Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Computergestützte Unterstützung von unerfahrenen Teams in der agilen Software-Entwicklung

Computer-based Support of Inexperienced Teams in Agile Software Development

Masterarbeit

im Studiengang Informatik

von

Maximilian Vinzenz Nixdorf

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüferin: Dr. Jil Klünder
Betreuerin: Dr. Jil Klünder

Hannover, 12.10.2022

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 12.10.2022

Maximilian Vinzenz Nixdorf

Zusammenfassung

In der agilen Software-Entwicklung spielen viele Faktoren eine Rolle für den Projekterfolg. Unklare und sich häufig ändernde Anforderungen, sowie Fehleinschätzungen im Bereich der Aufwandsschätzung, können Probleme im voranschreitenden Projektverlauf verursachen. Eine mangelhafte Planung zu Beginn eines Projekts kann schwerwiegende Konsequenzen, bis hin zum Fehlschlag, mit sich bringen. Speziell unerfahrene Entwicklerteams werden durch diese Tatsache vor ein großes Problem gestellt: Wie kann ein Projekt von Beginn an effizient gestaltet werden, wenn die dazu benötigte Erfahrung fehlt? Um diesen Problemen entgegenzuwirken, wird in dieser Masterarbeit ein Prototyp zur Unterstützung von unerfahrenen Teams in der agilen Software-Entwicklung vorgestellt.

Der entwickelte Prototyp begleitet Teams ab dem Vorliegen der User-Stories bis zum Ende des Projekts. Er bietet Unterstützung in den Bereichen der Aufwandsschätzung, der Rollenzuweisung von User-Stories zu spezifischen Entwicklerrollen, sowie Unterstützung für das Entwicklerteam und den Product Owner bei der Priorisierung, und Zuweisung zu Iterationen. Die implementierten Verfahren basieren auf aktueller Forschung und können nachgewiesenermaßen dabei helfen, Entwicklerteams allgemein zu unterstützen. Ziel dieser Arbeit ist zu untersuchen, ob auch explizit unerfahrene Entwicklerteams von einer Kombination dieser Verfahren profitieren können.

Mithilfe des vollendeten Prototyps wurde zum Abschluss eine Nutzerstudie mit elf Probanden durchgeführt. Das Ziel lag darin zu prüfen, ob der so entstandene Prototyp einen wahrgenommenen Nutzen bei der Projektplanung liefert. Die Ergebnisse weisen positive Tendenzen dahingehend auf.

Abstract

Computer-aided support of inexperienced teams in agile software development.

In agile software development, many factors influence project success. Unclear and frequently changing requirements, as well as misjudgements in the area of effort estimation, can cause problems as the project progresses. Poor planning at the beginning of a project can have serious consequences, including failure. Especially inexperienced developer teams are confronted with a big problem due to this fact: How can a project be designed efficiently from the beginning if the necessary experience is missing? To counteract these problems, this master thesis presents a prototype for supporting inexperienced teams in agile software development.

The developed prototype accompanies teams from the presentation of the user stories until the end of the project. It provides support in the areas of effort estimation, role assignment of user stories to specific developer roles, as well as support for the developer team and the product owner in prioritization, and assignment to iterations. The implemented methods are based on recent research and have been proven to help support developer teams in general. The goal of this work is to investigate whether explicitly inexperienced developer teams can also benefit from a combination of these procedures.

Finally, with the help of the completed prototype, a user study with eleven participants was conducted. Its goal was to determine whether the resulting prototype delivers a perceived benefit in project planning. The results show positive trends in this regard.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Um in der Industrie nicht von den sich rasant ändernden Anforderungen und Kundenwünschen überfordert zu werden, und gleichzeitig konkurrenzfähig zu bleiben, wird eine flexible Arbeitsweise benötigt [31]. Aus diesem Grund hat die agile Softwareentwicklung in den letzten Jahren immer mehr an Bedeutung gewonnen [31]. Durch die sich ständig anpassenden Anforderungen ergeben sich jedoch auch neue Probleme, die gerade unerfahrene Entwicklerteams vor große Herausforderungen stellen können [6]. Ohne ausreichende Planung, können agil durchgeführte Projekte schnell zum Fehlschlag werden [19].

Neben einer effektiven und effizienten Planung stellen die Anforderungen eine weitere Kernkomponente für erfolgreich durchgeführte Software-Projekte dar. Mehr als 50% aller Projektfehlschläge hängen mit ihnen zusammen [42]. In dieser Arbeit werden die Anforderungen, wie so oft in der agilen Software-Entwicklung, in Form von User-Stories abgebildet [42]. Um den angesprochenen Problemen entgegenzuwirken, wurde ein Prototyp zur Unterstützung in den Planungs- und Durchführungsphasen von agilen Softwareprojekten entwickelt. Der Fokus des Prototyps liegt in der Unterstützung von unerfahrenen Teams in den Bereichen der Aufwandsschätzung, der Rollenzuweisung und der Priorisierung von User-Stories.

1.1 Problemstellung

Fehlende Erfahrung des Entwicklerteams stellt in der agilen Software-Entwicklung ein nicht zu vernachlässigendes Problem dar [38]. Fehler in der initialen Planung eines solchen Projekts können schwerwiegende Folgen für den Projektverlauf haben. Das Entwicklerteam muss in der Lage sein, eine korrekte Abschätzung des Aufwands von User-Stories durchzuführen, um vorhersagen zu können, wie viele User-Stories voraussichtlich im Laufe einer Iteration vollendet werden können. Inkorrekte Schätzungen können den Erfolg eines Projekts stark beeinträchtigen [19]. Mögliche Folgen umfassen

Verzögerungen, eine geringere Softwarequalität, sowie erhöhte Entwicklungskosten [34].

Die Zuweisung von Entwicklern zu passenden User-Stories, sowie die Priorisierung dieser, ist ebenfalls aus zahlreichen Gründen relevant. Sie können zu erhöhter Software-Qualität, einer besser verteilten Arbeitslast unter den Entwicklern, und einem verringerten Gesamtrisiko führen [43].

Für unerfahrene Teams, mit wenig bis keiner vorhandenen Projekterfahrung, können die beschriebenen Tätigkeiten ein schwerwiegendes Problem darstellen.

1.2 Ziel der Arbeit

Im Rahmen dieser Arbeit wurde ein Prototyp entwickelt, der unerfahrenen Teams den Einstieg in die Welt der Software-Entwicklung vereinfachen soll. Zu diesem Zweck wurden, auf Basis der oben beschriebenen Problemstellung, Methoden zur Aufwandsschätzung, der Zuweisung von Entwicklerrollen, und der Priorisierung von User-Stories implementiert. Um identifizieren zu können, in welchem Umfang die implementierten Verfahren unerfahrenen Entwicklerteams Unterstützung liefern, werden die folgenden drei Forschungsfragen beantwortet:

Forschungsfrage 1

Inwiefern können existierende maschinelle Lernverfahren genutzt werden, um den Aufwand von User-Stories akkurat zu schätzen?

Forschungsfrage 2

Wie schneidet der Clustering-Ansatz mit kontextuellen BERT-Embeddings im Vergleich zu existierenden Ansätzen zur Aufwandsschätzung von User-Stories ab?

Forschungsfrage 3

Wie können unerfahrene Teams unterstützt werden, um die Planung von Softwareprojekten zu erleichtern?

1.3 Lösungsansatz

Zur Beantwortung der ersten Forschungsfrage werden die vier verschiedenen Kombinationen von zwei Clusterbildungsverfahren, in Kombination mit zwei Auswertungsstrategien beim Clustering mit kontextuellen

BERT-Embeddings in Kapitel 4 miteinander verglichen, um das Clustering-Verfahren mit den besten Resultaten zu erhalten. Im Anschluss wird das gewählte Verfahren mit einer Mittelwert- beziehungsweise Median-Baseline verglichen.

Lösungsansatz für die zweite Forschungsfrage: Das bereits bei der Beantwortung von *Forschungsfrage 1* gewählte Verfahren zur Clusterbildung in Kombination mit der passenden Auswertungsstrategie wird, analog zu dem Vergleich mit den Baselines, mit den State-of-the-Art Verfahren zur Aufwandsschätzung *Deep-SE* und LHC_{TC}*-SE* verglichen.

Die Beantwortung der dritten Forschungsfrage bringt eine Reihe von Teilfragen mit sich, die beantwortet werden müssen. In dieser Arbeit werden vier unterschiedliche Komponenten implementiert, die von potenziellem Nutzen für unerfahrene Entwicklerteams sein können: die automatisierte Aufwandsschätzung, die interaktive Aufwandsschätzung, die Rollenzuweisung und die Priorisierung von User-Stories. Für jedes dieser Verfahren wird der Grad der wahrgenommenen Unterstützung bei der Planung eines Softwareprojekts im Rahmen einer Nutzerstudie erhoben. Die Resultate dieser Unterfragen sollen Aufschluss darüber geben, wie stark die jeweiligen Verfahren als hilfreich wahrgenommen werden, und somit zur Beantwortung der Forschungsfrage beitragen. Diese Frage wird in Kapitel 7 behandelt.

Um die beschriebenen Lösungsansätze zu realisieren, wurde ein Prototyp implementiert, der unerfahrenen Teams die Planung von Software-Projekten erleichtern soll. Dazu wurden zahlreiche bestehende Ansätze als Grundlage genommen, kombiniert und zum Teil angepasst. Unerfahrenen Teams wird Unterstützung in den folgenden Bereichen angeboten:

- Automatisierte Aufwandsschätzung von User-Stories
- Interaktive Aufwandsschätzung von User-Stories
- Priorisierung von User-Stories
- Rollenzuweisung zu User-Stories

Im Bereich der automatisierten Aufwandsschätzung wurde ein bisher nicht erforschtes Clustering-Verfahren, das auf kontextuellen BERT-Embeddings basiert, durchgeführt und quantitativ ausgewertet. Die interaktive Aufwandsschätzung umfasst die Beantwortung von Fragen bezüglich der Fähigkeiten des Entwicklerteams, mit anschließender Berechnung durch ein Bayes-Netz. Die Rollenzuweisung erfolgt mittels eines neuronalen Netzes zur Zuweisung von Entwicklerrollen anstelle von expliziten Entwicklern. Im Rahmen der Priorisierung findet eine automatisierte Extraktion von Aktivitäten aus User-Stories statt, die im Anschluss zu einem Aktivitätsdiagramm zusammengefügt werden können.

Um den wahrgenommenen Nutzen der genannten Unterstützungsverfahren qualitativ zu analysieren, wurde eine Nutzerstudie mit elf Teilnehmern durchgeführt.

1.4 Ergebnisse der Arbeit

Im Rahmen dieser Arbeit wird ein Prototyp mit grafischer Benutzeroberfläche zur Unterstützung von unerfahrenen Teams in der agilen Softwareentwicklung implementiert. Das in der Software zur automatisierten Aufwandsschätzung genutzte Verfahren ist ein Clustering-Verfahren, welches auf einer Arbeit von Tawosi et al. [44] aufbaut, jedoch BERT-Embeddings zur Clusterbildung und somit zur Aufwandsschätzung nutzt. Im Rahmen einer Vergleichsauswertung liefert das implementierte Verfahren bessere Ergebnisse als zufälliges Raten, jedoch statistisch signifikant schlechtere Ergebnisse als State-of-the-Art Verfahren wie *Deep-SE* [4] und die Baselines.

In der durchgeführten Nutzerstudie mit elf Teilnehmern wurden die wahrgenommene Unterstützung durch die automatisierte Aufwandsschätzung, die interaktive Aufwandsschätzung sowie die Priorisierung der User-Stories als überwiegend positiv aufgenommen. Lediglich die Rollenzuweisung wurde als nicht hilfreich wahrgenommen. Die wahrgenommene Unterstützung durch den Prototyp als Ganzes wurde überwiegend als positiv angegeben. Diese Tendenzen lassen vermuten, dass der Prototyp unerfahrenen Teams bei der Software-Entwicklung helfen kann.

1.5 Struktur der Arbeit

In Kapitel 2 werden die für das Verständnis dieser Arbeit benötigten Grundlagen aus den Bereichen des Software Engingeerings, des maschinellen Lernens und des Natural Language Processings erläutert. Es folgt eine Betrachtung der verwandten Arbeiten aus den Bereichen der Aufwandsschätzung, Priorisierung und Zuweisung von User-Stories in Kapitel 3. Zu Beginn des Hauptteils der Arbeit (Kapitel 4) werden die Designentscheidungen, die Benutzeroberfläche, der Entwicklungsprozess sowie der Prototyp als Ganzes vorgestellt. In Kapitel 5 wird die Planung und Durchführung der Nutzerstudie erläutert. Die Studienergebnisse werden in Kapitel 6 dargestellt. Das Kapitel 7 befasst sich mit der Interpretation und Diskussion der neu implementierten Methoden und der Studienergebnisse. Am Ende dieser Arbeit findet sich eine Zusammenfassung der Ergebnisse in Kapitel 8.

Kapitel 2

Grundlagen

In diesem Kapitel werden die für das Verständnis dieser Arbeit benötigten Grundlagen aus den Bereichen des Software Engineerings (SE), des Maschinellen Lernens (ML), sowie des Natural Language Processings (NLP) erläutert.

2.1 Software Engineering

Da es bei einigen Begriffen aus dem Bereich des Software Engineerings keine bindenden Definitionen gibt, werden einige Aspekte wie User-Stories definiert, um ein einheitliches Verständnis zu erzeugen.

User-Story

Eine *User-Story* stellt eine kurze Beschreibung einer Funktionalität der zu entwickelnden Software dar. Eine User-Story wird als eine textuelle Beschreibung der Form „As a <type of user>, I want <capability> so that <business value>“ [5, S.25] definiert. User-Stories dürfen leicht von dieser Vorlage abweichen - zum Beispiel darf „so that“ durch „to/in order to“ ersetzt werden [5].

Eine beispielhafte User-Story für eine Bank-Software könnte folgendermaßen aussehen: „As a user I would like to log in, so that I can see my current account balance.“ Beim Betrachten dieser User-Story wird schnell klar, dass für die Implementierung viele Aspekte berücksichtigt werden müssen. Dem Nutzer muss unter anderem ein grafisches Benutzerinterface zur Verfügung gestellt werden, es muss eine Datenbank vorhanden sein, und zudem muss die Sicherheit des Kunden und der Bank gewährleistet werden. Um den Aufwand einer solch komplexen User-Story effektiv einschätzen zu können, werden Story-Points eingesetzt.

Story-Points

Story-Points sind ein relatives Mittel zur Aufwandsschätzung und dienen der Feststellung der Größe einer *User-Story*, dabei ist der zugewiesene Wert nur in Relation zu den anderen Werten relevant. Demnach sollte eine User-Story, die einen Aufwand von 2 Story-Points erhält, einen doppelt so großen Implementierungsaufwand besitzen wie eine User-Story mit nur einem Story-Point [5].

Velocity

Die *Velocity* beschreibt die Menge an Story-Points, die ein Entwicklerteam innerhalb einer einzelnen Iteration bearbeiten kann. Nach dem Durchführen einer Iteration lässt sich somit effektiv abschätzen, wie viele Story-Points, bei gegebener Iterationsanzahl, im Laufe des Projekts abgearbeitet werden können. Dies ermöglicht es dem Team eine passende Auswahl an User-Stories pro Iteration auszuwählen, sowie abzuschätzen, welche User-Stories sich im Laufe des Projekts zeitlich nicht implementieren lassen [5].

2.2 Maschinelles Lernen

Die Methoden des maschinellen Lernens dienen als Grundbaustein für einige der in dieser Arbeit genutzten Algorithmen. Es werden zu Beginn Verfahren aus dem Bereich des unüberwachten Lernens (Clustering), und im Anschluss, solche aus dem Bereich des überwachten Lernens (Neuronale Netze) erläutert.

Agglomeratives hierarchisches Clustering

Beim *agglomerativen hierarchischen Clustering* handelt es sich um ein Clustering-Verfahren, bei dem eine Menge von Datenpunkten schrittweise zu größeren Clustern zusammengefügt wird.

Sei $n \in \mathbb{N}$ die Anzahl an Elementen eines Datensatzes. Zu Beginn des Verfahrens sind n Cluster vorhanden. Mithilfe einer Distanzmetrik wird nun n mal die geringste Distanz zwischen zwei Clustern gebildet, wobei die jeweiligen Elemente mit der geringsten Distanz durch einen Fusionierungsalgorithmus zu einem gemeinsamen Cluster zusammengefasst werden, bis ein einzelner Cluster übrig ist. Wurde ein Element einem Cluster zugewiesen, kann es nicht mehr aus diesem entfernt werden. Es kann lediglich zu einem größeren Cluster kombiniert werden [11].

Seien $E, E_m, x_{m,l,k}, \overline{x}_{m,k} \in \mathbb{R}$ und $g, l, m, k \in \mathbb{N}$. Der in dieser Arbeit verwendete Fusionierungsalgorithmus ist das *Ward-Verfahren*, bei dem die Distanz der Cluster durch die von Ward [20] beschriebene Formel 2.1, bestimmt wird. Die aktuell maximale Anzahl an Clustern ist durch g

gegeben, m gibt den aktuell gewählten Cluster an. Die Variable E beschreibt die totale Fehler-Abweichungsquadratsumme (engl. error sum of squares) aller Cluster, und E_m die für den Cluster m. Die Anzahl der Datenpunkte im Cluster wird mit n_m angegeben. Die Zählvariable l gibt den jeweiligen Datenpunkt an. Die Anzahl an Einträgen (Dimensionen) pro Datenpunkt (Vektor) beträgt p_k, wobei k den aktuell betrachteten Eintrag angibt. Die Variable $x_{m,l,k}$ gibt somit den Wert des Datenpunktes des k-ten Eintrags des l-ten Datenpunktes im m-ten Cluster an. Der Median des m-ten Clusters für den k-ten Eintrag wird durch $\overline{x}_{m_k}$ angegeben. Der Ward-Algorithmus hat zum Ziel, den Fehler der Abweichungsquadratsumme E innerhalb der Cluster nach jedem Schritt so gering wie möglich zu halten [11].

$$\begin{aligned} E &= \sum_{m=1}^{g} E_m \text{ mit} \\ E_m &= \sum_{l=1}^{n_m} \sum_{k=1}^{p_k} (x_{m,l,k} - \overline{x}_{m,k})^2 \text{ und} \\ \overline{x}_{m,k} &= \frac{1}{n_m} \sum_{l=1}^{n_m} x_{m,l,k}. \end{aligned} \tag{2.1}$$

Im Anhang A.1 befindet sich eine anschauliche Beispielrechnung für die Bildung des ersten zusammengefügten Clusters.

Dendrogramm

Ein *Dendrogramm* stellt eine grafische Repräsentation, der durch eine hierarchische Clustering-Methode erzeugten Klassifikation, von Datenpunkten dar. Es weist eine ähnliche Struktur wie ein Baumdiagramm auf. Abbildung 2.1 zeigt ein Punktdiagramm mit den eingezeichneten Clustern (oben) und ein zugehöriges Dendrogramm (unten) mit vier Clustern. Aufgrund der Anordnungsmöglichkeiten der Blattknoten gibt es eine Vielzahl an äquivalenten Dendrogrammen für denselben Datensatz [11].

Der Abstand der Datenpunkte wird mithilfe der euklidischen Distanz berechnet. Auf der x-Achse sind die gelabelten Datenpunkte des Punktdiagramms aufgetragen. Die y-Achse stellt die euklidische Distanz zwischen zwei Datenpunkten bzw. Clustern dar. Die vier Cluster sind farblich markiert. Die horizontalen Linien geben den Abstand der durch sie verbundenen Cluster an. Als Abstandsmetrik wird der maximale Abstand der am weitesten entfernten Datenpunkte innerhalb der zu vergleichenden Cluster herangezogen. Die euklidische Distanz zwischen den nur aus (1) und (2) bestehenden Clustern beträgt somit 1. Die Distanz zwischen dem Cluster (7) und dem Cluster $(5, 6, 8)$ ist gleich der Distanz zwischen (7) und (6). Sie entspricht $\sqrt{(6-5)^2 + (7-9)^2} \approx 2.24$.

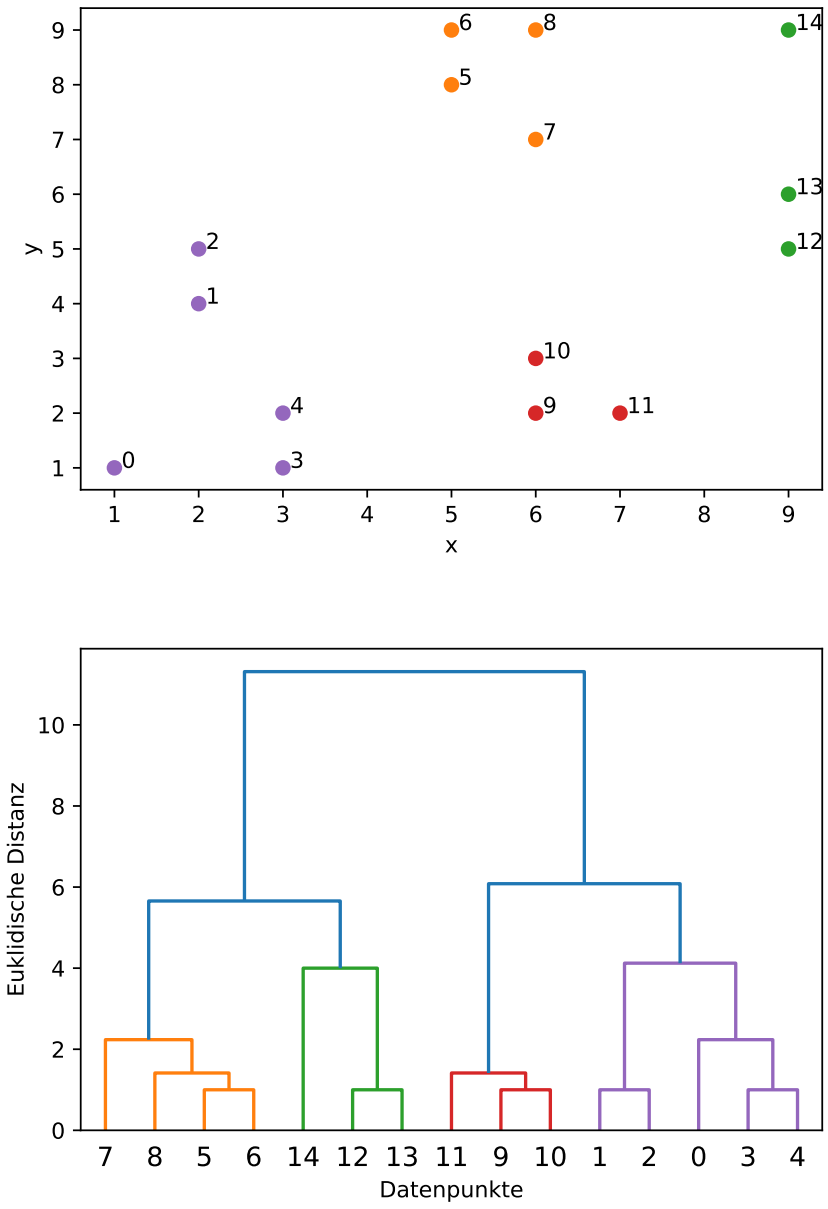


Abbildung 2.1: Beispiel: Punktdiagramm (oben) und mögliches zugehöriges Dendrogramm basierend auf der maximalen Distanz zwischen zwei Clustern (unten).

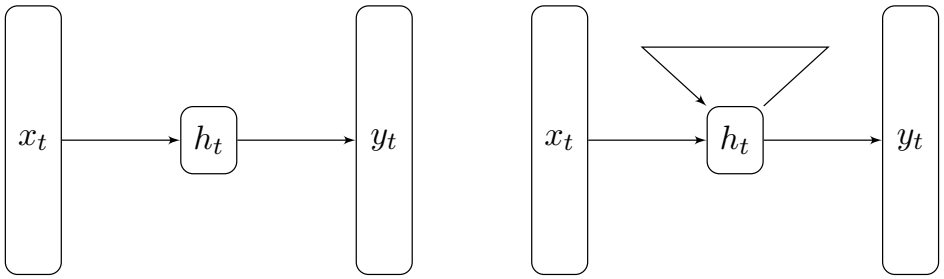


Abbildung 2.2: Schematischer Aufbau eines neuronalen (links), und eines rekurrenten neuronalen Netzes (rechts). (erweitert nach [21])

Rekurrente neuronale Netze

Neben dem Clustering kommen in dieser Arbeit auch spezielle neuronale Netze zum Einsatz. Eine Weiterentwicklung der grundlegenden neuronalen Netze sind die sogenannten *rekurrenten neuronalen Netze* (RNN), die im Bereich des Natural Language Processings für die Klassifizierung sowie für das Labeln von textuellen Datensätzen genutzt werden, da sie in der Lage sind, grundlegende kontextuelle Informationen zu verarbeiten [21]. Der Hauptunterschied zu den grundlegenden neuronalen Netzen liegt darin, dass sie einen Zyklus beinhalten [21].

Sei h_t der Wert der Aktivierungsfunktion für das Hidden-Layer zum Zeitpunkt t, y_t die Ausgabe des Hidden-Layer zum Zeitpunkt t, x_t die Eingabe des Hidden-Layers zum Zeitpunkt t und W die Gewichtsmatrix, die auch in nicht-rekurrenten neuronalen Netzen vorhanden ist und U die Gewichtsmatrix für h. Die Hidden-Layer des Netzes zum Zeitpunkt t erhalten den Ausgabewert des Hidden-Layers des Zeitpunkts $t-1$ bis $t=0$ als zusätzlichen Eingabeparameter [21].

Abbildung 2.2 zeigt den schematischen Aufbau eines neuronalen Netzes und eines rekurrenten neuronalen Netzes im Vergleich. Die Berechnungen von h_t und y_t im RNN erfolgen durch die in Formel 2.2 abgebildeten Gleichungen.

Eine Schwachstelle von RNNs wird beim Trainieren deutlich – verschwindende oder explodierende Gradienten. Des Weiteren sind die Informationen aus den vorangegangenen Hidden-Layern h_{t-u}, die aus weit zurückliegenden Zeitschritten stammen, bei der Verarbeitung im aktuellen Hidden-Layer nicht mehr hilfreich. Zur Lösung dieses Problems wurde eine Erweiterung der RNN-Architektur vorgenommen, die zu den Long Short-Term Memory-Netzen geführt hat [21].

$$\begin{aligned} h_t &= U \cdot h_{t-1} + W \cdot x_t \\ y_t &= V \cdot h_t \end{aligned} \tag{2.2}$$

LSTM

Long Short-Term Memory (LSTM) Netze wurden 1997 von Hochreiter et al. [16] als Lösung für das Problem von verschwindenden, beziehungsweise explodierenden Gradienten beim Trainieren von neuronalen Netzen vorgestellt [16].

