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

Quantitative Analyse von Einflussfaktoren auf die Lead Time mit Jira-Daten

Masterarbeit

im Studiengang Informatik

von

Daniel Wahlmann

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüferin: Dr. Jil Klünder
Betreuer: Nils Prenner

Hannover, 12. April 2021

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. April 2021

_____

Daniel Wahlmann

Kurzfassung

In der modernen Softwareentwicklung werden überwiegend agile Entwicklungsmethoden eingesetzt. Auch in der Automobilindustrie entwickeln sich die Prozesse zunehmend in Richtung agiler Methoden. Um die Effektivität dieser Prozesse zu messen, ist die Lead Time eine wichtige Metrik. Zur Verbesserung der Lead Time ist es notwendig, die Faktoren zu ermitteln, welche sie beeinflussen. In dieser Arbeit wurde eine Webanwendung entwickelt, mit der Jira-Daten untersucht werden können, um diese Einflussfaktoren zu ermitteln. Die Anwendung wurde genutzt, um die Daten des Projekts ConnectFleet von Volkswagen Nutzfahrzeuge zu analysieren. Zusätzlich wurde eine Übersicht der offenen Epics implementiert, die bei der Erkennung von Inaktivität und der Reduzierung der Lead Time helfen soll. Es wurden 20 Metriken auf ihren Einfluss untersucht, davon zeigten keine starke Korrelationen mit der Lead Time. Die größten Zusammenhänge haben sich bei der Anzahl der Autoren von Stories, der Anzahl von Änderungen an Stories und vor allem der Rolle des Erstellers einer Story gezeigt. So scheinen Stories, die von Personen erstellt wurden, die der Entwicklung nahe stehen, wie Entwickler oder Solution Engineers, im Vergleich schneller bearbeitet zu werden. Solche, die von weiter davon entfernten Personen erstellt wurden, wie dem Anforderungsmanagement, werden eher langsamer bearbeitet.

Abstract

Quantitative analysis of influencing factors on the lead time using Jira data

In modern software development, agile development methods are predominantly used. In the automotive industry, too, processes are increasingly developing in the direction of agile methods. The lead time is an important metric for measuring the effectiveness of these processes. To improve the lead time it is necessary to identify the factors that influence it. In this work, a web application was developed to explore Jira data to identify these influencing factors. The application was used to analyze data from the ConnectFleet project of Volkswagen Nutzfahrzeuge. In addition, an overview of open epics was implemented to help identify inactivity and reduce the lead time. Twenty metrics were examined for their impact, none of which showed strong correlations with the lead time. The strongest correlations were shown for the number of authors of a story, the number of changes to a story, and most significantly, the role of the creator of a story. Stories created by people close to the development, such as developers or solution engineers, seem to be processed faster in comparison. Stories created by people further away, such as requirements management, tend to be processed more slowly.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ein großer Teil aller Softwareprojekte werden nicht oder nur eingeschränkt erfolgreich abgeschlossen [4]. Besonders in traditionellen Softwareentwicklungsprozessen wie Wasserfall- [36] oder V-Modell [8] liegen oft Jahre zwischen Anforderungserhebung und Auslieferung der Software. Das führt dazu, dass sich die Kundenanforderungen während dieser Zeit ändern und das Produkt nicht mehr den Kundenwünschen entspricht. Ein weiteres Problem bei diesen Prozessen ist es, den Aufwand und die benötigten Ressourcen für die Projekte im Voraus zu schätzen, wodurch es häufig zu verspäteter Fertigstellung der Software kommt.

Um diese Probleme anzugehen, sind in den 1990er Jahren agile Methoden wie Extreme Programming [1] und Scrum [38] entstanden, deren gemeinsame Werte 2001 im agilen Manifest [2] festgehalten wurden. Diese Methoden setzen darauf, die Anforderungen nicht zu Beginn des Projekts festzulegen, sondern sie während der Entwicklung ständig an die sich ändernden Kundenwünsche anzupassen. Ein wesentliches Merkmal, das dies ermöglicht, ist das iterative Vorgehensmodell dieser Prozesse. Dadurch können vom Kunden priorisierte Features früh implementiert werden. Da der Kunde die Software früh testen kann, werden auch fehlerhafte oder geänderte Anforderungen gefunden, auf die flexibel reagiert werden können, ohne das gesamte Projekt zu verzögern.

In der Regel ist die Entwicklung nicht mit der Auslieferung der Software – der Version 1.0 – abgeschlossen, denn neben der Fehlerbehebung werden auch neue Funktionen entwickelt. Traditionell werden mehrere dieser Funktionen in mehr oder weniger regelmäßigen Intervallen als Patch ausgeliefert. Bei agilen Prozessen werden die neuen Versionen regelmäßig – z. B. am Ende jedes Sprints – ausgeliefert. Durch den Einsatz von Techniken wie *kontinuierlicher Bereitstellung* werden die Intervalle zwischen den Veröffentlichungen weiter reduziert, so dass eine bis Tausende Auslieferungen pro Tag möglich sind [37].

Die Automobilindustrie hat typischerweise lange Entwicklungszyklen. Es werden komplette Systeme – Automobile – entwickelt, bei denen die Hardware und die Software parallel umgesetzt werden. Dabei werden meist starre Entwicklungsprozesse eingesetzt, die keine agilen Methoden vorsehen [14].

Aber auch bei Automobilherstellern wächst der Druck, neue Technologien schneller auf den Markt zu bringen. Die Software ist ein wichtiger Teil davon, da moderne Autos komplett davon gesteuert werden. Dadurch ist es aber auch möglich, nachträglich neue Funktionen

bereitzustellen. Um das zu ermöglichen, werden zunehmend agile Prozesse erprobt und eingesetzt.

1.1 Problemstellung

Um ständig auf dem neuesten Stand zu sein, ist eine kontinuierliche Weiterentwicklung der Software nötig. Das erfordert auch, dass fortlaufend Anforderungen erhoben und umgesetzt werden.

Die Unternehmen sind stetig bemüht, die Entwicklungsprozesse zu verbessern. Dafür müssen laufend Metriken erhoben werden, mit denen ein Prozess evaluiert werden kann. Eine wichtige Metrik dabei ist die *Lead Time*, d. h. die Zeit von Anforderungserhebung bis zur Auslieferung des Features.

Nur die Metriken zu kennen reicht aber nicht aus, um die Entwicklungsprozesse zu verbessern. Zusätzlich ist es nötig zu wissen, welche Faktoren diese Metriken wie beeinflussen.

Gängige Entwicklungstools wie Jira [15] bieten zwar einige Metriken, die beschränken sich aber meist auf typische agile Metriken wie Burndown-Charts, Velocity oder Testabdeckung. Die für die kontinuierliche Auslieferung wichtige Lead Time wird nicht ermittelt, ebenso wenig mögliche Einflussfaktoren darauf.

1.2 Lösungsansatz

In dieser Arbeit soll untersucht werden, welche Faktoren die Lead Time beeinflussen. Dazu wird eine Webanwendung entwickelt, die auf einem vorhandenen Prototypen [18] basiert. Zusätzlich zum vorhandenen Dashboard wird die Anwendung um Funktionen zur statistischen Auswertung der Daten und zur grafischen Darstellung dieser Auswertung erweitert. Diese Anwendung wird genutzt, um die Jira-Daten des Projekts *ConnectFleet* von Volkswagen Nutzfahrzeuge (VWN) zu analysieren und mögliche Einflussfaktoren auf die Lead Time zu identifizieren. Zusätzlich wurde die Anwendung um ein Tool erweitert, mit dem der Fortschritt von Epics dargestellt werden kann. Damit sollen mögliche Probleme bei der rechtzeitigen Fertigstellung von Stories, aus denen die Epics bestehen, sichtbar gemacht werden.

1.3 Struktur der Arbeit

Diese Arbeit ist wie folgt strukturiert. In Kapitel 2 werden die für das Verständnis dieser Arbeit nötigen Grundlagen und bei der Implementierung genutzten Technologien erläutert. Anschließend werden in Kapitel 3 verwandte Arbeiten vorgestellt, die sich mit der Lead Time in der Softwareentwicklung befasst haben. In Kapitel 4 wird die Methodik erläutert,

mit der in dieser Arbeit vorgegangen wurde. Anschließend wird in Kapitel 5 die Implementierung der Webanwendung vorgestellt. Im folgenden Kapitel 6 werden die Ergebnisse der statistischen Auswertungen präsentiert. Diese Ergebnisse werden in Kapitel 7 diskutiert, bevor die Arbeit mit einer Zusammenfassung und einem Ausblick in Kapitel 8 abschließt.

Kapitel 2

Grundlagen

Für die Auswertung der im Rahmen dieser Arbeit gesammelten Daten werden statistische Methoden genutzt, die in Abschnitt 2.1 erläutert werden. Des Weiteren werden in Abschnitt 2.2 die Technologien vorgestellt, die bei der Implementierung des Prototypen genutzt wurden.

2.1 Statistische Methoden

Softwareprojekte erzeugen viele Daten, besonders wenn Projektmanagementtools wie Jira benutzt werden. Diese Daten müssen analysiert und interpretiert werden, um daraus Schlüsse ziehen zu können. Dazu benötigt man statistische Methoden. Der erste Schritt bei der Analyse von Daten ist es, einen Überblick über diese zu gewinnen, indem man mit *beschreibender Statistik* Lage-, Streuungs- und Zusammenhangsmaße bestimmt und visualisiert. Anschließend werden die Daten mit *Hypothesentests* analysiert, mit denen statistisch signifikante Aussagen über die Daten getroffen werden können. Dieser Abschnitt folgt dabei der Darstellung von Wohlin et al. [48].

2.1.1 Beschreibende Statistik

Nach dem Sammeln experimenteller Daten kann beschreibende Statistik genutzt werden, um interessante Aspekte des Datensatzes zu beschreiben und grafisch zu präsentieren. Das Ziel von beschreibender Statistik ist es, einen Überblick darüber zu gewinnen, wie die Daten verteilt sind. Beschreibende Statistik wird häufig vor Hypothesentests genutzt, um die Daten besser zu verstehen und um abnormale oder fehlerhafte Datenpunkte (so genannte *Ausreißer*) zu identifizieren.

In diesem Abschnitt werden einige Maße der beschreibenden Statistik und Techniken zur grafischen Darstellung von Daten vorgestellt, die in dieser Arbeit genutzt werden.

Lagemaße

Lagemaße werden genutzt, um die „Mitte“ eines Datensatzes zu bestimmen. Dieser „Mittelpunkt“ wird häufig als Durchschnitt oder Mittelwert bezeichnet und kann als *Erwar-*

tungswert der stochastischen Variable verstanden werden, von der die Datenpunkte des Datensatzes als Stichprobe entnommen wurden.

Im Folgenden wird angenommen, dass es n Datenpunkte $x_1, \ldots, x_n$ gibt, die aus einer stochastischen Variablen gezogen wurden. Das *(arithmetische) Mittel*, bezeichnet als $\bar{x}$, wird dann berechnet durch

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{2.1}$$

Zum Beispiel ist für den Datensatz $(1, 2, 2, 6, 7)$ der Mittelwert $\bar{x} = 3{,}6$.

Der *Median*, bezeichnet als $\tilde{x}$, repräsentiert den mittleren Wert des Datensatzes. Das bedeutet, dass die Anzahl der Datenpunkte, die größer als der Median sind, gleich der Anzahl der Datenpunkte ist, die kleiner als der Median sind. Der Median wird berechnet, indem die Datenpunkte aufsteigend sortiert werden und der mittlere Datenpunkt ausgewählt wird. Wenn n ungerade ist, ist dies wohldefiniert. Bei geradem n wird der Median als das arithmetische Mittel der beiden mittleren Werte definiert. Zum Beispiel ist der Median für den Datensatz $(1, 2, 2, 6, 7)$ $\tilde{x} = 2$.

Der Median ist ein Spezialfall des *Perzentils*, und zwar das 50 %-Perzentil, bezeichnet mit $x_{0,5}$. In einem Datensatz liegen 50 % der Werte unter $x_{0,5}$. Im Allgemeinen bezeichnet x_p das Perzentil, bei dem $p \cdot 100\,\%$ der Datenpunkte unter diesem Wert liegen. Weitere Spezialfälle sind die *Quartile*, wobei das erste Quartil $x_{0,25}$, das zweite Quartil dem Median und das dritte Quartil $x_{0,75}$ entspricht.

Der Mittelwert und der Median können zusätzlich Hinweise darauf geben, ob die Verteilung symmetrisch ist. Bei symmetrischen Verteilungen, wie z. B. $(2, 6, 9, 12, 16)$, haben Mittelwert und Median denselben Wert, hier $\bar{x} = \tilde{x} = 9$. Bei asymmetrischen Verteilungen, wie z. B. $(1, 2, 2, 3, 4, 12)$, können sich die beiden Maße unterscheiden, hier ist der Mittelwert $\bar{x} = 4$ und der Median $\tilde{x} = 2{,}5$. Außerdem ist der Mittelwert empfindlicher gegen Ausreißer als der Median.

Streuungsmaße

Zusätzlich zu den aus den Lagemaßen gewonnen Informationen ist es wichtig, die Verteilung der Datenpunkte zu kennen. Mit den Streuungsmaßen kann zusätzlich der Grad der Abweichung von den Lagemaßen bestimmt werden. Ein gebräuchliches Maß für die Streuung ist die (Stichproben-)*Varianz*, bezeichnet mit s^2, die berechnet wird mit

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 . \tag{2.2}$$

Die Division durch $n-1$ statt durch n verleiht der Varianz einige erwünschte Eigenschaften. Vor allem kann die Stichprobenvarianz damit als eine Schätzung der Varianz der betrachte-

ten stochastischen Variable genutzt werden. Im Falle des Datensatzes $(1, 2, 2, 6, 7)$ beträgt die Varianz $s^2 = 7{,}3$.

Die *Standardabweichung*, bezeichnet mit s, wird als die Quadratwurzel der Varianz definiert durch

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}. \tag{2.3}$$

Die Standardabweichung wird oft der Varianz vorgezogen, da sie die gleiche Einheit hat wie die betrachteten Daten. Für Zeiten in Tagen hätte z. B. die Varianz die Einheit d^2, während die Standardabweichung die Einheit d hätte. Für den Datensatz $(1, 2, 2, 6, 7)$ beträgt die Standardabweichung beispielsweise $s \approx 2{,}7$.

Varianz und Standardabweichung beziehen sich beide auf das arithmetische Mittel. Ein Streuungsmaß für den Median ist der Interquartilabstand, bezeichnet mit *IQR*, dieser wird berechnet durch

$$IQR = x_{0{,}75} - x_{0{,}25}. \tag{2.4}$$

Der Interquartilabstand ist also der Abstand vom ersten zum dritten Quartil, d. h. es befinden sich 50 % der Datenpunkte innerhalb des Interquartilabstands. Genau wie der Median ist er unempfindlich gegen Ausreißer. Für den Datensatz $(1, 2, 3, 3, 5, 7, 9, 10)$ ist zum Beispiel der Interquartilabstand $IQR = 8 - 2{,}5 = 5{,}5$, damit liegen die Datenpunkte $(3, 3, 5, 7)$ innerhalb des Interquartilabstands.

Zusammenhangsmaße

Für Datensätze, die aus paarweise zusammengehörigen Werten (x_i, y_i) von zwei Zufallsvariablen X und Y bestehen, ist es interessant, die Abhängigkeit der beiden Variablen zu untersuchen.

Ein Maß dafür, wie stark zwei Datensätze x_i und y_i zusammen variieren, ist die *Kovarianz*, bezeichnet mit c_{xy}, definiert durch

$$c_{xy} = \frac{S_{xy}}{n-1}, \tag{2.5}$$

dabei gilt

$$S_{xy} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = \left(\sum_{i=1}^{n} x_i y_i\right) - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right). \tag{2.6}$$

Die Kovarianz hängt von der Varianz der einzelnen Variablen ab. Um die Abhängigkeiten verschiedener Variablen vergleichen zu können, wird die Kovarianz mit den Standardab-

weichungen von x_i und y_i normalisiert, wodurch man den *Korrelationskoeffizienten* r erhält, gegeben durch

$$r = \frac{c_{xy}}{s_x s_y} = \frac{S_{xy}}{\sqrt{S_{xx} S_{yy}}} = \frac{\left(n \sum_{i=1}^{n} x_i y_i\right) - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sqrt{\left(n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right)\left(n \sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right)}}. \tag{2.7}$$

Der Korrelationskoeffizient liegt zwischen -1 und +1 und wenn es keine Korrelation gibt, gilt $r = 0$. Die Umkehrung gilt allerdings nicht, d. h. der Korrelationskoeffizient setzt einen linearen Zusammenhang der Daten voraus. Wenn die Daten $r = 0$ zeigen, kann aber auch eine nichtlineare Korrelation vorliegen, selbst wenn sie nicht linear korreliert sind.

Eine Möglichkeit, auch nichtlineare Korrelationen zu bestimmen, ist der *Rangkorrelationskoeffizient* r_s. Der Rangkorrelationskoeffizient wird genauso berechnet wie der Korrelationskoeffizient, allerdings werden die Werte der Daten durch ihren Rang, d. h. ihre Position nachdem der Datensatz sortiert wurde, ersetzt. Auch der Rangkorrelationskoeffizient setzt allerdings voraus, dass die Daten monoton sind, d. h. die Daten können auch korreliert sein, wenn der r_s-Wert nahe Null ist.

Für die Stärke der Korrelation werden in dieser Arbeit die Richtwerte aus Tabelle 2.1 verwendet. Der r-Wert allein reicht aber nicht aus, um Aussagen über die Korrelation treffen zu können. Zusätzlich muss die Aussagekraft dieses Wertes beurteilt werden können. Dazu wird ein t-Test (siehe Abschnitt 2.1.2) für den Korrelationswert durchgeführt. Der benötigte t-Wert wird berechnet mit [6]

$$t = |r| \sqrt{\frac{n-2}{1-r^2}}. \tag{2.8}$$

Tabelle 2.1: Richtwerte für die Beurteilung der Stärke der Korrelation berechnet mit dem Korrelationskoeffizienten. Die gleichen Werte gelten auch für der Rangkorrelationskoeffizienten.

Korrelationskoeffizient	Interpretation
$\lvert r\rvert < 0{,}2$	sehr schwache Korrelation
$0{,}2 \leq \lvert r\rvert < 0{,}4$	schwache Korrelation
$0{,}4 \leq \lvert r\rvert < 0{,}6$	mittlere Korrelation
$0{,}6 \leq \lvert r\rvert$	starke Korrelation

Für den Datensatz

$$X = ((2,3),(3,5),(4,8),(6,6),(8,7),(8,12),(10,14),(13,12),(13,14),(16,19))$$

(siehe Abbildung 2.1) ist der Korrelationskoeffizient $r(X) = 0{,}916$ und der Rangkorrelationskoeffizient $r_s(X) = 0{,}911$, also zeigen beide Koeffizienten eine ähnlich starke Korrelation. Allerdings gilt für den Datensatz

$$Y = ((2,2),(6,1),(10,1),(14,2),(18,2),(22,4),(25,6),(26,14),(27,20),(28,35))$$

(ebenfalls in Abbildung 2.1) $r(Y) = 0{,}696$ und $r_s(Y) = 0{,}923$, die Korrelation ist also stärker als der (lineare) Korrelationskoeffizient alleine vermuten lässt.

