorien ausleihen wollen. Nachdem von den Tutoren Leihanfragen gestellt wurden, können diese unter „Leihanfragen“ eingesehen werden. Sobald eine Leihanfrage akzeptiert wurde, kann über den Button „Leihe erstellen“ eine neue Leihe erstellt werden, bei der die von den Tutoren vorausgefüllten Feldern bereits eingetragen werden.

Shibboleth-Single-Sign-On

Die Möglichkeit, die Single-Sign-On Funktion über den Anschluss an die LDAP Server des LRZ zu verwenden, wurde in diesem Design Cycle vorbereitet. In der aktuellen Testversion über den Cloudanbieter ist dies jedoch noch nicht verfügbar, da für die Verwendung ein Antrag

[40] Leihschein
[41] Hardware

beim LRZ gestellt werden muss, um den Zugriff der Applikation auf die Server freizuschalten. Hierbei wurde auf eine vorzeitige Verbindung mit der LRZ verzichtet, da sonst potenziell reale Nutzerdaten über TUM-Fremde Server geschickt werden. Aus diesem Grund wurde lediglich eine Vorbereitung in den Quellcode eingefügt, der dann bei einer Inbetriebnahme auf einem TUM eigenen Server aktiviert wird.

5.6.2 Nutzertest Resultate

Abschließend zu diesem Design Cycle wurden den Mitarbeitern des Tutorbüros erneut SUS-Fragebögen zugeschickt, um die Benutzerfreundlichkeit der Integrierung des Verleihtools zu überprüfen. Die hierbei erzielten Ergebnisse sind mehr als zufriedenstellend mit einem Durchschnittswert von 97,5, der sich aus den Einzelwertungen 100 und 95 ergibt. Hier gab es auch in den Freitextfeldern keine weiteren Beanstandungen.

5.7 Zusammenfassung Design Cycles

Während der Design Cycles wurde in insgesamt vier verschiedenen Zyklen ein Softwareartefakt implementiert, getestet und auf Basis des erhaltenen Feedbacks verbessert. Das Softwareartefakt stellt die Neukonzeption des Tutortools dar. Dieses wurde zunächst auf Basis der im Relevance Cycle erarbeiteten Prozesse gestaltet und umfasst alle benötigten Kernfunktionalitäten, um den Tutorbetrieb bei der Erfüllung der täglichen Aufgaben zu unterstützen. Zusätzlich wurden weitere Funktionalitäten eingefügt, um die Benutzerfreundlichkeit zu steigern.

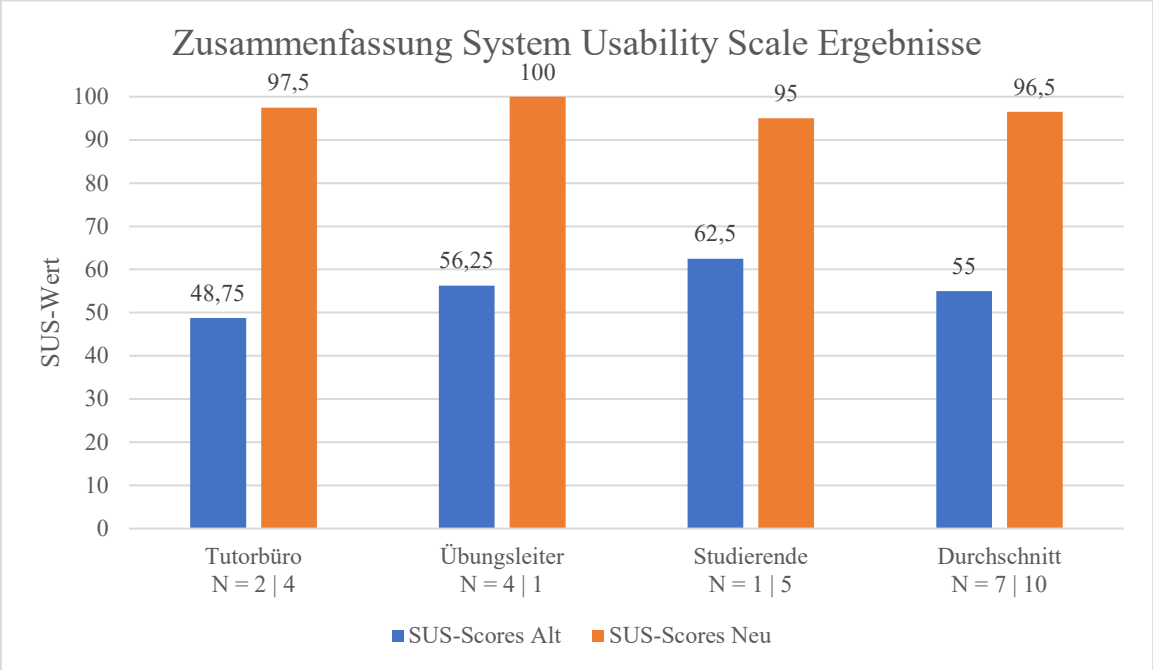


Abbildung 18: Zusammenfassung SUS Ergebnisse
Quelle: eigene Darstellung

Die Ergebnisse der Nutzertests zeigen hierbei deutlich überdurchschnittliche Ergebnisse. Vor allem im direkten Vergleich mit der alten Version des Tutortools fallen die Ergebnisse hierbei mehr als zufriedenstellend aus. Eine Zusammenfassung aller Ergebnisse der SUS-Fragebögen kann in Abbildung 18 betrachtet werden.

6 Schlussbetrachtung

In der vorliegenden Masterarbeit wurde eine Neukonzeption des Tutortools der Fakultät für Informatik an der TU München erarbeitet. Hierzu wurde auf Basis der Design-Science-Research Methodik nach Hevner et al. (2004) gearbeitet und die darin vorgestellten Richtlinien verfolgt. Auf dieser Grundlage wurde der Arbeitsprozess wie in Hevner (2007) beschrieben in drei Zyklen eingeteilt. In diesem Kapitel werden zunächst die Forschungsfragen im Fazit aufgegriffen und im Anschluss ein Ausblick auf mögliche zukünftige Forschungsfragen und das weitere Vorgehen mit der Neukonzeption beziehungsweise Neuimplementierung gegeben.