Wie bereits erwähnt, basiert ihre Struktur auf der von RNNs. Die Besonderheit von LSTM-Netzen liegt in der Fähigkeit, nicht benötigte kontextuelle Informationen aus vorherigen Zeitschritten zu entfernen und somit nur die wahrscheinlich benötigten Informationen in den Folgezeitschritten zu verwenden. Das bereits vorhandene RNN-Layer wird um ein vorangehendes Kontext-Layer, und um Gatter (engl. Gate), die den Informationskontrollfluss lenken, erweitert. Die innere Struktur einer LSTM Einheit besteht aus drei Gattern:

- Das *Forget-Gate* hat zur Aufgabe, nicht kontextrelevante Daten zu löschen.
- Das *Add-Gate* wählt die Informationen aus, für spätere Nutzung zum aktuellen Kontext hinzugefügt werden.
- Das *Output-Gate* wählt aus, welche Informationen für den aktuellen Hidden-State benötigt werden.

Der große Vorteil der LSTM-Einheiten liegt in der einfachen Substitutionsmöglichkeit in bereits bestehende Architekturen aus dem Bereich des maschinellen Lernens. Die Einheiten können mit nur minimalen Anpassungen, anstelle von grundlegenden Neuronen oder RNNs, in vorhandene Netzstrukturen eingesetzt werden. Abbildung 2.3 zeigt den Vergleich zwischen grundlegendem Neuron (links), RNN (mittig) und LSTM-Einheit (rechts). Je nachdem welche Art von Element ausgetauscht werden soll, werden nur die Kontextinformationen aus den vorherigen Layern c_{t-1}, oder eine Kombination aus diesen mit den Informationen aus den vorherigen Hidden-Layern h_{t-1} benötigt [21].

Ein Nachteil von LSTM-Netzen liegt im noch immer vorhandenen Informationsverlust, der bei der Verkettung von langen rekurrenten Verbindungen entsteht. Zusätzlich lassen sich Berechnungen mit LSTM-Netzen, aufgrund der benötigten Informationen aus allen vorherigen Layern, nicht parallelisieren [21].

2.3 Natural Language Processing

In dieser Arbeit spielen natürlichsprachige Sätze eine wichtige Rolle. Um algorithmisch in der Lage zu sein, semantische Informationen aus diesen Sätzen zu ziehen, werden Darstellungsformen und Methoden aus dem Bereich

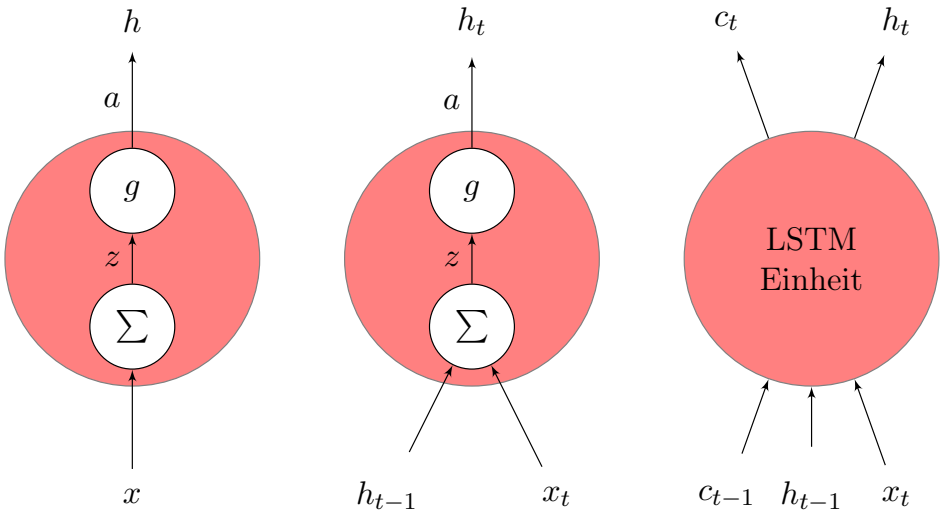


Abbildung 2.3: Vergleich zwischen grundlegendem Neuron (links), einfachem rekurrenten Netz (mittig) und LSTM-Einheit (rechts). Das Σ-Symbol stellt die Übertragungsfunktion und g die Aktivierungsfunktion dar. Die Eingabe x (bzw. x_t) ist der Eingabevektor (zum Zeitpunkt t), h, h_t und h_{t-1} sind die Zustände des Hidden-State zu den Zeitpunkten t bzw. $t-1$. Die Eingabe c_{t-1} und die Ausgabe c_t der LSTM Einheit, beschreiben den Kontextvektor zum jeweiligen Zeitpunkt t. [21]

des Natural Langauge Processings benötigt. Diese werden im Folgenden vorgestellt. Die Erläuterungen in diesem Abschnitt basieren auf dem Buch von Jurafsky et al. [21].

Embedding

Im Kontext des NLP wurde der Begriff *Embedding* zum ersten Mal von Landauer et al. [23] im Jahre 1997 genutzt. Angelehnt an die mathematische Bedeutung, beschrieb ein Embedding eine Zuweisung aus einem mathematischen Raum in einen anderen [21]. Die Bedeutung des Begriffs im Kontext des NLP hat sich jedoch im Laufe der Jahre verändert. Ein Embedding beschreibt einen vollbesetzten Vektor im latenten Raum, nicht die Abbildung, durch den dieser entsteht [21].

Sei $d \in \mathbb{N}$ mit $50 \leq d \leq 1000$ die Anzahl an Dimensionen eines vollbesetzten Vektors V mit Einträgen v_i im Wertebereich von $\mathbb{R}$, mit $i \in \mathbb{N}$ und $i \leq d$. Ein solcher Vektor wird als *Embedding* bezeichnet [21].

Seien s_1 und s_2 die folgenden Sätze:

s_1 : „Sie gingen zum Ball, um zu tanzen."

s_2 : „Sie haben den Ball ins Tor geschossen."

Seien nun V_1 und V_2 Embeddings eines beliebigen Wortes w der Sätze s_1 und s_2. Wird der durch s_i mit $i \in \{1, 2\}$ gegebene Kontext bei der Bildung von V_i berücksichtigt, wird V_i als *kontextuelles Embedding* bezeichnet. Aufgrund dieser Tatsache ist zu beachten, dass das gleiche Wort in beiden Sätzen ein jeweils unterschiedliches Embedding besitzen kann [21]. Das nachfolgende Beispiel soll dies verdeutlichen:
Sei w das Wort „Ball" sowie s_i mit $i \in \{1, 2\}$ die oben genannten Sätze: Betrachtet werden die kontextuellen Embeddings w_{s_i}. Da das Wort „Ball" im jeweiligen Satz s_i eine andere Bedeutung besitzt, gilt $w_{s_1} \neq w_{s_2}$.

Transformer

Bei *Transformern* handelt es sich um eine von Vaswani et al. [47] vorgeschlagene Methode des maschinellen Lernens, die speziell für NLP-Anwendungsgebiete entwickelt wurde [47].

Transformer bilden die Grundlage für komplexere Verfahren wie BERT, welches in Abschnitt 4.3.1 beschrieben wird. Transformer unterscheiden sich grundlegend von RNNs – sie besitzen keine rekurrenten Verbindungen, lassen sich jedoch parallelisieren. Ihre Kernkomponente sind *Transformerblöcke*, die sich aus *Self-Attention Layern*, Linearisierungslayern und vorwärts gerichteten Layern zusammen setzen [21].

Abbildung 2.4 zeigt ein schematisches Self-Attention Layer. Das Layer erhält eine Sequenz von Eingabevektoren $x_1, \cdots, x_5$ und erzeugt eine

Ausgabesequenz $y_1, \cdots, y_5$ gleicher Länge, wobei die Berechnungen der Ausgabevektoren parallel erfolgen kann. Bei der Verarbeitung eines Eingabevektors x_i mit $i \in 1, 2$ greift das Self-Attention Layer auf die Inhalte aller vorherigen Eingabevektoren (inklusive des aktuellen) zu. Dieser Zugriff erfolgt durch die Multiplikation der Eingabevektoren mit unterschiedlichen Gewichtsmatrizen, die diesen eine der folgenden Rollen zuweisen:

- Die *query*-Rolle mit der Gewichtsmatrix W^Q.[1]
- Die *key*-Rolle mit der Gewichtsmatrix W^K.
- Die *value*-Rolle mit der Gewichtsmatrix W^V.

Die parallele Berechnung der Ausgabevektoren ermöglicht das Zusammenfügen der Eingabevektoren zu einer Matrix X, bei der jede Zeile ein Embedding für ein Token der Eingabe darstellt. Durch Nutzung von Formel 2.3 können die Matrizen Q, K und V erzeugt werden, in denen sämtliche *key*-, *query*- und *value*-Vektoren gespeichert werden. Dies ermöglicht die Berechnung der *Self-Attention* nach Formel 2.4, wobei $d_k \in \mathbb{N}$ der Dimension des *key*-Vektors entspricht und K^T die transponierte Matrix K darstellt [21]. Die Formeln 2.3 und 2.4 stammen aus dem Buch von Jurafsky et al. [21].

$$Q = XQ^Q; \quad K = XW^K; \quad V = XW^V \tag{2.3}$$

$$\text{Self-Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.4}$$

Wie bereits erwähnt, bilden die Self-Attention Layer die Grundlage der Transformer-Blöcke. Abbildung 2.5 zeigt den schematischen Aufbau eines Transformer-Blocks für die Erzeugung des Ausgabetokens y_n. Da es sich bei der menschlichen Sprache um ein sehr komplexes Konstrukt handelt, stehen die Worte eines Satzes in der Regel mit mehreren anderen Worten auf unterschiedliche Weise im Zusammenhang. Ein einzelner Transformer-Block ist nicht in der Lage, alle diese Beziehungen zu erkennen und zu verarbeiten. *Multihead Self-Attention Layer* lösen dieses Problem durch die Verwendung von mehreren Self-Attention Layern, den sogenannten *heads*, die simultan die gleichen Eingabetokens mithilfe unterschiedlicher Gewichtsmatrizen verarbeiten, und eine gemeinsame Ausgabe erzeugen. Sei $h \in \mathbb{R}$ die Anzahl an heads und $W^O \in \mathbb{R}^{hd_v \times d}$ eine Matrix, um die konkatenierten Ausgaben von $\text{head}_1, \ldots, \text{head}_h$ mithilfe von Formel 2.5 auf Eingabelänge zu reduzieren, und um somit die *MultiHeadAttention* zu berechnen. Der Aufbau eines Multihead Self-Attention Layers ist in Abbildung 2.6 dargestellt [21].

[1] Die Wahl der hochgestellten Indizes Q, K und V erfolgt aufgrund der weiter unten vorgestellten Multihead Self-Attention Layer. Diese erhalten einen zusätzlichen tiefgestellten Index. Zum Erhalten der Lesbarkeit wurde hier der hochgestellte Index gewählt.

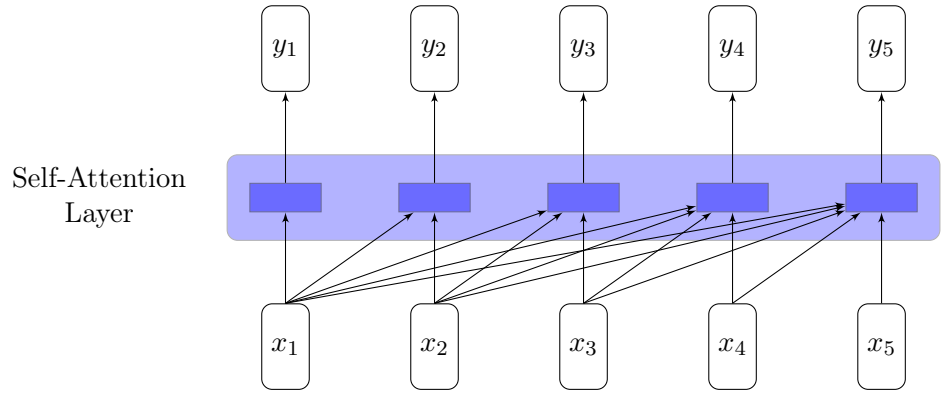


Abbildung 2.4: Informationsfluss eines kausalen Self-Attention Layers. Die Berechnungen innerhalb des Layers finden unabhängig voneinander statt und können parallel ausgeführt werden. [21]

$$\begin{aligned} \text{MultiHeadAttention}(X) &= (\text{head}_1 \oplus \text{head}_2 \oplus \text{head}_h)W^O \\ Q = XW_i^Q; \quad K = KW_i^K; \quad V &= XW_i^V \\ \text{head}_i &= \text{Self-Attention}(Q, K, V) \end{aligned} \tag{2.5}$$

Aufgrund der Tatsache, dass Transformer lediglich vorangehende Informationen in den Kontext eines betrachteten Tokens einbeziehen können, haben sie Probleme bei Lösen komplexeren NLP-Aufgaben. Aus diesem Grund bietet sich die Betrachtung der nachfolgenden Tokens an, was schließlich zur Entwicklung von bidirektionalen Transformern geführt hat [21].

Bidirektionale Transformer

Die *bidirektionalen Transformer* haben regulären Transformern gegenüber den Vorteil, dass sie bei der Verarbeitung eines Tokens in der Lage sind, sämtliche Tokens einer Eingabe zu betrachten. In Abbildung 2.7 ist das Self-Attention Layer eines solchen Transformers abgebildet. Die Abbildungen des Transformerblocks 2.5 und des Multihead Self-Attention Layers 2.6 lassen sich leicht zu Darstellungen für bidirektionale Transformer erweitern. Dazu müssen lediglich sämtliche nachfolgende Tokens der Eingabe $> n$ zur Liste von $x_1, x_2, x_3, \ldots, x_n$ hinzugefügt werden.

Durch das Training bidirektionaler Transformer (unter anderem) mittels *Next Sentence Prediction* auf natürlichsprachigen Datensätzen, lassen sich vortrainierte Sprachmodelle erzeugen, die bei Eingabe eines Strings in natürlicher Sprache, ein kontextuelles Embedding für jedes Token des Strings als Ausgabe liefern. Bei BERT handelt es sich um ein solches vortrainiertes Sprachmodell [21].

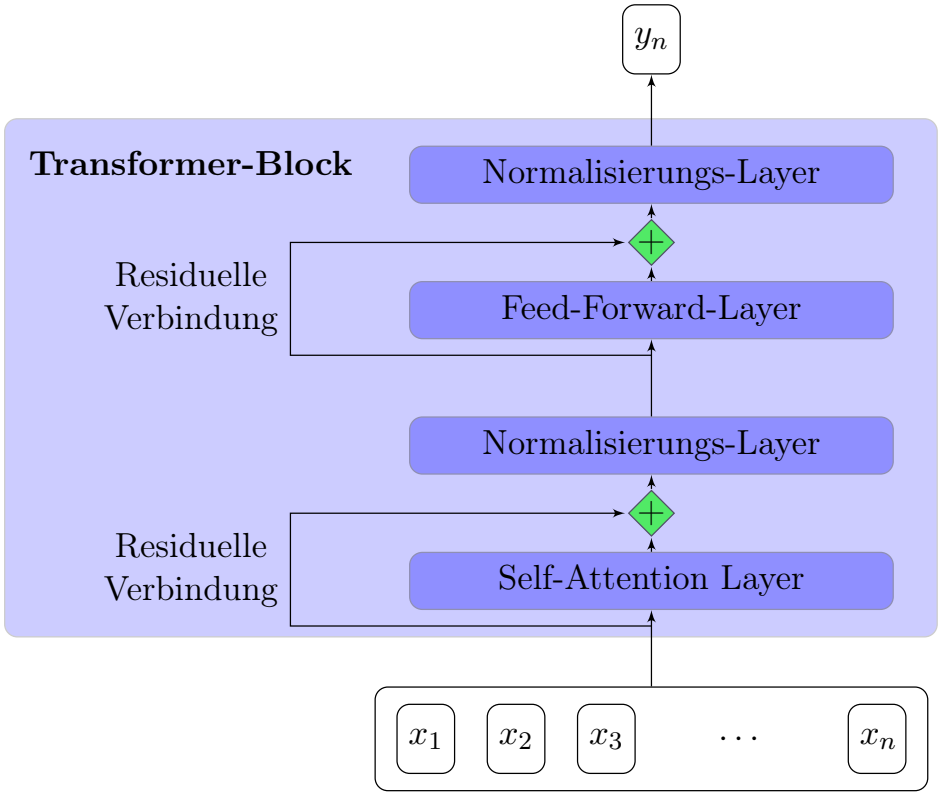


Abbildung 2.5: Schematischer Aufbau eines Transformer-Blocks für das Ausgabetoken y_n. Die +-Symbole führen eine Vektoraddition mit den eingehenden Vektoren aus. Die Normalisierungs-Layer dienen der Performanceverbesserung beim Training. [21]

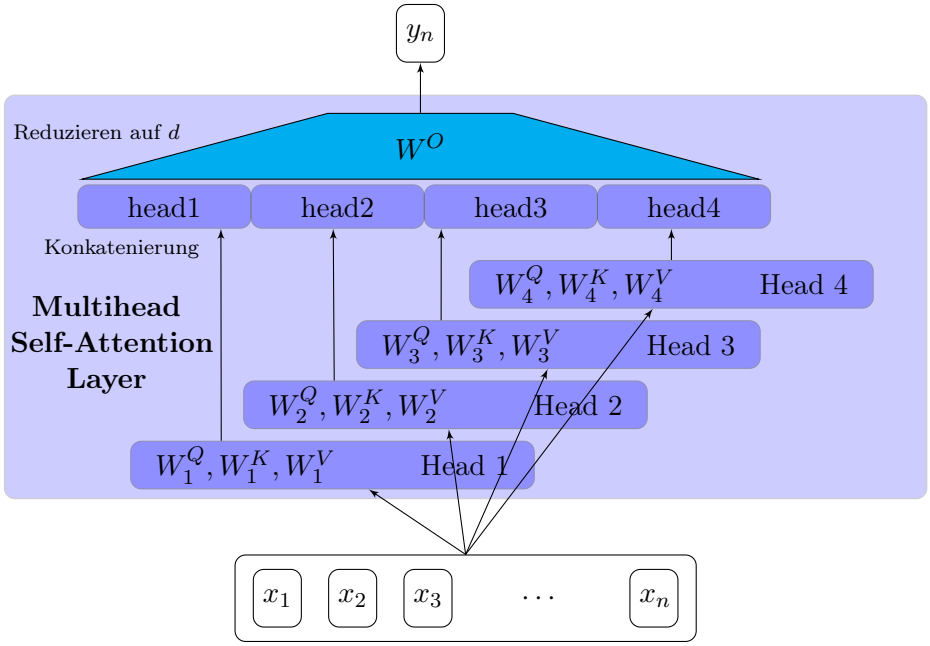


Abbildung 2.6: Schematischer Aufbau eines Multihead Self-Attention Layers zur Berechnung des Ausgabetokens y_n mit vier heads. Die Berechnungen der heads können parallel erfolgen. [21]

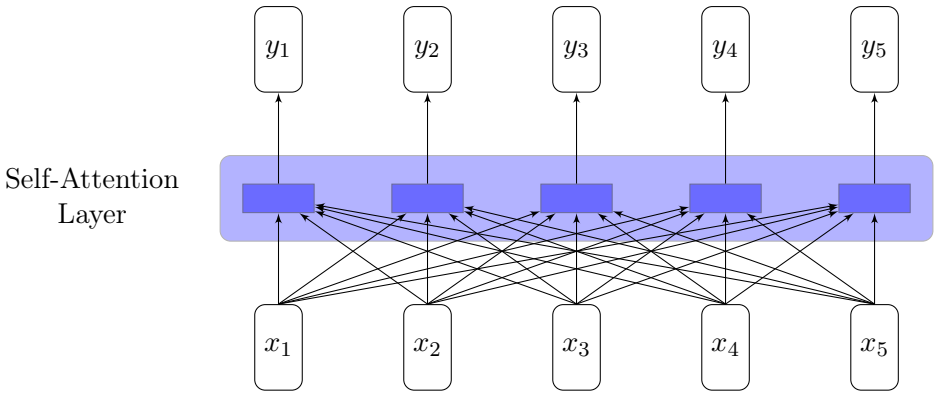


Abbildung 2.7: Informationsfluss eines bidirektionalen Self-Attention Layers. Im Gegensatz zu dem in Abbildung 2.4 dargestellten Layer, können die Berechnungen auch auf die Werte der Folgevektoren zugreifen. [21]

Kapitel 3

Verwandte Arbeiten

Diese Masterarbeit baut auf einigen Vorarbeiten auf, die im Folgenden beschrieben werden. Da der in dieser Arbeit implementierte Software-Prototyp über mehrere voneinander getrennte Komponenten verfügt, behandelt dieses Kapitel die Bereiche (1)Aufwandsschätzung von User-Stories, (2)Priorisierung von User-Stories und (3)Zuweisung von Benutzerrollen zu User-Stories separat. Der Software-Prototyp basiert in den drei genannten Teilpunkten auf jeweils einer der hier vorgestellten Arbeiten, die zum Teil angepasst, erweitert und kombiniert wurden, um unerfahrene Entwicklerteams bei der Projektplanung zu unterstützen. Die Aufwandsschätzung von Aufgaben sowie die Zuweisung von Entwicklern zu Aufgaben in der Software-Entwicklung sind Gebiete, in denen eine Vielzahl an Veröffentlichungen verfasst worden sind. Um die Übersicht gewähren zu können, werden in diesem Kapitel lediglich Veröffentlichungen aus dem Bereich der agilen Software-Entwicklung aufgeführt.

Aufwandsschätzung

Die Aufwandsschätzung von User-Stories in der agilen Software-Entwicklung ist ein Gebiet, indem zahlreiche Veröffentlichungen erschienen sind. Alsaadi et al. [1] haben ein systematisches Literaturreview veröffentlicht, in dem die für die Forschung zur Verfügung stehenden Datensätze und die genutzten Verfahren zur Aufwandsschätzung ermittelt wurden. Die Verfahren umfassen Entscheidungsbäume, Bayes-Netze, SVMs, LSTM-Netze, AutoEncoders, Ensemble-Modelle, Naive-Bayes Modelle, logistische Modelle, Random Forest, Fuzzy Modelle und schrittweise Regressionsmodelle [1].

Choetkiertikul et al. [4] haben ein komplexes Deep Learning Modell (*Deep-SE*) zur automatisierten Schätzung von Story-Points entwickelt, welches lediglich die textuelle Beschreibung einer User-Story als Eingabe benötigt. Die Eingabe wird mittels NLP-Verfahren angepasst und zu einem Embedding verarbeitet. Dieses Embedding wird von einem LSTM-Netz,

gefolgt von einem Reccurent Highway Netz (RHWN) verarbeitet, um mittels Regression einen Story-Point Schätzwert zu erzeugen [4]. Das entworfene Modell wird als eine State-of-the-Art Methode zur Aufwandsschätzung von User-Stories angesehen [44].

De Bortoli Fávero et al. [12] nutzen den von Choetkiertikul et al. [4] vorgestellten Einbettungsansatz als Grundlage zur automatisierten Aufwandsschätzung mittels vortrainierten BERT-Modellen (SE^3M). Nach dem Vorverarbeiten des Datensatzes wird ein Embedding mithilfe eines generischen vortrainierten BERT-Modells erzeugt. Dieses wird im Anschluss an ein auf den Kontext der Software-Entwicklung angepasstes BERT-Modell weitergereicht, welches einem Finetuning unterzogen worden ist. Das Weiterreichen des Embeddings an ein einfaches neuronales Netz liefert einen Story-Point Schätzwert. Dieses Modell bietet im Gegensatz zu dem von Choetkiertikul et al. [4] vorgestellten Verfahren den großen Vorteil, dass die Nutzung von vortrainierten BERT-Modellen tagelange Trainingszeit sparen kann [12].

Tawosi et al. [45] haben in einer Replikationsstudie von *Deep-SE* [4] gezeigt, dass das Verfahren nicht statistisch signifikant besser abschneidet als die zugrunde liegenden Basismethoden. Dementsprechend sollten *Deep-SE* und darauf aufbauende Methoden nicht zur vollständig automatisierten, sondern höchstens als Unterstützung zur Aufwandsschätzung eingesetzt werden [45, 44].

Die Grundlage für die in dieser Arbeit genutzte automatisierte Aufwandsschätzung ist durch ein von Tawosi et al. [44] vorgeschlagenes Clustering-Verfahren gegeben, in dem die Autoren User-Stories auf Basis von Tags clustern. Diese Tags werden mithilfe der Latent Dirichlet Allocation (LDA) aus den User-Stories erzeugt. Das Verfahren liefert vergleichbare Ergebnisse wie *Deep-SE* [4, 44].

Dragicevic et al. [8] haben ein Bayes-Netz (BN) zur Berechnung der Arbeitsstunden für eine User-Story entworfen, dessen Fokus auf den zugrunde liegenden Anforderungen liegt[1]. Das BN hat sechs Attribute, von denen vier zur Anforderungskomplexität kombiniert werden. In Kombination mit den Entwicklerfähigkeiten, und der Frage, ob es sich um einen neuen Aufgabentyp handelt, werden die Arbeitsstunden für eine Anforderung berechnet. Das BN liefert auf allen getesteten Datensätzen eine Genauigkeit von mindestens 90% und wurde zur Zeit der Veröffentlichung aktiv in einem Softwareentwicklungsunternehmen genutzt [8]. Es ist zu beachten, dass die für die effiziente Nutzung des BN geforderten Details bezüglich der Komplexität der Anforderung für unerfahrene Teams in der Regel schwierig zu bestimmen sind [9].

López-Martínez et al. [26] haben ein gewichtetes BN zur Unterstützung

[1] Dies bedeutet, dass zum Beispiel die Fähigkeiten der Entwickler zur Berechnung der Arbeitsstunden nicht beachtet werden.

von Studierenden, basierend auf der Arbeit von Dragicevic et al. [8], entworfen. Das BN setzt den Fokus auf die Dekomposition der Komplexität einer Anforderung, sowie auf die Relevanz einer Anforderung. Die Komplexität setzt sich aus den Attributen Erfahrung, Zeit und Aufwand zusammen. Die Relevanz besteht aus den Attributen Priorität und der Aufgaben-Schätzung. Das BN berechnet den Aufwand einer User-Story anhand von Story-Points basierend auf den Fibonacci-Zahlen mit einer Genauigkeit von 87% [26].

Die beiden BN besitzen zwar eine sehr hohe Genauigkeit in der Aufwandsschätzung, stellen die Fähigkeiten der Entwickler jedoch lediglich als ein einzelnes Attribut dar. Da die Fähigkeiten der Entwickler insbesondere in studentischen Teams divergieren und Kenntnis über diese zum Projekterfolg beitragen kann, sollten diese Fähigkeiten ebenfalls aufgeteilt werden [9]. Zu diesem Zweck haben Durán et al. [9] ein BN entworfen, welches als Grundlage für die in dieser Arbeit verwendete interaktive Aufwandsschätzung dient. Eine genaue Beschreibung des Verfahrens folgt in Kapitel 4.

Priorisierung

Im zweiten Teil der verwandten Arbeiten handelt es sich um Veröffentlichungen, die sich mit der Priorisierung von User-Stories beschäftigen:

Die Autoren Elsood et al. [10] haben eine ziel-basierte Methode zur Priorisierung von User-Stories entwickelt, die jeder User-Story ein relatives Gewicht, basierend auf den vom Product Owner (PO) festgelegten Zielen der zu entwickelten Software, zuweist.