Außer zu wissen, ob es eine Korrelation gibt und wie stark diese ist, ist es auch interessant zu untersuchen, welche Form dieser Zusammenhang annimmt. Angenommen, es besteht ein Zusammenhang zwischen X und Y, dann gibt es eine Funktion $y = f(x)$, die diesen Zusammenhang beschreibt. Wenn diese Funktion linear ist, also durch $y = \alpha + \beta x$, mit $\alpha, \beta \in \mathbb{R}$ ausgedrückt werden kann, dann kann *lineare Regression* genutzt werden, um diese Funktion zu bestimmen. Dabei wird eine Gerade so durch die Punkte gelegt, dass die Summe der quadratischen Abweichungen davon minimiert wird. Mit der Summe aus Gleichung 2.6 kann die Regressionsgerade $y = \bar{y} + \beta(x - \bar{x})$ bestimmt werden, wobei die Steigung durch $\beta = S_{xy} \,/\, S_{xx}$ und der Schnittpunkt mit der x-Achse durch $\alpha = \bar{y} - \beta\bar{x}$ gegeben sind.

Grafische Darstellungen

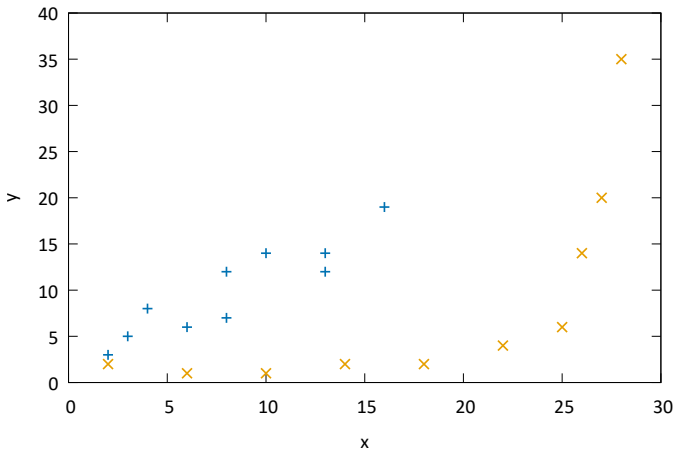

Abbildung 2.1: Beispiel für einen Scatterplot. Dargestellt sind die Datensätze aus den Beispielen für die Korrelation in Abschnitt 2.1.1.

Bei der Betrachtung eines Datensatzes sollten die bisher beschriebenen quantitativen Maße mit grafischen Darstellungen ergänzt werden. Graphen eignen sich gut, die Daten zu illustrieren und einen Überblick über den Datensatz zu gewinnen.

Ein geeigneter Graph für paarweise Daten (x_i, y_i) ist ein *Streudiagramm*, auch *Scatterplot*, bei dem die Daten als Punkte in zwei Dimensionen dargestellt werden. Ein Beispiel für ein Streudiagramm ist in Abbildung 2.1 dargestellt. Scatterplots sind gut dazu geeignet, Abhängigkeiten zwischen Variablen zu beurteilen. Anhand des Diagramms kann man erkennen, wie weit die Datenpunkte verstreut oder konzentriert sind und ob es Tendenzen zu Zusammenhängen gibt. Es ist auch möglich, Ausreißer zu erkennen und die Korrelation zu beurteilen.

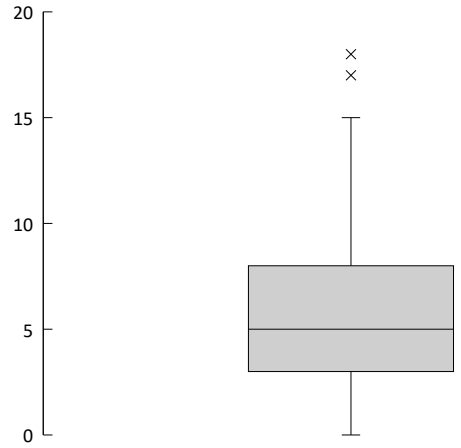

Abbildung 2.2: Beispiel für einen Boxplot.

Boxplots sind gut dazu geeignet, die Verteilung und die Schiefe der Verteilung darzustellen. In Abbildung 2.2 ist ein Beispiel für einen Boxplot zu sehen. Dabei werden verschiedene Quartile grafisch dargestellt. Der mittlere Strich in der Box repräsentiert den Median, die beiden Striche, die die Box nach unten bzw. nach oben begrenzen, repräsentieren das untere bzw. das obere Quartil. Die Box beinhaltet 50 % der Datenpunkte. Die Linien, die die Boxen nach unten und oben verlängern, werden *Antennen* oder *Whisker* genannt. Die Länge der Whisker ist durch den kleinsten bzw. größten Datenpunkt innerhalb des 1,5-Fachen des Interquartilabstands vom unteren bzw. oberen Quartil begrenzt. Werte, die außerhalb der Whisker liegen, werden einzeln gekennzeichnet und als Ausreißer aufgefasst.

2.1.2 Hypothesentests

Das Ziel von *Hypothesentests* ist es, zu prüfen, ob es möglich ist, eine bestimmte *Nullhypothese* H_0 abzulehnen, basierend auf einer Stichprobe aus einer statistischen Verteilung, und stattdessen eine *Alternativhypothese* H_1 zu akzeptieren. Dabei beschreibt die Nullhypothese einige Eigenschaften der Verteilung und der Experimentator möchte mit einer bestimmten Signifikanz ablehnen, dass diese Eigenschaften wahr sind. Häufig hängt die

Verteilung von nur einem Parameter ab. Die Nullhypothese aufzustellen bedeutet dann die Verteilung zu formulieren und dem Parameter, der getestet werden soll, einen Wert zuzuweisen.

Die Nullhypothese sollte negativ formuliert werden, da es die Absicht des Tests ist, sie abzulehnen. Wenn die Nullhypothese nicht abgelehnt wird, kann nichts über das Ergebnis gesagt werden. Wenn sie aber abgelehnt wird, kann mit einer gegebenen Signifikanz α gesagt werden, dass H_0 falsch ist. Wenn ein Test durchgeführt wird, ist es häufig möglich, die kleinste mögliche Signifikanz (bezeichnet als *p-Wert*) zu berechnen, mit der es möglich ist, die Nullhypothese abzulehnen.

Drei bedeutende Wahrscheinlichkeiten für Hypothesentests sind:

$$\alpha = P(\text{Typ-I-Fehler}) = P(\text{lehne } H_0 \text{ ab} | H_0 \text{ist wahr}) \tag{2.9}$$

$$\beta = P(\text{Typ-II-Fehler}) = P(\text{lehne } H_0 \text{ nicht ab} | H_0 \text{ ist falsch}) \tag{2.10}$$

$$\text{Trennschärfe} = 1 - \beta = P(\text{lehne } H_0 \text{ ab} | H_0 \text{ ist falsch}) \tag{2.11}$$

Ein Typ-I-Fehler tritt auf, wenn ein statistischer Test einen Zusammenhang feststellt, obwohl es keinen Zusammenhang gibt. Ein Typ-II-Fehler tritt auf, wenn ein Test keinen Zusammenhang feststellt, obwohl es einen gibt. Die Trennschärfe eines statistischen Tests ist die Wahrscheinlichkeit, dass er einen Zusammenhang aufdeckt, wenn es einen gibt. Vor der Durchführung des Tests muss festgelegt werden, mit welcher Wahrscheinlichkeit man bereit ist, Typ-I-Fehler zu akzeptieren. Ein üblicher Grenzwert für die Signifikanz ist $\alpha = 0{,}05$.

Hypothesentests können in *parametrische* und *nichtparametrische* Tests unterteilt werden. Parametrische Tests basieren auf einem Modell, das eine bestimmte Verteilung beinhaltet. In den meisten Fällen wird angenommen, dass einige Parameter dabei normalverteilt sind. Nichtparametrische Tests machen nicht die gleiche Art von Annahmen wie parametrische Tests. Nichtparametrische Tests sind allgemeiner als parametrische Tests, d. h. dass nichtparametrische Tests, wenn geeignet, anstelle von parametrischen Tests genutzt werden können, parametrische Tests können im Allgemeinen aber nicht anstelle von nichtparametrischen eingesetzt werden.

Bei der Entscheidung zwischen parametrischen und nichtparametrischen Tests müssen zwei Faktoren beachtet werden. Zum einen die Eignung des Tests. Es ist wichtig, dass die Annahmen zur Verteilung der Parameter realistisch sind. Zum anderen die Trennschärfe. Parametrische Tests haben im Allgemeinen eine größere Trennschärfe als nichtparametrische Test. Daher benötigen parametrische Test weniger Datenpunkte als nichtparametrische Tests, wenn die Annahmen zutreffen.

In den folgenden Abschnitten werden die in dieser Arbeit verwendeten Tests kurz vorgestellt.

t-Test

Der *t-Test* ist ein parametrischer Test, der genutzt wird, um zwei unabhängige Stichproben zu vergleichen. Der t-Test setzt voraus, dass beide Stichproben normalverteilt sind und die gleiche Standardabweichung haben. Der Test wird wie folgt durchgeführt:

Eingabe	Zwei unabhängige Stichproben $x_1, x_2, \dots x_n$ und $y_1, y_2, \dots y_m$.
H_0	$\mu_x = \mu_y$, d. h. die erwarteten Mittelwerte sind gleich.
Berechnung	Berechne $t_0 = \frac{\bar{x} - \bar{y}}{S_p \sqrt{\frac{1}{n} + \frac{1}{m}}}$ mit $S_p = \sqrt{\frac{(n-1)S_x^2 + (m-1)S_y^2}{n+m-2}}$ mit den Varianzen der Stichproben S_x^2 und S_y^2.
Kriterium	Zweiseitig ($H_1 : \mu_x \neq \mu_y$): lehne H_0 ab, wenn $\lvert t_0 \rvert > t_{\alpha/2;n+m-2}$. Dabei ist $t_{\alpha;f}$ das $(1-\alpha)$-Perzentil der t-Verteilung [42] mit f Freiheitsgraden, die hier gleich $n+m-2$ sind. Die Werte können aus Tabellen abgelesen werden oder mit Statistikbibliotheken berechnet werden. Einseitig ($H_1 : \mu_x > \mu_y$): lehne H_0 ab, wenn $t_0 > t_{\alpha,m+m-2}$.

Als Beispiel betrachten wir zwei Projekte, in denen die Lead Time für Stories in Tagen gemessen wurde. In einem Projekt ist das Ergebnis

$$x = 7{,}69; 4{,}67; 4{,}45; 4{,}61; 6{,}7; 6{,}63; 6{,}79; 6{,}51; 8{,}26; 5{,}04$$

und im anderen Projekt wurden folgende Werte gemessen

$$y = 8{,}28; 9{,}75; 9{,}49; 4{,}76; 7{,}42; 7{,}51; 5{,}98; 9{,}89; 5{,}5; 6{,}58; 9{,}79; 8{,}07.$$

Die Nullhypothese ist, dass die Lead Time in beiden Projekten gleich ist, die Alternativhypothese, dass sie es nicht ist. Aus den Daten ergibt sich $n = 10$ und $m = 12$. Die Mittelwerte sind $\bar{x} = 6{,}14$ und $\bar{y} = 7{,}75$. Damit ist dann $t_0 = -2{,}355$. Es gibt $f = 20$ Freiheitsgrade, damit ist $t_{0{,}05;22} = 2{,}086$. Da gilt $\lvert t_0 \rvert > t$, kann also die Nullhypothese mit einer Signifikanz von 0,05 abgelehnt werden. Der zu t_0 gehörende p-Wert ist $p = 0{,}0288$.

Mann-Whitney-Test

Der *Mann-Whitney-Test* ist eine nichtparametrische Alternative zum t-Test. Es ist immer möglich, diesen Test zu verwenden, wenn die Annahmen für den t-Test unsicher sind. Der Test wird wie folgt durchgeführt:

Eingabe	Zwei unabhängige Stichproben $x_1, x_2, \dots x_n$ und $y_1, y_2, \dots y_m$.
H_0	Die beiden Stichproben kommen aus der gleichen Verteilung.
Berechnung	Weise jedem Datenpunkt einen Rang zu und berechne $U = N_A N_B + \frac{N_A(N_A+1)}{2} - T$ und $U' = N_A N_B - U$, mit $N_A = \min(n,m)$ und $N_B = \max(n,m)$ und T ist die Summe der Ränge der kleinsten Stichprobe.

Kriterium Tabellen mit Kriterien zur Ablehnung der Nullhypothese gibt es z. B. von Milton [23].
Lehne H_0 ab, wenn $\min(U, U')$ kleiner oder gleich dem Wert in der Tabelle ist. Bei großen n, m kann die Verteilung von U durch die Normalverteilung genähert werden.

Mit den gleichen Daten wie aus dem Beispiel für den t-Test ergeben sich $N_A = 10$ und $N_B = 12$. Die Ränge der kleinsten Stichprobe (x) sind $15, 3, 1, 2, 11, 10, 12, 8, 17, 5$ und die Ränge der größten Stichprobe (y) sind $18, 20, 19, 4, 13, 14, 7, 22, 6, 9, 21, 16$. Aus den Rängen ergeben sich $T = 84$, $U = 91$ und $U' = 29$. Da der kleinere Wert von U und U' gleich 29, dem Wert aus der Tabelle [23], ist, kann die Nullhypothese mit einer Signifikanz von 0,05 abgelehnt werden.

ANOVA

Varianzanalyse (*ANOVA*) kann genutzt werden, um verschiedene Arten von Experimenten zu analysieren. Die Varianzanalyse umfasst eine Gruppe von Tests, in der einfachsten Form vergleicht der Test die Variabilität durch verschiedene Stufen mit der Variabilität durch zufällige Fehler. In dieser einfachsten Form kann der Test verwendet werden, um zu überprüfen, ob einige Stichproben den gleichen Mittelwert haben. Dieser Test mit einem Faktor und mehr als zwei Stufen (Treatments) wird wie folgt durchgeführt:

Eingabe a Stichproben: $x_{11}, x_{12}, \ldots x_{1n_1}; x_{21}, x_{22}, \ldots x_{2n_2}; \ldots; x_{a1}, x_{a2}, \ldots x_{an_a}$.

H_0 $\mu_{x_1} = \mu_{x_2} = \cdots = \mu_{x_a}$, d. h. alle Erwartungswerte sind gleich.

Berechnung
$$SS_{\mathrm{T}} = \sum_{i=1}^{a}\sum_{j=1}^{n_i} x_{ij}^2 - \frac{x_{\cdot\cdot}^2}{N}$$
$$SS_{\text{Treatment}} = \sum_{i=1}^{a} \frac{x_{i\cdot}^2}{n_i} - \frac{x_{\cdot\cdot}^2}{N}$$
$$SS_{\text{Error}} = SS_{\mathrm{T}} - SS_{\text{Treatment}}$$
$$MS_{\text{Treatment}} = SS_{\text{Treatment}} \,/\, (a-1)$$
$$MS_{\text{Error}} = SS_{\text{Error}} \,/\, (N-a)$$
$$F_0 = MS_{\text{Treatment}} \,/\, MS_{\text{Error}}$$
wobei N die Gesamtzahl der Messungen ist und ein Punkt als Index eine Summe über den gepunkteten Index bezeichnet, z. B. $x_{i\cdot} = \sum_j x_{ij}$.

Kriterium Lehne H_0 ab, wenn $F_0 > F_{\alpha,a-1,N-a}$. Dabei ist F_{α,f_1,f_2} das $(1-\alpha)$-Perzentil der F-Verteilung [11] mit f_1 und f_2 Freiheitsgraden. Die Werte für F können aus Tabellen abgelesen werden oder es kann mit Statistikbibliotheken direkt der p-Wert berechnet werden.

Kruskal-Wallis-Test

Der *Kruskal-Wallis-Test* ist eine nichtparametrische Alternative zur ANOVA mit einem Faktor. Dieser rangbasierte Test wird wie folgt durchgeführt [17]:

Eingabe	a Stichproben: $x_{11}, x_{12}, \dots x_{1n_1}; x_{21}, x_{22}, \dots x_{2n_2}; \dots; x_{a1}, x_{a2}, \dots x_{an_a}$.
H_0	Die Mediane der Populationen, aus denen die a Stichproben stammen, sind gleich.
Berechnung	Jedem Datenpunkt x_{ij} wird ein Rang $R(x_{ij})$ zugewiesen. Berechne $H = \left(\frac{12}{N(N+1)} \sum_{i=i}^{a} \frac{R_i^2}{n_i}\right) - 3(N+1)$, mit $R_i = \sum_{j=1}^{n_i} R(x_{ij})$ und $N = \sum_{i=1}^{a} n_i$. Wenn es mehrere Datenpunkte mit dem gleichen Rang gibt, muss der Wert korrigiert werden durch $\tilde{H} = \frac{H}{1 - \frac{\sum_{i=i}^{g}(t_i^3 - t_i)}{N^3 - N}}$, wobei g die Anzahl von Gruppen mit gleichem Rang ist und t_i die Dimension der i-ten Gruppe.
Kriterium	Bei drei ausreichend kleinen Strichproben, lehne H_0 ab, wenn $H > h_{n_1,n_2,n_3,1-\alpha}$, mit $h_{n_1,n_2,n_3,1-\alpha}$ aus einer Kruskal-Wallis-Tabelle [16]. Bei mehr oder größeren Stichprobe, lehne H_0 ab, wenn $H > \chi^2_{\alpha,a-1}$, dabei ist $\chi^2_{\alpha,f}$ das $(1-\alpha)$-Perzentil der Chi-Quadrat-Verteilung [5] mit f Freiheitsgraden.

Tukey-Test

Nachdem festgestellt wurde, ob es einen Unterschied zwischen Stichproben gibt, ist es interessant zu wissen, zwischen welchen Paaren von Stichproben es unterschiede gibt. Der *Tukey-Test* [44] ist ein parametrischer Test, der mehrere Stichproben auf einmal vergleicht. Dabei wird paarweise überprüft, ob die Mittelwerte der Stichproben übereinstimmen. Der Test wird wie folgt durchgeführt:

Eingabe	a Stichproben: $x_{11}, x_{12}, \dots x_{1n}; x_{21}, x_{22}, \dots x_{2n}; \dots; x_{a1}, x_{a2}, \dots x_{an}$.
H_0	Die Mittelwerte der Populationen, aus denen die a Stichproben stammen, sind gleich.
Berechnung	Berechne für jedes Paar $q_s = \frac{\bar{x_i} - \bar{x_j}}{s / \sqrt{n}}$, wobei $\bar{x_i}$ der größere und $\bar{x_j}$ der kleinere der beiden betrachteten und s die Standardabweichung der Summe aller Mittelwerte sind.
Kriterium	Lehne H_0 (paarweise) ab wenn $q_s > q_{\alpha,a,N-a}$, wobei N die Gesamtzahl der Datenpunkte ist und $q_{\alpha,k,\nu}$ das $(1-\alpha)$-Perzentil der studentized range-Verteilung [41] mit k und ν Freiheitsgraden.