6.1 Fazit

In Kapitel 1.2 wurden vier Forschungsfragen für die vorliegende Arbeit definiert, die allesamt beantwortet werden konnten. Zunächst wurde im Rigor Cycle auf die erste Forschungsfrage („Wie ist der aktuelle Stand der Literatur zum Thema Softwareevolution?“) eingegangen. Hierbei wurde eine Literaturrecherche zum Thema „Softwareevolution“ nach vom Brocke et al. (2009) durchgeführt. Die Ergebnisse wurden sowohl quantitativ als auch qualitativ ausgewertet. Insgesamt wurden 30 relevante Veröffentlichungen für zwei Konzeptfragen identifiziert und anschließend vorgestellt. Hierbei zeigte sich, dass das Thema der Softwareevolution in der Forschung stark diskutiert wird und vor allem in den letzten Jahren Methoden betrachtet wurden, um die Phase der Softwarewartung zu vereinfachen. Für den Implementierungsteil waren dabei insbesondere Veröffentlichungen interessant, die die Qualität des Quellcodes verbessern.

Der zweite Forschungsfrage („Wie lassen sich die aktuellen Prozesse innerhalb des Tutorbetriebs modellieren, wo treten Probleme auf und wie lassen sich die Prozesse optimieren?“) wurde zusammen mit der dritten Forschungsfrage („Welche Funktionalitäten muss die Neukonzeption des Tutortools für den Tutorbetrieb bereitstellen, damit sie alle (intendierten) Nutzergruppen optimal unterstützt?“) im Relevance Cycle dieser Arbeit betrachtet. In diesem Cycle wurden die Prozesse des Tutorbetriebs, die in Zusammenhang mit dem Tutortool stehen, erhoben, mit BPMN 2.0 abgebildet und auf dieser Basis für die Neuimplementierung optimiert. Hierbei lag der Fokus vor allem auf den Kernprozessen des Tutorbetriebs im Zusammenhang mit dem Tutortool. Später wurden auf Basis dieser Prozesse neben den Kernfunktionalitäten (die das Tutortool durch diese abgebildeten Prozesse beinhalten muss) weitere (optionale) Funktionalitäten für die Design Cycles definiert. Ebenso wurden Funktionalitäten des alten Tutortools betrachtet und begründet, weshalb diese in der Neukonzeption nicht mehr in Betracht gezogen wurden.

Die vierte Forschungsfrage („Wie können die erhobenen Funktionalitäten für das Tutortool implementiert werden?“) wurde durch die Implementierung der Neukonzeption beantwortet. Auf Grundlage der anderen Forschungsfragen wurde in insgesamt vier Design Cycles eine MERN-Stack basierte Webapplikation entwickelt und iterativ mit den intendierten Nutzergruppen getestet und verbessert. Die Nutzertests, die zum Abschluss jedes Design Cycles mit den jeweiligen Nutzergruppen durchgeführt wurden, ergaben eine überdurchschnittlich hohe Benutzerfreundlichkeit des Systems (hierbei sollte erwähnt werden, dass die Größe der Gruppe an Testpersonen dabei nicht repräsentativ war, sondern hauptsächlich dazu genutzt wurde, Feedback zur Neukonzeption während der Implementierungsphase zu sammeln).

Die Ergebnisse der Nutzertests geben Anlass zu der Vermutung, dass auch im Live Betrieb der Neuimplementierung nur wenige Probleme für die Nutzer auftreten werden. Dies kann jedoch zum Zeitpunkt der Abgabe dieser Masterarbeit noch nicht garantiert werden. Daher ist es für die Zeit nach dem Roll-Out der Neuimplementierung erforderlich, dass weiterhin mit den Nutzern des Tutortools (vor allem mit den Mitarbeitern des Tutorbüros) zusammengearbeitet wird, um eine hohe Qualität der Applikation sicherzustellen. Ein Nachteil der Neuimplementierung ist ferner, dass die Datenbank des alten Tutortools nicht kompatibel mit der der Neuimplementierung ist, wodurch zunächst keine Alt-Datensätze zur Verfügung stehen.

Durch die starke Einbindung der Nutzergruppen (vor allem der Mitarbeiter des Tutorbetriebs) durch die Design Science Methodik konnte ein starker Lern- und Trainingseffekt bei den Nutzern beobachtet werden. Dieser zeigte sich deutlich während des ersten Design Cycles. Zum Ende dieses Cylces konnte bereits eine gesteigerte Sicherheit bei der Bedienung durch die Nutzer beobachtet werden. Dieser positive Nebeneffekt ist (besonders bezüglich der bald nach Abgabe dieser Masterarbeit geplanten Übernahme der Neukonzeption in den Arbeitsbetrieb) von Vorteil, da hierdurch kein signifikantes zusätzliches Training der Nutzer notwendig wird.

Der wichtigste Punkt, den die vorliegende Neuimplementierung erfüllt, ist die Beendigung der unkontrollierten Softwareevolution des alten Tutortools durch ein Reengineering auf der Basis moderner Webtechnologien. Hinzu kommt (neben den strukturellen Verbesserungen durch das Reeingineering) auch die Einbringung funktionaler Optimierungen in das Tutortool. Die Neuimplementierung des Tutortools wurde zudem so konzipiert, dass mögliche Veränderungen in den Anforderungen oder Erweiterungen unkompliziert eingefügt werden können. Dies soll kontrollierte Softwareevolution unterstützen und die Minimierung negativer Effekte gewährleisten. Hierdurch steht dem Tutorbetrieb nach der Inbetriebnahme der Neukonzeption eine moderne, leicht wartbare und flexibel erweiterbare Lösung für die zukünftige Arbeit zur Verfügung.

6.2 Ausblick

Durch die Ergebnisse der verschiedenen Zyklen ergeben sich interessante Forschungsfragen für die Zukunft. Aus dem Rigor Cycle ging hervor, dass vor allem zur Vermeidung von schlechter Softwareevolution die Einführung von Methoden aus dem Knowledge Management eine gute Basis für Neuimplementierungen bildet. Hierbei kann vor allem die Verwendung von Knowledge Management Methoden in frühen Stadien der Implementierung (z.B.: Dokumentation von Entscheidungen) große Vorteile für die spätere Wartung der Software mit sich bringen (Johanssen et al., 2017, S. 1 ff.). In der Literaturrecherche wurden für die frühen Phasen des Software Development Lifecycles bislang nur zwei Veröffentlichungen (im Kontext des Knowledge Managements) identifiziert. Auf dieser Basis könnten für die Zukunft eine Vielzahl an Forschungsfragen interessant werden, die sich vor allem mit dem Einführen von Knowledge Management Methoden in die frühen Phasen des Software Development Lifecycles beschäftigen, um spätere Softwareevolution besser steuern und gleichsam die Arbeit von in der Softwarewartung tätigen Personen erleichtern zu können.