Hudda et al. [18] priorisieren User-Stories mittels einer Kombination aus Gewichtung durch Abstimmung durch die POs, und einer Aufwandsschätzung durch das Entwicklerteam mittels der Program Evaluation Review Technique (PERT)[2]. Eine algorithmische Lösung liefert die Prioritäten der User-Stories [18]. Eine mögliche Fehlerquelle beim Verfolgen dieses Ansatzes liegt in der Abstimmung durch die POs. Die Autoren gehen davon aus, dass eine Vielzahl an POs mit dem Entwicklerteam zusammenarbeiten, was in der Realität nicht der Fall ist.

Hoff et al. [17] haben ein Literaturreview zum Identifizieren der wichtigsten Einflussfaktoren bei der Priorisierung von User-Stories durch Stakeholder durchgeführt. Zu den identifizierten Faktoren zählen die Komplexität, die Anzahl von Abhängigkeiten sowie der Kostenvorteil für das Unternehmen.

Aufbauend auf den oben beschriebenen Einflussfaktoren haben Sachdeva et al. [36] einen Ansatz zur Priorisierung von User-Stories entwickelt. Sie nutzen die *Stanford CoreNLP* Software[3] [28] zur Extraktion von Aktivitäten aus den Titeln von User-Stories. Diese Aktivitäten dienen als

[2]PERT ist eine von der US Navy entwickelte Projektplanungstechnik, welche die inhärente Unsicherheit bei Schätzungen in der Projektplanung berücksichtigt [13].

[3]https://stanfordnlp.github.io/CoreNLP/

Basis zur automatisierten Erstellung eines Aktivitätsdiagramms, welches das Entwicklungsteam zusammen mit dem PO bei der Priorisierung der User-Stories unterstützen soll. Die Autoren berechnen die finale Priorität als eine gewichtete Kombination aus der Priorität des PO und des Entwicklerteams, geben jedoch keine konkrete Berechnungsvorschrift für die Gewichte an [36].

Rollenzuweisung

Der Großteil der Forschung im Bereich der Zuweisung von User-Stories fokussiert sich auf die explizite Zuweisung von Entwicklern zu User-Stories. Bereits 2006 haben Mak et al. [27] die Unterstützungssoftware *NextMove* für Projektleiter im Bereich der verteilten, agilen Softwareentwicklung vorgeschlagen. Die Software nutzt eine Kombination aus der Priorität einer Aufgabe und den benötigten Fähigkeiten der Entwickler, um einen konkreten Vorschlag für den Bearbeiter einer Aufgabe zu generieren [27].

Lin et al. [24] haben die HASE-Software[4] zur Durchführung einer Nutzerstudie mit 119 studentischen Teilnehmern in der agilen Softwareentwicklung entwickelt. In der Studie haben sie festgestellt, dass Entwickler mit einem geringen bis mittleren Vertrauen in die eigenen Fähigkeiten eher dazu neigen, ihr eigenes Können beim Bearbeiten von komplexen Aufgaben zu überschätzen [24].

Aslam et al. [2] haben Einflussfaktoren für die Zuweisung von User-Stories zu Entwicklern identifiziert und ein quantitatives, theoretisches Framework zur Zuweisung von User-Stories in der verteilten agilen Software-Entwicklung entwickelt.

Shafiq et al. [39] nutzten User-Stories aus dem Taiga.io-Repository[5], um die *TaskAllocator* Software zu entwickeln. Beim *TaskAllocator* handelt es sich um ein Jira-Plugin[6], mit dem einer Aufgabe aus dem Softwareentwicklungsprozess eine von sieben Rollen zugewiesen werden kann. Dazu gehören unter anderem Frontend Entwickler, der PO und Stakeholder. Die Klassifikation der vorverarbeiteten Aufgaben findet mittels eines LSTM-Netzes statt [39]. Das Verfahren dient als Grundlage für die in Kapitel 4 beschriebene Zuweisung von User-Stories.

Abgrenzung

Es gibt nur eine Arbeit, die zwei der vorgestellten Bereiche kombiniert. Die bereits vorgestellte *NextMove*-Software erzeugt neben dem Vorschlag eines Bearbeiters einer Aufgabe auch eine Priorität für diese Aufgabe. Diese basiert

[4] http://www.linjun.net.cn/hase/
[5] https://github.com/taigaio
[6] www.atlassian.com/software/jira

auf der Anzahl der zum Abschluss benötigten Teilaufgaben [27]. In dieser Arbeit wird erstmals versucht, alle Bereiche zu kombinieren.

Kapitel 4

Software-Prototyp

Die grundlegenden Ideen für die Planung des Software-Prototyps stammen aus den in Kapitel 3 vorgestellten verwandten Arbeiten. Die Veröffentlichungen von Tawosi et al. [44], Durán et al. [9], Sachdeva et al. [36] und Shafiq et al. [39] dienen als Grundlage für die vier Kernkomponenten der Software.

4.1 Konzept und Struktur

Der entworfene Software-Prototyp soll unerfahrene Entwicklerteams während des Planungsprozesses eines Projekts an die Hand nehmen und sie ab dem Moment, in dem die User-Stories feststehen, bis hin zur Implementierung begleiten. Der Unterstützungsprozess setzt dabei auf die folgenden Kernkomponenten, die sich mit der Verarbeitung von User-Stories befassen:

- **Automatisierte Aufwandsschätzung** von User-Stories durch ein Clustering-Verfahren.
- **Interaktive Aufwandsschätzung** von User-Stories mithilfe eines Bayes-Netzes.
- **Zuweisung von vordefinierten Entwickler-Rollen** (Frontend, Backend, Datenbank, Full Stack) zu einer User-Story.
- **Priorisierung** von User-Stories durch Erzeugung eines UML-Aktivitätsdiagramms und anschließender interaktiver Zuordnung zu Iterationen.

Für jede der Komponenten wird eine Benutzeroberfläche (GUI) bereitgestellt. Die Navigation der GUIs folgt einem größtenteils linearen Ablauf, der sich durch die in Abbildung 4.1 dargestellte Navigationsübersicht verfolgen lässt.

Die Startseite der Software ist die *Projektverwaltung*, die es Nutzern erlaubt, Projekte zu erstellen oder zu löschen. Zusätzlich wird die Navigation

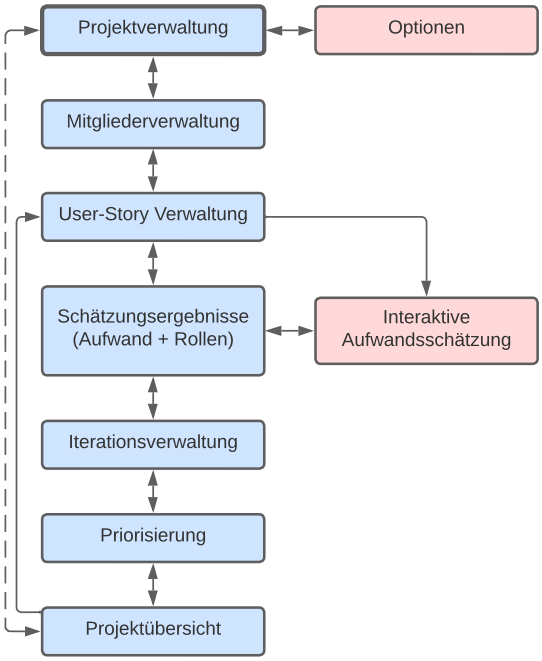


Abbildung 4.1: Navigationsübersicht des Prototyps. Die abgebildeten Elemente stellen die Benutzeroberflächen dar. Bei dem dick umrandeten Element handelt es sich um die Start-Oberfläche. Die rot markierten Elemente sind optionale Oberflächen. Die Verbindungen stellen die Navigationsmöglichkeiten dar, wobei die gepunktete Verbindung zwischen Projektverwaltung und -übersicht angibt, dass diese erst nach einmaligem Durchlaufen der vorherigen Oberflächen freigeschaltet wird.

zu einem *Optionsfenster* ermöglicht, in dem die Schriftgröße der Anwendung angepasst werden kann, um eine bessere Lesbarkeit bei Nutzung in großen Räumlichkeiten zu gewährleisten. Durch Auswahl eines Projekts wird die zugehörige *Mitgliederverwaltung* angezeigt.

Die *Mitgliederverwaltung* ermöglicht das Hinzufügen oder Entfernen von Entwicklern zum Projekt, sowie die Navigation zur *User-Story Verwaltung*.

In der *User-Story Verwaltung* wird das Erstellen beziehungsweise Entfernen von User-Stories unterstützt. Des Weiteren werden Optionen zur automatisierten Aufwandsschätzung und Rollenzuweisung aller User-Stories (Option 1), oder zur interaktiven Schätzung einer einzelnen User-Story (Option 2) angeboten. Der letztere Fall dient der zeitsparenden Schätzung bei nachträglichem Hinzufügen einer User-Story im weiteren Projektverlauf. Wurden die User-Stories mindestens einmalig automatisch geschätzt, wird eine dritte Option zur direkten Navigation der *Schätzungsergebnisoberfläche* angeboten, die sonst nur nach Ausführen von Option 1 oder 2 zu erreichen ist.

Die *Schätzungsergebnisoberfläche* zeigt eine Auflistung der User-Stories mit dem zugehörigen geschätzten Aufwand und der vorgeschlagenen Benutzerrolle an. Wird das Ergebnis der Schätzung als unpassend wahrgenommen, kann eine optionale *interaktive Aufwandsschätzung* erfolgen. Diese *interaktive Schätzung* erfolgt durch das Auswählen von vordefinierten Antwortoptionen bezüglich der Eigenschaften der User-Story sowie den Erfahrungen des Entwicklerteams. Neben dieser weiteren Schätzungsmethode wird eine zusätzliche Option zum Aufteilen einer User-Story angeboten, sofern diese über einen geschätzten Aufwand von 13 Story-Points verfügt. Wurde eine User-Story im vorherigen Interface interaktiv geschätzt, kann hier zudem eine Rollenzuweisung für diese User-Story erfolgen. Des Weiteren wird die Navigation zur *Iterationsverwaltung* angeboten.

In der *Iterationsverwaltung* kann die geplante Anzahl an Iterationen des Projekts und die maximale Anzahl von Story-Points pro Iteration festgelegt werden sowie zur *Priorisierungsoberfläche* navigiert werden.

In der *Priorisierungsoberfläche* ist eine kurze Erläuterung zu UML-Aktivitätsdiagrammen abgebildet, um den Nutzern die folgenden Funktionalitäten zu erleichtern. Das Interface ermöglicht das Öffnen eines separaten Fensters, in dem das eben genannte Diagramm per drag-and-drop erstellt werden kann. Nach erfolgreichem Erstellen wird die Navigation zur *Projektübersicht* freigeschaltet. Zur erleichterten Navigation beim Neustart der Anwendung, erlaubt das Vollenden des Diagramms zudem die Navigation von der *Projektverwaltung* direkt zur *Projektübersicht*.

Die *Projektübersicht* zeigt eine Tabelle mit sämtlichen zu einer User-Story gehörenden Informationen in Kombination mit dem zuvor erzeugten Aktivitätsdiagramm an. Die Oberfläche ermöglicht das Zuweisen von Iterationen und Entwicklern zu User-Stories, sowie die Navigation zurück zur *Projektverwaltung* (Startseite) und zur *User-Story Verwaltung*, um das

Erstellen neuer Stories zu vereinfachen.

4.2 Interfacedesign

Um die Nutzer des Prototyps effizient und effektiv unterstützen zu können, muss eine gute Usability gewährleistet werden. Aus diesem Grund wurde beim Design des Interfaces Wert auf die Einhaltung der acht goldenen Regeln des Interfacedesigns von Shneiderman et al. [41] sowie der Nielsen Heuristik [30] gelegt. Das Interface wird in einem konsistenten Stil dargestellt, Interaktionselemente wie Buttons und Textfelder sind klar als solche erkennbar. Durch das Hinzufügen von Icons zu den Interaktionselementen wird dem Nutzer die Auswirkung einer Aktion verdeutlicht. Ein konkretes Beispiel ist das grüne Plus-Symbol beim Hinzufügen einer User-Story. Sämtliche Benutzeroberflächen, die nicht in diesem Abschnitt abgebildet werden, können im Anhang A.4 eingesehen werden.

Die im vorherigen Abschnitt beschriebene Navigationsreihenfolge wird durch einen grünen, nach rechts gerichteten Pfeil unterstrichen. Dies erleichtert dem Nutzer auch auf Interface-Seiten mit vielen Interaktionselementen die Orientierung. Abbildung 4.2 zeigt die Benutzeroberfläche *Aufwandsschätzungsergebnisse und Rollenzuweisung*, in der ohne diese Unterstützung nicht sofort ersichtlich ist, wie der Nutzer in der logischen Hierarchie der Anwendung voranschreiten kann. Dies ist dadurch bedingt, dass neben der Vorwärtsnavigation die folgenden zusätzlichen Interaktionsmöglichkeiten angeboten werden:

- Die Zurücknavigation.
- Die Auswahl von User-Stories aus der Liste zum Anzeigen des geschätzten Aufwands und der zugewiesenen Rolle.
- Die interaktive Schätzung einer User-Story.
- Die automatisierte Rollenzuweisung für eine User-Story.
- Die Aufteilung einer User-Story.

Sämtliche Interaktionen mit den Benutzeroberflächen wurden ausgiebig getestet, um mögliche Fehler durch unerwartete Nutzerinteraktionen weitestgehend ausschließen zu können. Zudem wird eine konsistente Rücknavigation, hervorgehoben durch einen grünen, nach links gerichteten Pfeil, zur vorherigen Oberfläche bereitgestellt, um eventuelle Fehler auszubessern. Sämtliche Interaktionen – mit Ausnahme des Löschens eines Projekts, eines Mitglieds und einer User-Story – lassen sich rückgängig machen oder erneut ausführen.

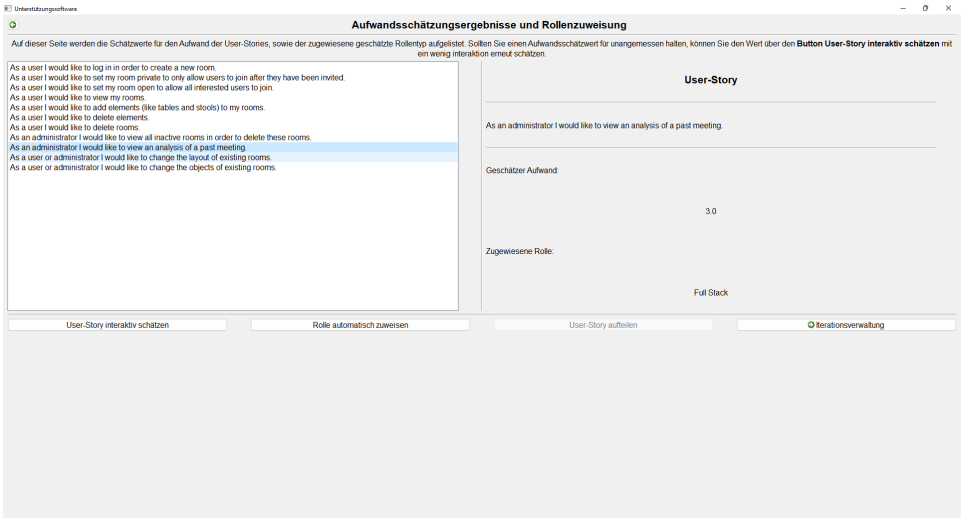


Abbildung 4.2: GUI: Schätzungsergebnisse

4.3 Implementierung und Entwicklung

Aufgrund der Tatsache, dass sich die Anforderungen von agilen Softwareprojekten während der Implementierungszeit durch das Entwicklerteam ändern können, wurde eine CSV-Datenbank zum Speichern und Laden von Projekten, Entwicklern, User-Stories und Iterationen angelegt. Dies ermöglicht zusätzlich die Verwendung der Software auf einem zentralen Gerät, welches nicht nur von einem einzelnen Team genutzt wird.

Das vollständige Datenbankschema ist im Anhang A.2 abgebildet. Bei der Auswahl des gewählten Schemas wurde darauf geachtet, dass es mehrere Projekte geben muss, denen jeweils mehrere Iterationen, User Stories und Teammitglieder zugewiesen werden können. Bei den Teammitgliedern musste zusätzlich sichergestellt werden, dass diese auch an einer Vielzahl von User Stories beteiligt sein können.

Der Software-Prototyp wurde in Python geschrieben. Die Erzeugung der grafischen Benutzerschnittstelle wurde mithilfe der *PyQt5*-Bibliothek[1] und dem in der Bibliothek enthaltenen Tool *QT Designer* erzeugt. Eine Installations- und Ausführungsanleitung für den Software-Prototyp, sowie eine Anleitung zum Verändern der Benutzeroberfläche befinden sich im Anhang A.3.

[1] https://pypi.org/project/PyQt5/

4.3.1 Aufwandsschätzung

Die Nutzung des Prototyps beginnt mit dem Anlegen eines Projekts durch einen Nutzer. Im Anschluss können Mitglieder und User-Stories zu diesem hinzugefügt werden. User-Stories können manuell eingegeben, oder aus einer Textdatei importiert werden. Im Anschluss wird die Aufwandsschätzung durchgeführt. Der automatisierte Teil der Aufwandsschätzung macht sich eine besondere Art von kontextuellen Embeddings zunutze: BERT.

BERT

Bei BERT handelt es sich um ein von Devlin et al. [7] entwickeltes Repräsentationsmodell für Sprachen. Die folgende Beschreibung des Modells und des Trainingsprozesses basiert auf der Arbeit der Autoren. BERT steht für „Bidirectional Encoder Representations from Transformers“. Wie der Name bereits vermuten lässt, nutzt es bidirektionale Transformer zur Erzeugung von kontextuellen Embeddings aus natürlichsprachigen Sätzen. Das Verfahren besteht aus zwei Phasen: einem Vortraining und einem Finetuning. Beim Finetuning handelt es sich um eine einfache Erweiterung des vortrainierten Modells, die es ermöglicht, eine Reihe von spezifischen NLP-Aufgaben, wie etwa das Beantworten von Fragen, effizient auszuführen [7]. Für die Zwecke dieser Arbeit wird das vortrainierte Modell *BERT-Base, Uncased*[2] verwendet, das den Finetuning-Prozess nicht durchlaufen hat. Das Modell verfügt über sieben Layer von Transformern, 768 Hidden-Layern und 12 Self-Attention-Heads [7]. Zur Erzeugung des Embeddings nutzt das BERT-Modell die von Wu et al. [49] vorgeschlagenen WordPiece-Embeddings [7].

Bevor die textuellen User-Stories vom BERT-Modell zu einem kontextuellen Embedding verarbeitet werden können, findet eine **Vorverarbeitung** statt. Die speziellen Tokens [CLS] und [SEP] müssen an den Anfang beziehungsweise an das Ende eines jeden Satzes gesetzt werden. Dies erlaubt es dem BERT-Modell, zwischen Sätzen unterscheiden zu können. Im Anschluss findet eine Vorverarbeitung durch den *BERT-Tokenizer* statt, der Großschreibung und Sonderzeichen, wie Befehle für einen Zeilenumbruch, entfernt, sowie für jedes Wort mindestens ein Token generiert. Die Prozedur wird anhand des folgenden Beispiels veranschaulicht:
Eingabestring:
„[CLS]As a user I would like to log in to the banking software, so that I can see my current account balance.[SEP]“
Tokenizer-Ausgabe:
[[CLS], ’as’, ’a’, ’user’, ’i’, ’would’, ’like’, ’to’, ’log’, ’in’, ’to’, ’the’, ’bank’, ’##ing’, ’soft’, ’so’, ’that’, ’i’, ’can’, ’see’, ’my’, ’current’, ’account’, ’balance’, [SEP]]

[2] https://github.com/google-research/bert

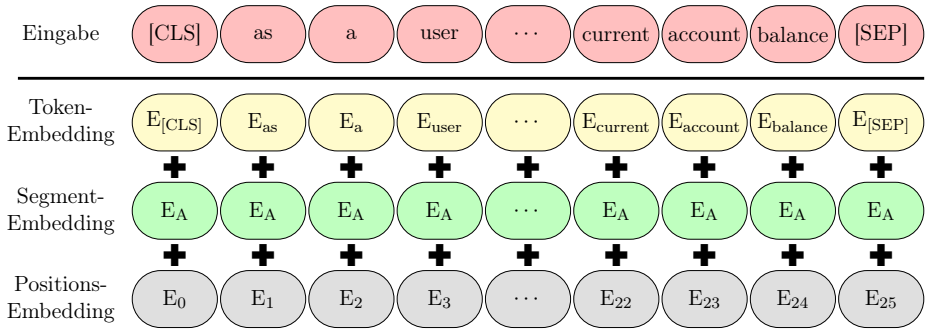


Abbildung 4.3: BERT-Eingaberepräsentation als Kombination des *Token-Embeddings*, des *Segments-Embeddings* und des *Positions-Embeddings*. (angepasst nach [7])

Aus diesen Tokens werden drei Embeddings erzeugt:

- Ein *Token-Embedding*
- Ein *Segment-Embedding* (für die Satzzugehörigkeit des Tokens)
- Ein *Positions-Embedding* (für die Position des Tokens in Eingabesequenz)

Das Eingabe-Embedding ist die Summe dieser drei Teil-Embeddings. Abbildung 4.3 verdeutlicht dies anhand des oben beschriebenen Beispiels. Es ist zu beachten, dass die jeweiligen *Segment-Embeddings* in diesem Fall für alle Tokens identisch sind, da lediglich ein einzelner Satz als Eingabe dient.

Zum **Vortraining** (engl. pre-training) des Modells wird der bereits von Zhu et al. [50] verwendete *Books*-Korpus, sowie der Textkorpus der englischsprachigen *Wikipedia*[3] genutzt [7]. Bei den genutzten Trainingsmethoden handelt es sich um das *Masked Language Model* (MLM) und um die *Next Sentence Prediction* (NSP). Für das MLM werden zufällige Tokens der Eingabe maskiert. Diese Maskierung geschieht in 80% der Fälle durch ein [MASK]-Token, zu 10% über ein zufälliges Token, und in den restlichen 10% bleibt das ursprüngliche Token erhalten. Im Trainingsprozess werden nur die [MASK]-Tokens vorhergesagt. Das Training findet trotz der Bidirektionalität nur jeweils von links nach rechts oder von rechts nach links statt. Dies verhindert, dass die bidirektionalen Transformer ansonsten trivialerweise in der Lage sind, das Token selbst zu betrachten, da sie den gesamten Token-Kontext einsehen können (siehe Kapitel 2). Bei der NSP handelt es sich um einen relativ einfachen Trainingsansatz: Dem Modell werden zwei Sätze

[3] https://en.wikipedia.org/wiki/Main_Page

gleichzeitig übergeben, wobei es sich beim zweiten Satz nur in 50% der Fälle um den korrekten Folgesatz handelt.

Das durch einen solchen Vortrainingsprozess erzeugte Modell ist in der Lage, bei Eingabe eines Satzes, kontextuelle Embeddings für jedes Token zu erzeugen. Diese BERT-Embeddings dienen als Grundlage für das in dieser Arbeit genutzte Verfahren zur automatisierten Aufwandsschätzung.

Automatisierte Aufwandsschätzung

Abbildung 4.4 zeigt den schematischen Prozess der Aufwandsschätzung. Jede durch einen Nutzer eingegebene User-Story wird durch das vortrainierte BERT-Modell *bert-base-uncased* zu einem kontextuellen Embedding verarbeitet. Durch den BERT-Tokenizer werden Tokens aus den Eingabestrings erzeugt. Für jedes dieser Tokens wird ein 768 Einträge langes Embedding erzeugt, welches direkt aus der Ausgabe des BERT-Modells stammt. Zur Erzeugung eines einzelnen Embeddings für eine User-Story, wird der Mittelwert aller Token-Embeddings einer Eingabe gebildet. Im Anschluss an diesen Prozess werden die Embeddings nacheinander an das vortrainierte Clustering-Model übergeben, um den ähnlichsten Cluster zu finden. Der finale geschätzte Wert für den Aufwand der User-Story wird aus den dem Cluster zugehörigen, im Trainingsprozess zugewiesenen User-Stories, gebildet.

Das hier genutzte Clustering-Verfahren basiert auf dem von Tawosi et al. [44] genutzten agglomerativen hierarchischen Clustering, wobei die genutzten Daten nach anderen Kriterien geclustert werden. Die Autoren haben mithilfe der Latent-Dirichlet-Allocation [3] Themen (engl. Topics) aus den User-Stories extrahiert, anhand derer das Clustering durchgeführt worden ist. In der hier genutzten Variante wird basierend auf der Ähnlichkeit der kontextuellen Embeddings geclustert. Zum Training des Modells wurde eine Untermenge der von Tawosi et al. [44] genutzten Datensätze gewählt. Das Kriterium für die Wahl der Datensätze bestand in einem maximalen Aufwand von 13 Story-Points pro User-Story. Zudem musste die Menge der Story-Points auf die Fibonacci-Zahlen begrenzt sein. Tabelle 4.1 zeigt eine Auflistung der zum Training genutzten Datensätze mit einer Gesamtanzahl von 8825 Einträgen. Im Rahmen des Trainingsprozesses werden zwei Verfahren genutzt, um den korrespondierenden Schätzwert eines Clusters zu erhalten – der Mittelwert (engl. mean) und der Median [44].

Für die Evaluierung der Datensätze wurden die von Tawosi et al. [44] und Choetkiertikul et al. [4] genutzten Maße gewählt: der Mean Absolute Error (MAE), der Median Absolute Error ($MdAE$) und die Standard Accuracy (SA). Bei der SA handelt es sich um eine von Shepperd et al. [40] vorgeschlagene Metrik zum Vergleich mehrerer Vorhersagemodelle. Die Formel 4.1 zeigt die genutzten Metriken. Die Funktion Median gibt den Median der Eingabeliste zurück. Die Variablen a_i und p_i beschreiben den

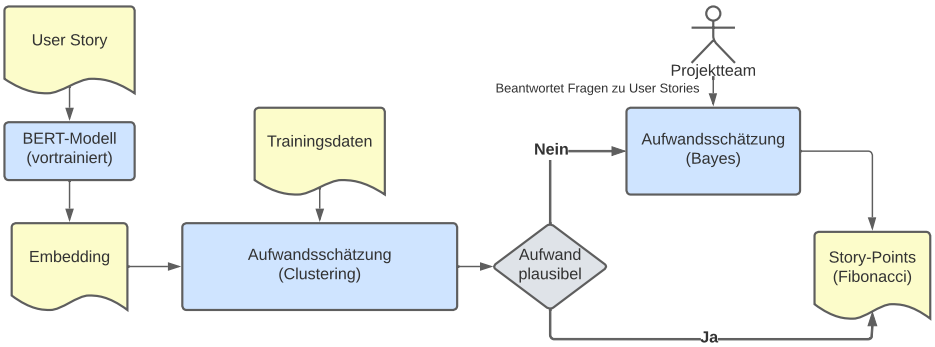


Abbildung 4.4: Makro-Ansicht der Aufwandsschätzung von User-Stories. Die gelben Elemente stellen Objekte, die blauen Elemente Aktivitäten beziehungsweise Funktionen dar. Bei den grauen Elementen handelt es sich um Entscheidungsknoten.

korrespondierenden *tatsächlichen Story-Point-Wert* und den *geschätzten Story-Point-Wert* des i-ten Datenpunkts. Die in der Berechnung von SA genutzte Variable MAE_{p_i} beschreibt den MAE für die jeweilige genutzte Methode zur Clusterbildung beziehungsweise der Clusteranzahl. Der Nenner MAE_{p_0} stellt den MAE für einen Datensatz der Größe n mit $n \approx 1000$ dar, wobei p_i zufällig[4] aus der Menge $\{1, 2, 3, 5, 8, 13\}$ gewählt wird [44]. Ist $SA > 0$, ist das getestete Verfahren besser als zufälliges Raten, für den umgekehrten Fall ist dies nicht der Fall [44].