Games-Howell-Test

Der *Games-Howell-Test* [13] ist eine nichtparametrische Alternative zum Tukey-Test, mit dem paarweise getestet wird, ob die Mittelwerte der Stichproben aus der gleichen Population stammen. Der Test wird wie folgt durchgeführt:

Eingabe	a Stichproben: $x_{11}, x_{12}, \dots x_{1n_1}; x_{21}, x_{22}, \dots x_{2n_2}; \dots; x_{a1}, x_{a2}, \dots x_{an_a}$.
H_0	Die Mittelwerte der Populationen, aus denen die a Stichproben stammen, sind gleich.
Berechnung	Berechne ür jedes Paar $q_s = \frac{\bar{x_i} - \bar{x_j}}{se \,/\, \sqrt{n}}$, wobei $\bar{x_i}$ der größere und $\bar{x_j}$ der kleinere der beiden Mittelwerte sind und $se = \sqrt{\frac{1}{2}\left(\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}\right)}$.
Kriterium	Lehne H_0 für das betrachtete Paar ab, wenn $q_s > q_{\alpha,a,df}$, mit $df = \frac{\left(s_i^2 \,/\, n_i + s_j^2 \,/\, n_j\right)^2}{\frac{\left(s_i^2/n_i\right)^2}{n_i - 1} + \frac{\left(s_j^2/n_j\right)^2}{n_j - 1}}$ und $q_{\alpha,k,\nu}$ das $(1-\alpha)$-Perzentil der studentized range-Verteilung [41] mit k und ν Freiheitsgraden.

Shapiro-Wilk-Test

Um zu entscheiden, ob ein parametrischer oder nichtparametrischer Test verwendet werden kann, muss unter anderem bekannt sein, ob die Stichprobe aus einer Normalverteilung stammt. Ein Test, der das überprüft, ist der *Shapiro-Wilk-Test* [39]. In seiner ursprünglichen Form konnte dieser Test nur für Stichproben mit bis zu 50 Datenpunkten verwendet werden, durch Erweiterungen [31] kann er auch für größere Stichproben mit mehreren Tausend Datenpunkten genutzt werden. Der Test wird wie folgt durchgeführt:

Eingabe	Eine Stichprobe $x_1, x_2, \dots x_n$.
H_0	Die Stichprobe stammt aus einer Normalverteilung.
Berechnung	Ordne die Stichprobe und weise den Datenpunkten einen Rang zu. Berechne $W = \frac{b^2}{(n-1)s^2}$, wobei $b = \sum_{i=1}^{k} a_i \left(x_{n+1-i} - x_i\right)$, mit $k = n \,/\, 2$ wenn n gerade bzw. $k = (n-1)/2$ wenn n ungerade, und a_i Koeffizienten, die sich aus der Ordnungsstatistik der Normalverteilung ergeben.
Kriterium	Lehne H_0 ab, wenn $W < W_{\text{crit}}$, wobei W_{crit} für kleine Werte von n aus Tabellen abgelesen werden kann.

2.2 Verwendete Technologien

In dieser Arbeit wurde eine Software entwickelt, mit der Daten von Jira ausgewertet werden können. Für die Entwicklung der Software wurden verschiedene Technologien und Softwarebibliotheken verwendet. In diesem Abschnitt wird ein Überblick über Jira gegeben und die genutzten Technologien und Bibliotheken kurz erläutert.

2.2.1 Jira

Jira [15] ist ein weitverbreitetes Entwicklungstool für agile Projekte von Atlassian. Jira bietet dabei Funktionen zur Planung und Verwaltung von agilen Softwareprojekten auf mehreren Ebenen, von Storycards über Scrum- oder Kanbanboards bis zu Roadmaps. Dabei unterstützt es auch die Integration von gebräuchlichen Entwicklungstools wie z. B. Versionskontroll- und Deploymentsystemen und weiteren Zusammenarbeitstools, sowie die Erweiterung durch sogenannte Apps. Zusätzlich bietet Jira eine Berichtsfunktion, die typische agile Visualisierungen wie verschiedene Burndown-Charts, Velocity oder kumulative Flussdiagramme erzeugt.

Ein zentrales Konzept von Jira sind *Issues*. Diese Issues sind Aufgabeneinheiten und können unter anderem Bugs, Stories oder Epics darstellen. Ein Issue beinhaltet dabei zumindest eine Zusammenfassung, einen Ersteller und einen Erstellungszeitpunkt. Weitere Felder, die üblicherweise vorhanden sind, sind eine ausführliche Beschreibung, der aktuelle Status im Workflow (s. u.), der Bearbeiter, dem das Issue zugewiesen wurde, die Priorität und die Lösung, wenn der Vorgang abgeschlossen wurde. Es können auch benutzerdefinierte Felder erstellt werden, beispielsweise für Storypoints. Außerdem können Kommentare hinterlassen werden. Zusätzlich gibt es Verknüpfungen zu anderen Issues. Stories sind oft Teil eines Epics und können auch Unteraufgaben haben. Wenn ein Versionsverwaltungssystem integriert ist, werden auch relevante Commits und Pull Requests angezeigt. Diese Issues können jederzeit von jedem berechtigten Entwickler bearbeitet werden, daher gibt es eine Änderungshistorie, in der alle Aktivitäten festgehalten werden.

Issues durchlaufen einen sogenannten *Workflow*, also einen Arbeitsablauf, der aus verschiedenen Status und Übergängen zwischen diesen Status besteht. Im einfachsten Fall besteht ein Workflow aus den Schritten „To Do“, „In Progress“ und „Done“, kann aber auch beliebig komplex sein. Dieser Workflow wird in Jira durch das Backlog und durch Scrum- oder Kanbanboards dargestellt. Das Backlog ist eine (teilweise) geordnete Liste von Issues, die noch zu erledigen sind. Die Boards beinhalten die Issues, die gerade bearbeitet werden, in Spalten nach Status geordnet.

Jira stellt außerdem REST-APIs bereit, mit denen viele Funktionen auch durch Fernzugriff genutzt werden können. Diese APIs wurden in dieser Arbeit genutzt, um auf die Issue-Daten zuzugreifen, um sie zu analysieren.

2.2.2 REST

Representational State Transfer (REST) [10] ist ein Architekturstil für verteilte Hypermediasysteme, der auf den Einschränkungen des World Wide Web basiert. Die „Webarchitektur mit ihrem wichtigsten Protokoll HTTP [ist eine] konkrete Ausprägung des Architekturstils REST." ([43])

Die fünf Kernprinzipien von REST sind [43]:

Eindeutige Identifikation Bei REST gibt es ein einheitliches Konzept zur Vergabe von eindeutigen Identifikatoren für Ressourcen, den Uniform Resource Identifiers (URIs). Diese URIs werden verwendet, um alle wesentlichen Abstraktionen einer Anwendung zu identifizieren. In der Regel sind URIs menschenlesbar, um eine einfache Interpretation zu ermöglichen.

Hypermedia Hypermedia bezeichnet das Konzept von Verknüpfungen. Dabei können Verknüpfungen einerseits zu weiteren Informationen führen, z. B. in einer Filmdatenbank hat ein Film Verknüpfungen zu den Schauspielern, die darin mitgewirkt haben. Andererseits können Verknüpfungen mitteilen, welche Aktionen als Nächstes möglich sind, z. B. das Verfassen einer Rezension zu einem Film.

Standardmethoden Hypertext Transfer Protocol (HTTP) bietet eine Schnittstelle, die einen festgelegten Satz von Methoden unterstützt. Diese Methoden sind GET, POST, PUT, DELETE, HEAD und OPTIONS. Die HTTP-Spezifikation beschreibt ihre Bedeutung und Garantien für ihr Verhalten.

Unterschiedliche Repräsentationen Für unterschiedliche Bedürfnisse ist es sinnvoll, verschiedene Repräsentationen der Ressourcen bereitzustellen. Die gleichen Daten können beispielsweise als HTTP zur Anzeige in einem Webbrowser bereitgestellt werden und als JavaScript Object Notation (JSON) zur Abfrage über eine API.

Statuslose Kommunikation Der Server hält keinen Zustand über mehrere Anfragen des Clients vor. Wenn ein Zustand benötigt wird, muss er entweder vom Client gespeichert werden und dem Server mit der nächsten Anfrage übermittelt werden, z. B. Logindaten, oder es muss eine neue Ressource erstellt werden, auf die der Client mit einer Verknüpfung zugreifen kann, z. B. ein Warenkorb.

2.2.3 Node.js

Node.js [25] ist eine JavaScript Laufzeitumgebung, die es ermöglicht, JavaScript Code außerhalb von Webbrowsern auszuführen. Node.js wurde entwickelt, um skalierbare Netzwerkanwendungen zu erstellen. Aufgrund des Fokus auf Streaming und niedrige Latenz eignet es sich gut als Grundlage von Web-Bibliotheken oder Frameworks [25].

Ein wichtiges Werkzeug für die Entwicklung mit Node.js ist *npm* [27]. npm ist ein Paketmanager für Node.js und stellt mit der npm Registry eine Sammlung von Open Source Bibliotheken und Frameworks bereit. Im Folgenden gibt es eine kurze Übersicht über die in dieser Arbeit verwendeten Bibliotheken.

Express

Express [9] ist ein Framework für die Entwicklung von Webanwendungen, das auf Node.js basiert. Express eignet sich vor allem für die Entwicklung von Webservern, also dem „Backend“ von Webanwendungen. Dazu bietet es Methoden zum Ausführen eines HTTP-Servers und zum asynchronen Empfangen, Verarbeiten und Beantworten von HTTP-Requests. Express kann auch durch Middlewares erweitert werden, um zusätzliche Funktionalitäten zu ermöglichen.

React

React [33] ist eine JavaScript-Bibliothek zur Entwicklung von Benutzeroberflächen. React kann – wie beliebiger JavaScript Code – im Webbrowser ausgeführt werden, es unterstützt aber auch serverseitiges Rendern mit Node.js. React setzt auf ein deklaratives Programmiermodell, dessen Kern Komponenten sind, die ihren eigenen Zustand verwalten und eine eigene Render-Funktion haben. Informationen fließen dabei von oben nach unten, der Zustand von höherliegenden Komponenten kann nur durch Callback-Funktionen verändert werden.

Um eine Anwendung zu ermöglichen, die aus mehr als einer Seite besteht, wurde zusätzlich *React Router* [32] verwendet, was Navigationskomponenten bietet. Für die Verwaltung von globalen Zuständen wurde *Recoil* [35] genutzt.

Mit *Material-UI* wurde eine Bibliothek von React-Komponenten für Benutzeroberflächen auf Basis des Material-Designs [21] genutzt. Material-UI bietet dabei responsives Layoutsystem, eine große Sammlung mit den üblichen HTML-Komponenten und von Material-Design definierte Komponenten wie der App Bar. Diese Komponenten können zusätzlich beliebig an die Bedürfnisse angepasst werden.

Axios

Das Backend der entwickelten Anwendung muss Anfragen an den Jira-Server senden können. Da es aber nicht im Browser ausgeführt wird, kann die von diesem bereitgestellte JavaScript Fetch-API nicht genutzt werden. Deshalb wird *Axios* [49] verwendet, ein HTTP-Client, der sowohl im Webbrowser als auch in Node.js genutzt werden kann.

Statistikbibliotheken

Für JavaScript gibt es keine umfangreiche Statistikbibliothek, die alle in dieser Arbeit genutzten statistischen Test bereitstellt. Es wurden daher für die verschiedenen Tests mehrere verschiedene JavaScript-Bibliotheken eingesetzt. *Simple Statistics* [20] ist eine Statistikbibliothek, die auf ausführliche Dokumentation und einfache Bedienung setzt. Simple Statistics wurde für beschreibende Statistik, lineare Regression und den t-Test genutzt. *jStat* [26] ist eine umfangreiche Statistikbibliothek, die viele gebräuchliche Verteilungen bietet. Sie wurde für ANOVA, den Tukey-Test und für die Berechnung des p-Werts aus dem t-Wert für den t-Test genutzt. Für den Shapiro-Wilk-Test wurde die Statistikbibliothek *Jerzy* [30] genutzt. *stdlib* [40] ist eine Standardbibliothek für JavaScript mit einem Fokus auf mathematische Funktionen. Daraus wurde der Kruskal-Wallis-Test verwendet. Für den Mann-Whitney-Test wurde eine JavaScript-Implementierung von Budak [3] verwendet.

Visualisierungsbibliotheken

Es gibt eine große Auswahl von Visualisierungsbibliotheken für JavaScript, die den Schwerpunkt in unterschiedlichen Bereichen setzten. Für diese Arbeit wurden zwei Bibliotheken gewählt, die React-Komponenten anbieten und daher leicht in React-Anwendungen integriert werden können. *Recharts* [34] ist eine Bibliothek, die auf Komponentenkomposition setzt, um Graphen zu erstellen. Recharts wurde für Balkendiagramme, Scatterplots und Liniendiagramme verwendet. *visx* [46] ist eine Sammlung von Low-Level-Visualisierungskomponenten, aus denen komplexe Visualisierungen aufgebaut werden können. visx wurde für Boxplots genutzt.

2.2.4 CouchDB

CouchDB [7] ist ein dokumentenorientiertes Datenbankmanagementsystem. Das Ziel von CouchDB ist es, ein einfach zu benutzendes Datenbankmanagementsystem zu entwickeln, das fehlertolerant und skalierbar ist. Dabei setzt es auf ein schemaloses Datenmodell, bei dem Dokumente unter einem eindeutigen Namen als JSON gespeichert werden. CouchDB bietet eine REST-API, um die Dokumente zu lesen oder zu verändern. Zusätzlich bietet es eine eingebaute Query-Engine, mit der beliebige JavaScript-Funktionen für Map/Reduce-Operationen genutzt werden können, um die Dokumente zu filtern. Durch ihre weite Verbreitung gibt es für viele Programmiersprachen Bibliotheken, mit denen Clients für CouchDB implementiert werden können. In dieser Arbeit wurde *Nano* [24] genutzt.

Kapitel 3

Verwandte Arbeiten

Produktivität in der Softwareentwicklung ist seit Jahrzehnten ein wichtiges Forschungsthema. Mit der zunehmenden Popularität von agilen Methoden hat auch die Lead Time an Bedeutung gewonnen. Auch Jira ist durch seine weite Verbreitung in der Softwareentwicklung zu einer wichtigen Quelle für Daten als Grundlage für weitere Analysen geworden. In diesem Kapitel wird dazu eine Auswahl relevanter Arbeiten vorgestellt.

Wagner und Ruhe [47] haben in einem Review Einflussfaktoren auf die Produktivität in der Softwareentwicklung untersucht. Dabei haben sie Studien aus den Jahren von 1970 bis 2007 betrachtet. Dabei haben sie zwischen „technical“ und „soft factors“ unterschieden, wobei Studien anfangs hauptsächlich technische Faktoren betrachtet haben und im Laufe der Zeit die „soft factors“ zunehmend an Bedeutung gewannen. Eine wichtige Erkenntnis ist, dass die Produktivität mit zunehmendem Kommunikationsaufwand steigt. Weitere viel genannte „technical factors“ sind die Komplexität des Produkts, Laufzeitbeschränkungen und der Grad der Nutzung von Entwicklungswerkzeugen. Für „soft factors“ wurden häufig die Kohäsion des Teams, die Fähigkeiten der Programmierer und die Teamgröße genannt.

Lagerström et al. [19] haben anhand von Daten aus 50 Projekten einer schwedischen Bank untersucht, welche Faktoren die Kosten und die Produktivität von Softwareentwicklung beeinflussen. Es wurden 31 Einflussfaktoren untersucht, von denen sechs einen signifikanten Einfluss auf die Kosten, allerdings nur einer einen signifikanten Einfluss auf die Produktivität hatte. Dieser Faktor war der Projekttyp, wobei Integrationsprojekte eine höhere Produktivität hatten als Entwicklungsprojekte. Zusätzlich zu diesem Faktor haben sie vier weitere Faktoren genannt, deren p-Wert unter 0,15 war. Das war zum einen die Güte von Schätzungen und Prognosen. Die Projekte, von denen angenommen wurde, dass bei ihnen die Güte von Schätzungen am schlechtesten war, waren hier die produktivsten. Weitere Faktoren waren die Präsentationsoberfläche, die Existenz eines Prüfleiters und die Anzahl von Budgetrevisionen.

Petersen [29] hat die Lead Time in agiler Softwareentwicklung in einer Fallstudie an zwölf Systemen bei Ericsson AB untersucht. Dabei hat er betrachtet, ob es Unterschiede zwischen verschiedenen Phasen des Entwicklungsprozesses gibt, zwischen Anforderungen, die mehrere Systeme betreffen, und solchen, die nur ein System betreffen und zwischen den Größen der Systeme. Von den drei betrachteten Parametern hatte nur die Größe des Systems einen signifikanten Einfluss auf die Lead Time.

Forsgren et al. [12] haben sich mit der Produktivität von Softwareentwicklern beschäftigt. Dabei haben sie zunächst vorherrschende Mythen und Irrglauben über die Produktivität behandelt. Diese Mythen beinhalten unter anderem, dass Produktivität nur von der Aktivität des Entwicklers abhängt, eine einzige Produktivitätsmetrik ausreicht, um die Leistung von Teams zu vergleichen oder dass Produktivitätsmessungen nur für Manager nützlich sind. Ausgehend von den Mythen haben sie ein Framework, SPACE, vorgestellt, das bei der Auswahl geeigneter Metriken für die Produktivität von Entwicklern helfen soll. Dabei werden Metriken aus den Bereichen „satisfaction and well-being“, „performance“, „activity“, „communication und collaboration“ und „efficency and flow“ vorgeschlagen. Es sollten Metriken aus mindestens drei der Bereiche erhoben werden, um signifikante Aussagen zu erhalten. Die gewählten Metriken können auch genutzt werden, um den Entwicklern die Werte des Unternehmens zu verdeutlichen.

Ortu et al. [28] haben Jira-Daten aus sieben Open-Source-Projekten genutzt. Dabei haben sie untersucht, ob sich unter den Entwicklern der Projekte Communities gebildet haben und ob diese Communities unterschiedlich lange benötigen, um Issues zu bearbeiten. Das Ergebnis der Studie war, dass in allen untersuchten Projekten Communities existieren. Die durchschnittliche Bearbeitungszeit unterschied sich zwischen den Communities, allerdings konnten sie zeigen, dass die Zeit unabhängig von der Größe der Community oder des Issue-Typs war.

Valdez et al. [45] haben Daten aus Jira für Sentiment Analysis genutzt. Sie haben die Issue-Kommentare von Open-Source-Projekten untersucht. Dabei fanden sie heraus, dass in den betrachteten Projekten Kommentare in ungelösten Issues negativer waren als die in abgeschlossenen. Außerdem wurden positive Kommentare häufiger am Morgen geschrieben, währen negative Kommentare am späten Nachmittag häufiger waren.

Kapitel 4

Methodik

In dieser Arbeit wurde eine Fallstudie an den Jira-Daten des Projekts ConnectFleet von Volkswagen Nutzfahrzeuge (VWN) durchgeführt. In diesem Kapitel wird das Vorgehen bei dieser Studie beschrieben. Dazu wird zunächst auf den Kontext der Studie eingegangen, anschließend wird erläutert, wie bei der Datenerhebung und -analyse vorgegangen wurde.