Für die Implementierung des Tutortools gibt es diverse Forschungsfragen, die für die Zukunft interessant werden könnten. So wäre es beispielsweise denkbar, die API so zu erweitern, dass den Mitarbeitern des Servicebüros für Hilfskräfte (SB-Z) ermöglicht wird, die benötigten

Informationen und Dokumente zur Vertragserstellung direkt aus dem Tutortool für ihr Verwaltungsprogramm zu erhalten. Eine weitere Erweiterungsmöglichkeit für das neue Tutortool wäre, die Zeitüberwachung von Tutoren direkt zu integrieren, um die bisher genutzten, separaten Formulare abzuschaffen.

Wie zuvor bereits erwähnt, ist eine Inbetriebnahme der Neuimplementierung im Anschluss an die Abgabe dieser Masterarbeit geplant. Hierfür wurde im Rahmen der seit der SARS-CoV-2 Pandemie üblichen Statusmeetings des Tutorbetriebs der folgende zeitliche Ablauf geplant:

Zunächst werden Anfang Juni 2021 die nötigen Vorkehrungen getroffen, um die Neukonzeption auf einem Server der TU München zum Testen der Serverumgebung in Betrieb zu nehmen. Sobald das System zuverlässig auf einem solchen Server läuft, werden die Zugänge für das alte Tutortool gesperrt und die Datenbank (des alten Tools) auf dem alten Testserver eingespielt. Anschließend (ca. Mitte Juni) wird der alte Liveserver mit einem neuen Betriebssystem ausgestattet und alle Einstellungen zurückgesetzt. Danach wird (unter Berücksichtigung der Erfahrungen aus der Inbetriebnahme auf dem Testserver) eine ordentliche Konfiguration des Tutortools eingespielt. Abschließend sollte die Neukonzeption pünktlich zu Beginn der Tutorenplanung für das Wintersemester 2021/2022 dem Tutorbetrieb vollumfänglich zur Verfügung stehen. Für den Livebetrieb wird auch weiterhin eng mit dem Tutorbetrieb zusammengearbeitet, um auftretende Fehler schnellstmöglich im Quellcode beseitigen zu können. Da die Datenbank des alten Tutortools nicht kompatibel mit der neuen Datenbank ist, wird (wie zuvor erwähnt) das alte Tutortool zu Archivzwecken für ungefähr ein Jahr parallel auf dem Testserver weiter aktiv sein. Der Hauptzweck des Parallelbetriebs ist vor allem das Überprüfen von Weiterbeschäftigungen. Der Parallelbetrieb wird zum Ende der Tutorenplanung für das Sommersemester 2022 eingestellt, das alte Tutortool vollständig aus dem Betrieb entfernt und die Datenbank offline archiviert. Ab diesem Zeitpunkt wird die Neukonzeption des Tutortools wie in dieser Masterarbeit erarbeitet im Alleinbetrieb von der Fakultät für Informatik für den Tutorbetrieb genutzt werden.

Literaturverzeichnis

Abramov, D. (2021). Redux. Bezogen von https://redux.js.org/ Zugegriffen am 21.04.2021

AIS. (2016). Senior Scholars' Basket of Journals, Assoziation for Information Systems (AIS). Bezogen von https://aisnet.org/page/SeniorScholarBasket Zugegriffen am 16.02.2021

Ajouli, A. (2015). *A Shadow Structure for Modularity of Java Program Evolution.* Paper presented at the 2015 41st Euromicro Conference on Software Engineering and Advanced Applications.

Ajouli, A. (2021). SEA: An UML Profile for Software Evolution Analysis in Design Phase. *Advances in Science, Technology and Engineering Systems Journal, 6*, 1334-1342. doi:10.25046/aj0601153

Ajouli, A., & Henchiri, K. (2019). *MODEM: an UML profile for MODEling and Predicting software Maintenance before implementation.* Paper presented at the 2019 International Conference on Computer and Information Sciences (ICCIS).

Augusten, S., & chrissikraus. (2018). Was ist ein Solution Stack? Bezogen von https://www.dev-insider.de/was-ist-ein-solution-stack-a-778556/ Zugegriffen am 02.03.2021

Awang, N. H., Wan-Kadir, W. M., & Shahibuddin, S. (2011). *A Middleware based, Policy Driven Adaptation Framework to Simplify Software Evolution.* Paper presented at the ENASE.

Axios. (2021). Axios. Bezogen von https://axios-http.com/ Zugegriffen am 21.04.2021

Bajaj, K., Patel, H., & Patel, J. (2015). *Evolutionary software development using test driven approach.* Paper presented at the 2015 International Conference and Workshop on Computing and Communication (IEMCON).

Bangor, A., Kortum, P., & Miller, J. (2009). Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of usability studies, 4*(3).

Bekanntmachung der Bayerischen Staatsregierung über die Pflicht zur Verfassungstreue im öffentlichen Dienst (Verfassungstreue-Bekanntmachung – VerftöDBek) vom 3. Dezember 1991 (AllMBl. S. 895, StAnz. Nr. 49), die zuletzt durch Bekanntmachung vom 27. September 2016 (AllMBl. S. 2138) geändert worden ist, (1991).

Berzins, L., Shing, M., Riehle, R., & Nogueira, J. (2000). *Evolutionary computer aided prototyping system (CAPS).* Paper presented at the Proceedings. 34th International Conference on Technology of Object-Oriented Languages and Systems-TOOLS 34.

Breivold, H. P., Crnkovic, I., & Larsson, M. (2012). A systematic review of software architecture evolution research. *Information and Software Technology, 54*(1), 16-40.

Brooke, J. (1996). SUS - A quick and dirty usability scale. *Usability evaluation in industry, 189*.

Chang, C.-H., Lu, C.-W., & Chu, W. C. (2008). *Improving software integration from requirement process with a model-based object-oriented approach.* Paper presented at the 2008 Second International Conference on Secure System Integration and Reliability Improvement.

Cho, E. S., Cha, J. E., & Yang, Y. J. (2004). *MARMI-RE: A method and tools for legacy system modernization.* Paper presented at the International Conference on Software Engineering Research and Applications.

Chu, W. C., Chang, C.-H., & Lu, C.-W. (2008). *Model-based Object-oriented Requirement Engineering and its Support to Software Documents Integration.* Paper presented at the Software Engineering Research and Practice.

Cobo, H., Mauco, V., Romero, M., & Rodriguez, C. (1999). *A tool to reengineer legacy systems to object-oriented systems.* Paper presented at the International Conference on Conceptual Modeling.