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |a_i - p_i|$$

$$MdAE = \text{Median}_{i=1}^{n} \{|a_i - p_i|\} \tag{4.1}$$

$$SA = \left(1 - \frac{MAE_{p_i}}{MAE_{p_0}}\right) \cdot 100$$

Sei $k \in \mathbb{N}$ mit $k \geq 3$ die Anzahl an Clustern und $l \in \mathbb{N}$ die Anzahl an Einträgen des korrespondierenden Trainingsdatensatzes. Um die optimale Clustergröße für die unterschiedlichen Datensätze zu finden, wurden sämtliche Trainingsdatensätze mehrfach nach dem in den Grundlagen 2 beschriebenen hierarchischen Clusteringverfahren geclustert. Die zu prüfenden Clustergrößen beginnen mit $k = 3$ und erhöhen sich jeweils um $\frac{l}{10}$, bis die maximale Clustergröße von $0.9 \cdot l$ erreicht wird [44].

[4] In der Implementierung der Software wurden pseudozufällige Zahlen verwendet.

Repository	Projekt	Schlüssel	#Einträge	Story-Points				
				Min	Max	Mean	Median	Std
Apache	Mesos	MESOS	1513	0	13	3.15	3	2.14
Appcelerator	Alloy	ALOY	241	0	13	3.71	3	2.23
	Command-Line Interface	CLI	293	0	13	3.18	3	2.30
	Daemon	DAEMON	205	1	13	5.58	5	3.76
Atlassian	Cofluence Cloud	CONFCLOUD	234	0	13	2.91	2	2.24
	Confluence Server and Data Center	CONFSERVER	456	0	13	3.12	3	1.93
Hyperledger	Fabric	FAB	303	0	13	2.69	2	3.20
MongoDB	Compass	COMPASS	260	1	8	3.55	3	1.85
MuleSoft	Mule	MULE	2935	0	13	3.88	3	3.46
Sonatype	Sonatype's Nexus	NEXUS	1425	0	13	1.70	1	1.82
Talendforge	Talend Data Preparation	TDP	471	0	13	2.31	2	1.84
	Talend ESB	TESB	730	0	13	2.13	2	1.45
Gesamt			8825					

Tabelle 4.1: Beschreibende Statistik der genutzten Datensätze, gruppiert nach den Repositories (angepasst nach [44]).

Die Durchführung sämtlicher in diesem Kapitel genutzten statistischen Auswertungen basiert auf ähnlichen Vergleichsstudien [4, 44, 37]. Für jeden Datensatz wird eine Menge an Werten erzeugt, bestehend aus den zu jeder Clustergröße gehörenden MAE-/$MdAE$-Werten, jeweils mit den zwei genutzten Clusterbildungsverfahren. Tabelle 4.2 zeigt einen Ausschnitt der Ergebnisse des ALOY-Datensatzes. Zur Prüfung, welches Verfahren die statistisch signifikant geringsten Fehlerwerte generiert, werden die Verfahren für jeden Datensatz mithilfe eines One-Tailed *Mann-Whitney U* Tests [29] mit einem Signifikanzniveau von $\alpha = 0.05$ miteinander verglichen, und auf Basis von Gewinn/Verlust/Gleich in Tabelle 4.3 aufgelistet. Es wird von den folgenden Hypothesen ausgegangen:

Nullhypothese H_0

Die absoluten Fehlerwerte durch die Vorhersagen von Modell m_i sind höher als die Vorhersagen des Modells m_j.

Alternativhypothese H_1

Die absoluten Fehlerwerte durch die Vorhersagen von Modell m_i sind kleiner als die Vorhersagen des Modells m_j.

Die Zuweisung z von Gewinn/Verlust/Gleich für das Verfahren m_i erfolgt nach der Fallunterscheidung 4.2. Die Funktion $\text{MWU}(m_i, m_j)$ liefert die Ergebnisse des Mann-Whitney U Tests, und die Funktion $p(\cdot)$ extrahiert den zugehörigen p-Wert.

$$z = \begin{cases} \text{Gewinn} & \text{falls } p(\text{MWU}(m_i, m_j)) < 0.05 \\ \text{Verlust} & \text{falls } p(\text{MWU}(m_i, m_j)) > 0.95 \\ \text{Gleich} & \text{sonst} \end{cases} \tag{4.2}$$

Das beste Verfahren wird auf Basis der Summe sämtlicher Gewinne gewählt. Es ist farblich hervorgehoben. Es handelt sich um die Fehlerberechnung mittels *MdAE*, in Kombination mit dem Mittelwert als Clusterbildungsverfahren mit Gewinn/Verlust/Gleich = [25/3/8]. Beide Verfahren, die den *MAE* als Maß zur Fehlerberechnung nutzen, schneiden vielfach schlechter ab als diese.

Mit dem *MdAE*-Mittelwert als bestem Clusterbildungsverfahren wird nun der Vergleich mit *Deep-SE*, *LHC$_{TC}$-SE*, der Mittelwert- und der Median-Baseline für jeden Datensatz durchgeführt. Für die Baselines wird der Mittelwert beziehungsweise Median der Story-Point-Werte des zu prüfenden Testdatensatzes gebildet. Dieser wird von den jeweiligen echten Story-Point-Werten abgezogen, um den absoluten Fehler für jede User-Story zu berechnen, der als Vergleichsbasis dient [40]. Tabelle 4.4 zeigt einen Vergleich

ALOY-Datensatz

MAE-Mittelwert	*MAE*-Median	*MdAE*-Mittelwert	*MdAE*-Median	#Cluster
2.202	1.480	2.070	2.000	3
2.216	1.480	2.164	2.000	8
2.161	1.460	2.170	2.000	13
2.066	1.480	1.995	2.000	18
2.237	1.540	2.208	2.000	23
2.206	1.640	2.139	2.000	28
2.191	1.620	2.111	2.000	33
2.235	1.920	1.936	2.000	38
2.280	2.020	2.000	2.000	43
⋮	⋮	⋮	⋮	⋮

Tabelle 4.2: Ausschnitt der *MAE*- und *MdAE*-Werte mit dem Mittelwert und dem Median als Clusterbildungsalgorithmus für den ALOY-Datensatz bei ansteigender Clusteranzahl. Das beste Ergebnis basierend auf dem *MdAE*-Mittelwert ist farblich hervorgehoben.

	Gewinn/Verlust/Gleich				
	MAE-Mit.	*MAE*-Med.	*MdAE*-Mit.	*MdAE*-Med.	Summe
MAE-Mit.	—	2/0/10	0/11/1	1/10/1	3/21/12
MAE-Med.	0/2/10	—	0/10/2	1/9/2	1/21/14
MdAE-Mit.	11/0/1	10/0/2	—	4/3/5	**25/3/8**
MdAE-Med.	10/1/1	9/1/2	3/4/5	—	22/6/8

Tabelle 4.3: Vergleich der Ergebnisse des Mann-Whitney U Tests bezüglich sämtlicher Kombinationen der Fehlerbestimmungsmetriken *MAE* und *MdAE* mit den Clusterbildungsverfahren Median (Med.) und Mittelwert (Mit.) für alle Datensätze. Die beste Kombination ist farblich hervorgehoben hinterlegt.

Projekt	Methode	*MAE*	*MdAE*	*SA*	#Cluster	Projekt	Methode	*MAE*	*MdAE*	*SA*	#Cluster
MESOS	BERT	1.48	1.28	63.06	93	ALOY	BERT	1.92	1.93	52.16	38
	LHC_{TC}-SE	**1.33**	**1.00**	34.63			LHC_{TC}-SE	2.28	2.00	9.01	
	Deep-SE	1.34	1.12	34.07			Deep-SE	1.51	**1.28**	39.67	
	Mittelwert	1.37	1.08	32.72			Mittelwert	2.23	2.17	10.84	
	Median	1.34	**1.00**	34.38			Median	**1.44**	2.00	42.53	
CLI	BERT	2.49	1.87	46.46	9	DAEMON	BERT	3.11	2.16	23.41	71
	LHC_{TC}-SE	**1.76**	2.00	33.35			LHC_{TC}-SE	**2.74**	3.00	33.81	
	Deep-SE	**1.76**	**1.30**	33.44			Deep-SE	3.29	**2.00**	20.55	
	Mittelwert	2.14	2.61	18.93			Mittelwert	2.75	2.75	33.53	
	Median	1.77	2.00	33.04			Median	**2.74**	3.00	33.81	
CONFCLOUD	BERT	1.50	0.87	62.51	3	CONFSERVER	BERT	1.20	1.00	67.71	12
	LHC_{TC}-SE	1.37	1.00	39.04			LHC_{TC}-SE	0.96	1.00	49.64	
	Deep-SE	1.48	**0.93**	33.89			Deep-SE	**0.91**	**0.64**	52.28	
	Mittelwert	1.49	1.23	33.65			Mittelwert	1.35	1.45	29.17	
	Median	**1.33**	1.00	40.87			Median	0.96	1.00	49.64	
FAB	BERT	1.85	1.00	54.37	141	COMPASS	BERT	1.57	1.20	61.73	23
	LHC_{TC}-SE	**0.65**	1.00	70.47			LHC_{TC}-SE	**1.30**	**1.00**	32.46	
	Deep-SE	0.86	**0.71**	61.06			Deep-SE	1.63	1.34	15.25	
	Mittelwert	1.19	1.10	46.21			Mittelwert	1.48	1.63	23.05	
	Median	0.67	1.00	69.75			Median	1.38	2.00	28.54	
MULE	BERT	2.42	2.29	39.40	119	NEXUS	BERT	1.10	**0.53**	72.97	3
	LHC_{TC}-SE	2.60	3.00	28.16			LHC_{TC}-SE	1.22	1.00	16.88	
	Deep-SE	**2.24**	**1.68**	37.95			Deep-SE	**1.08**	0.88	26.56	
	Mittelwert	2.79	3.18	22.68			Mittelwert	1.11	0.58	24.69	
	Median	**2.24**	2.00	38.05			Median	1.17	1.00	20.68	
TDP	BERT	**0.98**	0.93	**74.89**	30	TESB	BERT	**0.98**	**0.55**	74.93	31
	LHC_{TC}-SE	1.03	1.00	35.44			LHC_{TC}-SE	1.04	1.00	29.31	
	Deep-SE	0.99	**0.81**	37.69			Deep-SE	1.15	**0.73**	21.36	
	Mittelwert	1.17	1.38	26.26			Mittelwert	0.99	0.99	32.71	
	Median	0.99	1.00	37.74			Median	**0.98**	1.00	33.02	

Tabelle 4.4: Vergleichsdarstellung der durch das Clustering mit BERT erzielten *MAE*-, *MdAE*- und *SA*-Werte mit *Deep-SE*, *LHC$_{TC}$-SE*, Mittelwert und Median. Die besten Werte für *MAE* und *MdAE* sind fett hervorgehoben und grau gefärbt. Im Fall von BERT ist die Hervorhebung und Färbung blau. Die Grundlage für die abgebildete Clustergröße ist der *MdAE*-Wert.

der erzielten Fehlerwerte für die genannten Verfahren. Die Werte für *Deep-SE*, *LHC$_{TC}$-SE*, der Mittelwert- und der Median-Baseline sind aus der Arbeit von Tawosi et al. [44] übernommen worden. Aufgrund der Verwendung von zufälligen Zahlen in der Berechnung des SA-Werts weichen diese für die BERT-Methode von den SA-Werten für die anderen Methoden bei gleichem MAE-Wert leicht ab. Um die Reproduzierbarkeit zu vereinfachen, wird die jeweilige Clusteranzahl für die BERT-Methode angegeben. Da durch einen Vergleich auf Basis dieser Werte noch keine statistisch signifikanten Aussagen möglich sind, wird BERT mit allen Verfahren mithilfe des Mann-Whitney U Tests für ein Signifikanzniveau von $p = 0.05$ miteinander verglichen [44][5]. Um den Vergleich zu ermöglichen, wurden *Deep-SE* und *LHC$_{TC}$-SE* mithilfe der Reproduktions-Repositories ausgeführt, um die benötigte Datengrundlage zu generieren. Mithilfe dieser Daten wurde der Vergleich der Verfahren durchgeführt.

Für den Vergleich gelten die oben beschriebenen Hypothesen H_0 und

[5]Das ursprüngliche Github-Repository der Autoren von *Deep-SE* [4] ist nicht mehr verfügbar, die Daten waren jedoch noch auffindbar und sind auf der beigefügten CD zu finden. Aufgrund der nicht trivialen Nutzung des Verfahrens und fehlender Erläuterungen ist im Anhang A.3 eine Auflistung der benötigten Voraussetzungen abgebildet.

Projekt	BERT-SE verglichen mit			
	LHC_{TC}-SE	*Deep-SE*	Mittelwert	Median
MESOS	0.957	0.998	0.804	0.960
ALOY	0.992	0.997	0.871	0.995
CLI	0.944	0.999	0.934	0.944
DAEMON	0.638	0.389	0.635	0.638
CONFCLOUD	0.621	0.948	0.898	0.534
CONFSERVER	0.999	0.992	0.991	0.999
FAB	1.000	0.997	0.999	1.000
COMPASS	0.858	0.612	0.576	0.858
MULE	0.997	0.973	0.138	1.000
NEXUS	0.427	0.043	0.511	0.427
TDP	0.891	0.425	0.275	0.891
TESB	0.962	0.212	0.972	0.976

Tabelle 4.5: Vergleich von BERT-SE mit *LHC_{TC}-SE*, *Deep-SE*, der Mittelwert-Baseline und der Median-Baseline basierend auf dem Mann-Whitney U Test mit dem Signifikanzniveau $\alpha = 0.05$, in Form von Gewinn/Verlust/Gleich. Bei statistisch signifikanten Werten zeigen rot hinterlegte Zellen höhere *MdAE*-Werte für BERT-SE an (Verlust), grün hinterlegte kleinere Werte (Gewinn). Bei nicht farblich hervorgehobenen Feldern ist kein statistisch signifikanter Unterschied festzustellen. In diesem Fall wird ein Gleich gezählt.

H_1, wobei BERT-SE $= m_i$ gilt, da in allen Fällen BERT-SE mit den anderen Verfahren verglichen wird. Die Zuweisung von Gewinn/Verlust/Gleich erfolgt ebenfalls nach der oben beschriebenen Fallunterscheidung 4.2. Eine Auflistung der berechneten p-Werte findet sich in Tabelle 4.5. Die Ergebnisse der Vergleiche werden in Form von Gewinn/Verlust/Gleich in Tabelle 4.6 dargestellt.

Die Ergebnisse machen deutlich, dass BERT-SE weder besser als die State-of-the-Art Verfahren zur Aufwandsschätzung, noch als die Baselines abschneidet. Es wird lediglich ein statistisch signifikant besserer Wert für den NEXUS-Datensatz im Vergleich mit *Deep-SE* ermittelt.

	Gewinn/Verlust/Gleich				
	LHC_{TC}-SE	*Deep-SE*	Mittelwert	Median	Summe
BERT-SE	0/6/6	1/6/5	0/3/9	0/6/6	1/21/26

Tabelle 4.6: Vergleich von BERT-SE mit *LHC_{TC}-SE*, *Deep-SE*, der Mittelwert-Baseline und der Median-Baseline basierend auf dem Mann-Whitney U Test in Form von Gewinn/Verlust/Gleich.

Interaktive Aufwandsschätzung

Nach dem Durchführen der *automatisierten Aufwandsschätzung* kann bei Bedarf ein Verfahren zur *interaktiven Aufwandsschätzung* durchgeführt werden. Es dient als Fallback-Methode für den Fall, dass die von der *automatisierten Aufwandsschätzung* vorgeschlagenen Werte von den Nutzern als unpassend empfunden werden. Diese Sicherheitsvorkehrung wurde aufgrund der Tatsache getroffen, dass automatisierte Verfahren, die ihre Entscheidungen auf Basis von historischen Daten treffen, nur eine unterstützende Rolle haben sollen, und nicht ohne Supervision durch Nutzer eingesetzt werden sollten [44].

Die *interaktive Aufwandsschätzung* der User-Stories nutzt das von Durán et al. [9] vorgeschlagene Bayes-Netz zur Aufwandsschätzung. Abbildung 4.5 zeigt die Struktur des Netzes, wobei die Knoten ohne eingehende Kanten als First-Level-Variablen bezeichnet werden. Jeder der Knoten des Diagramms stellt eine (bedingte) Wahrscheinlichkeitstabelle dar. Das Bayes-Netz schlüsselt die *Komplexität einer User-Story* (Gesamtkomplexität) in die vier Aspekte *Erfahrung*, *Entwicklerfähigkeiten*, *Projektwissen* und die *technische Komplexität* auf [9]. Diese werden teilweise weiter aufgeschlüsselt, sodass die Gesamtkomplexität durch unterschiedliche Kombinationen der acht zugrunde liegenden Variablen gebildet wird. Die Wahl der Gewichte der First-Level-Variablen erfolgte durch eine von den Autoren durchgeführte Umfrage mithilfe eines Teams von 21 Softwareentwicklungs-Experten. Auf Basis einer 5-stufigen Likert-Skala wurde die Akzeptanz der jeweiligen First-Level-Variablen bezüglich des Einflusses auf die Gesamtkomplexität erhoben [9].

Zur Berechnung der Gewichte zwischen den Wahrscheinlichkeitstabellen werden die von López-Martínez et al. [26] genutzten Formeln verwendet. Sei $V = \{v_1, v_2, \ldots, v_{|V|}\}$ mit $v = (v_w, v_f)$ die Menge der Expertenmeinungen, die durch die Likert-Skalenwerte ermittelt worden sind. Das Gewicht des jeweiligen Likert-Skalenwerts wird durch $v_w \in \{0, 0.25, 0.5, 0.75, 1\}$ und dessen Frequenz mit $v_f \in \mathbb{N}$ beschrieben. Sei $S = \{s_1, s_2, s_3\}$ die Menge der gewichteten Werte für die Zustände der Variablen mit $s_1 = 0.5, s_2 = 0.33$ und $s_3 = 0.17$. Sei $G = \{g_1, g_2, \ldots, g_{|G|}\}$ die Menge an Variablen (Knoten), die Einfluss auf einen Kindknoten haben[6]. Die Berechnung der Gewichte zwi-

[6]Ein Kindknoten ist ein Knoten mit einer eingehenden Verbindung. Der Knoten, von dem die Verbindung ausgeht, wird als Elternknoten bezeichnet

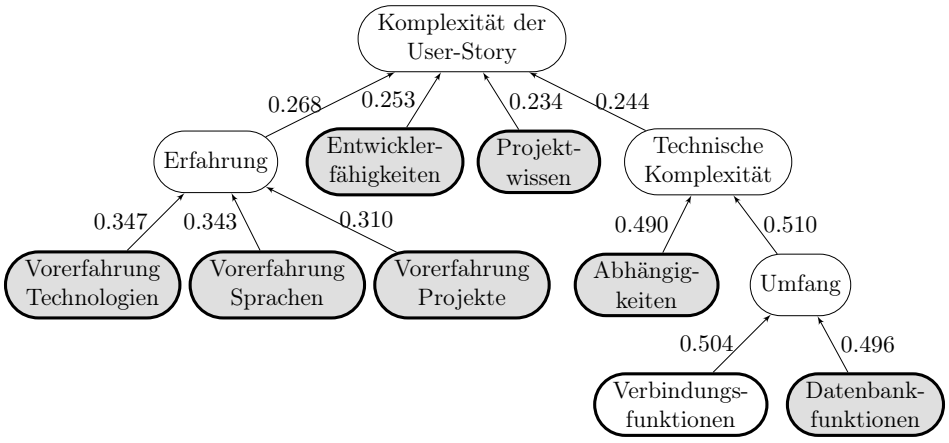


Abbildung 4.5: Struktur des Bayes-Netzes zur interaktiven Aufwandsschätzung. Die First-Level-Variablen sind farblich hervorgehoben. Die Verbindungen stellen die Abhängigkeiten zwischen den Knoten dar. Die Kantenbeschriftungen stellen die Gewichtung dar, mit denen der Wert eines Elternknotens zur Berechnung eines Kindknotens beiträgt. (erweitert nach [9])

schen den Knoten erfolgt mittels Formel 4.3.

$$\underset{g \in G}{G} = \frac{\sum_{v \in V}(v_w \cdot v_f)}{\sum_{v \in V} v_f}$$

$$G_{\underset{g \in G}{norm}} = \frac{G}{\sum_{g \in G} g} \tag{4.3}$$

$$\underset{a \in A}{a} = \frac{\sum_{g \in G} g}{|G|}$$

Zur Veranschaulichung wird das Gewicht des Knotens *Umfang* berechnet: Zur Berechnung werden die Gewichte der Knoten *Verbindungsfunktionen* und *Datenbankfunktionen* benötigt. Dazu wird die Auswahlfrequenz des jeweiligen Likert-Elements benötigt. Die Anzahl der durch die Autoren erhobenen Stimmen bezüglich der Akzeptanz der Knoten sind die Folgenden:

Knoten	Likert-Skala				
	−−	−	○	+	++
Verbindungsfunktionen	0	1	3	9	8
Datenbankfunktionen	0	1	5	6	9

Anhand dieser Werte folgen die Berechnungen der nicht normierten Gewichte

$$\underset{Verb.}{G} = \frac{(0 \cdot 0) + (0.25 \cdot 1) + (0.5 \cdot 3) + (0.75 \cdot 9) + (1 \cdot 8)}{21} \approx 0.786,$$
$$\underset{Dat.}{G} = \frac{(0 \cdot 0) + (0.25 \cdot 1) + (0.5 \cdot 5) + (0.75 \cdot 6) + (1 \cdot 9)}{21} \approx 0.774$$

gefolgt von der Berechnung der normierten Gewichte

$$\underset{Verb._{norm}}{G} = \frac{0.786}{0.786 + 0.774} \approx \mathbf{0.504},$$
$$\underset{Dat._{norm}}{G} = \frac{0.774}{0.786 + 0.774} \approx \mathbf{0.496}.$$

Somit sind die finalen Gewichte für die ersten beiden Knoten berechnet. Mithilfe dieser wird nun das Gewicht und das normierte Gewicht des Knotens *Umfang* berechnet. Da dieser jedoch Einfluss auf den Kindknoten *Technische Komplexität* hat, müssen sämtliche Gewichte der Elternknoten bei der Berechnung betrachtet werden. Somit wird das nicht-normierte Gewicht von *Abhängigkeiten* benötigt. Die Berechnung erfolgt analog zu den oben durchgeführten Berechnungen, das Gewicht ist $\underset{Abh.}{G} = 0.75$.

$$\underset{Umf.}{a} = \frac{0.786 + 0.774}{2} \approx 0.78,$$
$$\underset{Umf._{norm}}{G} = \frac{0.78}{0.78 + 0.75} \approx \mathbf{0.51}.$$

Die Zuweisung von Initialwerten für die First-Level-Variablen erfolgt durch das Beantworten einer Reihe an vorgefertigten Fragen. Die Entwickler (Nutzer des Prototyps) werden nach ihren *Vorerfahrungen bezüglich der genutzten Technologie*, der *Programmiersprachen*, ihren *Vorerfahrungen zu bereits durchgeführten Softwareprojekten*, zum *Projekt im Allgemeinen* sowie zu *Details der User-Story* befragt [9]. Eine Auflistung der Fragen mit der zugehörigen Vorauswahl an Antworten ist in Tabelle 4.7 dargestellt. Die Nutzer haben die Wahl zwischen drei Antwortmöglichkeiten, bestehend aus *Niedrig (0.17)*, *Regulär (0.33)* oder *Hoch (0.5)*. Je nach gewählter Antwort wird der zugehörige Wert als Basiswert für die zur Frage passende First-Level-Variable verwendet. Dieser wird für die Berechnung der verbundenen Kindknoten durch Kombination mit den Kantengewichten genutzt. Die Berechnung des Knotens *Erfahrung* wird beispielhaft im Folgenden erläutert:

Zur Berechnung werden alle Gewichte der Kanten zu den Elternknoten *Vorerfahrung Technologie*, *Vorerfahrung Sprachen* und *Vorerfahrung Projekte* mit den möglichen Werten der gewählten Antworten multipliziert. Tabelle 4.8 zeigt die resultierenden Produkte. Nach der Berechnung dieser Werte folgt die Bildung der konditionellen Wahrscheinlichkeitstabelle für

Vorerfahrung Technologien	
Wie viel Erfahrung haben Sie mit der für die User Story benötigten Technologie (IDE, Hardware, etc.)?	
Niedrig	Ich habe noch nie mit den benötigten Technologien gearbeitet.
Regulär	Ich habe bereits mit den Technologien gearbeitet, neige in der Nutzung aber noch zu Fehlern.
Hoch	Ich habe die benötigten Technologien gemeistert.
Vorerfahrung Sprachen	
Wie viel Erfahrung haben Sie mit den in der User-Story benötigten Programmiersprachen?	
Niedrig	Ich habe die für diese User-Story benötigten Programmiersprachen noch nie verwendet.
Regulär	Ich habe die für die User-Story benötigten Programmiersprachen verwendet, mache aber noch Fehler.
Hoch	Ich habe die benötigte Technologie gemeistert.
Vorerfahrung Projekte	
Wie viel Erfahrung haben Sie in Projekten, in denen ähnlich User-Stories bearbeitet wurden?	
Niedrig	Ich habe noch nie mit den benötigten Technologien gearbeitet.
Regulär	Ich habe bereits mit den Technologien gearbeitet, neige in der Nutzung aber noch zu Fehlern.
Hoch	Ich habe die benötigte Technologie gemeistert.
Entwicklerfähigkeiten	
Wie bewerten Sie Ihre eigenen Fähigkeiten, die User-Story implementieren zu können?	
Niedrig	Ich habe noch nie mit den benötigten Technologien gearbeitet.
Regulär	Ich habe bereits mit den Technologien gearbeitet, neige in der Nutzung aber noch zu Fehlern.
Hoch	Ich habe die benötigte Technologie gemeistert.
Projektwissen	
Wie viel wissen Sie über das zugehörige Projekt?	
Niedrig	Ich habe noch nie mit den benötigten Technologien gearbeitet.
Regulär	Ich habe bereits mit den Technologien gearbeitet, neige in der Nutzung aber noch zu Fehlern.
Hoch	Ich habe die benötigte Technologie gemeistert.
Abhängigkeiten	
Wie viele Abhängigkeiten zu anderen User-Stories gibt es?	
Niedrig	Ich habe noch nie mit den benötigten Technologien gearbeitet.
Regulär	Ich habe bereits mit den Technologien gearbeitet, neige in der Nutzung aber noch zu Fehlern.
Hoch	Ich habe die benötigte Technologie gemeistert.
Verbindungsfunktionen	
Wie komplex ist die Logik die für diese User-Story benötigt wird?	
Niedrig	Ich habe noch nie mit den benötigten Technologien gearbeitet.
Regulär	Ich habe bereits mit den Technologien gearbeitet, neige in der Nutzung aber noch zu Fehlern.
Hoch	Ich habe die benötigte Technologie gemeistert.
Datenbankfunktionen	
Wie kompliziert sind die Datenbankinterfaces, die für diese User-Story benötigt werden?	
Niedrig	Ich habe noch nie mit den benötigten Technologien gearbeitet.
Regulär	Ich habe bereits mit den Technologien gearbeitet, neige in der Nutzung aber noch zu Fehlern.
Hoch	Ich habe die benötigte Technologie gemeistert.