Bei der Auswahl von Metriken wurde explorativ vorgegangen. Die Studie hat also nicht das Ziel, vorher festgelegte Hypothesen zu überprüfen, sondern möglichst viele Faktoren zu untersuchen und dabei herauszufinden, welche davon einen Einfluss auf die Lead Time haben. Aus diesem Grund werden die betrachteten Metriken in Kapitel 6 vorgestellt.

4.1 Kontext

Die Studie wurde im Projekt *ConnectFleet* bei VWN durchgeführt. In diesem Projekt wird ein Flottenmanagementsystem entwickelt. Zur Verwaltung von Aufgaben, also Epics, Stories und Bugs, wird Jira genutzt. Die Aufgaben werden auf sieben Teams aufgeteilt, die nach Farben benannt sind. Die Teams haben unterschiedliche Größen und setzen sich aus Mitarbeitern von VWN und von externen Dienstleistern zusammen.

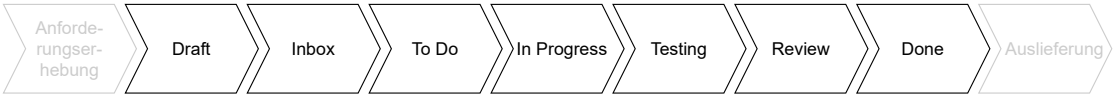

Abbildung 4.1: Der Entwicklungsprozess bei VWN. In dieser Arbeit werden die Phasen Draft bis Done betrachtet, die den Status entsprechen, die eine Story in Jira durchläuft. Die Prozessschritte davor und danach wurden zu Anforderungserhebung bzw. Auslieferung zusammengefasst.

Je nach Team wird mit verschiedenen agilen Methoden gearbeitet, Scrum oder Kanban. Unabhängig von der genutzten Methode durchlaufen die Aufgaben bestimmte Prozessschritte, im Folgenden Status genannt. Abbildung 4.1 zeigt den Ablauf des Entwicklungsprozesses. Bevor die Aufgaben in Jira erfasst werden, werden Anforderungen durch Feedback von Kunden und anderen Stakeholdern erhoben. Nachdem die Anforderungen gesammelt wurden, werden in Jira Epics erstellt und mit Stories gefüllt. Neue Epics und

Stories erhalten den Status *Draft*. Darin werden die Stories noch ohne Rückmeldung der Entwickler ausformuliert. Wenn dies abgeschlossen ist, kommen sie in den nächsten Status, *Inbox*. Darin sind Diskussionen mit den Entwicklern vorgesehen, um Unklarheiten zu beseitigen. Wenn die Story bereit für die Entwicklung ist, kommt sie in den Status *To Do*. Dieser Status entspricht dem Backlog. Die eigentliche Entwicklung findet im Status *In Progress* statt. Hier werden die erforderlichen Funktionalitäten implementiert. Anschließend folgt der Status *Testing*. Darin werden die vorher implementierten Funktionen teamintern getestet. Wenn die Tests keine Fehler gefunden haben, folgt der Status *Review*. Dieser dient der Qualitätssicherung, indem ein zweiter Entwickler die Implementierung reviewt. Diese Reviews werden regelmäßig an festgelegten Terminen durchgeführt. War das Review erfolgreich, ist die Story für das Team abgeschlossen und erhält den Status *Done*. Damit werden die Änderungen in den Development-Branch gemerged, wo weitere Tests durchgeführt werden und schließlich werden sie ins Produktivsystem ausgeliefert.

Der hier beschriebene Prozess stellt den idealen Ablauf dar, es kann aber davon abgewichen werden. Wenn in einem Status – z. B. in *Testing* – Probleme festgestellt werden, kann die Story in einen vorherigen Status zurückgesetzt werden – z. B. in *To Do*. Zusätzlich ist es möglich, dass eine Story nicht implementiert wird – z. B. wenn sich Anforderungen ändern und sie nicht mehr benötigt wird – und sie direkt den Status *Done* erhält, versehen mit einer Anmerkung wie „Won't Do".

4.2 Datenerhebung

Die in dieser Arbeit verwendeten Daten stammen aus dem Jira-Repository des Projekts ConnectFleet. Diese Daten wurden über die von Jira bereitgestellten REST-APIs abgefragt. Die Lead Time wird in dieser Arbeit als die Zeit von der Erstellung der Story in Jira bis zu ihrem Abschluss definiert, deshalb wurden nur Stories betrachtet, die abgeschlossen sind. Das Projekt enthält zum Zeitpunkt dieser Arbeit 4116 Stories[1], die diese Bedingung erfüllen.

Um die Stories zu erhalten, wurde die Ressource `api/2/search` genutzt. Diese Ressource akzeptiert Suchanfragen in der Jira Query Language (JQL). Um alle abgeschlossenen Stories zu erhalten, wurde der Suchstring

```
project = "ConnectFleet" AND type = story AND status = done
```

verwendet. Bei der Betrachtung eines einzelnen Teams wurde die Anfrage um den String `cf[24121] = "{TEAM}"` erweitert, wobei `cf[24121]` das Feld in den Jira-Issues für das Team ist, das die Story bearbeitet und `{TEAM}` durch das betrachtete Team ersetzt wurde, z. B. `Team Black`. Zusätzlich wurde der Parameter `expand = ['changelog']` gesetzt, um die Änderungshistorie für jede Story zu erhalten.

[1] Stand 06.04.2021

Die Antwort auf die Anfrage ist ein JSON-Objekt, das ein Array mit den Issues enthält, die die Kriterien erfüllen. Ein Issue ist ein Objekt, das einige Metadaten wie eine eindeutige Id und mögliche Expansionsoptionen, ein Objekt mit sämtlichen Feldern der Story und ein optionales Objekt mit der Änderungshistorie enthält. Das Feld-Objekt hat 553 Einträge, von denen die meisten leer sind. Es wird zwischen Standardfeldern, die von Jira vorgegeben sind, und benutzerdefinierten Feldern, den Custom Fields, unterschieden. Zu den Standardfeldern gehören unter anderem das Erstellungsdatum, das Abschlussdatum, die Zusammenfassung, die Beschreibung und der Assignee, also die Person, die die Story implementiert. Die Custom Fields enthalten unter anderem das Team, die Storypoints und Kommentare. Das Änderungshistorie-Objekt enthält ein Array mit sämtlichen Änderungen, die nach dem Erstellen am Issue vorgenommen wurden. Jede der Änderungen enthält einen Zeitstempel, den Autor der Änderung und sämtliche Felder, die geändert wurden mit ihrem Zustand vor und nach der Änderung.

Nach Auswahl der Metriken wurden die erforderlichen Daten mit der in dieser Arbeit entwickelten Anwendung aus den Issue-Objekten ausgelesen und in eine für die Analyse geeignete Form gebracht.

4.3 Datenanalyse

Zur Auswertung der gesammelten Daten wurde zunächst beschreibende Statistik genutzt. Je nach Art der Daten wurden sie als Boxplot oder Scatterplot dargestellt, um die Verteilung der Daten zu illustrieren. Anschließend wurde je nach Art der Daten unterschiedlich vorgegangen.

Wenn die betrachtete Metrik in einer Intervall- oder Rationalskala vorliegt, wurden der Korrelationskoeffizient und der Rangkorrelationskoeffizient berechnet. Zusätzlich wurde mit linearer Regression eine Regressionsgerade bestimmt. Dadurch kann ermittelt werden, ob es einen Zusammenhang gibt und wie groß dieser ist.

Bei Metriken in Nominal- oder Ordinalskala wurden statistische Tests genutzt, um zu bestimmen, ob es einen signifikanten Zusammenhang gibt. Dabei wurde zunächst für jede Kategorie geprüft, ob die Daten normalverteilt sind. Dafür wurde der Shapiro-Wilk-Test genutzt. Bei normalverteilten Daten wurden anschließend parametrische Tests verwendet, ansonsten wurden nichtparametrische Tests genutzt. Für Metriken mit zwei Kategorien wurde der t-Test als parametrischer und der Mann-Whitney-Test als nichtparametrischer Test verwendet. Wenn die Metriken mindestens drei Kategorien hatten, wurde zunächst ANOVA als parametrischer bzw. der Kruskal-Wallis-Test als nichtparametrischer Test genutzt, um festzustellen, ob es signifikante Unterschiede zwischen den Kategorien gibt. Wenn signifikante Abweichungen festgestellt wurden, wurde anschließend der Tukey-Test bzw. der Games-Howell-Test verwendet, um die Kategorien paarweise zu vergleichen und zu ermitteln, zwischen welchen Paaren es signifikante Unterschied gibt. Bei allen Test wurde die Signifikanz $\alpha = 0,05$ gewählt. In allen Fällen wurden zweiseitige Tests genutzt,

es wurde also ermittelt, ob es einen signifikanten Unterschiede gibt, nicht ob der Mittelwert einer Kategorie größer oder kleiner als der der anderen ist.

Kapitel 5

Implementierung

In dieser Arbeit sollte neben der Untersuchung von Einflussfaktoren auf die Lead Time auch der Prototyp der Webanwendung von Kudravzev [18] weiterentwickelt werden. Diese sollte einen Überblick über den aktuellen Stand des Prozesses und die Lead Time geben. Der Prototyp wurde um Funktionen erweitert, um verschiedene Metriken mit Bezug zur Lead Time erfassen, statistisch auswerten und darstellen zu können. Zusätzlich wurde er um ein Tool erweitert, mit dem der Status von Epics dargestellt werden kann. Bei der Entwicklung wurde iterativ vorgegangen.

In diesem Kapitel werden zunächst die Anforderungen vorgestellt, die sich für die Erweiterung des Prototypen ergeben. Anschließend wird ein Überblick über die Architektur der Anwendung gegeben. Im folgenden Abschnitt wird auf die Umsetzung der Architektur eingegangen.

5.1 Anforderungen

Die Anforderungen ergeben sich einerseits aus den bisherigen Funktionen des Prototypen, die erhalten bleiben sollen, und andererseits aus den für diese Arbeit nötigen Erweiterungen. Es werden auch nichtfunktionale Anforderungen erfasst, die die Usability der Anwendung verbessern und die weitere Entwicklung vereinfachen sollen. Nachfolgend werden die Anforderungen mit einer kurzen Erläuterung aufgelistet.

Zusätzlich haben sich aus Gesprächen mit Solution Engineers von VWN weitere Anforderungen ergeben. Bei einem ersten Gespräch hat sich herausgestellt, dass das Tool in seiner bisherigen Form keinen Mehrwert bietet. Als ein wichtiges Problem bei der Entwicklung wurde genannt, dass die Epics bei Beginn der Umsetzung nicht ausreichend Stories enthalten. Daraus haben sich die Anforderungen zur Erweiterung der Anwendung um ein Tool ergeben, mit dem der aktuelle Zustand der Epics visualisiert werden kann. Diese Anforderungen wurden in einem weiteren Gespräch nach Vorstellung eines Prototypen weiter verfeinert, sodass mehr Details zu den Stories in den Epics angezeigt werden sollen. Es wurde auch erwähnt, dass Stories, die nicht dem aktuellen Sprint zugeordnet sind, in Jira praktisch unsichtbar sind. Deshalb soll angezeigt werden, ob die Epics inaktive Stories beinhalten.

[RE01] Der Benutzer meldet sich mit seinem Benutzernamen und Passwort für Jira in der Anwendung an.

Die REST-API von Jira verlangt eine Authentifizierung. Dafür werden die gleichen Zugangsdaten wie für das Jira-Webinterface genutzt.

[RE02] Die Anwendung enthält die Ansichten aus dem Dashboard des vorherigen Prototyps [18].

Das Dashboard bestand aus vier Tabs. Der Tab *Aktuelles* enthielt eine Übersicht über die in einem selektierbaren Zeitraum abgeschlossen Stories und deren Lead Time. Er enthielt auch ein Balkendiagramm, das die durchschnittlichen Verweildauern der Stories in den Status darstellt. Der Tab *Bottleneck* enthielt ein Balkendiagramm, das die aktuelle Anzahl der Stories pro Status anzeigt. Der Tab *Batchsize* enthielt Balkendiagramme, die die Anzahl von wöchentlichen Statusübergängen für die letzten vier Wochen visualisieren. Der Tab *History* enthielt Balkendiagramme, die für das letzte Jahr die monatlich fertiggestellten Stories, durchschnittliche monatliche Lead Time und die durchschnittliche Verweildauer der Stories pro Status darstellen.

[RE03] Der Nutzer kann das Team auswählen, für das die Daten angezeigt werden.

Für alle Ansichten soll es möglich sein, die Daten einzelner Teams, aller Teams oder solche, die keinem Team zugeordnet sind, anzuzeigen.

[RE04] Die Anwendung enthält eine Ansicht, die alle Epics im Status *To Do* auflistet. Zu jedem dieser Epics wird der Key, die Zusammenfassung, die Anzahl der Stories, die Anzahl der Storypoints, die Anzahl der inaktiven Stories und eine Fortschrittsanzeige angezeigt.

[RE05] Für jedes Epic aus der Übersicht ([RE04]) gibt es eine Detailansicht. Diese Seite enthält eine Auflistung aller Stories des Epics mit Erstellungsdatum, Key, Zusammenfassung, Storypoints, Status, (letztem) Sprint, Inaktivität und offenen Pull Requests.

[RE06] Der Nutzer kann in der Detailansicht ([RE05]) ein Zieldatum und eine Anzahl von Stories oder Storypunkte angeben. Der Nutzer kann diese Daten in der Anwendung speichern. Zusätzlich zeigt die Anwendung eine Visualisierung des bisherigen, des vorhergesagten und des erwünschten Fortschritts.

Aus den Gesprächen mit den Entwicklern hat sich ergeben, dass Epics häufig nicht rechtzeitig mit Stories gefüllt werden. Die Übersicht ([RE04]) soll einen Überblick über die aktuell offenen Epics geben. Die Detailansicht ([RE05]) dient dazu, den aktuellen Zustand der Epics zu visualisieren und zu verdeutlichen, ob Handlungsbedarf besteht. Stories sind inaktiv, wenn die letzte Änderung länger als vier Wochen in der Vergangenheit liegt.

[RE07] Die Anwendung kann die für bestimmte Metriken benötigten Daten aus der Jira REST-API extrahieren.

[RE08] Die Anwendung kann für die Daten ([RE07]) die angemessenen Maße der beschreibenden Statistik berechnen und statistische Tests durchführen.

[RE09] Die Anwendung kann die Ergebnisse der statistischen Auswertung ([RE08]) visualisieren.

Aus den Jira-Daten müssen die für die betrachtete Metrik relevanten Daten extrahiert werden. Je nach Art der Daten werden anschließend eine Regressionsanalyse oder statistische Tests durchgeführt. Bei den Tests müssen abhängig von der Anzahl an Kategorien und der Normalität der Daten die passenden Tests ausgewählt werden. Aufgrund der explorativen Vorgehensweise werden die Metriken nicht in den Anforderungen festgelegt, sondern die Anwendung wird bei Bedarf um die gewünschte Metrik erweitert.

[RE10] Die Erweiterung der Anwendung um neue Metriken erfordert Änderungen an nicht mehr als drei Stellen im bestehenden Quellcode.

Da die Anwendung aufgrund der Vorgehensweise häufig um neue Metriken erweitert wird, soll die Entwicklung nicht durch umfangreiche Änderungen erschwert werden. Wenn die Metrik neue statistische Methoden oder Visualisierungen erfordert, darf diese Anforderung verletzt werden.

[RE11] Die Anwendung soll modular aufgebaut sein.

Da die Anwendung ein Prototyp ist, kann nicht ausgeschlossen werden, dass große Teile davon verändert werden. Um das zu erleichtern, soll die Anwendung so aufgebaut sein, dass sich Module unabhängig voneinander wiederverwenden lassen.

[RE12] In 95 % der Fälle soll die Anwendung die gewünschte Ansicht in unter 5 Sekunden nach dem Aufruf darstellen.

Der vorherige Prototyp [18] hatte Ladezeiten von bis zu mehreren Minuten. Diese sind für das beabsichtigte Einsatzgebiet dieser Anwendung nicht akzeptabel.

5.2 Architektur

Die Architektur der Anwendung ist darauf ausgelegt, die Erweiterung der Anwendung um zusätzliche Metriken zu vereinfachen. Dazu wird die Anwendung zunächst in ein Backend und ein Frontend aufgeteilt. Das ist auch erforderlich, da direkte Zugriffe auf die Jira REST-API aus dem Webbrowser durch die Same-Origin-Policy verhindert werden. Das Backend ist für die Beschaffung der Jira-Daten und die Berechnung der betrachteten

Metriken und den zugehörigen statistischen Maßen und Tests zuständig. Das Frontend visualisiert die bereitgestellten Daten im Webbrowser.

Die Kommunikation zwischen Frontend und Backend findet über ein REST-ähnliches Interface statt. Das Backend bietet die nachfolgenden Ressourcen. Bei allen Ressourcen außer `/login` muss ein `Authorization` Header mit einem Autorisierungstoken gesetzt sein.

`POST /login` Überprüft die Zugangsdaten des Nutzers. Benötigt die Parameter `user` für den Nutzernamen, `password` für das Passwort und `projekt` für das betrachtete Projekt. Gibt bei Erfolg einen Autorisierungstoken, bei Misserfolg eine Fehlermeldung zurück.

`GET /{project}/{team}/current/{period}` Gibt die Daten für die Aktuelles-Ansicht für das angegebene Projekt, Team und den ausgewählten Zeitraum zurück.

`GET /{project}/{team}/bottleneck` Gibt die Daten für die Bottleneck-Ansicht für das angegebene Projekt und Team zurück.

`GET /{project}/{team}/batchsize` Gibt die Daten für die Batchsize-Ansicht für das angegebene Projekt und Team zurück.

`GET /{project}/{team}/history` Gibt die Daten für die History-Ansicht für das angegebene Projekt und Team zurück.

`GET /{project}/{team}/epics` Gibt die Daten für die Übersicht der Epics des angegebenen Projekts und Teams zurück.

`GET /epic/{epicKey}` Gibt die Daten für die Detailansicht des angegebenen Epics zurück.

`GET /epicgoal/{epicKey}` Gibt die Zieldaten für das angegebene Epic zurück.

`POST /epicgoal` Speichert die Zieldaten für ein Epic. Benötigt die Parameter `epicKey` für den Key des Epics, `type` für die Art des Ziels, Anzahl Stories oder Storypoints, `goal` für den Zielwert und `date` für das Zieldatum.

`GET /statistics/{project}/{metric}/{team}` Gibt die Daten der statistischen Auswertung der gewählten Metrik für das angegebene Projekt und Team zurück.