Cooper, H. M. (1988). Organizing knowledge syntheses: A taxonomy of literature reviews. *Knowledge in society, 1*(1), 104-126.

de Vasconcelos, J. B., Kimble, C., Carreteiro, P., & Rocha, Á. (2017). The application of knowledge management to software evolution. *International Journal of Information Management, 37*(1), 1499-1506.

Eitan, A. T., Smolyansky, E., Harpaz, I. K., & Perets, S. (2021a). Connected Papers. Bezogen von https://www.connectedpapers.com/about Zugegriffen am 21.04.2021

Eitan, A. T., Smolyansky, E., Harpaz, I. K., & Perets, S. (2021b). Connected Papers. Bezogen von https://www.connectedpapers.com/ Zugegriffen am 20.04.2021

Facebook. (2021). Create React App - Getting Started. Bezogen von https://create-react-app.dev/docs/getting-started Zugegriffen am 21.04.2021

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (Vol. 7): University of California, Irvine Irvine.

Godfrey, M. W., & German, D. M. (2008). *The past, present, and future of software evolution.* Paper presented at the 2008 Frontiers of Software Maintenance.

Gonçalves, R., Lima, I., & Costa, H. (2015). *Using TDD for developing object-oriented software—A case study.* Paper presented at the 2015 Latin American Computing Conference (CLEI).

Haitzer, T., Navarro, E., & Zdun, U. (2017). Reconciling software architecture and source code in support of software evolution. *Journal of Systems and Software, 123*, 119-144.

Harzing, A.-W. (2020). Journal Quality List. (Sixty-seventh Edition).

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems, 19*(2), 4.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 75-105.

Hewing, M. (2014). A Blueprint of the Customer–Design of a Method for an extended View on Customer Processes in BPM. In *Business Process Blueprinting*: Springer.

IEEE. (1990). IEEE standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, 1-84.

IEEE Computer Society. (2021). IEEE Transactions on Software Engineering. Bezogen von https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=32 Zugegriffen am 16.02.2021

IEEE Xplore. (2021). IEEE Xplore. Bezogen von https://ieeexplore.ieee.org/ Zugegriffen am 17.02.2021

Johanssen, J. O., Kleebaum, A., Bruegge, B., & Paech, B. (2017). *Towards a Systematic Approach to Integrate Usage and Decision Knowledge in Continuous Software Engineering.* Paper presented at the CSE@ SE.

Khan, R., Azam, F., Maqbool, B., & Anwar, M. W. (2020). *A Framework for Automated Reengineering of BPMN Models by Excluding Inefficient Activities.* Paper presented at the Proceedings of the 2020 9th International Conference on Software and Computer Applications.

Kimura, S., Hotta, K., Higo, Y., Igaki, H., & Kusumoto, S. (2014). *Does return null matter?* Paper presented at the 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE).

Konersmann, M. (2018). *On executable models that are integrated with program code.* Paper presented at the MODELS Workshops.

Konersmann, M., & Goedicke, M. (2012). A conceptual framework and experimental workbench for architectures. In *Software Service and Application Engineering* (pp. 36-52): Springer.

Kontogiannis, K., & Patil, P. (1999). *Evidence driven object identification in procedural code.* Paper presented at the STEP'99. Proceedings Ninth International Workshop Software Technology and Engineering Practice.

Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE, 68*(9), 1060-1076.

Lehman, M. M., & Belady, L. A. (1985). *Program evolution: processes of software change*: Academic Press Professional, Inc.

Lehman, M. M., Ramil, J. F., Wernick, P. D., Perry, D. E., & Turski, W. M. (1997). *Metrics and laws of software evolution-the nineties view.* Paper presented at the Proceedings Fourth International Software Metrics Symposium.

Liu, X., Yang, H., Zedan, H., & Cau, A. (2000). *Speed and scale up software reengineering with abstraction patterns and rules.* Paper presented at the Proceedings International Symposium on Principles of Software Evolution.

Ludewig, J., & Lichter, H. (2013). *Software Engineering: Grundlagen, Menschen, Prozesse, Techniken*: dpunkt. verlag.

Mohan, A., Gold, N., & Layzell, P. (2004). *An initial approach to assessing program comprehensibility using spatial complexity, number of concepts and typographical style.* Paper presented at the 11th Working Conference on Reverse Engineering.

Mokni, A., Huchard, M., Urtado, C., Vauttier, S., & Zhang, H. Y. (2014a). *Formal rules for reliable component-based architecture evolution.* Paper presented at the International Conference on Formal Aspects of Component Software.

Mokni, A., Huchard, M., Urtado, C., Vauttier, S., & Zhang, H. Y. (2014b). *A three-level formal model for software architecture evolution.* Paper presented at the SATToSE: Seminar on Advanced Techniques and Tools for Software Evolution.

MongoDB. (2021). MERN Stack. Bezogen von https://www.mongodb.com/mern-stack Zugegriffen am 02.03.2021

Mongoose. (2021). Bezogen von https://mongoosejs.com/ Zugegriffen am 21.04.2021

Newcomb, P., & Kotik, G. (1995). *Reengineering procedural into object-oriented systems.* Paper presented at the Proceedings of 2nd Working Conference on Reverse Engineering.

Node.js. (2019). Overview of Blocking vs Non-Blocking. Bezogen von https://nodejs.org/en/docs/guides/blocking-vs-non-blocking/ Zugegriffen am 21.04.2021

Node.js. (2020). About NodeJS. Bezogen von https://nodejs.org/en/about/ Zugegriffen am 02.03.2021

OMG Group. (2011). About the Business Process Model And Notation Specification Version 2.0. Bezogen von https://www.omg.org/spec/BPMN/2.0/PDF Zugegriffen am 28.02.2021

OpenJS. (2020). Express.js. Bezogen von https://expressjs.com/de/ Zugegriffen am 05.10.2020

Opferkuch, S., & Ludewig, J. (2004). Software-Wartung–Eine Taxonomie. *Softwaretechnik-Trends, 24*(2), 35-36.

Paech, B., Apel, S., Grunske, L., & Prehofer, C. (2016). Empirische Forschung zu Software-Evolution. *Informatik-Spektrum, 39*(3), 186-193.

Parnas, D. L. (1994). *Software aging.* Paper presented at the Proceedings of 16th International Conference on Software Engineering.