Tabelle 4.7: Fragen und zugehörige Antworten zur Bestimmung der First-Level Attribute. (angepasst nach [9])

Erfahrung, aufbauend aus allen möglichen Kombinationen der First-Level-Variablen sowie der Gewichte. Für jede Kombination wird eine Summe aus den drei Werten gebildet. Nachdem alle Kombinationen berechnet wurden, werden diese durch den maximalen Wert geteilt. Mithilfe der folgenden Matrix wird die beschriebene Berechnung mathematisch dargestellt:

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{pmatrix}$$

Die Matrixeinträge werden folgendermaßen bestimmt:

$$\begin{aligned} w_{11} &= ((g_1 \cdot s_1) + (g_2 \cdot s_1) + \cdots + (g_n \cdot s_1)) \\ w_{12} &= ((g_1 \cdot s_1) + (g_2 \cdot s_1) + \cdots + (g_n \cdot s_2)) \\ w_{1m} &= ((g_1 \cdot s_1) + (g_2 \cdot s_1) + \cdots + (g_n \cdot s_m)) \\ &\cdots \\ w_{n1} &= ((g_1 \cdot s_m) + (g_2 \cdot s_m) + \cdots + (g_n \cdot s_1)) \\ w_{n2} &= ((g_1 \cdot s_m) + (g_2 \cdot s_m) + \cdots + (g_n \cdot s_2)) \\ w_{nm} &= ((g_1 \cdot s_m) + (g_2 \cdot s_m) + \cdots + (g_n \cdot s_m)). \end{aligned}$$

Die normierte Matrix W_{norm} wird mittels

$$W_{norm} = \frac{W}{\max(W)}$$

ermittelt, wobei $\max(W)$ den höchsten Eintrag der Matrix W zurückgibt. Sämtliche so berechneten Kombinationen sind in Tabelle 4.9 abgebildet. Der höchste Additionswert liegt bei 0.502. Die Einträge der so resultierenden normierten Matrix werden in die Spalte *Hoch* für den Knoten *Erfahrung* eingetragen. Für die Werte der Spalte *Niedrig* wird die korrespondierende Gegenwahrscheinlichkeit eingetragen. Zur Berechnung des Kindknotens *Komplexität der User-Story* müssen nun wieder die Produkte mit dem Gewicht gebildet werden, wobei in diesem Fall die Werte für s_1 bis s_{27} durch die Werte der Spalte *Hoch* gegeben sind. Gleiches muss für alle anderen Knoten durchgeführt werden.

	g_1 Vorerfahrung Technologien (0.347)	g_2 Vorerfahrung Sprachen (0.343)	g_3 Vorerfahrung Projekte (0.310)
s_1 Hoch (0.5)	0.174	0.172	0.156
s_2 Regulär (0.33)	0.115	0.113	0.102
s_3 Niedrig (0.17)	0.059	0.058	0.053

Tabelle 4.8: Kombination der gewählten Werte für die First-Level-Variablen S_i *Vorerfahrung Technologien*, *Vorerfahrung Sprachen* und *Vorerfahrung Projekte* mit den zugehörigen Gewichten G_i zur Berechnung der *Erfahrung*.

Kombinationen				Werte $Erfahrung$	
Vorerfahrung Technologien	Vorerfahrung Sprachen	Vorerfahrung Projekte	Addition	Hoch	Niedrig
$g_1 \cdot s_1$	$g_2 \cdot s_1$	$g_3 \cdot s_1$	$0.174 + 0.172 + 0.156 = 0.502$	1.00	0.00
$g_1 \cdot s_1$	$g_2 \cdot s_1$	$g_3 \cdot s_2$	$0.174 + 0.172 + 0.102 = 0.448$	0.89	0.11
$g_1 \cdot s_1$	$g_2 \cdot s_1$	$g_3 \cdot s_3$	$0.174 + 0.172 + 0.053 = 0.399$	0.79	0.21
$g_1 \cdot s_1$	$g_2 \cdot s_2$	$g_3 \cdot s_1$	$0.174 + 0.113 + 0.156 = 0.443$	0.88	0.12
$g_1 \cdot s_1$	$g_2 \cdot s_2$	$g_3 \cdot s_2$	$0.174 + 0.113 + 0.102 = 0.389$	0.77	0.23
$g_1 \cdot s_1$	$g_2 \cdot s_2$	$g_3 \cdot s_3$	$0.174 + 0.113 + 0.053 = 0.340$	0.68	0.32
$g_1 \cdot s_1$	$g_2 \cdot s_3$	$g_3 \cdot s_1$	$0.174 + 0.058 + 0.156 = 0.388$	0.77	0.23
$g_1 \cdot s_1$	$g_2 \cdot s_3$	$g_3 \cdot s_2$	$0.174 + 0.058 + 0.102 = 0.334$	0.67	0.33
$g_1 \cdot s_1$	$g_2 \cdot s_3$	$g_3 \cdot s_3$	$0.174 + 0.058 + 0.053 = 0.285$	0.57	0.43
$g_1 \cdot s_2$	$g_2 \cdot s_1$	$g_3 \cdot s_1$	$0.115 + 0.172 + 0.156 = 0.443$	0.88	0.12
$g_1 \cdot s_2$	$g_2 \cdot s_1$	$g_3 \cdot s_2$	$0.115 + 0.172 + 0.102 = 0.389$	0.77	0.23
$g_1 \cdot s_2$	$g_2 \cdot s_1$	$g_3 \cdot s_3$	$0.115 + 0.172 + 0.053 = 0.340$	0.68	0.32
$g_1 \cdot s_2$	$g_2 \cdot s_2$	$g_3 \cdot s_1$	$0.115 + 0.113 + 0.156 = 0.384$	0.76	0.24
$g_1 \cdot s_2$	$g_2 \cdot s_2$	$g_3 \cdot s_2$	$0.115 + 0.113 + 0.102 = 0.330$	0.66	0.34
$g_1 \cdot s_2$	$g_2 \cdot s_2$	$g_3 \cdot s_3$	$0.115 + 0.113 + 0.053 = 0.281$	0.56	0.44
$g_1 \cdot s_2$	$g_2 \cdot s_3$	$g_3 \cdot s_1$	$0.115 + 0.058 + 0.156 = 0.329$	0.66	0.34
$g_1 \cdot s_2$	$g_2 \cdot s_3$	$g_3 \cdot s_2$	$0.115 + 0.058 + 0.102 = 0.275$	0.55	0.45
$g_1 \cdot s_2$	$g_2 \cdot s_3$	$g_3 \cdot s_3$	$0.115 + 0.058 + 0.053 = 0.226$	0.45	0.55
$g_1 \cdot s_3$	$g_2 \cdot s_1$	$g_3 \cdot s_1$	$0.059 + 0.172 + 0.156 = 0.387$	0.77	0.23
$g_1 \cdot s_3$	$g_2 \cdot s_1$	$g_3 \cdot s_2$	$0.059 + 0.172 + 0.102 = 0.333$	0.66	0.34
$g_1 \cdot s_3$	$g_2 \cdot s_1$	$g_3 \cdot s_3$	$0.059 + 0.172 + 0.053 = 0.284$	0.57	0.43
$g_1 \cdot s_3$	$g_2 \cdot s_2$	$g_3 \cdot s_1$	$0.059 + 0.113 + 0.156 = 0.328$	0.65	0.35
$g_1 \cdot s_3$	$g_2 \cdot s_2$	$g_3 \cdot s_2$	$0.059 + 0.113 + 0.102 = 0.274$	0.55	0.45
$g_1 \cdot s_3$	$g_2 \cdot s_2$	$g_3 \cdot s_3$	$0.059 + 0.113 + 0.053 = 0.225$	0.45	0.55
$g_1 \cdot s_3$	$g_2 \cdot s_3$	$g_3 \cdot s_1$	$0.059 + 0.058 + 0.156 = 0.273$	0.54	0.46
$g_1 \cdot s_3$	$g_2 \cdot s_3$	$g_3 \cdot s_2$	$0.059 + 0.058 + 0.102 = 0.219$	0.44	0.56
$g_1 \cdot s_3$	$g_2 \cdot s_3$	$g_3 \cdot s_3$	$0.059 + 0.058 + 0.053 = 0.170$	0.34	0.66

Tabelle 4.9: Kombination der gewählten Werte für die First-Level-Variablen S_i *Vorerfahrung Technologien*, *Vorerfahrung Sprachen* und *Vorerfahrung Projekte* mit den zugehörigen Gewichten G_i zur Berechnung der *Erfahrung*.

Durch Eingabe sämtlicher First-Level-Variablen kann auf Basis der gezeigten Berechnungen ein individueller Wert für die *Komplexität der User-Story* gebildet werden. Sei Y ein Story-Point-Wert, Y_1 der kleinst mögliche Story-Point-Wert (1), und Y_2 der größtmögliche Wert (13). Der niedrigste mit dem Bayes-Netz zu erreichende Wert (0.34) entspricht X_1, und der höchste (1.0) X_2. Formel 4.4 wandelt das zu den gewählten First-Level-Variablen passende Resultat in Story-Points um, die im Anschluss zu Fibonacci-Zahlen

aufgerundet werden.

$$Y = Y_1 + \left[\left(\frac{X - X_1}{X_2 - X_1} \right) (Y_2 - Y_1) \right] \tag{4.4}$$

4.3.2 Rollenzuweisung

Direkt im Anschluss an die automatisierte Aufwandsschätzung des Prototyps findet die automatisierte Rollenzuweisung statt. Die Zuweisung der User-Stories zu bestimmten Rollen erfolgt auf Basis des von Shafiq et al. [39] vorgeschlagenen *TaskAllocator* Jira-Plugins. Beim *TaskAllocator* handelt es sich um ein Plugin, das die Zuweisung von aufgabenspezifischen Rollen zu den in Jira eingetragenen Issues ermöglicht. Die möglichen Rollen umfassen dementsprechend nicht nur Aufgaben aus dem Bereich der Implementierung von Software, sondern auch vom umgebenden Prozess. Es handelt sich bei den Rollen um *Frontend*, *Backend*, *Full Stack*, *PO*, *Projektmanager*, *Inhalt* und *Stakeholder*. Bei den Aufgaben der Rolle *Projektmanager* handelt es sich um Planungsaufgaben, wie beispielsweise die Terminfestlegung der Teammeetings. Die Rolle *Inhalt* beinhaltet Aufgaben wie das Verfassen von Texten oder das Erzeugen von Abbildungen. Für die Zwecke dieser Arbeit wurde das von den Entwicklern bereitgestellte Python-Skript[7] genutzt und angepasst. Der schematische Ablauf der Rollenzuweisung ist in Abbildung 4.6 zu sehen. Die manuelle Anpassung durch das Projektteam wurde im Anschluss an die in Kapitel 5 beschriebene Studie, aufgrund von Rückmeldungen der Studienteilnehmer, implementiert.

Der *TaskAllocator* erhält als Eingabe eine textuelle User-Story, die von einem LSTM-Netz verarbeitet wird und die Rolle mit der höchsten passenden Wahrscheinlichkeit zurückgibt. Das zugrunde liegende LSTM-Netz ist in Abbildung 4.7 abgebildet. Der Eingabestring wird durch ein Embedding-Layer zu einem kontextuellen Embedding[8] verarbeitet und an ein Spatial Dropout 1D-Layer weitergegeben, bevor es im LSTM-Layer weiterverarbeitet wird. Im Anschluss erfolgt die Klassifikation durch ein Dense-Layer mit der Ausgabegröße $n \in \mathbb{N}$, wobei n der Anzahl der möglichen Rollen entspricht.

Da in dieser Arbeit lediglich die Software-Entwicklung betrachtet wird, wurden die Rollen *PO*, *Projektmanager*, *Inhalt* und *Stakeholder* entfernt. Zusätzlich wurde die neue Rolle *Datenbank* hinzugefügt. In dieser angepassten Form kann das von Shafiq et al. [39] genutzte vortrainierte Modell nicht mehr verwendet werden. Bevor mit dem Training des angepassten Modells begonnen werden kann, musste der zugrunde liegende Datensatz aus dem *Taiga.io*[9] Repository durch folgenden Schritte angepasst werden:

[7] https://github.com/jku-isse/TaskAllocator
[8] Es handelt sich nicht um ein BERT-Embedding.
[9] https://taiga.io/

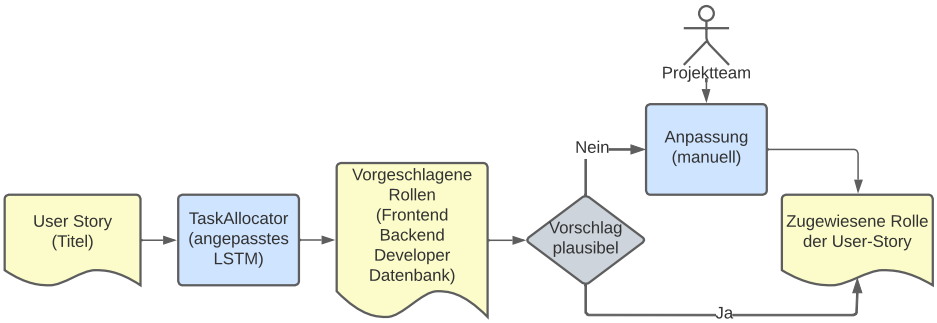


Abbildung 4.6: Makro-Ansicht der Zuweisung von Rollen zu User-Stories. Die gelben Elemente stellen Objekte, die blauen Elemente Aktivitäten beziehungsweise Funktionen dar. Bei den grauen Elementen handelt es sich um Entscheidungsknoten.

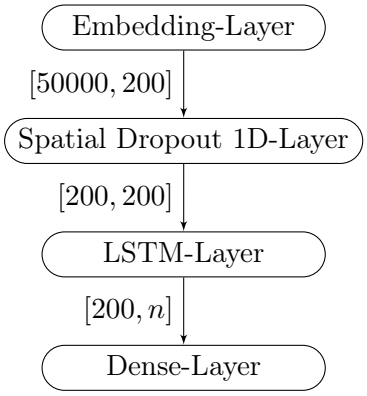


Abbildung 4.7: Darstellung des zur Klassifikation genutzten Netzes. Die Tupel neben den Verbindungen geben die Eingabe- beziehungsweise Ausgabegröße des jeweiligen Layers an. Die Größe der Ausgabe des Dense Layers entspricht der Anzahl der zu klassifizierenden Rollen. (angepasst nach [39])

1. Die ursprünglich genutzten Trainings-, Test- und Validierungsdatensätze werden kombiniert.

2. Die Rollen *Content*, *PO*, *Stakeholder* und *Team-Katalyst* werden aus den Datensätzen entfernt.

3. Manuelles Re-Labeling sämtlicher Einträge mit den Labels *Entwickler* und *Backend-Entwickler*, um die beiden Labels zusätzlich in *Datenbank* zu unterteilen.

Das durchgeführte Labeling wurde durch Mithilfe von zwei Kommilitonen durchgeführt. Das resultierende Label wurde durch einen einfachen Mehrheitsentscheid festgelegt. Bei drei unterschiedlichen Labels wurde das Label nach einer Diskussion gewählt. Die Größe und Aufteilung der Datensätze (links), sowie die Verteilung des resultierenden Gesamtdatensatzes (rechts) ist in Abbildung 4.8 dargestellt. Es ist zu beachten, dass die Anzahl an *Datenbank* Labels in den Trainingsdaten unterrepräsentiert ist, was zu einem Problem bei der Erzeugung von ausreichenden Features führen kann [15].

Beim Training des LSTM-Netzes mit den neu gelabelten Datensätzen wurden zusätzlich zu jeder Trainingsepoche die Validierungs-Genauigkeit und der Validierungs-Verlust bestimmt. Um auf Basis des Verlusts den Trainingsprozess frühzeitig beenden zu können, sollte ein Overfitting auf den Trainingsdaten stattfinden [32]. Die Epochenanzahl liegt bei 15. Das Training wird nach drei konsekutiven Epochen, in denen der Validierungs-Verlust zunimmt, frühzeitig beendet. Das so erzeugte Model liefert eine Genauigkeit auf dem Testdatenset von 64.38%. Die angegebene Genauigkeit ist der jeweilige Mittelwert der erzielten Genauigkeit von fünf Trainingsdurchläufen. Diese rangierten im Intervall von 60.00% bis 69.99%. Diese Abweichungen lassen sich durch die kleine Datenmenge in Kombination mit der unausgeglichenen Verteilung der Klassen erklären. Das angepasste neuronale Netz liefert eine $\sim$ 5 % geringere Genauigkeit als das ursprüngliche, von Shafiq et al. [39] vorgeschlagene Netz (69.30%). Die Fehler, die bei der Klassifizierung vom Netz gemacht werden, können durch die Konfusionsmatrix in Abbildung 4.9 betrachtet werden. Es ist zu erkennen, dass *Full Stack* am häufigsten von allen Klassen vorhergesagt wird. Die Label *Datenbank*, *Backend* und *Frontend* werden in vier, fünf und zwei Fällen als *Full Stack* fehl klassifiziert. Das Label *Frontend* wird mit insgesamt zwei falschen Vorhersagen am wenigsten oft falsch eingeschätzt.

4.3.3 Priorisierung

Im Anschluss an die Aufwandsschätzung und die Rollenzuweisung findet die interaktive Priorisierung statt. Sie basiert auf dem von Sachdeva et al. [36] vorgeschlagenen Verfahren. Die Autoren nutzen die von Manning et al. [28] veröffentlichte *Stanford CoreNLP*-Bibliothek zur Extraktion von Aktivitäten

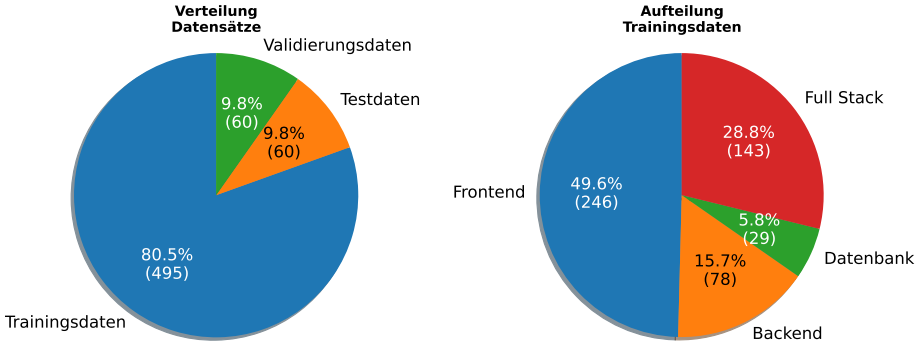


Abbildung 4.8: Größe der Trainings-, Test- und Validierungsdatensätze (links) sowie die Rollenverteilung des Trainingsdatensatzes (rechts).

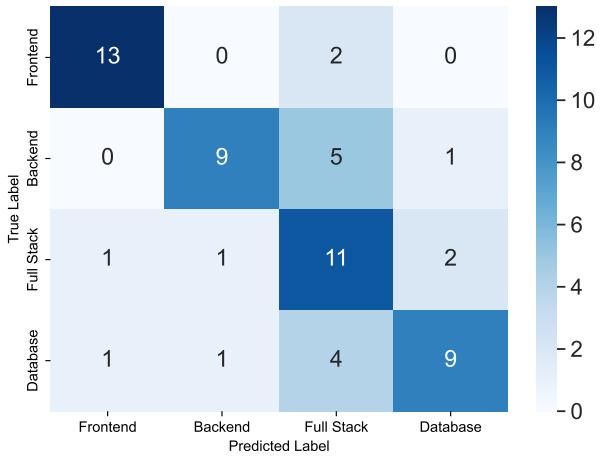


Abbildung 4.9: Konfusionsmatrix der Vorhersagen auf dem Testset mit $n = 60$, wobei $n_{FE} = n_{BE} = n_{FS} = n_{DB} = 15$.

und Akteuren aus textuellen User-Stories. Zur Extraktion werden die User-Stories in Tokens aufgeteilt. Diesen Tokens werden unter Zuhilfenahme von *Part-of-Speech-Tagging* (POS) sogenannte *Tags* zugewiesen. Ein *POS-Tag* enthält Informationen über die Wortart des jeweiligen Wortes, wie zum Beispiel *NOMEN* oder *VERB* [28]. Mittels dieser Tags, sowie der Erkennung von kurzen Phrasen und Markierungen wie „*as a*“ oder „*if*“, werden Aktivitäten gebildet. Im Anschluss an die Extraktion werden diese in einer Tabelle aufgelistet, um daraus ein UML-Aktivitätsdiagramm zu erzeugen.

In dieser Arbeit wird das beschriebene Verfahren in leicht abgeänderter Form implementiert. Abbildung 4.10 stellt den schematischen Ablauf dar. Anstelle der nur in Java verfügbaren *Stanford CoreNLP*-Bibliothek wird die von Qi et al. [33] entwickelte Python-Bibliothek *Stanza*[10] verwendet. *Stanza* ist eine NLP-Bibliothek, die zusätzlich zu NLP-Methoden des maschinellen Lernens, auch über ein Interface zur *Stanford CoreNLP*-Bibliothek verfügt, um webbasiert auf dessen Funktionalität zugreifen zu können [33].

Auf die Extraktion der Akteure wird verzichtet. Sollten mehrere User-Stories mit ähnlichen Aktivitäten vorhanden sein, werden diese zusammengefasst, wie im folgenden Beispiel dargestellt:

1. „As a user I would like to log in, in order to view my data.“

2. „As an administrator I would like to log in, in order to change my password.“

Aus den beiden gegebenen Sätzen werden die Aktivitäten *log in*, *view data* und *change password* extrahiert. Die Aktivität *log in* wird trotz mehrfachen Vorkommens, nur ein einziges Mal extrahiert.

Die beschriebene Extraktion erfolgt mithilfe eines regelbasierten Systems. Da sämtliche User-Stories nach demselben Schema verfasst wurden, können die ersten sieben Wörter vernachlässigt werden. Die Kombinationen aus POS-Tags, die zur Erzeugung einer Aktivität führen, sind in Tabelle 4.10 aufgelistet. Nach der Extraktion der Aktivitäten können diese interaktiv zur Erstellung eines Aktivitätsdiagramms genutzt werden. Die Autoren beschreiben in ihrer Veröffentlichung [36] kein genaues Verfahren, wie sie das Aktivitätsdiagramm bilden. Gulia et al. [14] haben ein Verfahren zur automatisierten Erzeugung eines UML-Aktivitätsdiagramms aus natürlichsprachigen Spezifikationen entwickelt, das jedoch keine nachträglichen Änderungen zulässt. Aus diesem Grund wurde in dieser Masterarbeit eine interaktive Variante konzeptioniert, mit der nachträgliche Änderungen und Anpassungen im Projektverlauf durchgeführt werden können. Abbildung 4.11 zeigt die zugehörige Benutzeroberfläche. Die Implementierung dieser interaktiven Anwendung ist vom eher statischen *PyQt5*-Interface

[10] https://stanfordnlp.github.io/stanza/

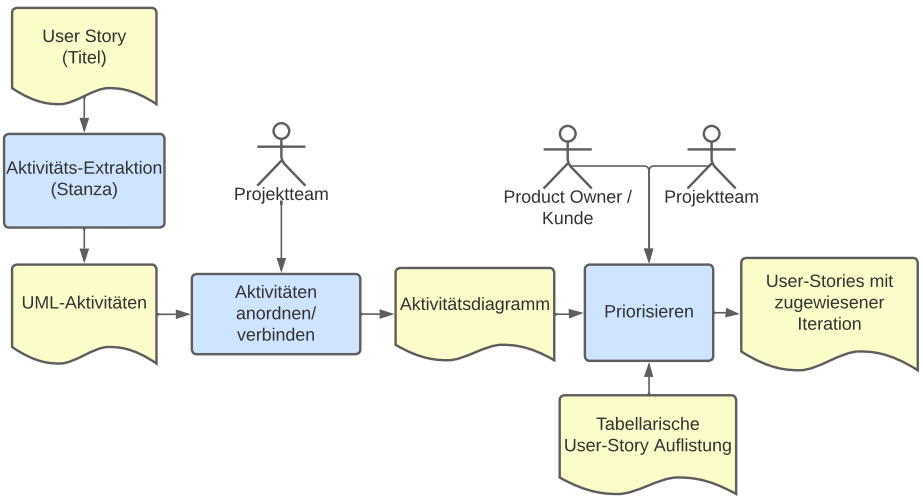


Abbildung 4.10: Makro-Ansicht der Priorisierung von User-Stories. Die gelben Elemente stellen Objekte, die blauen Elemente Aktivitäten beziehungsweise Funktionen dar.

losgelöst, um die benötigten Interaktionsmöglichkeiten zu erlauben. Die Oberfläche wurde mithilfe der *Pygame*[11]-Bibliothek erstellt. Auf ihr werden sämtliche extrahierten Aktivitäten angezeigt, mit der Option, unpassende zu entfernen, oder neue hinzuzufügen. Die Aktivitäten können in einem Grid angeordnet, miteinander verbunden, und mit den UML-Symbolen für *Start*, *Ende*, *Fallunterscheidung* und *parallele Ausführung* ergänzt werden. Zur Ergänzung der *Fallunterscheidung* können Konditionen an den jeweiligen Kanten ergänzt werden.

Es gibt zwar bereits viele Software-Lösungen, die das Erstellen von UML-Aktivitätsdiagrammen ermöglichen, die Besonderheit an dieser Implementation ist jedoch die Anbindung an die Aktivitätsextraktion. Somit ersparen sich die Nutzer die zeitaufwändige manuelle Übertragung in ein Drittprogramm, und sie können das erzeugte Diagramm nach Fertigstellung direkt zur Priorisierung der User-Stories nutzen.

4.3.4 Kombination der Komponenten

Die kombinierten Resultate der Komponenten des Prototyps münden schließlich in der *Projektübersicht*, die in Abbildung 4.12 dargestellt wird. Das Entwicklerteam erhält einen Überblick über sämtliche zuvor erstellten User-Stories, inklusive des geschätzten Aufwands und der vorgeschlagenen Entwicklerrolle. Anhand dieser Vorschläge kann einer User-Story nun ein

[11] https://www.pygame.org/news

Regel und Beispiel
VERB + ADP „As a student I would like to ***log in***, in order to view my account.“
$\text{VERB}_{\text{head}_{\text{NOUN}}}$ + [. . .] + NOUN „As a student I would like to log in, in order to ***view*** my ***account***.“
$\text{VERB}_{\text{head}_{\text{NOUN}}}$ + [. . .] + ADJ + NOUN „As a student I would like to log in, in order to ***create*** a ***big room***.“
$\text{VERB}_{\text{head}_{\text{NOUN}}}$ + [. . .] + NOUN + ADJ „As a student I would like to log in, to ***set*** a ***room private***.“

Tabelle 4.10: Regeln zur Extraktion von Aktivitäten aus User-Stories, basierend auf den POS-Tags und dem syntaktischen Kopf des jeweiligen Wortes. Ein NOUN steht für ein Nomen, ADP für eine Adposition und VERB für ein Verb. $\text{A}_{\text{head}_{\text{B}}}$ gibt an, dass es sich bei A um den syntaktischen Kopf des Wortes B handelt. Zwischen den Tags liegende Symbole werden mit [. . .] dargestellt.