Eine Übersicht der Architektur des Backends ist in Abbildung 5.1 dargestellt. Die Index-Komponente fungiert als Router und nimmt alle Anfragen entgegen, ruft die zuständigen Komponenten auf und gibt deren Antwort zurück. Die Login-Komponente überprüft, ob die Zugangsdaten den Zugriff auf das gewählte Projekt gewähren. Die Komponenten Current, Batchsize, Bottleneck und History senden Anfragen an Jira, verarbeiten die Antworten

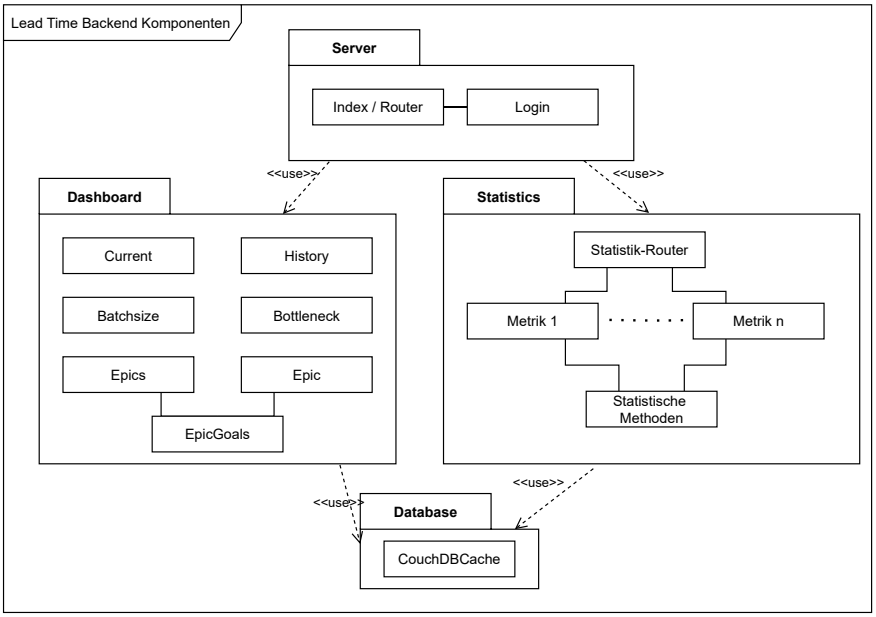

Abbildung 5.1: Übersicht der Backend-Architektur.

und erzeugen die Daten, die für die jeweilige Ansicht benötigt werden. Die EpicGoals-Komponente speichert die Zielvorgaben zu bestimmten Epics in der Datenbank und lädt diese daraus. Die Epics- und die Epic-Komponenten verarbeiten die Antworten auf Anfragen nach allen Epics eines Teams bzw. ein bestimmtes Epic, zusätzlich integrieren sie die Zieldaten für die Epics aus der Datenbank. Die Statistics-Komponente bereitet die Anfrage an die Jira-API vor und übergibt sie der gewünschten Metrik. Die Metrik-Komponenten führen diese Anfragen aus, extrahieren aus den Antworten die benötigten Daten und führen die geeigneten statistischen Methoden auf diesen Daten aus. Die Statistische-Methoden-Komponente stellt Funktionen für beschreibende Statistik und statistische Tests bereit.

Die oben genannten Komponenten sind ausreichend für die Funktionalität der Anwendung, allerdings kann damit die Anforderung [RE12] nicht erfüllt werden. Dafür wird die Datenbank-Komponente benötigt, die von allen andern Komponenten als Cache genutzt wird. In diesem Cache werden die Ergebnisse der anderem Komponenten und die Antworten auf Jira-Anfragen zwischengespeichert. Wenn jetzt eine Anfrage an das Backend gestellt wird, werden zuerst die Daten aus dem Cache geladen und zurückgegeben. Anschließend wird überprüft, ob die Daten veraltet sind und gegebenenfalls neu berechnet. Das erspart das minutenlange Warten auf die Jira-Anfragen. Der Ablauf ist in Abbildung 5.2 dargestellt.

Für die Umsetzung des Frontends wurde React[33] genutzt, es besteht also aus React-Komponenten. Eine vereinfachte Übersicht der Architektur ist in Abbildung 5.3 dargestellt. Um das Aufrufen der verschiedenen Ansichten durch Eingabe einer URL zu ermöglichen,

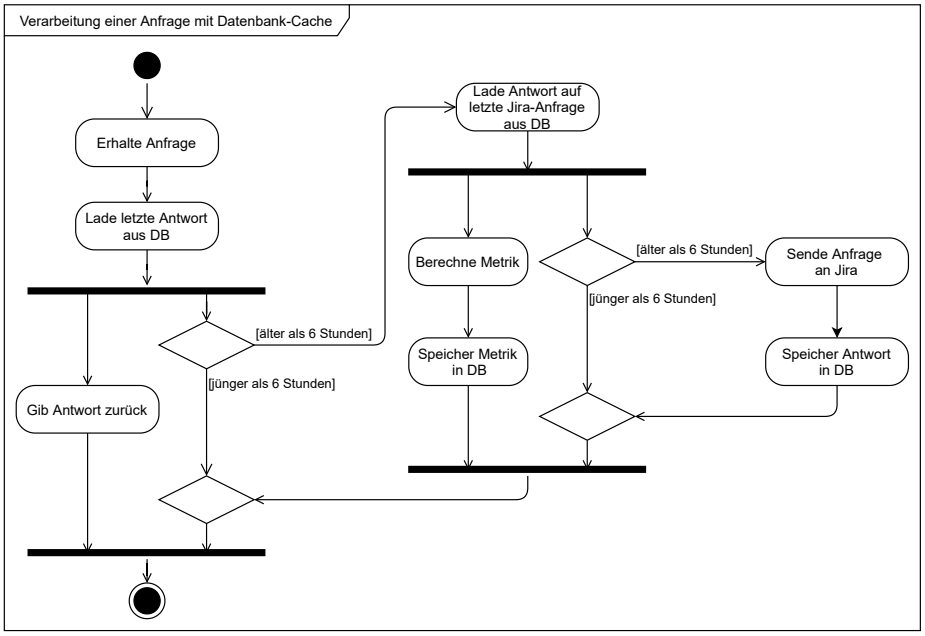

Abbildung 5.2: Ablauf der Verarbeitung einer Anfrage mit Nutzung des Datenbank-Caches.

wird ein Router benötigt. Dieser wird in den verschiedenen Switch-Komponenten genutzt. Die Context-Komponenten verwalten den globalen Zustand der Anwendung und stellen Methoden bereit, mit denen er verändert werden kann. Die Login-Komponente stellt Eingabefelder für Nutzername und Passwort bereit und führt den Login-Vorgang durch. Die Komponenten zur Darstellung der Daten vom Backend wurden in die Pakete Dashboard und Statistics aufgeteilt. Beide Pakete enthalten Menü-Komponenten, die die Navigation zu den verschiedenen Ansichten ermöglichen, und eine Switch-Komponente, die die gewählte Ansicht ausliefert. Das Dashboard-Paket enthält die Current-, Batchsize-, Bottleneck- und History-Komponenten für die Ansichten aus dem vorherigen Prototypen. Zusätzlich enthält es die Epics- und EpicDetails-Komponenten für die Übersicht der offenen Epics bzw. die Detailansicht eines Epics. Das Statistics-Paket enthält MetricView-Komponenten, die Ansichten einer oder einiger zusammenhängender Metriken bereitstellen. Außerdem enthält es verschiedene ChartPanel-Komponenten für die Visualisierung der Daten und StatisticsPanel-Komponenten, um die Ergebnisse der statistischen Tests anzuzeigen.

5.3 Umsetzung der Architektur

Die im vorherigen Abschnitt vorgestellte Architektur wurde sowohl für das Backend als auch für das Frontend als JavaScript-Anwendungen implementiert, die die Node.js [25]

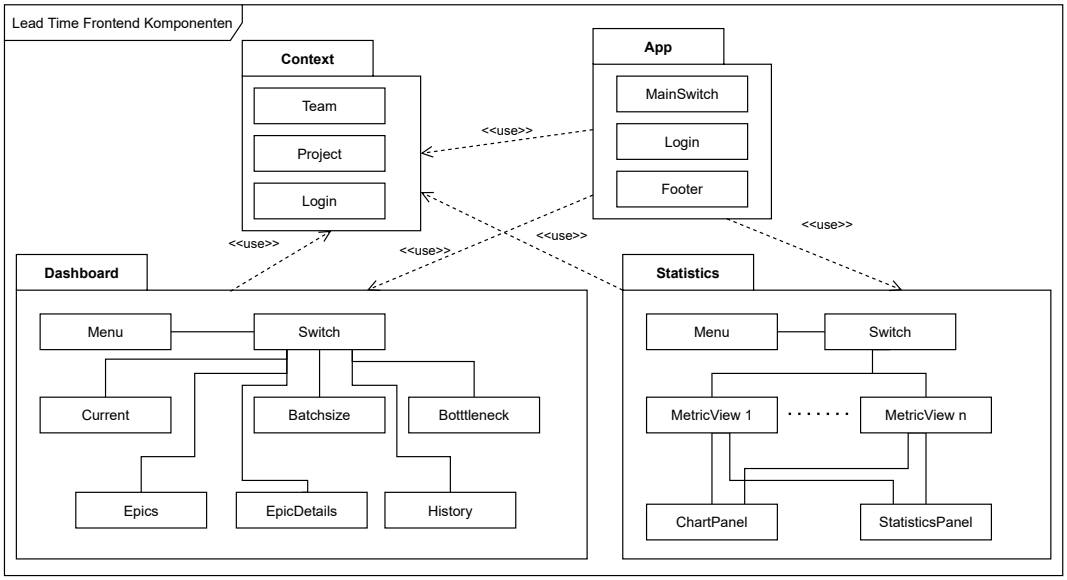

Abbildung 5.3: Übersicht über die Frontend-Architektur.

Laufzeitumgebung nutzen. Im Folgenden werden die wichtigsten Aspekte der Implementierung der Webanwendung und die resultierenden Oberflächen vorgestellt.

5.3.1 Backend

Das Backend wurde als Webserver auf Basis von Express [9] entwickelt. Express wird in der Index-Komponente für einen HTTP-Server und zur Annahme von eingehenden Anfragen verwendet. Zur Verarbeitung der Anfragen werden die enthaltenen Parameter extrahiert und an die zuständige Komponente weitergeben. Die Antwort der Komponente bzw. eine möglicherweise auftretende Fehlermeldung wird dann als Antwort auf die Anfrage zurückgegeben. Der Express-Server ist in der Lage, mehrere Anfragen parallel zu verarbeiten. Dies ist zum Einen wichtig, damit mehrere Nutzer gleichzeitig darauf zugreifen können, zum anderen da einige Ansichten des Frontends mehrere Anfragen parallel senden.

Bei den Komponenten für das Dashboard wurden aus dem vorherigen Prototypen die Funktionen für die Verarbeitung der Jira-Daten für die Current-, Batchsize-, Bottleneck- und History-Ansichten mit wenigen Änderungen übernommen. Die wichtigste Änderung daran war die Erweiterung um den Datenbank-Cache. Die Epics-Komponente sendet eine Suchanfrage für alle Epics im Status *To Do* an die Jira REST-API. Zusätzlich werden für jedes Epic Anfragen nach den enthaltenen Stories gesendet, da diese nicht im Epic enthalten sind. Aus den Antworten werden dann für die Epics jeweils die relevanten Daten extrahiert. Die Daten werden mit den in den Datenbank gespeicherten Werten für die Ziele der Epics

kombiniert und in einem Antwort-Objekt zurückgegeben. Die Epic-Komponente fragt die gleichen Daten an, allerdings nur für ein Epic. Es werden zusätzliche Daten zu den Stories bereitgestellt, u. a. auch der Status von Merge-Requests aus dem Versionsverwaltungssystem. Die EpicsGoal-Komponente lädt die Werte der Ziele aus der Datenbank bzw. speichert diese.

Im Statistics-Paket werden alle Anfragen zunächst von der getStatistics-Komponente angenommen. Diese erzeugt die für die gewünschte Metrik passende Anfrage, dies ist, bis auf wenige Ausnahmen, die in Abschnitt 4.2 erwähnte. Anschließend führt sie die Funktion aus, die die Metrik berechnet und gibt die Antwort an die aufrufende Index-Komponente zurück. Zum Hinzufügen einer neuen Metrik muss diese nur in der getStatistics-Komponente importiert werden und an der entsprechenden Stelle mit der passenden Anfrage aufgerufen werden. Die Metric-Komponenten führen die ihnen übergebenen Anfragen aus, extrahieren aus den Antworten die erforderlichen Daten, führen auf diesen Daten die angemessenen Statistik-Funktionen aus und geben die Ergebnisse als Antwort-Objekt zurück. Die Statistik-Funktionen werden von der Statistics-Komponente bereitgestellt. Diese bietet zum Einen Funktionen, die für die Metrik-Daten die Maße der beschreibenden Statistik berechnen und statistische Tests durchführen. Zum anderen stellt sie eine einheitliche Schnittstelle für die benötigten Funktionen aus den verschiedenen genutzten Statistikbibliotheken bereit.

Mit den bisher beschriebenen Komponenten können die funktionellen Anforderungen erfüllt werden. Für die Erfüllung der Anforderung [RE12] muss die Antwortzeit der Jira REST-API berücksichtigt werden. Diese Antwortzeit kann mehrere Minuten betragen, bei mehreren parallelen Anfragen wurden teilweise mehr als zehn Minuten beobachtet. Das kann neben der langen Wartezeit für den Nutzer auch zu unerwünschten Verbindungsabbrüchen durch Timeouts führen. Um das zu verhindern, wurde mit der Datenbank-Komponente ein Cache für die Antworten der anderen Komponenten und von Jira-Anfragen implementiert. Dabei wird CouchDB [7] als Datenbank genutzt, da es eine leichtgewichtige Dokumentendatenbank ist, die die Dokumente als JSON speichert. Das Format wird auch für die Kommunikation mit der Jira REST-API und mit dem Frontend benutzt. Wenn beispielsweise die getStatistics-Komponente aufgerufen wird, übergibt sie die Anfrage zunächst an die Datenbank-Komponente und diese lädt die letzte Antwort aus dem Cache. Diese Antwort wird sofort an die Index-Komponente zur Beantwortung der Anfrage zurückgegeben. Wenn die Antwort älter als sechs Stunden ist, erzeugt die getStatistics-Komponente asynchron eine Funktion, die die gewählte Metrik berechnet, und übergibt sie der Datenbank-Komponente. Diese führt dann die Funktion aus und speichert das Ergebnis im Cache. Ähnlich wird auch bei Anfragen an die Jira REST-API, die während der Ausführung der Funktion gesendet werden, vorgegangen.

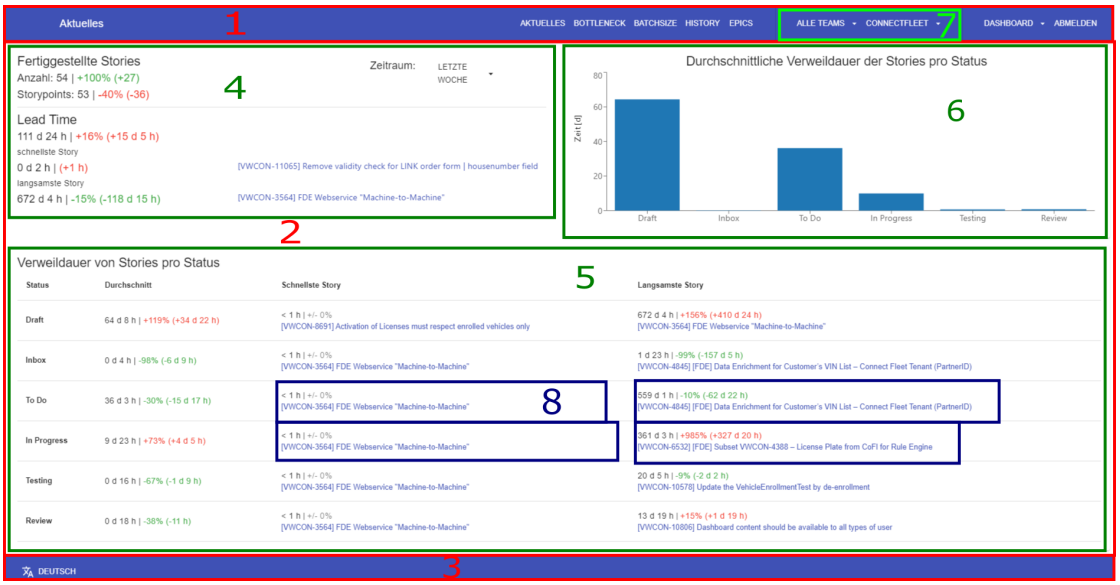

Abbildung 5.4: Zusammensetzung der Aktuelles-Ansicht aus Komponenten. Die einzelnen Komponenten sind: 1) Navigationsmenü, 2) Current-Ansicht, 3) Footer-Komponente, 4) Übersichts-Panel, 5) Detail-Panel, 6) Diagramm-Panel, 7) Team- und Projekt-Auswahl, 8) Detail-Zelle. Diese Komponenten sind wiederum aus weiteren Komponenten, z. B. von Material-UI oder von Visualisierungsbibliotheken, und aus HTML-Elementen zusammengesetzt.

5.3.2 Frontend

Für die Benutzeroberfläche der Anwendung wurde eine Webanwendung basierend auf React [33] entwickelt. React-Anwendungen sind typischerweise aus Komponenten zusammengesetzt, die den Zustand der Komponente verwalten und einen Teil der Ansicht rendern. Abbildung 5.4 zeigt als Beispiel, wie die Aktuelles-Ansicht aus weiteren Komponenten zusammengesetzt ist. Für ein einheitliches Erscheinungsbild wurde Material-UI [22] verwendet, das viele Komponenten im Material Design [21] bereitstellt. Es wurde außerdem eine Internationalisierungsbibliothek verwendet, mit der sämtliche Texte sowohl auf Deutsch als auch auf Englisch angezeigt werden können. Die Auswahl der Sprache ist jederzeit im Footer möglich.

Für die Verwaltung des globalen Zustands wurde Recoil [35] verwendet. Recoil stellt Datenstrukturen zum atomaren Speichern eines Zustands bereit, auf den aus beliebigen React-Komponenten zugegriffen werden kann. Globale Zustände wurden für den Login-Status, das ausgewählte Team und Projekt und die gewählte Zeitspanne in der Aktuelles-Ansicht verwendet.

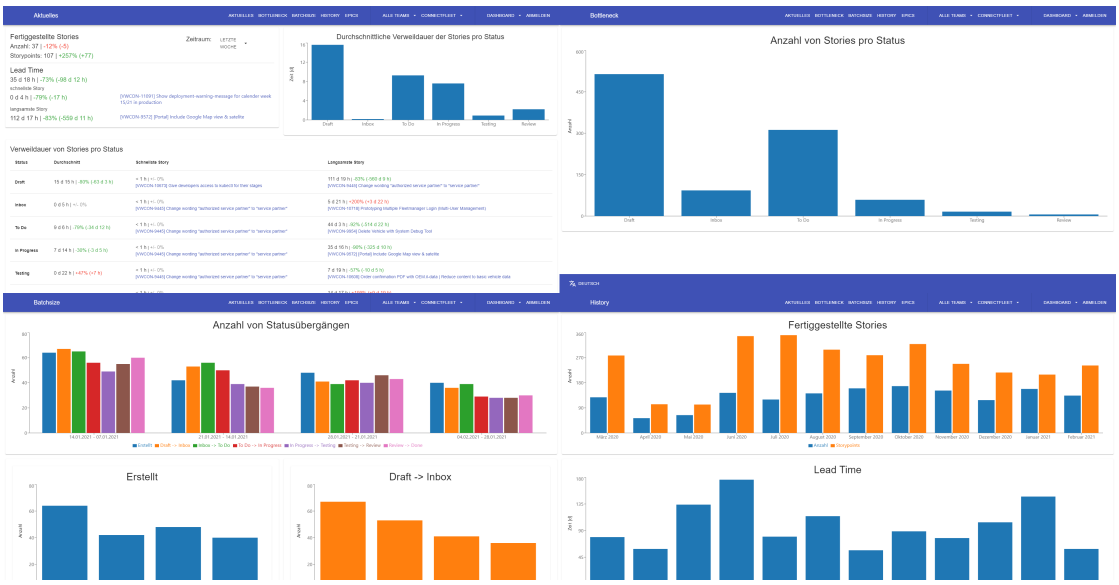

Abbildung 5.5: Die Ansichten Aktuelles, Bottleneck, Batchsize und History.