Piantadosi, V., Fierro, F., Scalabrino, S., Serebrenik, A., & Oliveto, R. (2020). How does code readability change during software evolution? *Empirical Software Engineering, 25*(6), 5374-5412.

Qureshi, M. K., Karidis, J., Franceschini, M., Srinivasan, V., Lastras, L., & Abali, B. (2009). *Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling.* Paper presented at the 2009 42nd Annual IEEE/ACM international symposium on microarchitecture (MICRO).

Ruiz, F. J. B., Molina, J. G., & García, O. D. (2017). On the application of model-driven engineering in data reengineering. *Information Systems, 72*, 136-160.

Sahoo, A., Kung, D., & Gupta, S. (2016). *An Agile Methodology for Reengineering Object-Oriented Software.* Paper presented at the SEKE.

Signavio. (2021). Signavio. Bezogen von https://www.signavio.com/ Zugegriffen am 20.04.2021

Subramanian, V. (2017). *Pro MERN Stack*: Springer.

Tamzalit, D., & Mens, T. (2016). Evolution patterns: Designing and reusing architectural evolution knowledge to introduce architectural styles. *arXiv preprint arXiv:1605.06289*.

Tsoukalas, D., Kehagias, D., Siavvas, M., & Chatzigeorgiou, A. (2020). Technical debt forecasting: an empirical study on open-source repositories. *Journal of Systems and Software, 170*, 110777.

Van Someren, M., Barnard, Y., & Sandberg, J. (1994). The think aloud method: a practical approach to modelling cognitive. *London: AcademicPress*.

VHB e.V. (2019). VHB-JOURQUAL 3. Bezogen von https://vhbonline.org/vhb4you/vhb-jourqual/vhb-jourqual-3 Zugegriffen am 21.02.2021

vom Brocke, J., Simons, A., Niehaves, B., Niehaves, B., Reimer, K., Plattfaut, R., & Cleven, A. (2009). Reconstructing the giant: On the importance of rigour in documenting the literature search process.

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, xiii-xxiii.

Zou, Y., & Kontogiannis, K. (2001). *A framework for migrating procedural code to object-oriented platforms.* Paper presented at the Proceedings Eighth Asia-Pacific Software Engineering Conference.

Легалов, А., Легалов, И., & Матковский, И. (2018). Instrumental Support of the Evolutionary Expansion of Programs Using a Incremental Development.

Anhang

Anhang A Abbildungen Allgemein

Anhang A.1 Richtlinien für Design Science Research nach Hevner

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Abbildung 19: Richtlinien für Design-Science Research
Quelle: Hevner, March, Park, und Ram (2004, S. 83)

Anhang A.2 Software Lifecycle

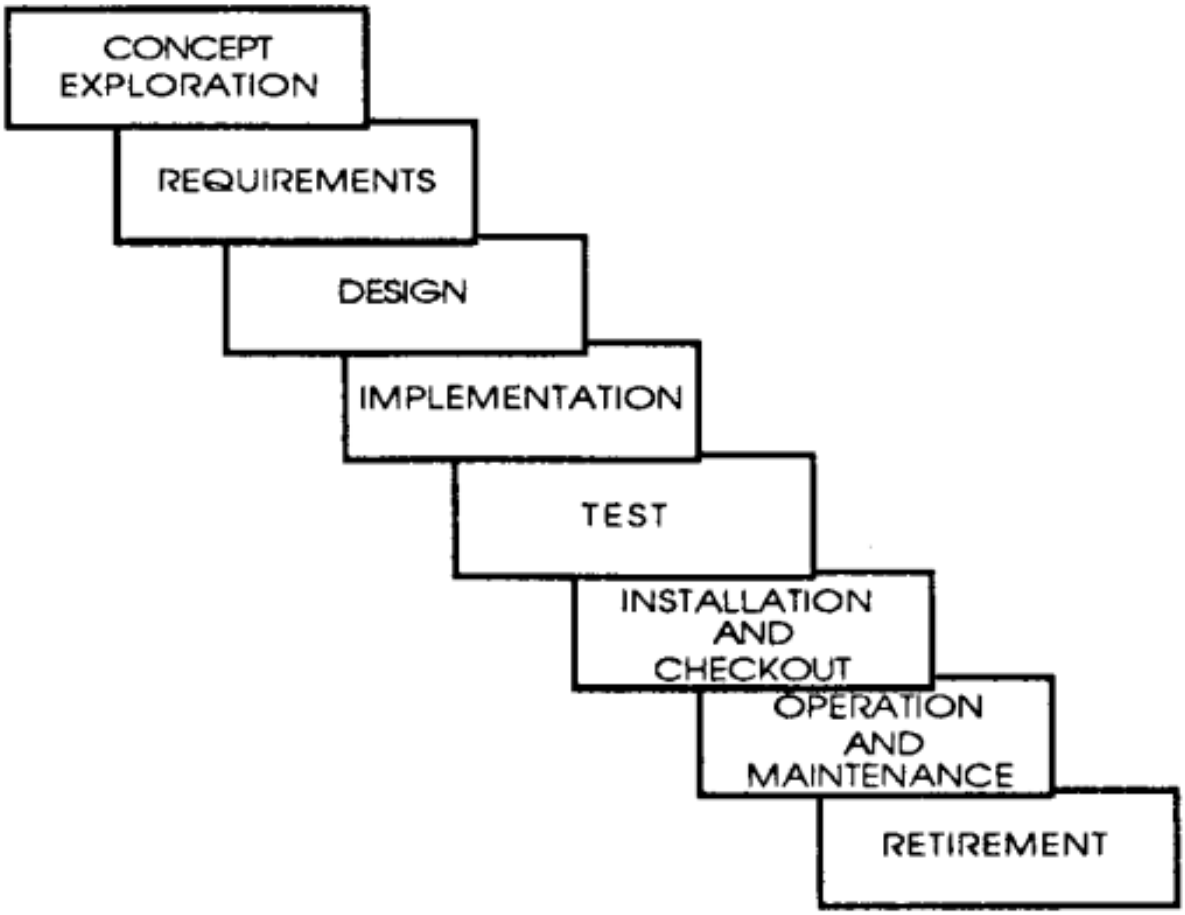


Abbildung 20: Software Lifecycle
Quelle: IEEE (1990, S. 68)