Teammitglied zugewiesen werden, das die Anforderungen der User-Story am besten erfüllt. In Kombination mit dem erstellten Aktivitätsdiagramm kann das Team in Zusammenarbeit mit dem PO die User-Stories zu Iterationen zuweisen. Das Aktivitätsdiagramm hilft beiden Parteien dabei, die benötigten Abhängigkeiten besser überblicken und verstehen zu können [17].

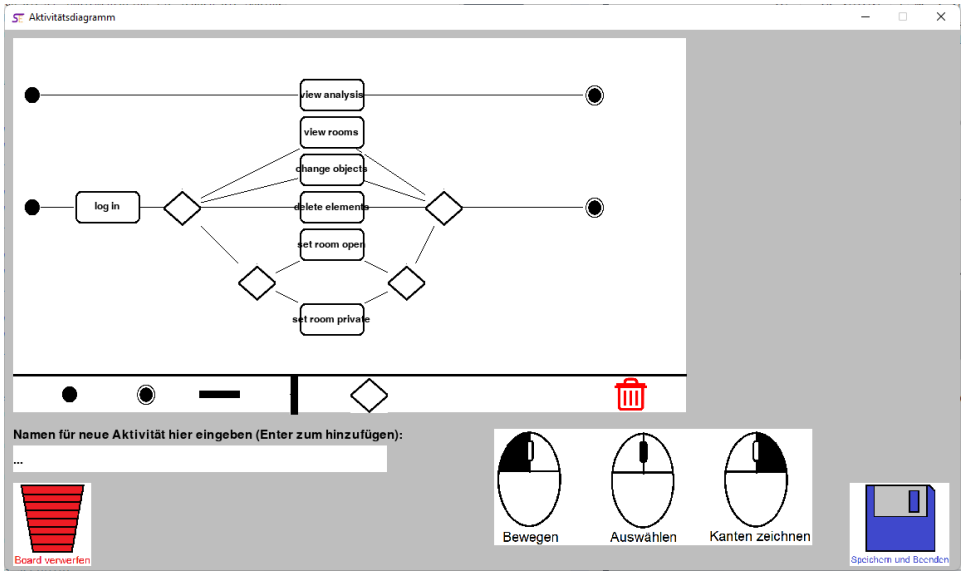


Abbildung 4.11: GUI: Aktivitätsdiagramm

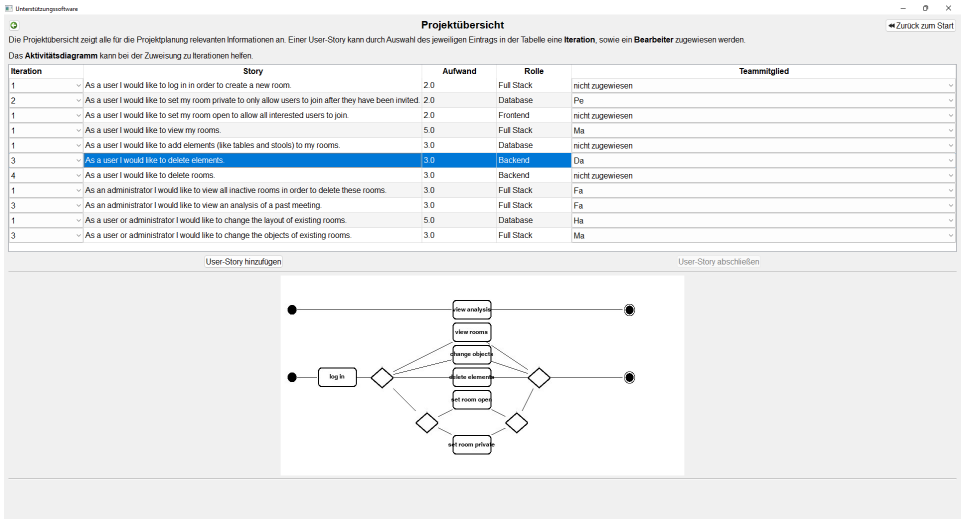


Abbildung 4.12: GUI: Projektübersicht

Kapitel 5

Studiendesign

Um den Nutzen der in Kapitel 4 vorgestellten Software zu überprüfen, wurde eine Nutzerstudie durchgeführt. Das Vorgehen wird in diesem Kapitel erläutert.

5.1 Ziel der Studie

Das Forschungsziel dieser Studie wird nach dem von Wohlin et al. [48] vorgeschlagenen *Goal-Definition-Template* formuliert:

Forschungsziel

Analysiere *den entwickelten Software-Prototyp*
mit dem Ziel*, das Potenzial zu untersuchen,*
in Bezug auf *die Unterstützung eines unerfahrenen Entwicklerteams*
aus Sicht *von unerfahrenen Entwicklern*
im Rahmen von *einer Nutzerstudie.*

5.2 Vorbereitung

Das Ziel der Nutzerstudie ist es zu prüfen, ob der Software-Prototyp als Ganzes eine wahrgenommene Unterstützung für die Studienteilnehmer bietet. Zudem soll der jeweilige Nutzen der Komponenten (Aufwandsschätzung, Rollenzuweisung und Priorisierung) für sich allein genommen evaluiert werden. Zu diesem Zweck wird eine Within-Subjects-Studie geplant, in der die Teilnehmer, in Gruppen von bis zu 8 Personen, User-Stories ohne Unterstützung schätzen sollen. Im Anschluss sollen sie die entwickelte Software als Unterstützung nutzen. Die verschiedenen Bedingungen beziehen sich lediglich auf die Komponente der Aufwandsschätzung. Bei der zweiten Bedingung sollen zusätzlich die anderen Komponenten des Prototyps zur weiteren Projektplanung genutzt werden.

Mit diesen Hintergedanken wurden demografische sowie inhaltliche Fragen konzipiert, die von den Teilnehmern beantwortet werden sollen. Folgende demografische Aspekte werden abgefragt:

- Identifizierungscode
- Geschlecht
- Muttersprache
- Studiengang
- Fachsemester
- Vorherige Projekterfahrung

Der Identifizierungscode dient dazu, die Antworten der zu dem Code gehörenden Person aus dem Datensatz zu entfernen, sofern dies gefordert wird. Eine Abfrage der Muttersprache ist vorteilhaft, da die zu beantwortenden Fragen bei der Nutzung der interaktiven Aufwandsschätzung auf Deutsch gestellt sind. Werden diese Fragen nicht korrekt verstanden und beantwortet, kann dies den resultierenden geschätzten Aufwand beeinflussen. Die Fragen bezüglich des Studiengangs, des Fachsemesters und der vorherigen Projekterfahrung sollen Aufschluss darüber geben, ob es sich bei den Teilnehmern um die Zielgruppe von unerfahrenen Entwicklern handelt.

Für die Durchführung der Studie werden User-Stories und der dazugehörige Kontext benötigt. Zu diesem Zweck wurden, basierend auf dem SWP-Projekt *VirtuHoS-1*[1], eine Beschreibung der gewünschten Software sowie zugehörige fiktive User-Stories erstellt, die im Anhang A.9 - A.10 abgebildet sind. Um den Zeitaufwand während der Durchführung zu verringern, wurden die User-Stories, die während der Nutzung des Prototyps verwendet werden sollten, als Textdatei gespeichert, sodass sie in die Software importiert werden können.

Zur Erhebung des *wahrgenommenen Nutzens* des Prototyps und seiner Komponenten wurden jeweils Fragen nach dem Beantwortungsmuster einer Likert-Skala mit vier Antwortmöglichkeiten konzipiert, bei denen auf das neutrale Element verzichtet wurde, um eine definitiv positive oder negative Tendenz als Antwort zu erhalten. Zudem wurden Fragen zur allgemeinen Nutzbarkeit des Prototyps und zur wahrgenommenen Hilfe durch diesen, im Vergleich zu der manuellen Schätzung, hinzugefügt. Zu jeder Frage gehört ein Kommentarfeld für optionale Erläuterungen. Zuletzt werden die Teilnehmer nach möglichen Verbesserungsvorschlägen oder Anmerkungen, sowie nach eigene Ideen, wie unerfahrene Teams in der Software-Entwicklung unterstützt werden können, gefragt. Eine Auflistung sämtlicher Fragen befindet sich im Anhang A.5. Zur Durchführung der Nutzerstudie werden die Fragen mit der Software *LimeSurvey*[2] dargestellt. Für den Fall, dass Teilnehmer im Zeitraum der Studiendurchführung keinen Zugang zu einem Gerät haben, das dem Ausfüllen des Fragebogens dient, wurden Kopien des Fragebogens

[1] https://www.pi.uni-hannover.de/de/se/lehre/swp/swp-wise-2020-2021/
[2] https://survey.uni-hannover.de/

auf Papier erstellt. Zudem wurde eine Datenschutzerklärung verfasst, durch die die Weiterverarbeitung der erhobenen Daten durch mögliche Folgestudien oder das SE sichergestellt werden soll, sowie den Teilnehmern die Möglichkeit geben soll, ihre erhobenen Daten im Nachhinein löschen zu lassen. Die Einverständniserklärung ist im Anhang A.8 zu finden.

5.3 Teilnehmerakquise

Die Rekrutierung der Studienteilnehmer erfolgte auf mehreren Wegen. Es wurde ein Rekrutierungsanschreiben (siehe Anhang A.7) verfasst und am *Schwarzen Brett*[3] von *StudIP* eingetragen. Selbiges Anschreiben wurde personalisiert und zusätzlich an alle Studierenden geschickt, die zu dem damaligen Zeitpunkt eine Abschlussarbeit am SE geschrieben haben. Bei dem dritten Rekrutierungsweg hat es sich um persönliche Anfragen zu Kommilitonen gehandelt. Als Anreiz zur Studienteilnahme wurde unter allen Teilnehmern ein Amazongutschein im Wert von 20,00 € zufällig verlost. Es konnten insgesamt 11 Studienteilnehmer rekrutiert werden. Dementsprechend wurde die Studie zweifach durchgeführt.

5.4 Ablauf

Für die Studie wurde eine Zeit von 45 Minuten angesetzt. Der Ablauf beider Studiendurchgänge war folgendermaßen geplant:

1. Aushändigen und Ausfüllen der Datenschutzerklärung (5 Min.)
2. Erläuterung des Studienablaufs (3 Min.)
3. Ausfüllen des demografischen Teils des Fragebogens (3 Min.)
4. Vorstellung des fiktiven Softwareprojekts (5 Min.)
5. Aufwandsschätzung zu drei User-Stories ohne Softwareunterstützung (5 Min.)
6. Nutzung der Software zur Projektplanung (15 Min.)
    - Anlegen des Projekts
    - Hinzufügen der Mitglieder
    - Importieren der User-Stories
    - Automatisierte Aufwandsschätzung und Rollenzuweisung
    - Interaktive Aufwandsschätzung (bei Bedarf)
    - Aktivitätsdiagramm erstellen
    - Priorisieren

[3] https://studip.uni-hannover.de/plugins.php/schwarzesbrettplugin/category/view/

7. Ausfüllen des inhaltlichen Teils des Fragebogens (9 Min.)

Nach dem Durchführen des ersten Durchgangs wurde das Feedback initial gesichtet. Das so gewonnene Usability-Feedback wurde vor dem Durchführen des zweiten Durchlaufs in den Prototyp eingepflegt, um potenzielle, auf schlechter Usability basierende negative Bewertungen, in diesem zu reduzieren. Das Feedback beinhaltete unter anderem die Anpassbarkeit der Schriftgröße für eine bessere Lesbarkeit bei großem Abstand zum Display sowie eine intuitivere Bedienung bei der Erzeugung des Aktivitätsdiagramms zur Priorisierung von User-Stories.

Der zweite Durchgang der Studie wurde Online mit drei Teilnehmern durchgeführt. Um den Probanden die Nutzung der Software zu ermöglichen, wurden die Anwendungen *AnyDesk*[4] und *Discord*[5] verwendet. *Discord* ist eine Software, die Audio- und Video-Kommunikation, sowie das Teilen des Bildschirms ermöglicht. Sie wurde zur Kommunikation mit den Studienteilnehmern verwendet. *Anydesk* erlaubt den Fernzugriff einer Person auf den Rechner einer anderen. Mit dieser wurde einem Teilnehmer Zugriff auf den Rechner gewährt, auf dem der Prototyp ausgeführt wurde. Für die anderen Teilnehmer wurde lediglich der Bildschirm geteilt, sodass sie die Interaktionen verfolgen konnten. Die Durchführung verlief nach dem oben beschriebenen Ablauf, mit dem Unterschied, dass die Datenschutzerklärung lediglich von den Teilnehmern gelesen, jedoch nicht unterschrieben wurde.

[4] https://anydesk.com/de
[5] https://discord.com/

Kapitel 6

Studienergebnisse

In diesem Kapitel werden die Studienergebnisse dargestellt. Eine Interpretation der Ergebnisse findet in Kapitel 7 statt. Die Ergebnisse setzen sich aus den Antworten der demografischen und der inhaltlichen Fragen zusammen.

6.1 Demografische Fragen

Die Studie wurde in zwei Durchgängen mit insgesamt elf Teilnehmern durchgeführt, wobei es sich bei neun Teilnehmern um männliche und bei zwei Teilnehmern um weibliche Personen gehandelt hat. Für einen einzelnen Teilnehmer war Deutsch nicht die Muttersprache. Sämtliche Teilnehmer waren in einem Informatikstudiengang eingeschrieben, oder hatten einen solchen bereits erfolgreich abgeschlossen. Die Anzahl der absolvierten Fachsemester rangiert von drei bis zehn. Alle Teilnehmer verfügten über vorherige Projekterfahrung im Bereich der Softwareentwicklung, zum Beispiel aus dem SWP, dem XP-Labor, eigenen Projekten oder durch eine berufliche Tätigkeit. Aufgrund dieser Vorerfahrungen fallen die Probanden genau in die gewünschte Zielgruppe. Sie verfügen über eine gewisse Projekterfahrung und kennen sich bereits mit den Grundlagen und Fachbegriffen aus, haben aber keine langjährige Praxiserfahrung in der Software-Entwicklung.

6.2 Inhaltliche Fragen

Die Antworten der inhaltlichen Fragen dienen der Beantwortung der ersten Forschungsfrage. Sie sind in den Abbildungen 6.1 – 6.3 dargestellt. Bezüglich der Verständlichkeit und Nutzbarkeit des Software-Prototyps wurden acht positive und drei negative Antworten gegeben.

Die **wahrgenommene Unterstützung** durch die **automatisierte und interaktive Aufwandsschätzung** wurde ebenfalls jeweils drei-mal negativ und acht-mal positiv beantwortet. Die automatisierte Aufwandsschätzung wurde zwei-mal, die interaktive drei-mal als *sehr hilfreich* bewertet. Zudem

wurde diese jedoch einmal als *überhaupt nicht hilfreich* bewertet. Für die **automatisierte Rollenzuweisung** wurden neun negative und zwei positive Antworten gegeben. Die negativen Bewertungen beinhalten fünf *überhaupt nicht hilfreich*-Stimmen.

Die **Schätzung mithilfe des Software-Prototyps wurde im Vergleich mit der zu Beginn manuell durchgeführten Schätzung** siebenmal als unterstützend, und viermal als nicht unterstützend wahrgenommen. Bezüglich der **Weiterempfehlung des Prototyps an unerfahrene Teams** wurden drei negative und acht positive Antworten gegeben. Die zu dieser Frage abgegebenen Kommentare kritisierten die verbesserungswürdige Usability, den hohen Zeitaufwand bei der Verwendung des Prototyps, und die nicht hilfreiche Rollenzuweisung. Die folgenden zwei Kommentare sollen dies verdeutlichen:

- „Wenn die GUI verbessert wird und etwas mehr Usability reinkommt, auf jeden Fall.“ **[Teilnehmer-ID: 0001]**

- „Ist ein ordentlicher Anhaltspunkt wahrscheinlich, wenn man gar keine Ahnung hat. Allerdings müssen die Rollen überarbeitet werden.“ **[Teilnehmer-ID: 0000]**

Die Nummer hinter dem Zitat gibt die jeweilige selbstgewählte Nutzer-ID des Teilnehmers an.

Die Abschlussfragen bezüglich **Anmerkungen und Verbesserungsvorschlägen** haben zahlreiche wertvolle Hinweise geliefert. Die folgende Auflistung zeigt sämtliche Vorschläge, unterteilt in solche, die bereits vor dem zweiten Studiendurchlauf in den Prototyp implementiert wurden, und solche, die noch nicht implementiert wurden:

Implementierte Vorschläge:

- *Möglichkeit zum interaktiven Anpassen der Schriftgröße.* Implementiert durch zusätzliche Oberfläche zur dynamischen Anpassung der Schriftgröße während der Nutzung des Prototyps.

- *Texte in Kästen integrieren, bessere Skalierbarkeit der Benutzeroberfläche.* Implementiert durch dynamisches Anpassen der GUI-Elemente bei Änderung der Fenstergröße sowie Optimierung der Darstellung im Vollbildmodus.

- *Pop-up bei Ladezeiten.* Implementiert durch eine zusätzliche informative Oberfläche während der Berechnung der Story-Points, der Rollenzuweisung und der Extraktion der Aktivitäten aus den User-Stories. Während dies angezeigt wird, ist keine Interaktion mit dem Prototyp möglich.

- *Speichern-Button, erläuternde Abbildungen und bessere Usability beim Aktivitätsdiagramm.* Das Speichern war zuvor nur über Schließen des Fensters möglich, was ohne Erklärung während der Studie zu Verwirrung geführt hat. Es wurden zusätzlich zum Speichern weitere Funktionalitäten wie das Löschen der gesamten Aktivitäten und die Option zur Mehrfachauswahl dieser implementiert.

Nicht implementierte Vorschläge:

- *Kombination der Verfahren für interaktive und automatisierte Aufwandsschätzung.* Eine Kombination der beiden Verfahren würde quantitative Auswertungen der ermittelten Story-Points, und eine Umstrukturierung des Prototyps voraussetzen. Es handelt sich um einen interessanten Ansatz, der im Rahmen von weiteren Abschlussarbeiten erforscht werden könnte.

- *Explainability bezüglich der Schätzungsergebnisse.* Um Explainability zu sämtlichen Komponenten des Prototyps zu ermöglichen, wäre eine Überarbeitung der meisten Benutzeroberflächen notwendig. Dies wäre jedoch ebenfalls ein interessanter Ansatz für künftige Forschung.

Zusätzlich zu den soeben dargestellten Verbesserungsvorschlägen haben die Studienteilnehmer noch zahlreiche zusätzliche Ideen für eine mögliche Unterstützung von unerfahrenen Teams beigetragen:

- Kurze einschlägige Tutorials der Art „Auf diese Aspekte muss beim Schätzen von User-Stories geachtet werden...“.

- Auflistung von passenden Frameworks nach Beantwortung einer Reihe von Fragen.

- Jira-Plugins mit den im Prototyp implementierten Funktionalitäten (nicht step-by-step).

- Unerfahrenen Entwicklern Beispielprojekte mit den zugehörigen Schätzwerten zeigen.

- Unerfahrenen Entwicklern Diskussionsvideos von der manuellen Schätzung von User-Stories anzeigen.

- Unterstützung durch erfahrene Entwickler beim Planungsprozess.

Nachfolgend werden die wichtigsten Ergebnisse der Studie noch einmal aufgelistet. Die vierstufigen Likertskalenelemente werden zusammengefasst mit − und + abgekürzt:

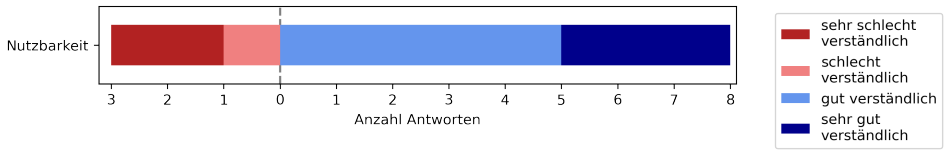


Abbildung 6.1: Antworten der Studienteilnehmer ($n = 11$) zur Verständlichkeit der Nutzung der Software auf Basis der Antworten einer Likert-Skala.

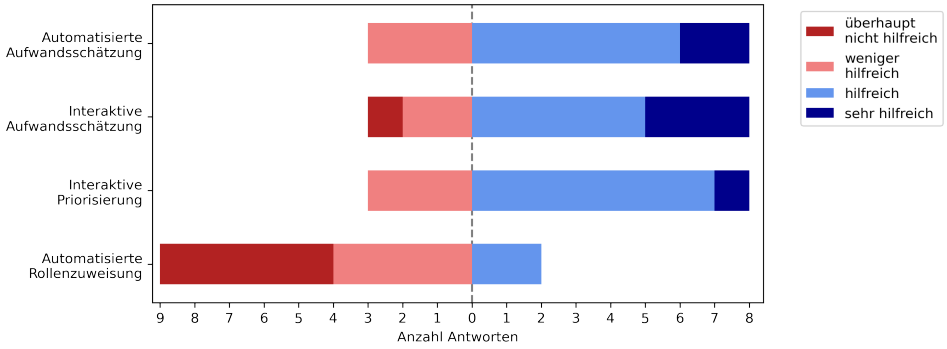


Abbildung 6.2: Antworten der Studienteilnehmer ($n = 11$) bezüglich der wahrgenommenen Unterstützung durch die aufgelisteten Kernkomponenten des Software-Prototyps auf Basis der Antworten einer Likert-Skala.

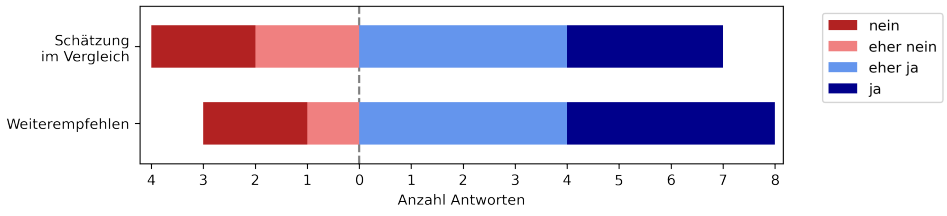


Abbildung 6.3: Antworten der Studienteilnehmer ($n = 11$) bezüglich der Unterstützung durch die softwaregestützte Aufwandsschätzungen im Vergleich mit der manuellen Aufwandsschätzung, sowie die Weiterempfehlung des Software-Prototyps, jeweils auf Basis der Antworten einer Likert-Skala.

Wichtigste Studienergebnisse

Die Freitextantworten haben deutlich gemacht, dass eine bessere Usability des Prototyps benötigt wird, um die Interaktionszeit zu reduzieren. Die Teilnehmer haben zusätzlich zahlreiche Vorschläge für weitere Unterstützung von unerfahrenen Teams vorgeschlagen. Bezüglich der wahrgenommenen Unterstützung und der Weiterempfehlung des Prototyps wurden die folgenden Antworten gegeben:

- Weiterempfehlung des Prototyps: − (3), + (8)
- Als unterstützend wahrgenommene Komponenten:
  - Automatisierte Aufwandsschätzung + (8)
  - Interaktive Aufwandsschätzung + (8)
  - Interaktive Priorisierung + (8)
- Als nicht unterstützend wahrgenommene Komponenten:
  - Automatisierte Rollenzuweisung − (9)

Kapitel 7

Diskussion und Interpretation

In diesem Kapitel der Arbeit werden einerseits die Ergebnisse der in Kapitel 4 vorgestellten Verfahren zur automatisierten Aufwandsschätzung, Zuweisung und Priorisierung von User-Stories, und andererseits die Ergebnisse der in Kapitel 5 dargestellten Nutzerstudie, interpretiert und gedeutet.

7.1 Beantworten der Forschungsfragen

Dieser Abschnitt befasst sich mit der Beantwortung der folgenden, bereits in Kapitel 1 vorgestellten, Forschungsfragen:

Forschungsfrage 1

Inwiefern können existierende maschinelle Lernverfahren genutzt werden, um den Aufwand von User-Stories akkurat zu schätzen?

Zur Beantwortung der ersten Forschungsfrage wurde in Kapitel 4 ein Clustering-Verfahren basierend auf kontextuellen BERT-Embeddings implementiert und auf mehreren Datensätzen ausgewertet. Zu diesem Zweck wurde der *MdAE* ermittelt, und mit dem *MdAE* der Mittelwert-Baseline und der Median-Baseline für jeden Datensatz verglichen, die auch zur Evaluation von LHC_{TC}-*SE* genutzt worden sind [44]. Zusätzlich wurde zu jedem Datensatz der *SA*-Wert ermittelt. Dieser dient als Indikator, ob das Verfahren besser als zufälliges Raten abschneidet. Im Vergleich mit der Mittelwert-Baseline hat das Clustering-Verfahren in drei Fällen statistisch signifikant schlechtere, und in den restlichen neun Fällen, gleich gute Ergebnisse erzielt. Bei dem Vergleich mit der Median-Baseline wurden jeweils sechs signifikant schlechtere und sechs gleich gute Ergebnisse erzielt. Das genutzte Verfahren liefert somit keine besseren Schätzwerte als bei Verwendung einer der beiden Baselines. Diese Ergebnisse bezüglich der Median-Baseline decken sich mit den Erkenntnissen von Tawosi et al. [44] im Bezug auf LHC_{TC}-*SE* [44]. Die Ergebnisse des jeweils ermittelten *SA*-Wertes liegen in allen Fällen deutlich

(>27) über einem negativen Wert, was verdeutlicht, dass das Verfahren besser als zufälliges Raten abschneidet. Die Kombination dieser Resultate legt nahe, dass das Clustering mittels kontextueller BERT-Embeddings in dieser Form nur eingeschränkt dabei helfen kann, User-Stories akkurat zu schätzen.

Forschungsfrage 2

Wie schneidet der Clustering-Ansatz mit kontextuellen BERT-Embeddings im Vergleich zu existierenden Ansätzen zur Aufwandsschätzung von User-Stories ab?

Das implementierte Clustering-Verfahren dient auch zur Beantwortung der zweiten Forschungsfrage als Grundlage. Zur Klärung wurde eine Vergleichsauswertung des Verfahrens (auf Basis des *MdAE*) mit der State-of-the-Art Methode *Deep-SE* und dem auf dem *Latent Dirichlet Allocation* (LDA) basierenden Clustering-Verfahren LHC_{TC}-*SE* durchgeführt. Das Verfahren hat in sechs von zwölf Fällen statistisch signifikant schlechter abgeschnitten als *Deep-SE*, in einem Fall statistisch signifikant besser, und in fünf Fällen gleich gut. Der Vergleich mit LHC_{TC}-*SE* hat in sechs Fällen statistisch signifikant schlechtere, und in den restlichen sechs Fällen gleich gute Ergebnisse erzielt. Das BERT-Clustering Verfahren liefert somit schlechtere Ergebnisse als die getesteten Verfahren. Aufgrund der Ergebnisse bietet es sich an, alternative Clustering-Methoden zu erforschen, und währenddessen auf andere Methoden zur Aufwandsschätzung auszuweichen.