Die Login-Ansicht wird aufgerufen, wenn eine beliebige Ansicht der Anwendung aufgerufen werden soll und der Nutzer nicht eingeloggt ist. Sie besteht aus Eingabefeldern für Nutzernamen und Passwort und Auswahlelementen für das Team und das Projekt. Zum einloggen wird eine POST-Anfrage an die `/login` Ressource gesendet. Wenn die Anfrage erfolgreich ist, wird der Autorisierungstoken aus der Antwort im Browser gespeichert, sodass beim nächsten Aufrufen der Anwendung kein erneutes Einloggen erforderlich ist. Wenn die Anfrage fehlschlägt, wird eine Fehlermeldung angezeigt. Wenn der Nutzer angemeldet ist, kann er sich jederzeit mit dem Abmelden-Button im Menü ausloggen, wodurch der gespeicherte Token gelöscht wird.

Nach dem Einloggen wird zunächst das Dashboard aus dem vorherigen Prototypen mit der Aktuelles-Ansicht angezeigt. Das Menü erlaubt die Auswahl der anderen Ansichten des Dashboards, zusätzlich die Auswahl des Teams und des Projekts. Die Aktuelles-, Batchsize-, Bottleneck- und History-Ansichten (siehe Abbildung 5.5) sind vom Aufbau und der Funktionalität identisch mit dem vorherigen Prototypen. Sie wurden auf das Komponentensystem von React und die Nutzung von Material-UI angepasst. Außerdem wurde für die Visualisierungen Recharts [34] statt der ursprünglich verwendeten Bibliothek genutzt. Die Daten für die Ansichten werden von den entsprechenden Ressourcen des Backends bezogen.

Das Dashboard wurde um die Epics-Ansicht erweitert, die in Abbildung 5.6 dargestellt ist. In dieser Ansicht wird eine Übersicht aller Epics angezeigt, die sich im Status *To Do* befinden. Um die Daten der dargestellten Epics zu erhalten, wird eine GET-Anfrage an die Ressource `/{project}/{team}/epics` im Backend gesendet. Für jedes Epic wird der Key,

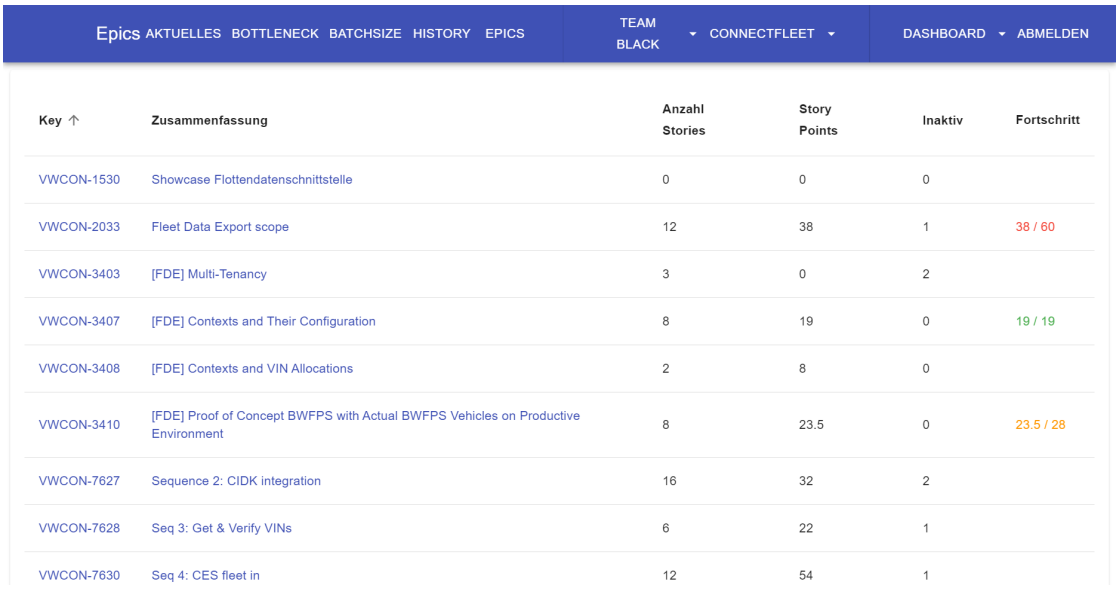

Abbildung 5.6: Die Epics-Ansicht zeigt die Übersicht aller Epics im Status *To Do*.

die Zusammenfassung, die Anzahl von Stories im Epic, die Summe der Storypoints dieser Stories, die Anzahl der inaktiven Stories und der Fortschritt zum gesetzten Ziel angezeigt. Der Fortschritt wird nur angezeigt, wenn für das Epic ein Ziel festgelegt wurde. Das Ziel dieser Ansicht ist es, mögliche Verzögerungen beim Füllen der Epics mit Stories frühzeitig zu erkennen. Durch Auswählen eines Epics gelangt der Benutzer zur Epic-Ansicht.

Die Epic-Ansicht ist in Abbildung 5.7 dargestellt. Sie zeigt die Details eines bestimmten Epics. Für die erforderlichen Daten wird eine GET-Anfrage an die Ressource `/epic/{epicKey}` gesendet. Die Ansicht enthält eine Visualisierung des Fortschritts des Epics, d. h. es wird der Verlauf der enthaltenen Storypoints bzw. Stories über die Zeit seit Erstellung des Epics angezeigt. Es ist auch möglich, ein Ziel festzulegen, bestehend aus Zieldatum, Wert, der erreicht werden soll und Art des Ziels, also Anzahl der Storypoints oder der Stories. Wenn ein Ziel festgelegt wurde, wird auch ein idealer Verlauf zum Erreichen des Ziels und ein projizierter Verlauf der bisherigen Entwicklung zum Zieldatum angezeigt. Außerdem enthält die Ansicht eine Auflistung aller Stories des Epics mit Erstellungsdatum, Key, Zusammenfassung, Storypoints, Status, letztem zugeordneten Sprint, ob die Story in den letzten vier Wochen aktiv war und dem Status von zugehörigen Pullrequests aus der Versionsverwaltung.

Außer den Ansichten des Dashboards können vom Menü auch Statistik-Dashboards erreicht werden. Die Statistik-Ansichten wurden auf mehrere Dashboards aufgeteilt, da sonst in der Menüleiste nicht ausreichend Platz für die Buttons aller Ansichten ist. Die Statistik-Ansichten zeigen Visualisierungen der betrachteten Metriken und die Ergebnisse

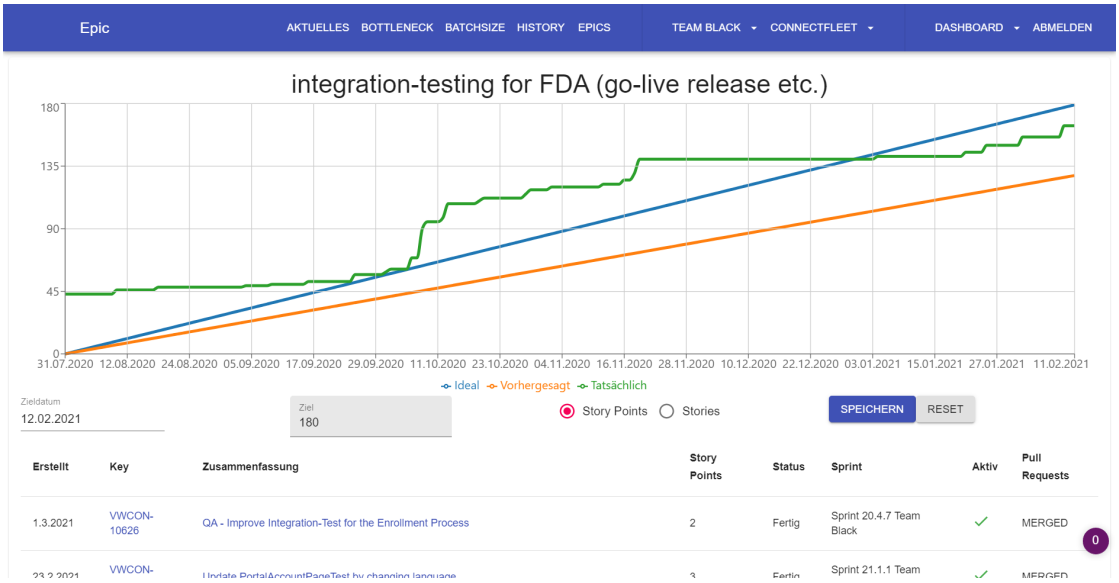

Abbildung 5.7: Die Epic-Ansicht zeigt die Detailansicht eines Epics.

der statistischen Analysen. Es werden teilweise mehrere verwandte Metriken in einer Ansicht angezeigt. Abbildung 5.8 zeigt ein Beispiel für die Statistik-Ansicht.

Für die Visualisierungen wurden mehrere Komponenten implementiert. Die Boxplot-Komponente basiert auf visx [46] und zeigt Boxplots für alle Kategorien der Metrik. Es gibt auch eine Variante, die die Ausreißer ausblendet, da durch diese in manchen Fällen die Größe der Boxen nicht mehr zu erkennen ist. Für Metriken in Intervall- oder Rationalskalen gibt es eine Scatterplot-Komponente, die auf Recharts basiert. Zusätzlich gibt es noch eine Variante des Scatterplots mit veränderlicher Größe der dargestellten Punkte, um die Abhängigkeit eines Werts von zwei Variablen zu visualisieren und ein Balkendiagramm für Größen, bei denen die Bildung von Mittelwerten nicht sinnvoll ist.

Abgesehen von den Visualisierungen werden in den Statistik-Ansichten auch die Ergebnisse der statistischen Analysen angezeigt. Dafür gibt es zwei Komponenten für die verschiedenen Arten von Metriken. Das CorrelationPanel zeigt die Ergebnisse der Korrelationsanalyse, also die den Korrelationskoeffizienten, den Rangkorrelationskoeffizienten und die Werte der linearen Regression. Das StatisticsPanel zeigt die Ergebnisse der statistischen Tests, dabei wird der durchgeführte Test und der dazugehörige p-Wert angezeigt. Zusätzlich gibt es ein Panel für beschreibende Statistik, das Mittelwert, Standardabweichung und Median anzeigt.

Eine Statistik-Ansicht für eine neue Metrik kann implementiert werden, indem eine neue Komponente aus den passenden Visualisierungs- und Statistik-Komponenten zusammengesetzt wird. Sofern die Metrik im Backend berechnet wurde, muss noch eine Anfrage an die

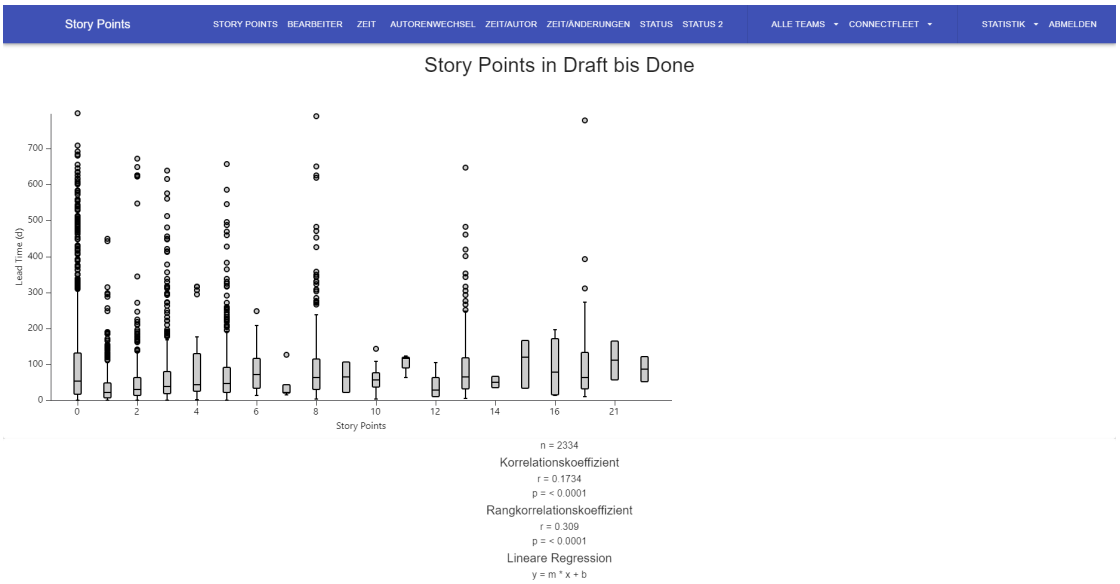

Abbildung 5.8: Ein Beispiel für die Statistik-Ansicht mit einer Darstellung der betrachteten Metrik als Boxplots und den Ergebnissen der statistischen Auswertung, in diesem Fall eine Korrelationsanalyse. Es können auch mehrere verwandte Metriken in einer Ansicht anzeigt werden.

passende Ressource gesendet werden. Um auf die Ansicht zugreifen zu können, muss noch ein Eintrag im Switch hinzugefügt werden, zusätzlich wird ein Button im Menü benötigt, der auf diesen Eintrag verweist.

5.4 Empfehlungen für die Weiterentwicklung des Prototypen

Die Webanwendung wurde nicht mit dem Ziel entwickelt, in einer Produktionsumgebung eingesetzt zu werden. Deshalb gibt es einige Aspekte, die bei der Entwicklung eine geringere Priorität hatten. Nachfolgend gibt es eine kurze Übersicht der Bereiche, die vor einem produktiven Einsatz der Anwendung Aufmerksamkeit erfordern.

Zunächst sollte sowohl im Frontend als auch im Backend HTTPS eingesetzt werden. Das kann für das Backend erreicht werden, indem der von Express bereitgestellte HTTP-Server durch einen HTTPS-Server ersetzt wird, für das Frontend muss der Production-Build von einem HTTPS-Server ausgeliefert werden.

Die Fehlerbehandlung wurde im Prototypen vernachlässigt. Es werden nur erwartete Fehler abgefangen, wie ein „Not Found Error“ der Datenbank, alle anderen werden ans Frontend weitergegeben, wo es zu unerwartetem Verhalten kommen kann, da die Fehler auch da nicht behandelt werden. Eine Überprüfung, ob die Antworten auf Anfragen das

erwartete Format haben, sollte zusätzlich implementiert werden. Bei zu vielen parallelen Anfragen kann es vorkommen, dass das Backend aufgrund des Überschreitens des Heap-Speichers abstürzt. Um das zu verhindern, könnte die Anzahl der gleichzeitig möglichen Verbindungen begrenzt werden, was aber wiederum zu Problemen mit der Kommunikation zwischen Frontend und Backend führen kann.

Im Frontend kann man Teams und Projekte auswählen. Allerdings werden die Teams nicht nach Projekten unterschieden. Das ist in der aktuellen Implementierung kein Problem, weil nur ein Projekt betrachtet wird. Wenn mehrere Projekte unterstützt werden sollen, dann sollten die Teams nach Projekten getrennt ausgewählt werden können, da nicht davon ausgegangen werden kann, dass alle Projekte die gleichen Namenskonventionen für Teams haben. Dafür muss im Frontend die Team-Komponente angepasst werden, das Backend muss nicht verändert werden.

Die Zeit zwischen Anfrage und Antwort des Servers konnte durch den Datenbank-Cache auf einige Millisekunden reduziert werden, dadurch werden die Seiten ohne merkliche Verzögerung aufgebaut. Allerdings benötigt der Seitenaufbau in einigen Fällen deutlich länger, vor allem bei Scatterplots für alle Teams. In diesen Fällen müssen mehr als 4000 Punkte gerendert werden, was den Prozessor auslastet, wodurch die Seite während dieser Zeit nur langsam auf Eingaben reagiert. Das kann möglicherweise durch serverseitiges Rendern behoben werden.

In der aktuellen Implementierung werden vier verschiedene Statistikbibliotheken genutzt, die teilweise nicht mehr gepflegt werden. Vor diesem Hintergrund könnte es sinnvoll sein, das Backend in einer Programmiersprache zu implementieren, für die es umfangreichere Bibliotheken gibt, wie z. B. Python. Durch die Aufteilung in Frontend und Backend müsste das Frontend dabei nicht verändert werden, solange das Backend weiterhin die gleichen Schnittstellen bereitstellt.

Kapitel 6

Ergebnisse

Nachdem im vorherigen Kapitel die Implementierung der Anwendung vorgestellt wurde, wird sie nun genutzt, um die Jira-Daten auszuwerten. Das Ziel dieser Studie ist es, herauszufinden, welche Faktoren Einfluss auf die Lead Time haben. Die Lead Time wird hier als die Zeit vom Erstellungszeitpunkt einer Story bis zu dem Zeitpunkt, zu dem sie in den Status *Done* versetzt wird, festgelegt. Das widerspricht der Definition der Lead Time als Zeit von der Äußerung einer Anforderung bis zu deren Auslieferung, aber die Prozessschritte außerhalb von Jira können mit der Anwendung nicht gemessen werden.

Aufgrund des explorativen Vorgehens in dieser Arbeit wurden die untersuchten Metriken nicht im Vorhinein festgelegt, sondern basierend auf den vorherigen Ergebnissen ausgewählt. Für jede Metrik wurde die Anwendung um Komponenten erweitert, um sie zu berechnen und zu visualisieren. In diesem Kapitel werden diese Metriken und die statistischen Auswertungen von diesen vorgestellt. Im Folgenden werden Personen, die Änderungen an der Story in Jira vornehmen, als Autoren bezeichnet. Die Entwickler, denen die Story zugewiesen wurde, werden Bearbeiter genannt.

6.1 Storypoints

Der erste Bereich, der betrachtet wurde, sind Storypoints. Storypoints sind Abstraktionen des erwarteten Aufwands zur Umsetzung einer Story. Daher war die Vermutung, dass die Lead Time mit der Anzahl von Storypoints korreliert. Für die Analyse der Metrik wurden die Anzahl von Storypoints und die Lead Time aus den Jira-Daten extrahiert. Es wurde eine Korrelationsanalyse durchgeführt. Bei der Analyse wurden die Stories ignoriert, denen keine Punkte zugewiesen wurden.