Anhang B Abbildungen Neukonzeption

Anhang B.1 Beispiel Aufenthaltstitel abgelaufen

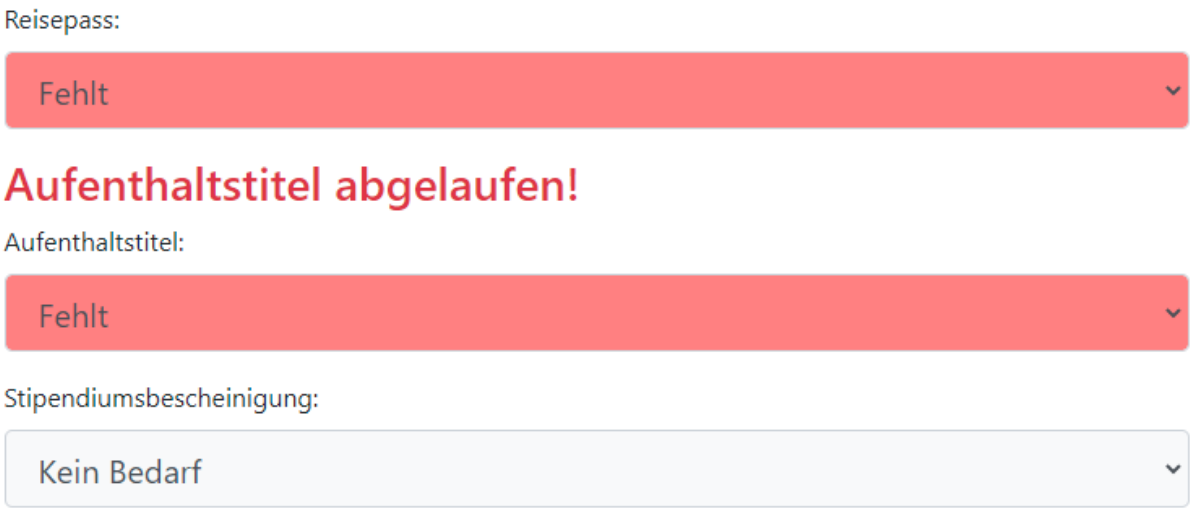


Abbildung 21: Beispiel Aufenthaltstitel abgelaufen
Quelle: eigene Darstellung

Anhang B.2 Dashboard Mitarbeiter Tutorbetrieb

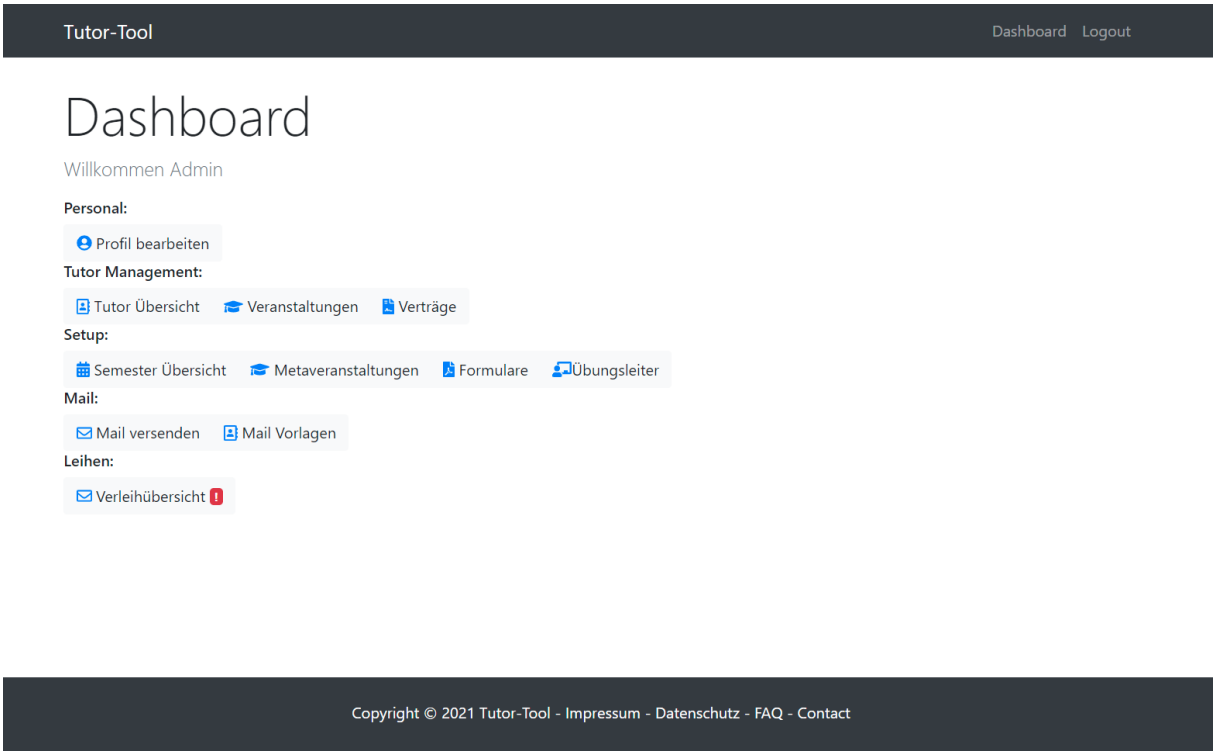


Abbildung 22: Dashboard Mitarbeiter Tutorbetrieb
Quelle: eigene Darstellung

Anhang B.3 Beispiel Verfassungsprüfung notwendig

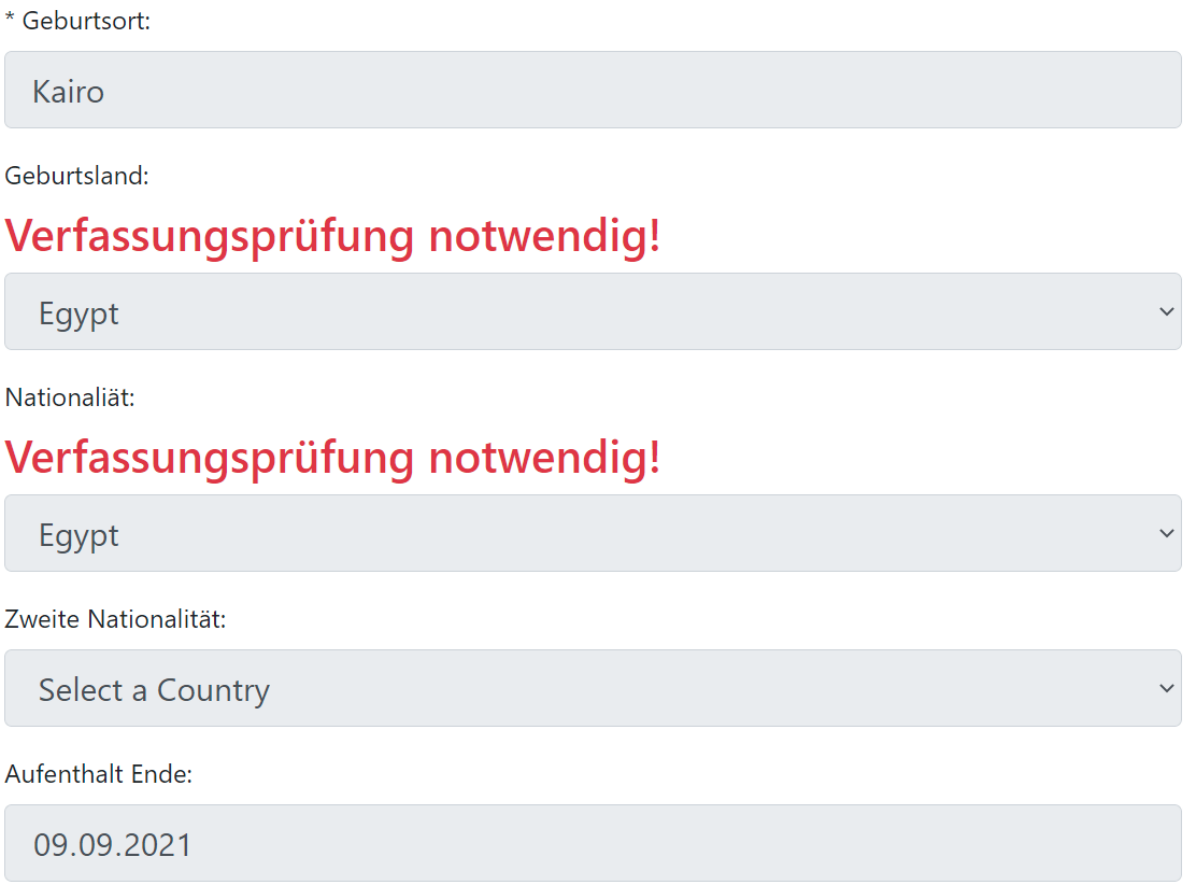


Abbildung 23: Beispiel Verfassungsprüfung notwendig
Quelle: eigene Darstellung

Anhang B.4 Warnung Überschreitung Wochenstunden

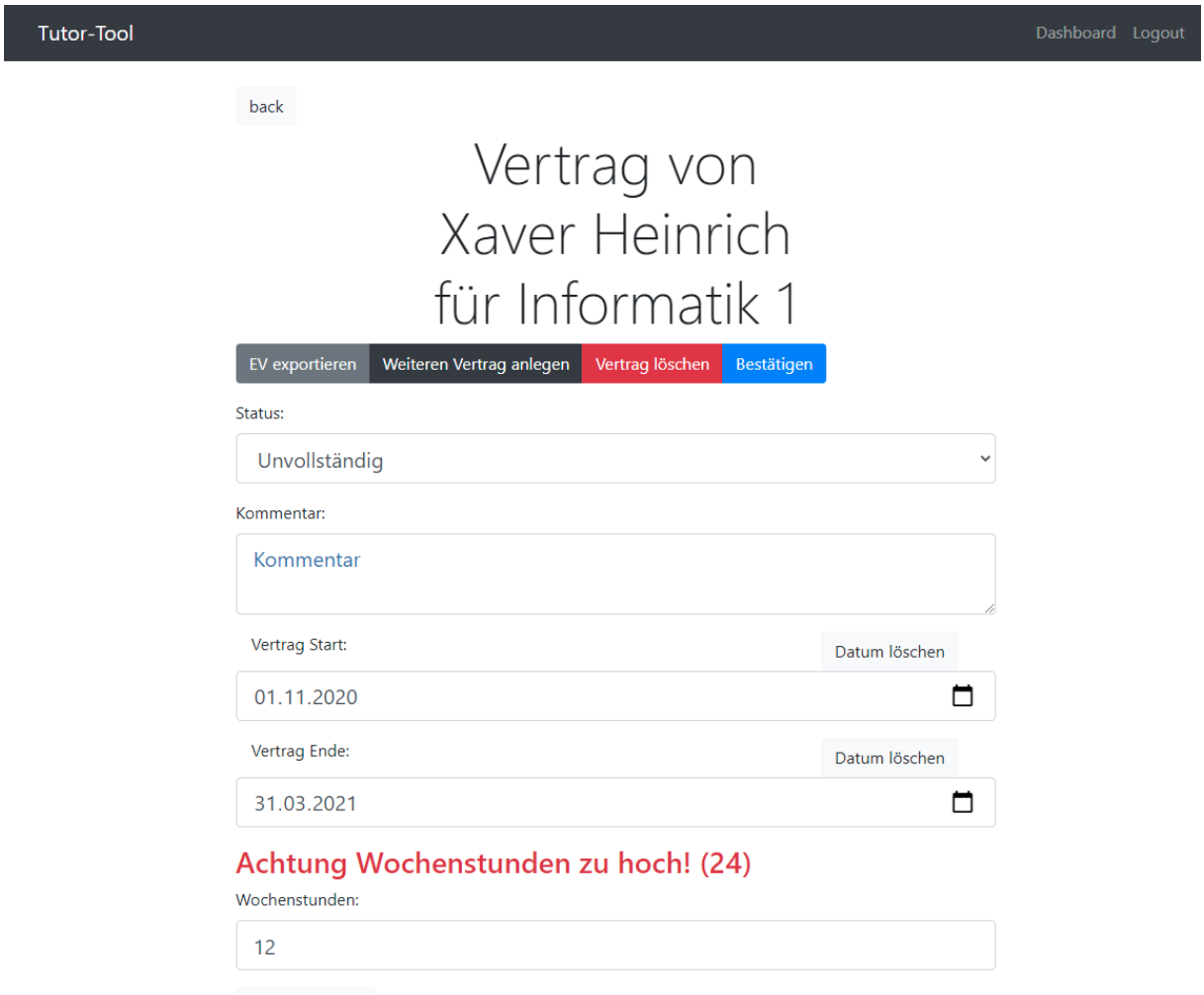


Abbildung 24: Warnung Überschreitung Wochenstunden
Quelle: eigene Darstellung

Anhang B.5 Dashboard Studierende/Tutoren

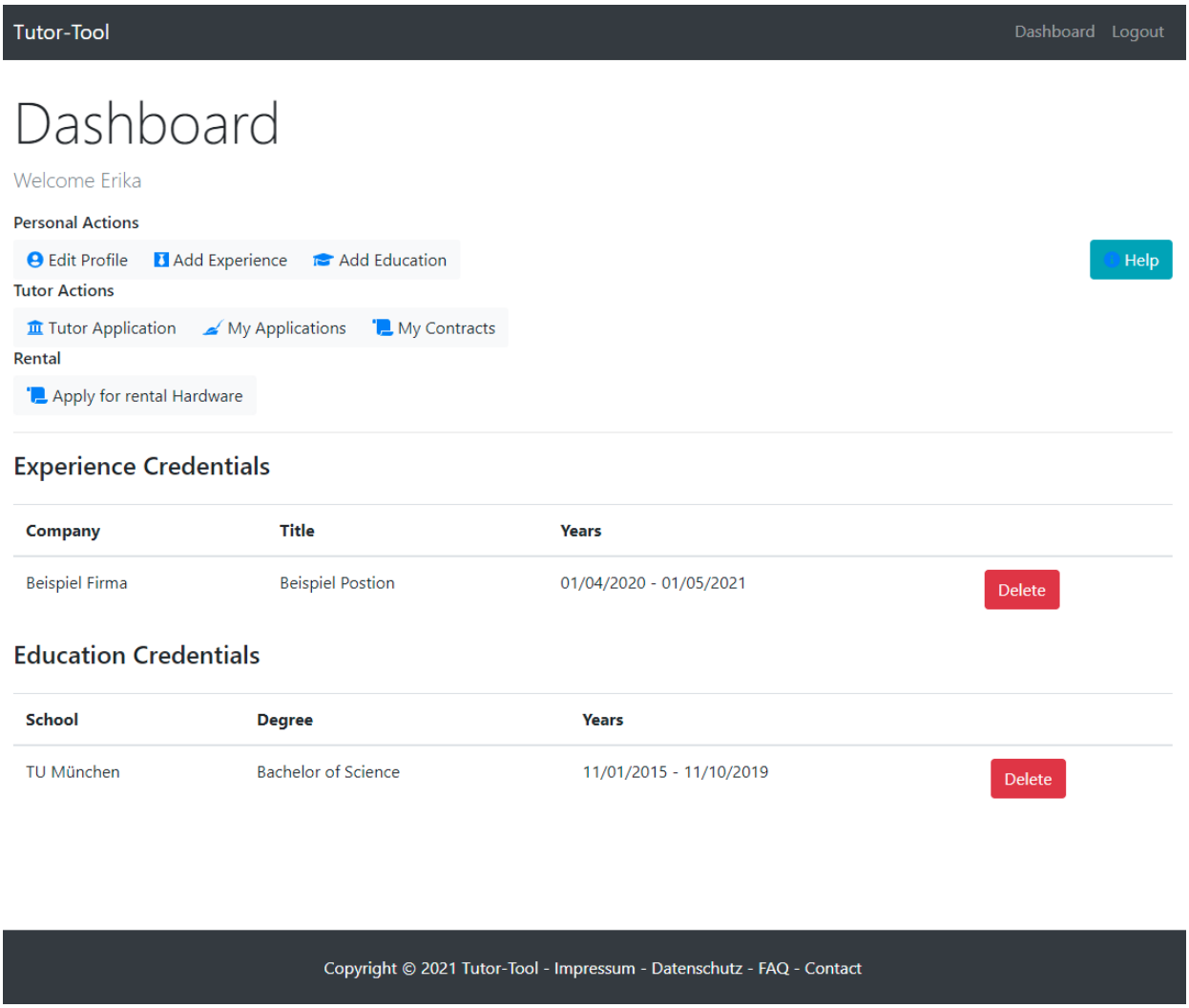


Abbildung 25: Dashboard Studierende/Tutoren
Quelle: eigene Darstellung

Anhang C Inhalt des elektronischen Anhangs

Datei	Dateiname	Inhalt