Forschungsfrage 3

Wie können unerfahrene Teams unterstützt werden, um die Planung von Softwareprojekten zu erleichtern?

Zur Beantwortung der dritten Forschungsfrage wurde eine Nutzerstudie durchgeführt, um den Grad der wahrgenommenen Unterstützung durch die Komponenten des Prototyps bei der Projektplanung zu erheben. Zudem sollte die Unterstützung der Software als Ganzes bewertet werden. Die Kombinationen der Antworten der folgenden Fragen haben zur Beantwortung der Forschungsfrage beigetragen:

1. Wie hilfreich war die automatisierte Aufwandsschätzung bei der Planung des Software-Projekts?

2. Wie hilfreich war die interaktive Aufwandsschätzung bei der Planung des Software-Projekts?

3. Wie hilfreich war die interaktive Priorisierung bei der Planung des Software-Projekts?

4. Wie hilfreich war die automatisierte Rollenzuweisung bei der Planung des Software-Projekts?
5. War die softwaregestützte Aufwandsschätzung, im Vergleich zu der zu Beginn durchgeführten, manuellen Aufwandsschätzung, eine Unterstützung?
6. Würden Sie unerfahrenen Teams die Verwendung der Software empfehlen? Bitte begründen Sie Ihre Entscheidung.

Mithilfe der Fragen 1 bis 4 sollten die Komponenten der Software identifiziert werden, mit denen unerfahrene Teams unterstützt werden können. Die gegebenen Antworten haben deutlich gemacht, dass lediglich die automatisierte Rollenzuweisung als nicht unterstützend empfunden wurde. Durch die Frage 5 sollte geprüft werden, ob eine computergestützte Aufwandsschätzung vorteilhaft gegenüber der standardmäßig durchgeführten, manuellen Aufwandsschätzung ist. Die Antworten legen nahe, dass dies der Fall ist. Schlussendlich wird mit der 6-ten Frage nach der Unterstützung des Prototyps als Ganzes gefragt – auch hier zeigen sich positive Tendenzen dahingehend, dass der Prototyp eine unterstützende Wirkung besitzt.

7.2 Interpretation

Die quantitative Auswertung des Clustering-Verfahrens hat deutlich gemacht, dass das Verfahren für die Aufwandsschätzung von User-Stories nur bedingt hilfreich ist. Es wird ein *MdAE* ermittelt, der schlechter als die State-of-the-Art Methoden abschneidet. Das BERT-Clustering-Verfahren schneidet nicht besser als die Median-Baseline ab, genau wie bereits für die betrachteten Verfahren *Deep-SE* und LHC_{TC}*-SE* von Tawosi et al. [44, 45] angemerkt worden ist. Das Verfahren liefert jedoch dank des vortrainierten BERT-Modells den Vorteil, dass sich die Vorbereitungszeit zur Clusterbildung und zur Findung der optimalen Clustergröße im Bereich von wenigen Stunden bewegt, und sich somit im gleichen zeitlichen Rahmen wie die Trainingszeit von LHC_{TC}*-SE* bewegt. Zum Vergleich: das Trainieren von *Deep-SE* auf neuen Datensätzen kann Tage in Anspruch nehmen [4]. Aufgrund der schlechteren Ergebnisse kann das Verfahren nicht ohne Vorbehalte zur Aufwandsschätzung empfohlen werden. Es kann jedoch als Grundlage für weitere Forschung im Bereich der Aufwandsschätzung genutzt werden. Mögliche Erweiterungen und Anpassungen werden in Kapitel 8 beschrieben.

Die qualitativen Ergebnisse der Nutzerstudie zeigen klare positive Tendenzen bezüglich der wahrgenommenen Unterstützung durch den Prototyp in der Projektplanung. Von den elf Studienteilnehmer würden acht den Prototyp an unerfahrene Teams weiterempfehlen, wobei vier diese Empfehlung nur schwach zum Ausdruck bringen. Jeweils acht der Teilnehmer empfanden die automatisierte und die interaktive Aufwandsschätzung sowie

die interaktive Priorisierung als *hilfreich* oder *sehr hilfreich*. Diese positiven Tendenzen zeigen sich jedoch nur in der Aufwandsschätzung und der Priorisierung von User-Stories. Die Rollenzuweisung wurde von vier Studienteilnehmern als *nicht hilfreich*, und von fünf weiteren als *überhaupt nicht hilfreich* wahrgenommen. Dies lässt sich zum Teil mithilfe der Genauigkeit des in Kapitel 4 trainierten LSTM-Netzes erklären: Das Netz hat lediglich eine Genauigkeit von 64.38% – somit wird etwa jede dritte Rolle falsch zugewiesen, was einen nicht zu vernachlässigenden Teil der Zuweisungen darstellt. Die Resultate dieses Abschnitts werden nachfolgend noch einmal zusammengefasst:

Interpretationsergebnisse

Der Prototyp zeigt positive Tendenzen bezüglich der wahrgenommenen Unterstützung von unerfahrenen Entwicklerteams, die jedoch in einer ausführlicheren Studie bestätigt werden müssen.
Das BERT-Clustering Verfahren schneidet besser als zufälliges Raten, jedoch statistisch signifikant schlechter als die Vergleichsmethoden *Deep-SE*, LHC_{TC}*-SE* und die Baselines ab.
Die automatisierte Rollenzuweisung liefert aufgrund der niedrigen Genauigkeit nur bedingte Unterstützung und bedarf einer Verbesserung.

7.3 Einschränkung der Validität

In diesem Abschnitt wird auf die Einschränkungen der Validität basierend auf den vier von Wohlin et al. [48] vorgeschlagenen Kategorien – der *Conclusion Validity*, *Internal Validity*, *Construct Validity* sowie der *External Validity* – eingegangen.

Conclusion Validity: Bei den durch die Studie erhaltenen qualitativen Ergebnissen handelt es sich aufgrund der geringen Teilnehmerzahl von $n = 11$ um Ergebnisse, deren Aussagekraft stark eingeschränkt ist. Um die qualitativen Tendenzen zu bestätigen, müsste eine weitere Studie mit großer Teilnehmerzahl ($n \approx 60$) durchgeführt werden, zu der die hier durchgeführte Studie als Vorstudie fungieren könnte.

Internal Validity: Aufgrund der Studiendurchführung mit einer großen Gruppe und einer weiteren kleinen Gruppe von lediglich drei Personen ist eine nur sehr kleine Kontrollgruppe vorhanden. Die Resultate des zweiten Durchgangs zeigen zwar zum ersten Durchgang passende Tendenzen, zur Prüfung der Resultate wäre jedoch eine größere Anzahl an Teilnehmern vonnöten. Zudem erfolgte die Teilnehmerrekrutierung nicht zufällig.

Die gegebenen Informationen bezüglich des Projektkontexts und der

User-Stories für die zu planende Software in der Studie waren nicht ausführlich genug, um eine korrekte Schätzung zu ermöglichen. Sie haben zu viel Interpretationsspielraum zugelassen, um in der gegebenen Zeit eine vollständige Analyse durchführen zu können, was zu Unklarheiten während der Studiendurchführung geführt hat. Es wurde versucht, eine möglichst gute Beschreibung zu liefern, die es den Teilnehmern in der Kürze der Zeit ermöglicht, einen Überblick über das Projekt zu erhalten. Aufgrund der bereits hoch angesetzten Zeitdauer von 45 Minuten war eine umfassendere Beschreibung nicht möglich, ohne die Gesamtdauer der Studie weiter zu erhöhen. Eine solche Erhöhung hätte die Aufmerksamkeit der Probanden negativ beeinflussen können. Im Rahmen eines echten Software-Projekts haben die Teilnehmenden in der vorangehenden Phase der Anforderungserhebung deutlich mehr Zeit, um sich mit dem Projektkontext auseinander zu setzen und um Abhängigkeiten zwischen, sowie Voraussetzungen für die User-Stories zu identifizieren.

Construct Validity: Die für die Entwicklung des Clustering-Verfahrens genutzten Datensätze stammen aus Open-Source Projekten, deren Verlässlichkeit bezüglich dem zu einer User-Story gehörenden Aufwand nicht ohne Einschränkungen zu vertrauen ist. Zudem stimmen die Formulierungen der User-Stories nicht in sämtlichen Fällen mit der Definition einer User-Story nach Cohn [5] überein. Idealerweise wäre ein Datensatz, der aus nach obiger Definition geschriebener User-Stories besteht, die aus einem Team von unerfahrenen Entwicklern im Rahmen eines Universitätsprojekts (wie dem SWP) geschätzt wurden.

Gleiches gilt für die Datensätze, die zur Implementierung der automatisierten Rollenzuweisung genutzt wurden. Zudem handelt es sich bei den für das Training und die Auswertung der angepassten Rollen genutzten Datensätzen um zum Teil selbst gelabelte Datensätze. Es liegt zudem eine ungleichmäßige Verteilung der verschiedenen Labels in den Datensätzen vor. Das Label *Datenbank* ist in geringerer Anzahl vertreten als die anderen drei Labels, was dem LSTM-Netz die Zuweisung des Labels erschwert [15]. Die entwickelte Software ist relativ komplex. Es sind viele Funktionalitäten und Benutzerschnittstellen vorhanden. Lediglich ein Teilnehmer war für die Interaktion mit dem Prototyp verantwortlich. Während der Nutzung stand ich für Rückfragen zur Verfügung. Eine längere Einarbeitungszeit, eventuell in Form einer kurzen Experimentierphase von etwa zehn Minuten, in der jeder Teilnehmer eine eigene Version der Software ausprobieren darf, wäre jedoch eine Option, um die Komplexität der Software besser überblicken zu können. Um die Belastung des interagierenden Probanden im zweiten Studiendurchgang zu reduzieren, wurde nach dem ersten Durchlauf der Studie Usability-Feedback eingepflegt.

External Validity: Während des Rekrutierungsprozesses wurde explizit nach Teilnehmern mit geringer oder nicht vorhandener Vorerfahrung im Bereich der Software-Entwicklung gesucht. Sämtliche Studienteilnehmer verfügen jedoch über ein gewisses Maß Vorerfahrung im Bereich der Software-Entwicklung, wodurch sie Teil der Zielgruppe sind, jedoch zum Teil über mehr Erfahrung verfügen, als Studenten während der Durchführung des SWPs haben. Somit kann eine Studie mit völlig unerfahrenen Teilnehmern grundlegend andere Resultate mit sich bringen.

Kapitel 8

Zusammenfassung und Ausblick

In diesem abschließenden Kapitel werden die wichtigsten Ergebnisse dieser Arbeit zusammengefasst. Zudem werden zahlreiche weiterführende Forschungsansätze präsentiert, die auf Basis des in dieser Arbeit implementierten Prototyps weitere Unterstützung, nicht nur beschränkt auf unerfahrene Entwicklerteams, bieten.

8.1 Zusammenfassung

Das Ziel dieser Arbeit ist es herauszufinden, wie unerfahrene Teams in der agilen Software-Entwicklung unterstützt werden können. Zu diesem Zweck wurde ein Software-Prototyp mit grafischer Benutzeroberfläche implementiert. Die Haupt-Funktionalitäten des Prototyps liegen in der Verarbeitung von textuellen User-Stories. Sie basieren auf aktueller Forschung und umfassen die Aufwandsschätzung, die Zuweisung zu vier Entwicklerrollen, und die Priorisierung von User-Stories durch die Erzeugung eines Aktivitätsdiagramms mit anschließender Zuweisung zur passenden Iteration.

Die Aufwandsschätzung besteht aus zwei Teilen. Im interaktiven Teil wird der Aufwand von User-Stories durch Beantwortung von spezifischen Fragen mithilfe eines Bayes-Netzes geschätzt. Der automatisierte Teil nutzt eine neue Clustering-Methode auf Basis eines vortrainierten BERT-Modells. Sie hat dem State-of-the-Art Verfahren zur Aufwandsschätzung, *Deep-SE*, gegenüber den Vorteil, dass sie eine stark reduzierte Trainings- beziehungsweise Vorbereitungszeit benötigt [44]. Das genutzte Verfahren wurde im Rahmen einer Vergleichsauswertung mit einer Mittelwert- und einer Median-Baseline, sowie zwei aktuellen Verfahren zur Aufwandsschätzung verglichen. Die Verfahren wurden, auf Basis des berechneten MdAE, auf zwölf Datensätzen aus dem *Tawos-Datensatz* miteinander verglichen. Das implementierte Verfahren liefert bessere Ergebnisse als zufälliges Raten,

schneidet jedoch statistisch signifikant schlechter als die Mittelwert- und Median-Baseline, *Deep-SE* und *LHC$_{TC}$-SE* ab. Die angepasste Rollenzuweisung ermöglicht eine Zuweisung der User-Stories zu den Klassen *Frontend*, *Backend*, *Full Stack* und *Datenbank* mit einer Genauigkeit von 64,38 %. Die letzte Komponente des Prototyps unterstützt bei der Priorisierung von User-Stories. Um diese zu erleichtern, werden automatisiert Aktivitäten aus den User-Stories extrahiert, die manuell zu einem Aktivitätsdiagramm zusammengefügt werden können. Auf Basis dieses Diagramms findet die Zuweisung der User-Stories zur passenden Iteration statt.

Um identifizieren zu können, welche Komponenten des Prototyps unerfahrene Entwickler bei der Projektplanung unterstützen können, wurde eine Nutzerstudie mit elf Teilnehmern durchgeführt. Die Ergebnisse weisen positive wahrgenommene Tendenzen bezüglich der Unterstützung durch die Aufwandsschätzung und die Priorisierung von User-Stories auf. Die Rollenzuweisung wird tendenziell nicht als unterstützend empfunden (neun negativ, zwei positiv). Der entwickelte Prototyp als Ganzes wird von acht der elf Teilnehmenden als hilfreich wahrgenommen. Die Ergebnisse zeigen das mögliche Potenzial des Ansatzes, das im Rahmen einer Weiterentwicklung ausgeschöpft und evaluiert werden kann.

8.2 Ausblick

Aufgrund der Struktur des Software-Prototyps können die genutzten Methoden mit wenig Aufwand verändert, ersetzt oder erweitert werden. Die in dieser Arbeit geschaffene Software kann als Grundgerüst für weitere Arbeiten im Bereich der Unterstützung von unerfahrenen Teams, oder auch zur Nutzung in erfahreneren Teams, genutzt werden.

In einer kürzlich erschienenen Veröffentlichung von Kuhrmann et al. [22] wurden interessante Parallelen zwischen den Schwierigkeiten beim Software-Entwicklungsprozess von Startups zu unerfahrenen Teams aufgezeigt. Dies ermöglicht eine potenzielle Ausweitung der Zielgruppe des Software-Prototyps in einen wirtschaftlichen Bereich.

Die Nutzerstudie hat Verbesserungspotential bezüglich der Usability des Prototyps aufgezeigt, das zum Teil bereits implementiert worden ist. Es kann überprüft werden, ob eine umfassende Überarbeitung der Benutzeroberfläche, mit dem expliziten Fokus auf Usability, die wahrgenommene Unterstützung im Vergleich zu der jetzigen Version des Prototyps verbessert.

Eine weitere zu prüfende Option ist die Weiterentwicklung des Prototyps zur Mobilanwendung, mit dem Ziel, die interaktiven Elemente des Prototyps von mehreren Nutzern simultan bedienen zu lassen. Dies würde im Bereich der interaktiven Aufwandsschätzung die Diskussion im Team redundant machen, sowie eine Zusammenarbeit bei der Erzeugung des Aktivitätsdiagramms ermöglichen, wodurch der Zeitaufwand reduziert werden würde.

Die durch die Nutzerstudie aufgezeigten positiven Tendenzen haben aufgrund der geringen Teilnehmerzahl nur eine eingeschränkte Aussagekraft. Aus diesem Grund bietet sich die Durchführung einer größeren Nutzerstudie, zum Beispiel im Rahmen des Software-Projekts, an.

Die Implementierung der Kernkomponenten des Prototyps in Form von mehreren Jira-Plugins bietet sich an, um die durch Jira bereits vorhandene effiziente Infrastruktur zur Projektverwaltung durch zusätzliche Unterstützungsmaßnahmen zu erweitern.

Im Bereich der Aufwandsschätzung wurde der Mittelwert aller Wort-Embeddings eines Satzes als Grundlage für das Clustering-Verfahren, und somit zur Schätzung des Aufwands einer User-Story, genutzt. Die genutzten kontextuellen BERT-Embeddings bieten jedoch eine Vielzahl an Möglichkeiten zur Bildung eines Gesamt-Embeddings aus den zugrunde liegenden Worten, wie zum Beispiel die Nutzung mehrerer Ausgabe-Layer, oder die Verwendung von *SBERT* oder *RoBERTa* [35, 25]. Es kann erforscht werden, ob andere Embeddings bessere Ergebnisse liefern, als das genutzte Verfahren.

In kürzlich veröffentlichten Arbeiten von Tawosi et al. [45, 46] wurde ein möglicher Bias in den durch Menschen in Story-Points geschätzten Datensätzen entdeckt, der die Genauigkeit von Aufwandsschätzungsverfahren beeinflusst, die auf diesen Datensätzen aufbauen. Die Autoren schlagen vor, dass zukünftige Verfahren die Entwicklungszeit anstelle von Story-Points berücksichtigen und berechnen sollen.

Das Training der genutzten Algorithmen zur Aufwandsschätzung und zur automatisierten Rollenzuweisung mit Datensätzen, die nur auf User-Stories aus Software-Projekten basieren, wäre eine Möglichkeit, die Software optimal für künftige Softwareprojekte zu trainieren, um sie den Studierenden für die Planung langfristig zur Verfügung zu stellen. Durch das Aufzeichnen der Entwicklungszeit kann zudem ein Datensatz erzeugt werden, mit dem der oben angesprochene Bias reduziert werden kann.

Anhang A

Appendix

A.1 Ward-Beispielrechnung

Zur Veranschaulichung des Ward-Verfahrens werden die Tupel $p_1 = (1, 2), p_2 = (3, 4), p_3 = (8, 9)$ betrachtet. Es sind somit zu Beginn drei Cluster vorhanden. Es folgt, dass die beste Clustering-Option in einer der Kombinationen $p_{1,2} = p_1 + p_2, p_{1,3} = p_1 + p_3, p_{2,3} = p_2 + p_3$ liegen muss. Es muss nun geprüft werden, in welchem der drei Fälle E am geringsten ist.

Fall $p_{1,2}$:

$$
\begin{aligned}
&\overline{x}_{1,1} = \frac{1}{2} \cdot (1 + 3) = 2, \quad \overline{x}_{1,2} = \frac{1}{2} \cdot (2 + 4) = 3 \\
&E_1 = ((1 - 2)^2 + (2 - 2)^2) + ((3 - 3)^2 + (4 - 3)^2) = 2 \\
&\overline{x}_{2,1} = 8, \quad \overline{x}_{2,2} = 9 \\
&E_2 = ((8 - 8)^2 + (9 - 9)^2) = 0 \\
&E = 2 + 0 = 2
\end{aligned}
$$

Fall $p_{1,3}$:

$$
\begin{aligned}
&\overline{x}_{1,1} = \frac{1}{2} \cdot (1 + 8) = 4.5, \quad \overline{x}_{1,2} = \frac{1}{2} \cdot (1 + 9) = 5 \\
&E_1 = ((1 - 4.5)^2 + (2 - 4.5)^2) + ((8 - 5)^2 + (9 - 5)^2) = 43.5 \\
&\overline{x}_{2,1} = 3, \quad \overline{x}_{2,2} = 4 \\
&E_2 = ((3 - 3)^2 + (4 - 4)^2) = 0 \\
&E = 43.5 + 0 = 43.5
\end{aligned}
$$

Fall $p_{2,3}$:

$$\overline{x}_{1,1} = 1, \quad \overline{x}_{1,2} = 2$$
$$E_1 = ((1-1)^2 + (2-2)^2) = 0$$
$$\overline{x}_{2,1} = \frac{1}{2} \cdot (3+8) = 5.5, \quad \overline{x}_{2,2} = \frac{1}{2} \cdot (4+9) = 6.5$$
$$E_2 = ((3-5.5)^2 + (4-5.5)^2) + ((8-6.5)^2 + (9-6.5)^2) = 17$$
$$E = 0 + 17 = 17$$

Die Funktion $max(\cdot)$ gibt den maximalen Eintrag einer Liste zurück. Es folgt $max([2, 43.5, 17]) = 2$. Somit wird im ersten Schritt der Cluster $p_{1,2}$ gebildet.

A.2 Datenbankschema

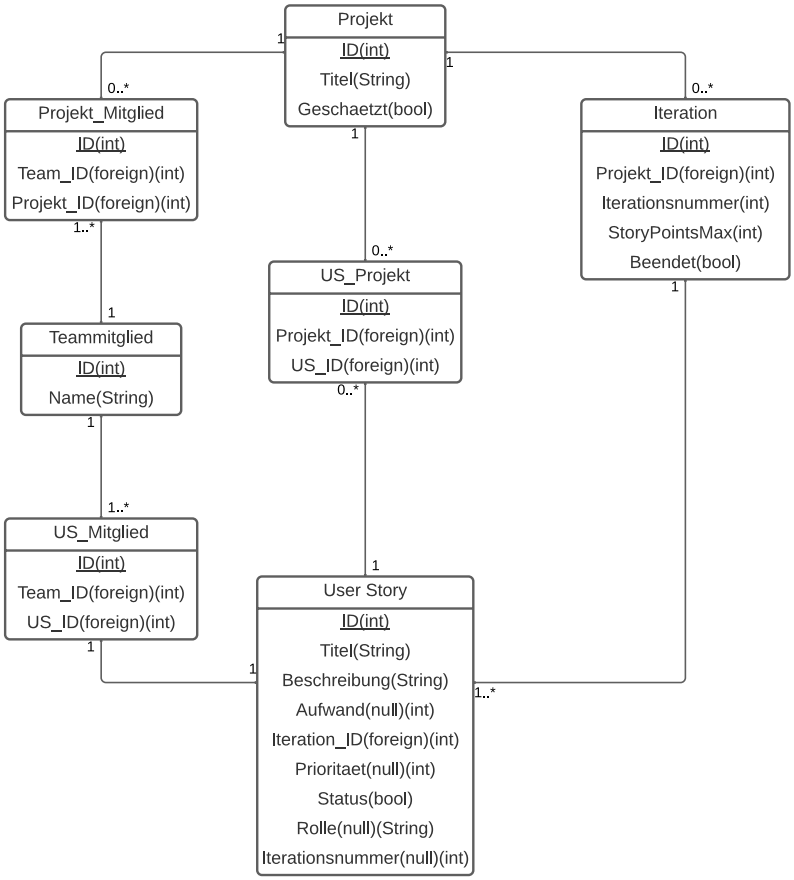


Abbildung A.1: Datenbankschema

A.3 Anleitungen

Die entwickelte Software muss mit Python Version 3.8 ausgeführt werden, um die Funktionsweise aller in der *requirements.txt* aufgeführten Bibliotheken zu gewährleisten! Die Software wurde auf Windows 10 und Windows 11 getestet. Eine fehlerfreie Funktionsweise auf anderen Betriebssystemen kann nicht gewährleistet werden! Zur Nutzung der Software wird aufgrund der Funktionsweise einer Bibliothek eine Internetverbindung benötigt.

Installation und Ausführung

Zum Installieren der Software müssen die folgenden Schritte ausgeführt werden:

1. Python Version 3.8 von https://www.python.org/downloads/ herunterladen und installieren.
2. Alle benötigten Anforderungen über die Befehlszeile installieren:
   $ pip install requirements.txt
   (Die Installation kann wenige Minuten dauern, da größere Dateien heruntergeladen werden müssen.)