Tabelle 6.1: Ergebnisse der Korrelationsanalyse für Lead Time abhängig von den Storypoints, $n = 2313$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	0,1764	<0,0001
Rangkorrelationskoeffizient	0,3101	<0,0001

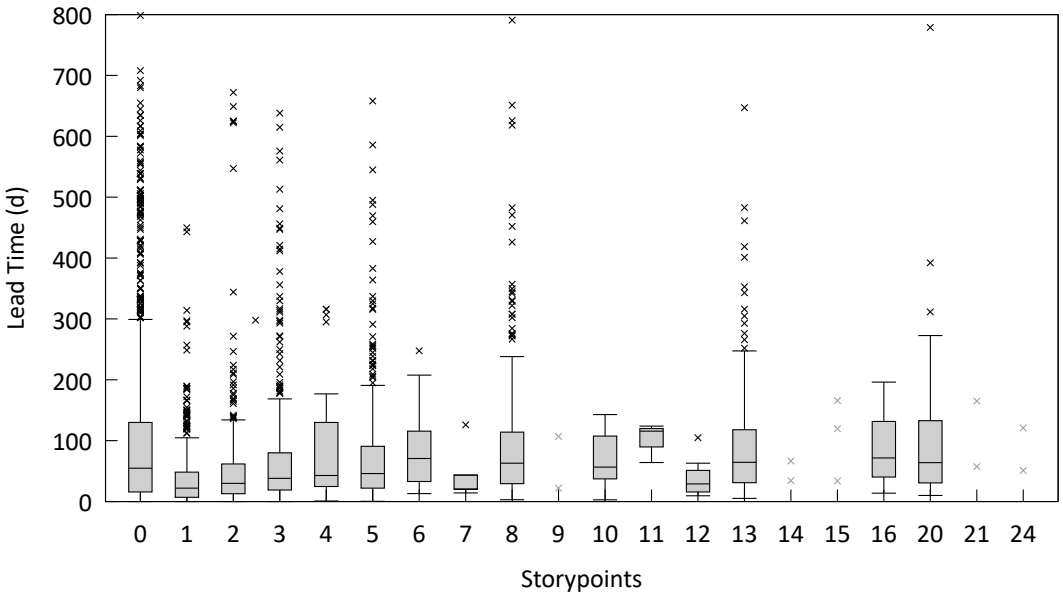

Abbildung 6.1: Boxplots der Lead Time abhängig von den Storypoints. Stories, denen keine Punkte zugewiesen wurden, werden mit null Punkten dargestellt.

In Abbildung 6.1 ist die Lead Time abhängig von den Storypoints dargestellt. Die Ergebnisse der Korrelationsanalyse können der Tabelle 6.1 entnommen werden. Man sieht für beide Koeffizienten eine statistisch signifikante Korrelation. Für den Korrelationskoeffizienten ist es eine sehr schwache, für den Rangkorrelationskoeffizienten eine schwache Korrelation. Die stärkere nichtlineare Korrelation lässt sich dadurch erklären, dass die Storypoints als abstrakte Maße keiner linearen Skala folgen. Zusätzlich kann der Wert eines Punkts sich zwischen den Teams unterscheiden. Die lineare Regression hat die Gerade $\mathit{LeadTime} = 3{,}8785\,\mathrm{d} \cdot \mathit{StoryPoints} + 50{,}8357\,\mathrm{d}$ ergeben, also wäre die Lead Time für jeden Punkt um knapp vier Tage länger. Allerdings ist dieser Wert aufgrund der schwachen linearen Korrelation nur bedingt aussagefähig.

Die Storypoints werden erst bei der Sprintplanung vergeben und schätzen den Aufwand der Umsetzung der Story. In der Lead Time sind aber auch Prozessschritte enthalten, die nicht direkt mit der Implementierung verbunden sind. Deshalb wurde als Nächstes die Zeit betrachtet, die die Story im Status *In Progress* verbringt, da dies der Zeit entsprechen sollte, in der aktiv an der Umsetzung der Story gearbeitet wird. Für diese Metrik wurde die Zeitspanne zwischen dem ersten Mal, wenn die Story auf *In Progress* gesetzt wird und dem letzten Mal, wenn sie von *In Progress* auf einen anderen Status gesetzt wird, ermittelt. Stories, die nie in den Status *In Progress* gesetzt wurden, wurden herausgefiltert, weshalb die Anzahl der Stories niedriger ist als bei der vorherigen Metrik.

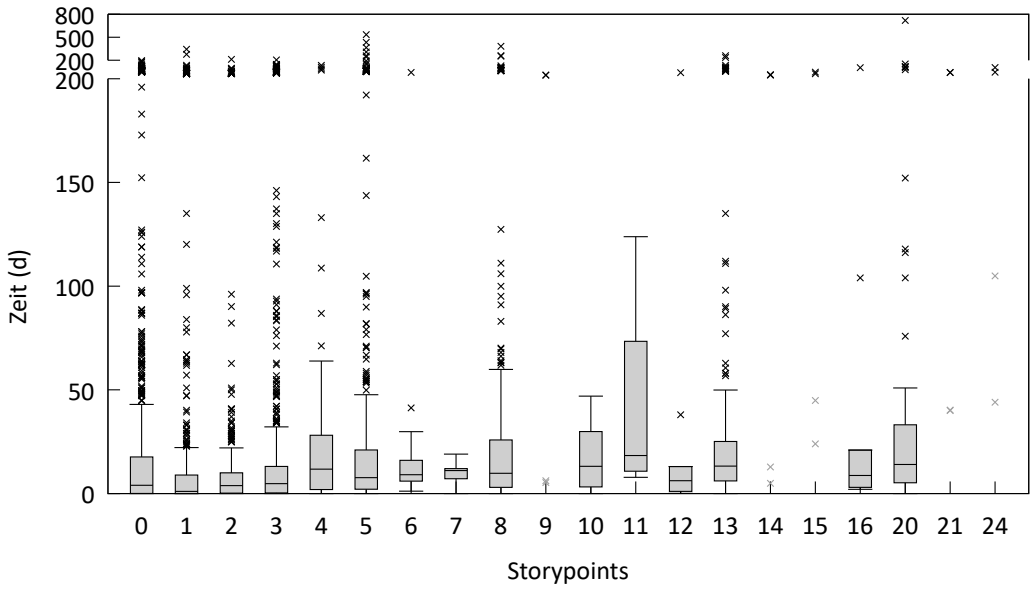

Abbildung 6.2: Boxplots für die Zeit der Story im Status *In Progress* abhängig von den Storypoints. Stories, denen keine Punkte zugewiesen wurden, werden mit null Punkten dargestellt.

Abbildung 6.2 zeigt die Zeit, die die Stories im Status *In Progress* waren, abhängig von der Anzahl der Storypoints. In Tabelle 6.2 sind die Ergebnisse der Korrelationsanalyse aufgelistet. Beide Korrelationskoeffizienten sind – entgegen der Erwartung – geringer als bei der Betrachtung des gesamten Prozesses. Aus der linearen Regression folgt die Funktion $Zeit_{\text{InProgress}} = 1{,}5309\,\text{d} \cdot StoryPoints + 9{,}2769\,\text{d}$, die aber wieder nur bedingte Aussagekraft hat. Bei manchen Teams lassen sich allerdings stärkere Korrelationen beobachten. Bei Team Black ist der Rangkorrelationskoeffizient $r = 0{,}4458$ ($p < 0{,}0001$, $n = 321$), d. h. bei diesem Team hängt die Anzahl der Storypoints stärker mit der Implementierungszeit zusammen.

Tabelle 6.2: Ergebnisse der Korrelationsanalyse für die Zeit der Story im Status *In Progress* abhängig von den Storypoints, $n = 2047$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	0,1647	<0,0001
Rangkorrelationskoeffizient	0,2989	<0,0001

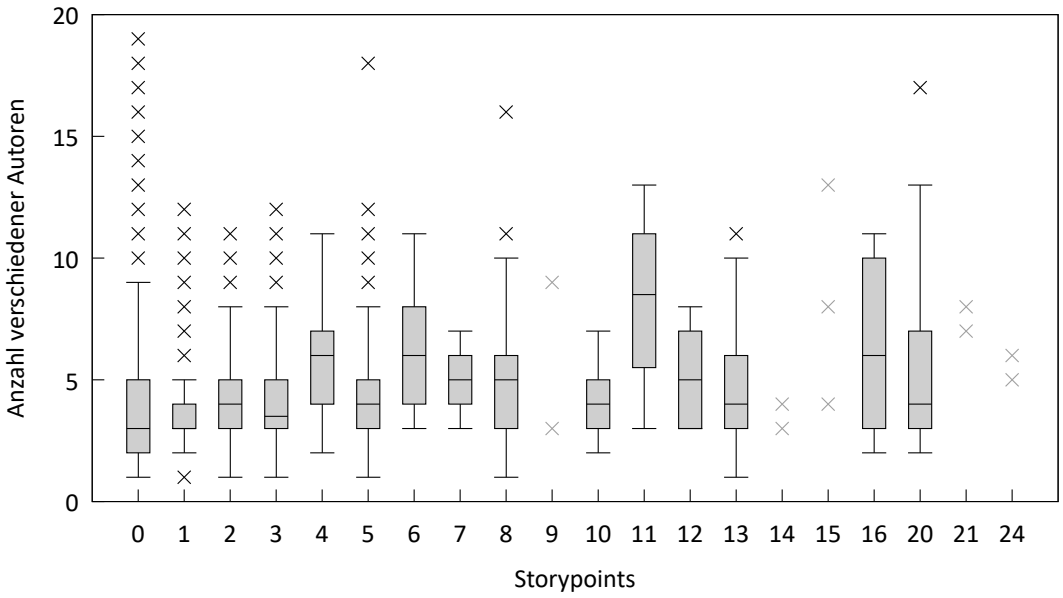

Abbildung 6.3: Boxplots der Anzahl der unterschiedlichen Autoren von Stories abhängig von den Storypoints.

Da die Anzahl der Storypoints nur schwach mit der Lead Time zusammenhängt, wurde als Nächstes betrachtet, ob bei Stories mit mehr Punkten mehr Personen mitarbeiten. Dazu wurden der Ersteller und die Autoren von Änderungen gezählt. Personen, die mehrfach vorkamen, wurden nur einmal gezählt. Die Stories, denen keine Punkte zugewiesen wurden, wurden wieder ignoriert.

Tabelle 6.3: Ergebnisse der Korrelationsanalyse für die Anzahl der unterschiedlichen Autoren von Stories abhängig von den Storypoints, $n = 2316$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	0,1932	<0,0001
Rangkorrelationskoeffizient	0,1933	<0,0001

In in Abbildung 6.3 sind die Anzahl der verschiedenen Autoren einer Story abhängig von den Storypoints zu sehen. Tabelle 6.3 zeigt das Ergebnis der Korrelationsanalyse. Man sieht, dass es auch hier nur eine sehr schwache, aber statistisch signifikante, Korrelation zwischen den beiden Größen gibt. Die Regressionsgerade ist durch $\#Autoren = 0{,}1009 \cdot StoryPoints + 3{,}7781$ gegeben. Es kann also auch nicht davon

ausgegangen werden, dass an komplexeren Stories mehr Personen mitarbeiten und der Mehraufwand dadurch kompensiert wird.

6.2 Bearbeiter und Autoren

Da die Storypoints nur einen geringen Zusammenhang mit der Lead Time haben, wurden als Nächstes die Personen betrachtet, die an den Stories mitarbeiten. Dabei wurde zwischen den Bearbeitern, die den aktuellen Prozessschritt umsetzen, und Autoren, die Änderungen an der Story in Jira vornehmen, unterschieden. Bearbeiter und Autoren können auch identisch sein.

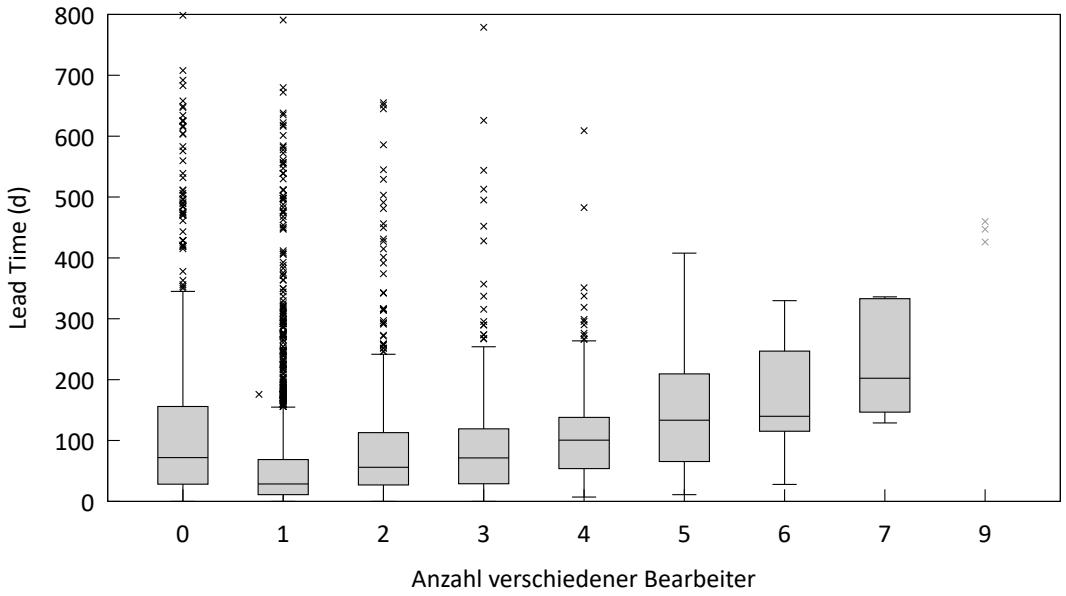

Abbildung 6.4: Boxplots der Lead Time abhängig von der Anzahl unterschiedlicher Bearbeiter einer Story.

Zunächst wurde untersucht, wie die Anzahl unterschiedlicher Bearbeiter mit der Lead Time zusammenhängt. Dabei wurden aus dem Changelog, wenn das Assignee-Feld geändert wurde, und dem Assignee-Feld der Key des Bearbeiters extrahiert. Doppelte Keys wurden nur einmal gezählt. Für die Analyse wurden, wie bei den Storypoints, die Stories ignoriert, denen nie ein Bearbeiter zugewiesen wurde.

Die Lead Time abhängig von der Anzahl unterschiedlicher Bearbeiter einer Story ist in Abbildung 6.4 dargestellt. In Tabelle 6.4 sind die dazugehörigen Ergebnisse der Korrelationsanalyse aufgelistet. Hier gibt es eine schwache, statistisch signifikante Korrelation zwischen den Bearbeitern und der Lead Time. Die lineare Regression ergibt die Gerade

Tabelle 6.4: Ergebnisse der Korrelationsanalyse für die Lead Time abhängig von der Anzahl unterschiedlicher Bearbeiter einer Story, $n = 3450$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	0,2197	<0,0001
Rangkorrelationskoeffizient	0,3036	<0,0001

$LeadTime = 21{,}6519\,\text{d} \cdot \#Bearbeiter + 40{,}0568\,\text{d}$. Mögliche Schlüsse daraus sind, dass durch mehr Bearbeiter die benötigte Zeit steigt, andererseits kann es auch sein, dass Stories, die längere Zeit nicht bearbeitet wurden, häufiger einen neuen Bearbeiter zugewiesen bekommen.

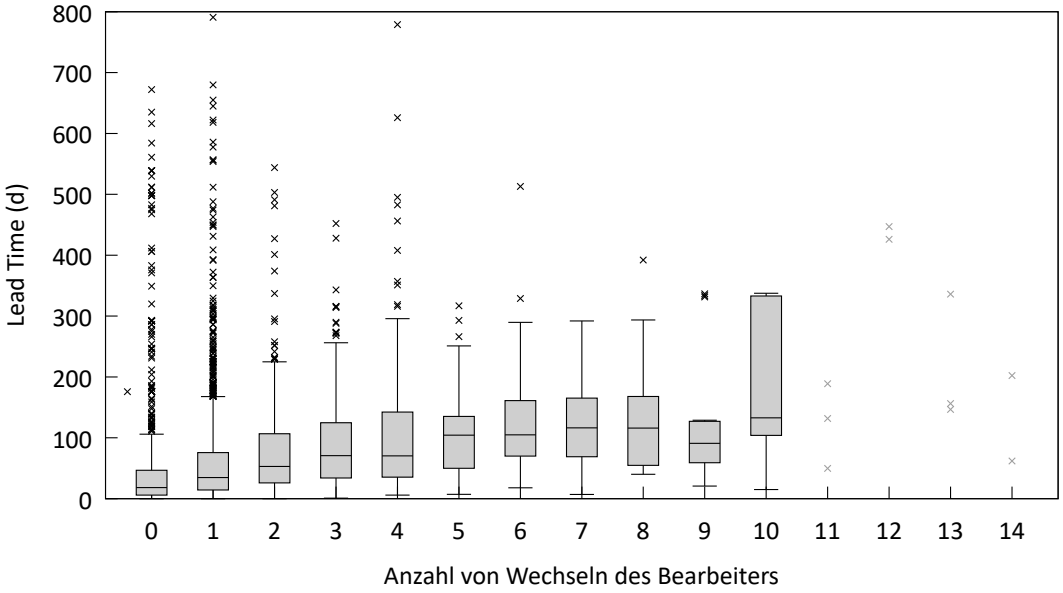

Abbildung 6.5: Boxplots der Lead Time abhängig von davon, wie oft der Bearbeiter einer Story wechselt.

Um das näher zu untersuchen, wurde betrachtet, wie oft der Bearbeiter der Story geändert wurde. Dafür wurde die Anzahl der Änderungen des Felds Assignee im Changelog gezählt. Dabei wurden die Stories, denen nie ein Bearbeiter zugewiesen wurde, herausgefiltert.

Abbildung 6.5 zeigt die Lead Time abhängig von der Anzahl der Wechsel des Bearbeiters einer Story. Tabelle 6.5 listet die Ergebnisse der Korrelationsanalyse auf. Es gibt eine schwache, statistisch signifikante Korrelation zwischen der Anzahl

Tabelle 6.5: Ergebnisse der Korrelationsanalyse für die Lead Time abhängig davon wie oft der Bearbeiter einer Story geändert wird, $n = 3288$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	0,225	<0,0001
Rangkorrelationskoeffizient	0,3551	<0,0001

von Änderungen und der Lead Time. Die Korrelation ist etwas stärker als die der Anzahl der verschiedenen Bearbeiter. Die Regressionsgerade ist gegeben durch $LeadTime = 12{,}1546\,\mathrm{d} \cdot \#Änderungen_{\mathrm{Bearbeiter}} + 52{,}0916\,\mathrm{d}$. Es kann sein, dass der Wechsel von Bearbeitern zu längeren Lead Times führt, möglicherweise weil sich der neue Bearbeiter in die Story einarbeitern muss. Es kann aber auch sein, dass bei Stories, die länger inaktiv waren, häufiger der Bearbeiter wechselt.