Gesamtprozess Tutorbetrieb altes Tutortool	Gesamtprozess-Tutorbetrieb-alt.pdf	BPMN 2.0 Diagramm des Gesamtprozesses einer Einstellung von Studierenden als Tutoren unter Verwendung des alten Tutortools.
Gesamtprozess Tutorbetrieb Neukonzeption	Gesamtprozess-Tutorbetrieb-Neukonzeption.pdf	BPMN 2.0 Diagramm des Gesamtprozesses einer Einstellung von Studierenden als Tutoren unter Verwendung der Neukonzeption.
Prozess Leihe Verleihtool	Prozess-Leihe-Verleihtool.pdf	BPMN 2.0 Diagramm des Prozesses einer Leihe unter Verwendung des Verleihtools.
Prozess Leihe Neukonzeption	Prozess-Leihe-Neukonzeption.pdf	BPMN 2.0 Diagramm des Prozesses einer Leihe unter Verwendung der Neukonzeption.
Prozess Veranstaltung anlegen	Prozess-Veranstaltung-anlegen.pdf	BPMN 2.0 Diagramm des Anlegens einer Veranstaltung unter Verwendung des alten Tutortools.
Prozess Übungsleiter anlegen altes Tutortool	Prozess-Übungsleiter-anlegen-alt.pdf	BPMN 2.0 Diagramm des Anlegens einer Veranstaltung unter Verwendung der Neukonzeption.
Prozess Übungsleiter anlegen Neukonzeption	Prozess-Übungsleiter-anlegen-Neukonzeption.pdf	BPMN 2.0 Diagramm des Anlegens eines neuen Übungsleiters unter Verwendung des alten Tutortools.
Prozess Vertrag anlegen altes Tutortool	Prozess-Vertrag-anlegen-alt.pdf	BPMN 2.0 Diagramm des Anlegens eines neuen Übungsleiters unter Verwendung der Neukonzeption.
Prozess Vertrag anlegen Neukonzeption	Prozess-Vertrag-anlegen-Neukonzeption.pdf	BPMN 2.0 Diagramm des Anlegens eines Vertrags unter Verwendung des alten Tutortools.
Prozess neues Formular altes Tutortool	Prozess-neues-Formular-alt.pdf	BPMN 2.0 Diagramm Änderns eines Formulars unter Verwendung der Neukonzeption.
Prozess neues Formular Neukonzeption	Prozess-neues-Formular-Neukonzeption.pdf	BPMN 2.0 Diagramm des Änderns eines Formulars unter Verwendung der Neukonzeption.

Tabelle 12: Inhalt des elektronischen Anhangs
Quelle: eigene Darstellung