Zum Ausführen der Software muss folgender Befehl in das CLI eingegeben werden:

```
$ python Launch.py
```

Deep-SE Voraussetzungen

Das ursprünglich in der Arbeit von Choetkiertikul et al. [4] genannte Github-Repository steht nicht mehr zur Verfügung. Die zur Implementierung des Verfahrens genutzten Daten stammen aus dem Repository eines Mitglieds des Entwicklerteams, das aktuell ebenfalls nicht mehr zur Verfügung steht. Das Verfahren befindet sich auf der beiliegenden CD. Es kann aufgrund von Versionsinkompatibilitäten (die benötigte *Tensorflow*-Version ist mit der benötigten *Python*-Version nicht kompatibel) nicht unter *Microsoft Windows* ausgeführt werden. Zur Ausführung der Klassifizerung muss *Theano* als Backend (nicht *Tensorflow*) genutzt werden. Es werden die folgenden spezifischen Versionen von *Python* und der zugehörigen Bibliotheken benötigt:

- Python 2.7
- Keras 1.0.2
- Theano 0.8.0
- scikit-learn 0.13

- Numpy 1.16.0
- pandas 0.24.2
- scipy 1.2.3

Anpassung der Benutzeroberfläche

Die GUI der Anwendung wurde mithilfe von PyQt5[1] erzeugt. Zur Erzeugung wurde der integrierte *Designer* genutzt, durch den eine UI-Datei erzeugt wird, in der grafischen Elemente der GUI gespeichert sind. Zum Verändern der UI-Datei müssen die folgenden Schritte durchgeführt werden:

1. Startbefehl in die Kommandozeile eingeben:
   $ pyqt5-tools designer

2. In dem Pop-up der nun geöffneten Anwendung den *Öffnen...*-Button betätigen und die UI-Datei auswählen.

3. Nach dem Bearbeiten der Datei das *Speichern*-Symbol (Diskette) betätigen.

A.4 Benutzeroberfläche

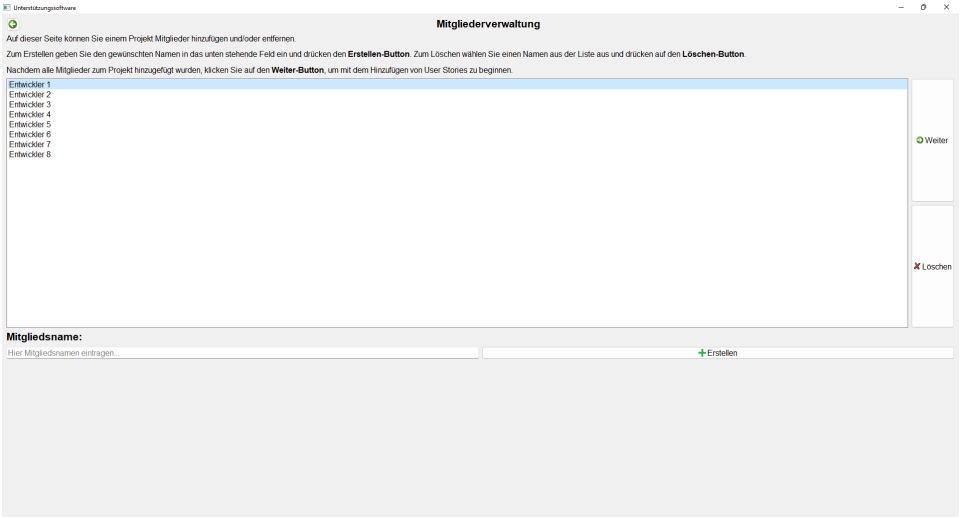


Abbildung A.2: GUI: Mitgliederverwaltung

[1] https://pypi.org/project/PyQt5/

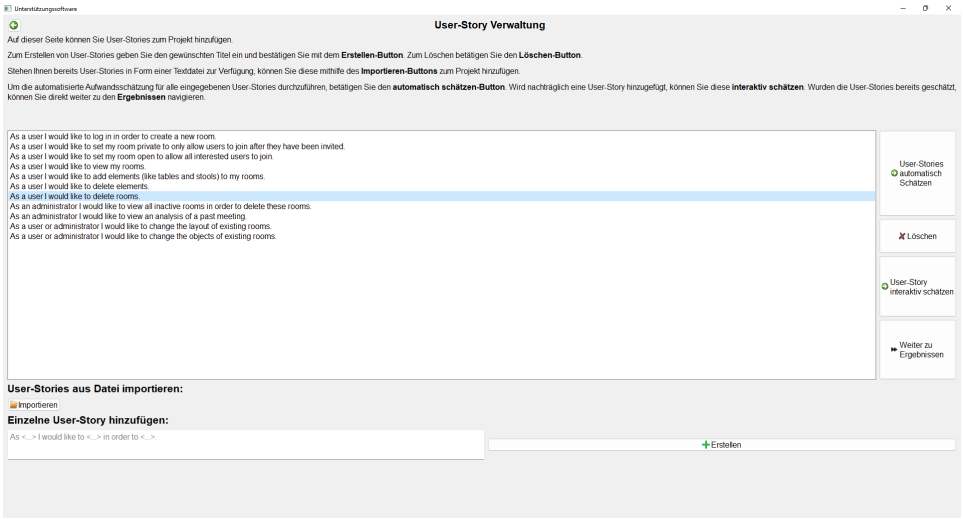


Abbildung A.3: GUI: User-Story Verwaltung

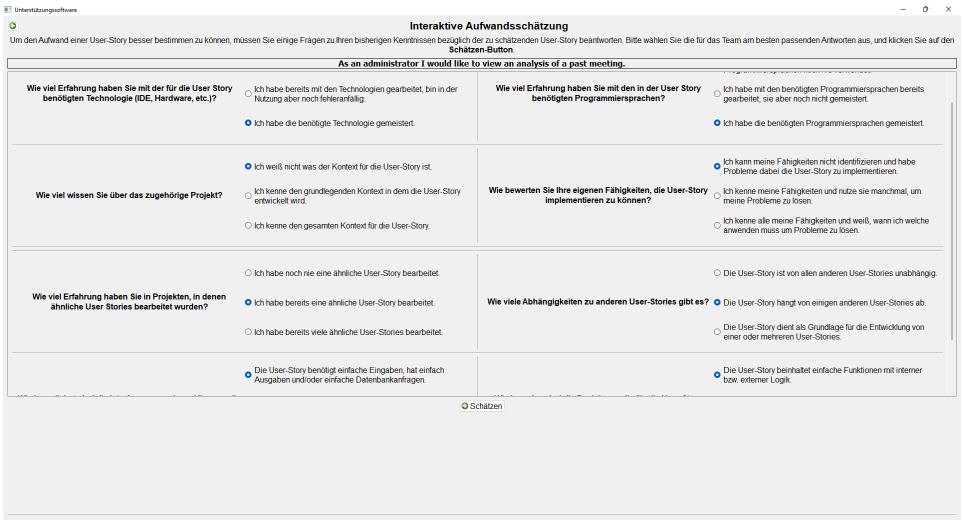


Abbildung A.4: GUI: Interaktive Aufwandsschätzung

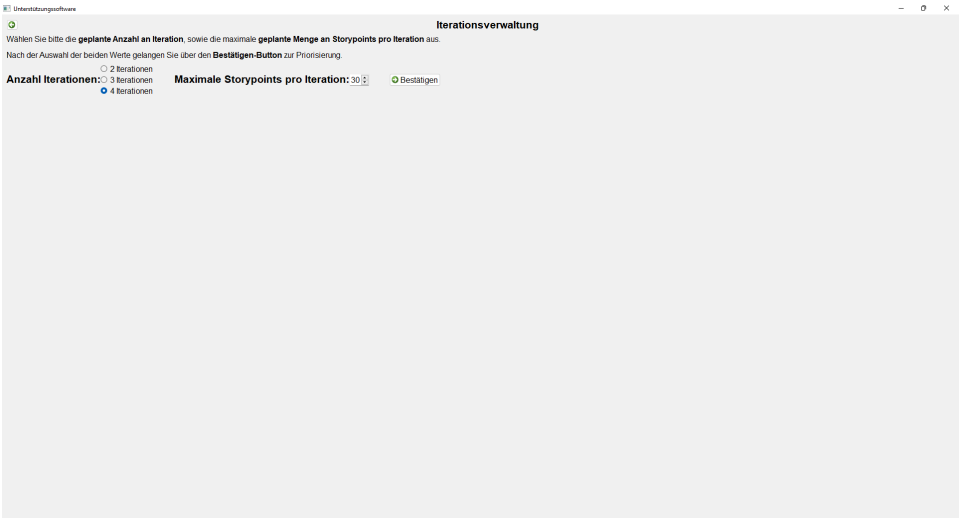


Abbildung A.5: GUI: Iterationsverwaltung

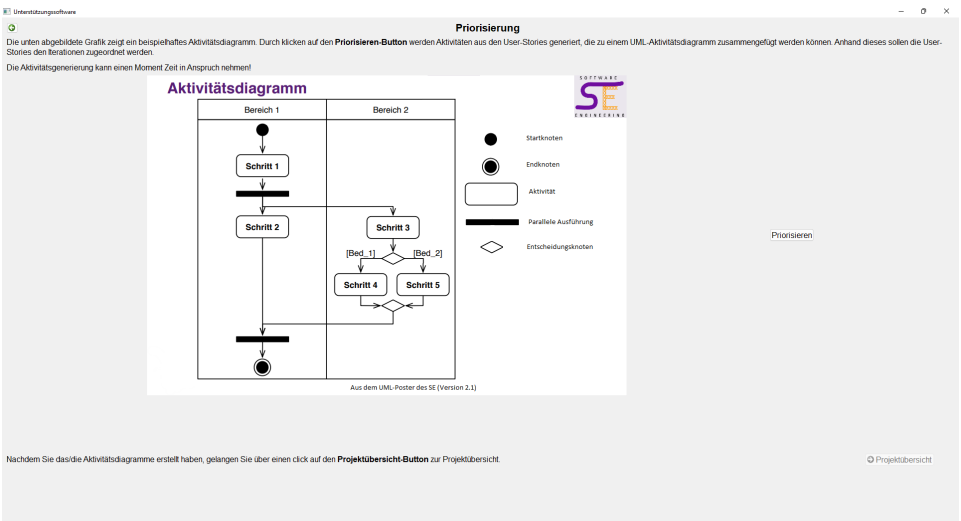


Abbildung A.6: GUI: Priorisierungserläuterung

A.5 Studie

In diesem Abschnitt werden sämtliche zur Studie gehörenden Inhalte abgebildet. Dazu zählen die Rekrutierungsanschreiben, die in der Studie genutzten Informationstexte, die Liste der gestellten Fragen sowie die Einverständniserklärung.

Rekrutierung

Hallo Studies,

im Rahmen meiner Masterarbeit suche ich **Informatikstudierende** für eine Studie. Es geht dabei um die Unterstützung von studentischen Teams in der agilen Software-Entwicklung.

Für die Teilnahme an der Studie solltet ihr über **sehr gute Deutsch- und grundlegende Englischkenntnisse** verfügen, müsst aber **keine weiteren Vorkenntnisse** mitbringen - es sind auch gerne Studierende aus dem Bachelor willkommen, die das **SWP noch nicht** mitgemacht haben.

Während der Studie sollt ihr eine von mir entwickelte Software nutzen, um die Planung für ein Software-Projekt durchzuführen, und eine anschließende Umfrage durchführen. Keine Sorge, ihr müsst **nichts programmieren**!

Die Studie soll im Zeitraum der **letzten beiden Vorlesungswochen (12.07 - 22.07)** in Teams von idealerweise 8 Personen, in Präsenz in den Räumen des SE durchgeführt werden (der genaue Raum wird noch bekannt gegeben).

Die Studiendauer wird in etwa **45 Minuten** betragen.
Für Rückfragen stehe ich euch natürlich gerne zur Verfügung (maximilian.nixdorf@stud.uni-hannover.de).

Bei Interesse könnt ihr euch gerne für alle passenden Termine in der unten verlinkten **Doodle-Umfrage** eintragen. Ich gebe euch so früh wie möglich, spätestens jedoch bis zum 12.07. Bescheid, ob und an welchem Termin die Studie stattfinden wird.

https://doodle.com/meeting/participate/id/aO7GYZGa

Sämtliche erhobenen Daten, die während der Umfrage gesammelt werden, werden anonymisiert!
Die E-Mailadressen und Namen, die bei der Wahl eines Termins in der Doodle-Umfrage angegebene werden, werden lediglich zur Kontaktaufnahme verwendet und nicht an Dritte weitergegeben.

Als Dankeschön gibt es nach der Studie einige Süßigkeiten und es wird ein 20 € Amazongutschein unter allen Teilnehmenden verlost!

Ich würde mich sehr über eure Unterstützung freuen :)

Liebe Grüße

Maximilian

Abbildung A.7: Rekrutierungsanschreiben am Schwarzen Brett

Durchführung

Fragenliste

Die folgende Auflistung zeigt sämtliche Fragen aus dem den Studienteilnehmern vorgelegten Fragebogen. Eine Druckversion des LimeSurvey Online-Fragebogens findet sich auf der beiliegenden CD.

1. Bitte geben Sie Ihre selbst gewählte 4-stellige Zahl ein.
   (Textfeld)
2. Bitte geben Sie Ihr Geschlecht an.
   (männlich, weiblich, divers, keine Antwort)
3. Ist Deutsch Ihre Muttersprache?
   (ja, nein, keine Antwort)
4. In welchem Studiengang sind sie eingeschrieben?
   (Textfeld)
5. In welchem Fachsemester befinden Sie sich?
   (Textfeld)
6. Haben Sie bereits Erfahrungen in anderweitigen Software-Projekten gesammelt? Falls ja, wo?
   (ja, nein, keine Antwort, Kommentarfeld)
7. Wie verständlich war die Nutzung des Prototyps?
   (sehr gut verständlich, gut verständlich, schlecht verständlich, sehr schlecht verständlich, keine Antwort, Kommentarfeld)
8. Wie hilfreich war die automatisierte Aufwandsschätzung bei der Planung des Software-Projekts?
   (sehr hilfreich, hilfreich, weniger hilfreich, überhaupt nicht hilfreich, Kommentarfeld)
9. Wie hilfreich war die interaktive Aufwandsschätzung bei der Planung des Software-Projekts?
   (sehr hilfreich, hilfreich, weniger hilfreich, überhaupt nicht hilfreich, Kommentarfeld)
10. Wie hilfreich war die interaktive Priorisierung bei der Planung des Software-Projekts?
    (sehr hilfreich, hilfreich, weniger hilfreich, überhaupt nicht hilfreich, Kommentarfeld)
11. Wie hilfreich war die automatisierte Rollenzuweisung bei der Planung des Software-Projekts?
    (sehr hilfreich, hilfreich, weniger hilfreich, überhaupt nicht hilfreich, Kommentarfeld)

12. War die softwaregestützte Aufwandsschätzung, im Vergleich zu der zu Beginn durchgeführten, manuellen Aufwandsschätzung, eine Unterstützung?
(ja, eher ja, eher nein, nein, Kommentarfeld)

13. Würden Sie unerfahrenen Teams die Verwendung der Software empfehlen? Bitte begründen Sie Ihre Entscheidung.
(ja, eher ja, eher nein, nein, Kommentarfeld)

14. Haben Sie Anmerkungen, Vorschläge, Wünsche oder Verbesserungsvorschläge für den Prototypen?
(Textfeld)

15. Haben Sie Ideen, was unerfahrene Teams bei der Softwareentwicklung unterstützen könnte?
(Textfeld)

Einverständniserklärung zur Studie: “Unterstützung von unerfahrenen Teams in der agilen Software-Entwicklung”

Die Studie wird von Maximilian Nixdorf, einem Studenten der Leibniz Universität Hannover im Rahmen seiner Masterarbeit durchgeführt.

Ich nehme freiwillig an der Studie teil. Die Studienteilnahme wird nicht vergütet. Ich habe das Recht, die Teilnahme an der Studie jederzeit und ohne Angabe von Gründen abzubrechen.

Diese Einwilligung ist freiwillig. Ich kann sie ohne Angabe von Gründen verweigern, ohne dass ich dadurch Nachteile zu befürchten habe.

Datenerfassung, -speicherung und -verwendung

Ich wurde darüber informiert, dass die Angaben in dem (Online-)Fragebogen erfasst, elektronisch für einen unbegrenzten Zeitraum gespeichert und zur Auswertung der Studie herangezogen werden. Die erfassten Daten werden alleinig für die wissenschaftliche Forschung genutzt, ausschließlich anonymisiert ausgewertet und nur für das Fachgebiet Software Engineering der Leibniz Universität Hannover zugänglich sein. Die Daten, sowie die daraus basierenden Auswertungen, werden im Rahmen der Masterarbeit von Maximilian Nixdorf in anonymisierter Form veröffentlicht. Die Daten werden nur im Rahmen dieser Masterarbeit ausgewertet, können aber auch für wissenschaftliche Publikationen des Fachgebiets Software Engineering herangezogen werden.

Sie haben gemäß Datenschutz gegenüber dem Informationsträger das Recht auf Auskunft, Berichtigung, sowie Löschung ihrer personenbezogenen Daten. Sie können diese Einwilligungserklärung jederzeit schriftlich widerrufen. Nach erfolgtem Widerruf werden Ihre personenbezogenen Daten gelöscht und ab diesem Zeitpunkt für keine weiteren Publikationen mehr verwendet.

Ich bin mit der Einverständniserklärung einverstanden und nehme freiwillig an dieser Studie teil. Meine Studienteilnahme wird nicht vergütet.

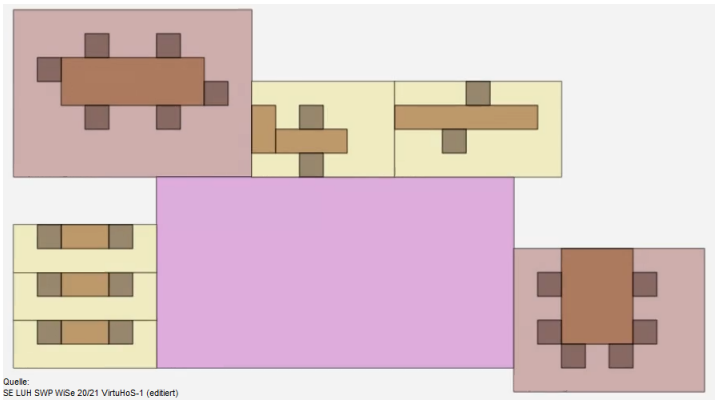
Nachname, Vorname selbst gewählte 4-stellige Zahl

Ort, Datum, Unterschrift

Abbildung A.8: Einverständniserklärung

Folgendes Projekt dient als Grundlage der zur Verfügung gestellten User-Stories:

Ich leite die WorkFromHome GmbH und benötige eine Software, die das Homeoffice erleichtert. Aufgrund der Corona-Pandemie arbeiten unsere Mitarbeiter nun fast ausschließlich im Homeoffice und müssen zur Koordination von Meetings E-Mails nutzen. Dies führt zu überfüllten Postfächern und es ist bereits vorgekommen, dass wichtige E-Mails nicht beantwortet worden, weil sie im Postfach untergegangen sind.
Ich hätte gerne eine virtuelle Kopie unserer ehemaligen Büroräume, in denen sichtbar ist, welcher Nutzer sich aktuell in ihnen befindet. Zudem soll ersichtlich sein, ob die jeweiligen Personen bereit für Gespräche (gegebenenfalls über eine Anbindung an bestehende Software wie BBB, Skype oder Teams) sind, und wie viele Personen in einem Raum Platz haben. Die Nutzer sollen ihre Kollegen in Räume einladen können, oder ihnen Anfragen schicken, um die Räume ihrer Kollegen betreten zu dürfen. Zudem soll eine Option zur Webcamfreigabe integriert werden.
Der Versammlungsraum soll Platz für beliebig viele Teilnehmer bieten und soll nur von Administratoren gebucht werden dürfen. Es wäre sinnvoll, wenn alle Mitarbeiter vor einem solchen Meeting eine Erinnerung erhalten würden, da es ab und zu vorkommt, dass einige Teilnehmer zu spät erscheinen.



Quelle:
SE LUH SWP WiSe 20/21 VirtuHoS-1 (editiert)

Schätzung ohne Software

Nun würde ich euch bitten, die folgenden zwei User-Stories im Team zu schätzen: sprich den benötigten Aufwand in Story-Points angeben und einen Bearbeiter für die User-Story auswählen - am besten jemanden, der über genügend Kenntnisse verfügt.

As a user I would like to log in in order to create a new room.
As an administrator I would like to view an analysis of the past meetings.

Abbildung A.9: Studienerläuterung

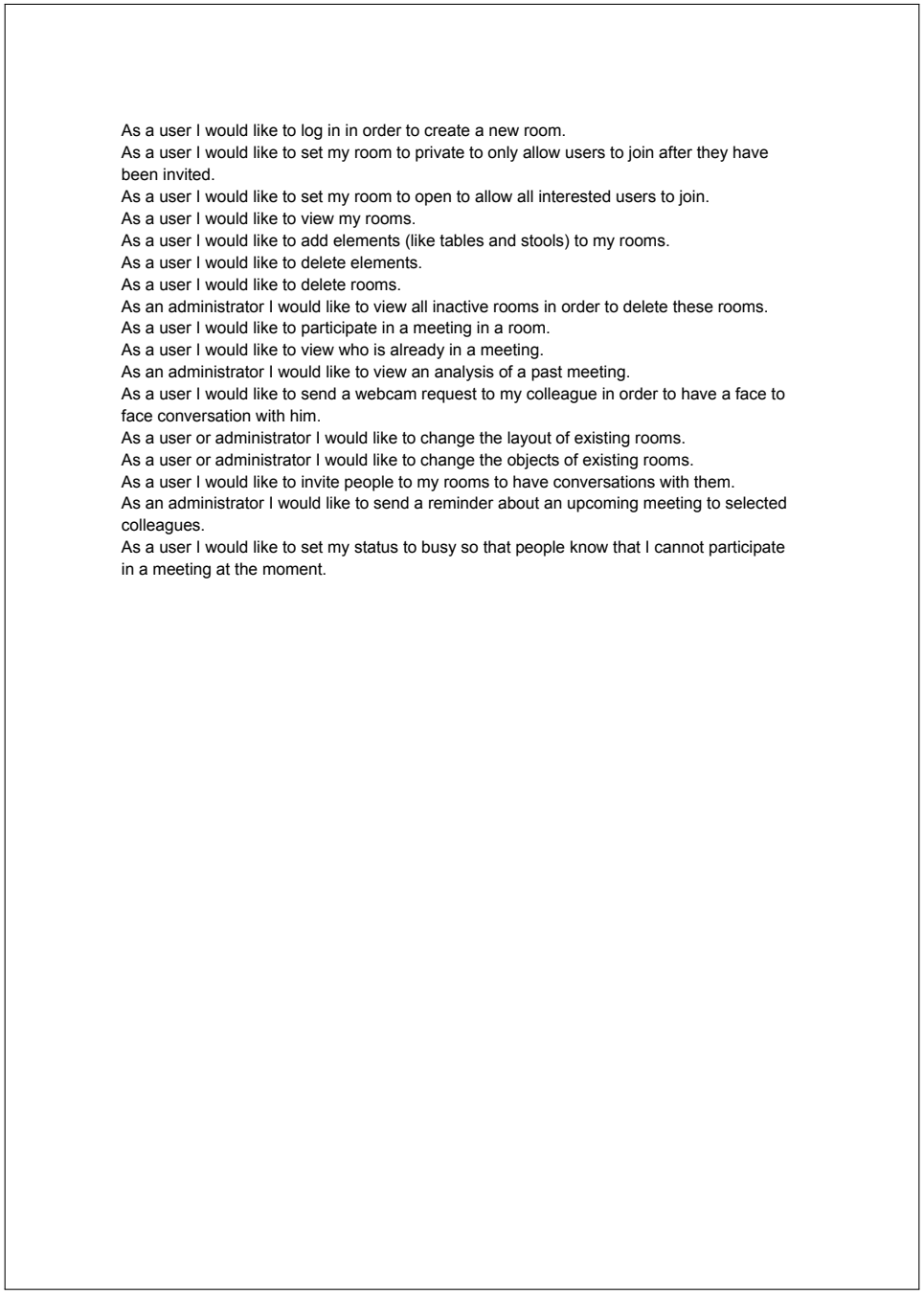


Abbildung A.10: User-Stories zum Schätzen in der Studie

Literaturverzeichnis

[1] B. Alsaadi and K. Saeedi. Data-driven effort estimation techniques of agile user stories: a systematic literature review. *Artificial Intelligence Review*, 2022.

[2] W. Aslam and F. Ijaz. A quantitative framework for task allocation in distributed agile software development. *IEEE Access*, 6:15380–15390, 2018.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[4] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies. A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, 45(7):637–656, 2018.

[5] M. Cohn. *Agile Estimation and Planning*, chapter 6, pages 163–199. John Wiley & Sons, Ltd, 2006.

[6] K. Conboy, S. Coyle, X. Wang, and M. Pikkarainen. People over process: Key challenges in agile development. *IEEE Software*, 28(4):48–57, 2011.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[8] S. Dragicevic, S. Celar, and M. Turic. Bayesian network model for task effort estimation in agile software development. *Journal of systems and software*, 127:109–119, 2017.

[9] M. Durán, R. Juárez-Ramírez, S. Jiménez, and C. Tona. User story estimation based on the complexity decomposition using bayesian networks. *Programming and Computer Software*, 46(8):569–583, 2020.

[10] M. A. A. Elsood, H. A. Hefny, and E. S. Nasr. A goal-based technique for requirements prioritization. In *2014 9th International Conference on Informatics and Systems*, pages SW–18–SW–24, 2014.

[11] B. S. Everitt, S. Landau, M. Leese, and D. Stahl. *Hierarchical Clustering*, chapter 4, pages 71–110. John Wiley & Sons, Ltd, 2011.

[12] E. M. D. B. Fávero, D. Casanova, and A. R. Pimentel. Se3m: A model for software effort estimation using pre-trained embedding models. *Information and Software Technology*, 147:106886, 2022.

[13] J. Frame. *The New Project Management: Tools for an Age of Rapid Change, Complexity, and Other Business Realities*. Jossey-Bass business & management series. Wiley, 2002.

[14] S. Gulia and T. Choudhury. An efficient automated design to generate uml diagram from natural language specifications. In *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pages 641–648, 2016.

[15] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[17] G. Hoff, A. Fruhling, and K. Ward. Requirement prioritization decision factors for agile development environments. volume 5, page 66, 2008.

[18] S. Hudda, R. Mahajan, and S. Chopra. Prioritization of user-stories in agile environment. *Indian Journal of Science and Technology*, 9(10.17485), 2016.

[19] A. Idri, M. Hosni, and A. Abran. Systematic literature review of ensemble effort estimation. *Journal of Systems and Software*, 118:151–175, 2016.

[20] J. H. W. Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[21] D. Jurafsky and J. H. Martin. *Speech & language processing*. 2021.

[22] M. Kuhrmann, J. Muench, and J. Klunder. Hacking or engineering? towards an extended entrepreneurial software engineering model. In *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering*, ICSSP'22, page 66–76, New York, NY, USA, 2022. Association for Computing Machinery.

[23] T. K. Landauer, D. Laham, B. Rehder, and M. E. Schreiner. How well can passage meaning be derived without using word order? a comparison of latent semantic analysis and humans. 1997.

[24] J. Lin, H. Yu, Z. Shen, and C. Miao. Studying task allocation decisions of novice agile teams with data from agile project management tools. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, page 689–694, New York, NY, USA, 2014. Association for Computing Machinery.

[25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[26] J. López-Martínez, A. Ramírez-Noriega, R. Juárez-Ramírez, G. Licea, and S. Jiménez. User stories complexity estimation using bayesian networks for inexperienced developers. *Cluster computing*, 21(1):715–728, 2018.

[27] D. K. Mak and P. B. Kruchten. Task coordination in an agile distributed software development environment. In *2006 Canadian Conference on Electrical and Computer Engineering*, pages 606–611, 2006.

[28] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[29] N. Nachar. The mann-whitney u: A test for assessing whether two independent samples come from the same distribution. *Tutorials in Quantitative Methods for Psychology*, 4, 2008.

[30] J. Nielsen. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, page 152–158, New York, NY, USA, 1994. Association for Computing Machinery.

[31] G. Papadopoulos. Moving from traditional to agile software development methodologies also on large, distributed projects. *Procedia - Social and Behavioral Sciences*, 175:455–463, 2015. Proceedings of the 3rd International Conference on Strategic Innovative Marketing (IC-SIM 2014).

[32] L. Prechelt. *Early Stopping — But When?*, pages 53–67. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[33] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.

[34] A. Ramirez-Noriega, R. Juarez-Ramirez, R. Navarro, and J. Lopez-Martinez. Using bayesian networks to obtain the task's parameters for schedule planning in scrum. In *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pages 167–174, 2016.

[35] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[36] S. Sachdeva, A. Arya, P. Paygude, S. Chaudhary, and S. Idate. Prioritizing user requirements for agile software development. In *2018 International Conference On Advances in Communication and Computing Technology (ICACCT)*, pages 495–498. IEEE, 2018.

[37] F. Sarro and A. Petrozziello. Linear programming as a baseline for software effort estimation. *ACM Trans. Softw. Eng. Methodol.*, 27(3), 2018.

[38] E. Scott and D. Pfahl. Using developers' features to estimate story points. In *Proceedings of the 2018 International Conference on Software and System Process*, pages 106–110, 2018.

[39] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed. Taskallocator: A recommendation approach for role-based tasks allocation in agile software development. In *2021 IEEE/ACM Joint 15th International Conference on Software and System Processes (ICSSP) and 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE)*, pages 39–49, 2021.

[40] M. Shepperd and S. MacDonell. Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8):820–827, 2012. Special Issue: Voice of the Editorial Board.

[41] B. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, and N. Elmqvist. *Designing the User Interface: Strategies for Effective Human-Computer Interaction: Sixth Edition.* Pearson, 2016.

[42] A. Sillitti and G. Succi. *Requirements Engineering for Agile Methods*, pages 309–326. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[43] Á. Szőke. Decision support for iteration scheduling in agile environments. In F. Bomarius, M. Oivo, P. Jaring, and P. Abrahamsson, editors, *Product-Focused Software Process Improvement*, pages 156–170, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[44] V. Tawosi, A. Alsubaihin, and F. Sarro. Investigating the effectiveness of clustering for story point estimation. In *SANER*. IEEE, 2022.

[45] V. Tawosi, R. Moussa, and F. Sarro. Deep learning for agile effort estimation have we solved the problem yet?, 2022.

[46] V. Tawosi, R. Moussa, and F. Sarro. On the relationship between story points and development effort in agile open-source software, 2022.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[48] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Computer Science. Springer Berlin Heidelberg, 2012.

[49] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[50] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.