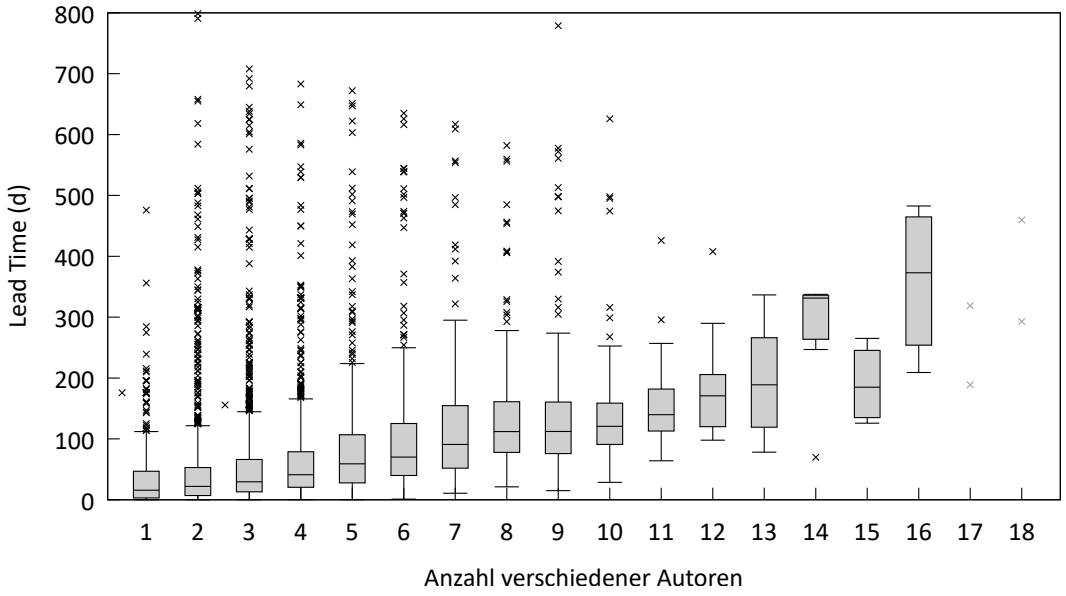

Abbildung 6.6: Boxplots der Lead Time abhängig von der Anzahl unterschiedlicher Autoren einer Story.

Zusätzlich zu den Bearbeitern sind die Personen interessant, die mit der Story interagieren. Diese Autoren beinhalten alle Personen, die etwas an der Story in Jira ändern und den Ersteller der Story. In der Regel sind die Bearbeiter auch Autoren. Für diese Metrik wurden der Ersteller der Story und die Ersteller von Einträgen des Changelogs gezählt, doppelte Personen wurden nur einmal gezählt.

Tabelle 6.6: Ergebnisse der Korrelationsanalyse für die Lead Time abhängig von der Anzahl unterschiedlicher Autoren einer Story, $n = 4106$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	0,3134	<0,0001
Rangkorrelationskoeffizient	0,4567	<0,0001

In Abbildung 6.6 ist die Lead Time abhängig von der Anzahl unterschiedlicher Autoren dargestellt. In Tabelle 6.6 ist das Ergebnis der zugehörigen Regressionsanalyse zu sehen. Der Korrelationskoeffizient zeigt einen schwachen Zusammenhang, der Rangkorrelationskoeffizient einen mittleren, beide sind statistisch signifikant. Die lineare Regression für die Daten ergibt $\mathit{LeadTime} = 12{,}495\,\mathrm{d} \cdot \#\mathit{Autoren} + 26{,}4122\,\mathrm{d}$. Mögliche Gründe für die Korrelation sind, dass bei aufwendigeren Stories mehr Personen involviert sind. Es kann aber auch sein, dass Stories, nachdem sie länger nicht bearbeitet wurden, von anderen Entwicklern weitergeführt werden. Der Rangkorrelationskoeffizient ist deutlich höher als bei den bisher betrachteten Metriken, weshalb der Wechsel von Autoren weiter untersucht wurde.

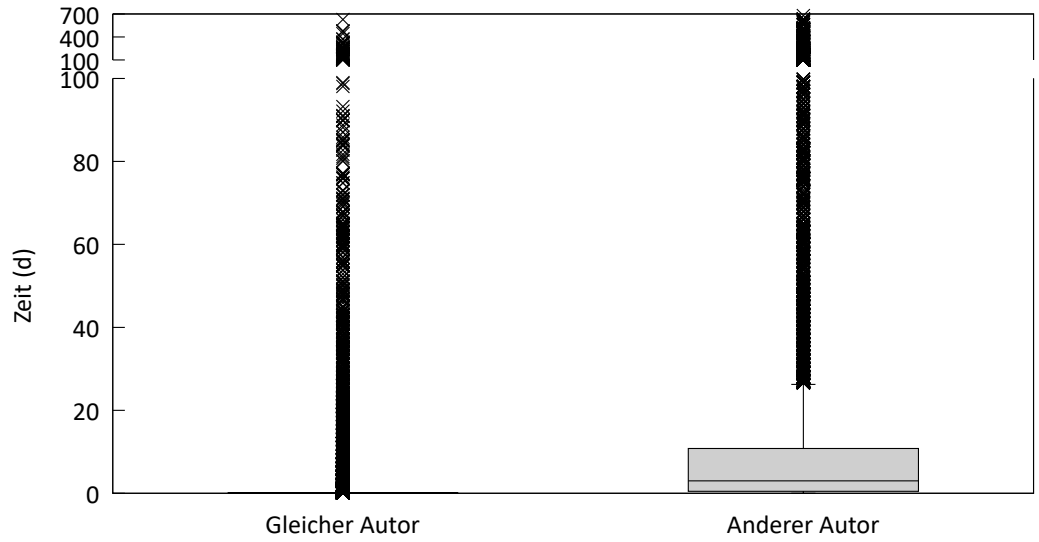

Abbildung 6.7: Boxplots der Zeit zwischen Änderungen an einer Story vom gleichen Autor wie dem der vorherigen Änderung oder von einem anderen Autor.

Als Nächstes wurde die Zeit zwischen Änderungen betrachtet. Dabei wurde dazwischen unterschieden, ob die Änderung vom gleichen Autor wie die Vorherigen oder von einem anderen Autor durchgeführt wurde. Für diese Metrik wurden die Zeitstempel der Einträge des Changelogs verwendet und der Autor der aktuellen Änderung mit dem der letzten verglichen.

Tabelle 6.7: Beschreibende Statistik für die Zeit zwischen Änderungen an einer Story vom gleichen Autor wie dem der vorherigen Änderung oder von einem anderen Autor.

Lösung	n	Mittelwert d	Standardabweichung d	Median d
Gleicher Autor	72 070	1,8438	10,9013	0,0009
Anderer Autor	25 483	14,2473	44,1702	3,0057

In Abbildung 6.7 ist die Zeit zwischen Änderungen für die beiden Kategorien dargestellt. Tabelle 6.7 zeigt die dazugehörigen Ergebnisse der beschreibenden Statistik. Aus den Unterschieden zwischen Mittelwert und Median wird erwartet, dass die Verteilungen nicht der Normalverteilung folgen, was durch den Shapiro-Wilk-Test für beide Kategorien mit $p < 0{,}0001$ statistisch signifikant bestätigt wird. Der nachfolgend durchgeführte Mann-Whitney-Test lehnt die Nullhypothese, dass beide den gleichen Erwartungswert haben, mit $p < 0{,}0001$ ab. Man sieht also einen deutlichen Unterschied in der Zeit zwischen Änderungen vom gleichen bzw. von verschiedenen Autoren. Das unterstützt die auch vorherigen Ergebnisse, dass Stories mit mehr Autoren eine längere Lead Time haben.

Um den Einfluss von unterschiedlichen Autoren weiter zu untersuchen, wurden Stories betrachtet, die mehrmals vom gleichen Autor und dazwischen von einem anderen Autor bearbeitet wurden. Dazu wurden die Ersteller von Änderungen im Changelog mit vorherigen Autoren verglichen. Es wurden die Fälle gezählt, in denen der Ersteller ein vorheriger Autor, aber nicht der Autor der letzten Änderung ist. Für diese Metrik wurden nur Stories betrachtet, die mindestens zwei verschiedene Autoren haben.

Tabelle 6.8: Ergebnisse der Korrelationsanalyse für die Lead Time abhängig von der Anzahl der Fälle, in denen ein vorheriger Autor die Story erneut bearbeitet, nachdem ein anderer sie bearbeitet hat, $n = 3786$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	0,2814	<0,0001
Rangkorrelationskoeffizient	0,4766	<0,0001

In Abbildung 6.8 ist die Lead Time abhängig von wiederholten Autoren mit Unterbrechung durch einen andern Autor dargestellt. Tabelle 6.8 zeigt die

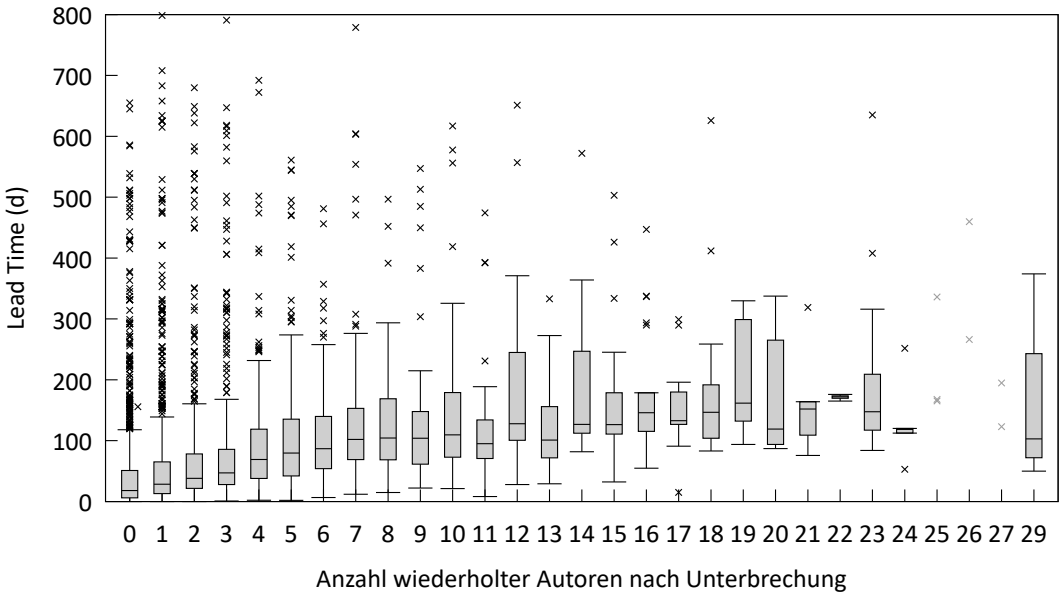

Abbildung 6.8: Boxplots der Lead Time abhängig von der Anzahl von Fällen, in denen ein Autor, der schon an der Story gearbeitet hat, die Story wieder bearbeite, nachdem sie von einem anderen Autoren bearbeitet wurde.

dazugehörigen Ergebnisse der Korrelationsanalyse. Der lineare Korrelationskoeffizient zeigt einen schwachen Zusammenhang, der Rangkorrelationskoeffizient einen mittleren, beide sind statistisch signifikant. Die Regressionsgerade ist gegeben durch $\mathit{LeadTime} = 7{,}0107\,\mathrm{d} \cdot \#\mathit{Änderungen}_{\text{nach Unterbrechung}} + 62{,}5384\,\mathrm{d}$. Dieser Zusammenhang kann möglicherweise dadurch erklärt werden, dass der erste Autor auf eine Antwort auf einen Kommentar wartet, bevor er weiterarbeiten kann.

Um den Einfluss der Anzahl von Autoren von dem wiederholter Autoren nach Unterbrechungen unterscheiden zu können, wurde die letzte Metrik für verschiedene Anzahlen von Autoren der Story betrachtet. Dabei wurden Stories mit zwei bis drei, vier bis fünf und mindestens sechs Autoren untersucht. In Tabelle 6.9 sind die Ergebnisse der Korrelationsanalyse aufgelistet. Alle drei Kategorien zeigen ähnlich große leichte Korrelationen. Die lineare Regression ergibt $\mathit{LeadTime} = 11{,}1888\,\mathrm{d} \cdot \#\mathit{Änderungen}_{\text{nach Unterbrechung}} + 49{,}9149\,\mathrm{d}$ für zwei oder drei, $\mathit{LeadTime} = 6{,}8534\,\mathrm{d} \cdot \#\mathit{Änderungen}_{\text{nach Unterbrechung}} + 60{,}331\,\mathrm{d}$ für vier oder fünf und $\mathit{LeadTime} = 3{,}7972\,\mathrm{d} \cdot \#\mathit{Änderungen}_{\text{nach Unterbrechung}} + 104{,}4548\,\mathrm{d}$ für sechs oder mehr Autoren. Diese Werte implizieren, dass Stories mit mehr Autoren grundsätzlich länger brauchen (oder es mehr Autoren gibt, weil sie länger inaktiv waren). Die geringere Steigung impliziert aber auch, dass bei Stories mit mehr Autoren deren Wechsel geringere Auswirkungen hat.

Tabelle 6.9: Ergebnisse der Korrelationsanalyse für die Lead Time abhängig von der Anzahl der Fälle, in denen ein vorheriger Autor die Story erneut bearbeitet, nachdem ein anderer sie bearbeitet hat, unterteilt nach Anzahl der Autoren der Story.

Anzahl Autoren	n	Statistik	Wert	p-Wert
zwei bis drei	1750	Korrelationskoeffizient	0,1578	<0,0001
		Rangkorrelationskoeffizient	0,3108	<0,0001
vier bis fünf	1071	Korrelationskoeffizient	0,1667	<0,0001
		Rangkorrelationskoeffizient	0,2903	<0,0001
sechs oder mehr	965	Korrelationskoeffizient	0,2009	<0,0001
		Rangkorrelationskoeffizient	0,353	<0,0001

Wenn die letzte Vermutung zutrifft, sollte die mittlere Zeit zwischen Änderungen größer werden, wenn die Autoren öfter wechseln. Das wurde als Nächstes betrachtet, indem statt der Lead Time die mittlere Zeit zwischen Änderungen abhängig von der Anzahl der mehrfachen Autoren nach Unterbrechung untersucht wurde. Die mittlere Zeit zwischen Änderungen wurde ermittelt, indem die Lead Time durch die Anzahl der Einträge im Changelog geteilt wurde.

Tabelle 6.10: Ergebnisse der Korrelationsanalyse für die mittlere Zeit zwischen Änderungen abhängig von der Anzahl der Fälle, in denen ein vorheriger Autor die Story erneut bearbeitet, nachdem ein anderer sie bearbeitet hat, $n = 3786$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	−0,096	<0,0001
Rangkorrelationskoeffizient	0,0697	<0,0001

Die mittlere Zeit zwischen Änderungen abhängig von der Anzahl der wiederholten Autoren nach einer Unterbrechung ist in Abbildung 6.9 zu sehen. Tabelle 6.10 zeigt die Ergebnisse der Korrelationsanalyse dazu. Man sieht, dass es praktisch keinen Zusammenhang zwischen den beiden Größen gibt. Die Zeit zwischen Änderungen kann also nicht der Grund sein, dass die Lead Time steigt, wenn Autoren die Story nach Unterbrechungen wiederholt bearbeiten. Es ist möglich, dass die längere Lead Time durch die höhere Anzahl an Änderungen zustande kommt. In Tabelle 6.11 sind die Ergebnisse der Korrelationsanalyse wie bei der letzten Metrik aufgeteilt nach Anzahl der Autoren aufgelistet. Auch hier zeigen sich nur sehr schwache Zusammenhänge, die stärksten bei mindestens sechs Autoren. Dabei ist die Korrelation allerdings negativ, was bedeutet, dass die Zeit zwischen Änderungen bei häufigerem Wechsel von Autoren kleiner wird.

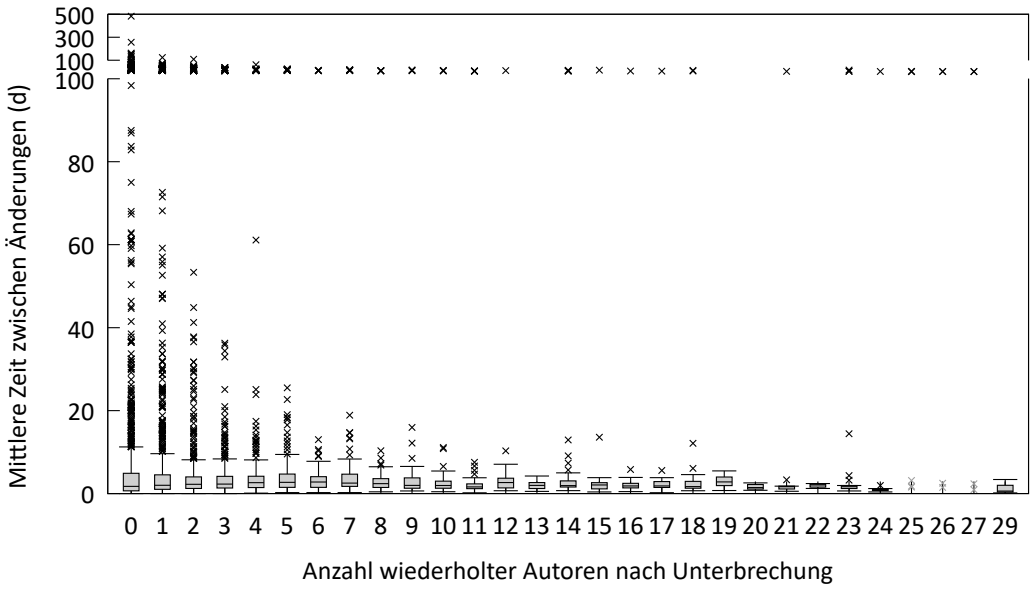

Abbildung 6.9: Boxplots der mittleren Zeit zwischen Änderungen abhängig von der Anzahl von Fällen, in denen ein Autor, der schon an der Story gearbeitet hat, die Story wieder bearbeite, nachdem sie von einem anderen Autoren bearbeitet wurde.

Wenn die mittlere Zeit zwischen Änderungen nicht mit dem Wechsel der Autoren zusammenhängt, kann sie noch mit der Anzahl der Autoren zusammenhängen. Für diese Metrik wurden die Anzahl der unterschiedlichen Autoren und die mittlere Zeit zwischen Änderungen wie oben beschrieben ermittelt.

Abbildung 6.10 zeigt die mittlere Zeit zwischen Änderungen abhängig von der Anzahl unterschiedlicher Autoren. In Tabelle 6.12 sind die dazugehörigen Ergebnisse der Korrelationsanalyse aufgelistet. Man sieht auch hier nur sehr schwache Korrelationen. Die Anzahl der Personen, die an einer Story mitarbeiten, hat anscheinend kaum Einfluss auf die mittlere Zeit zwischen Änderungen.

Tabelle 6.11: Ergebnisse der Korrelationsanalyse für die mittlere Zeit zwischen Änderungen abhängig von der Anzahl der Fälle, in denen ein vorheriger Autor die Story erneut bearbeitet, nachdem ein anderer sie bearbeitet hat, unterteilt nach Anzahl der Autoren der Story.

Anzahl Autoren	n	Statistik	Wert	p-Wert
zwei bis drei	1750	Korrelationskoeffizient	−0,0662	0,0056
		Rangkorrelationskoeffizient	0,1436	<0,0001
vier bis fünf	1071	Korrelationskoeffizient	−0,0984	0,0012
		Rangkorrelationskoeffizient	0,0618	0,0432
sechs oder mehr	965	Korrelationskoeffizient	−0,2143	<0,0001
		Rangkorrelationskoeffizient	−0,1834	<0,0001

Tabelle 6.12: Ergebnisse der Korrelationsanalyse für die mittlere Zeit zwischen Änderungen abhängig von der Anzahl an verschiedenen Autoren, $n = 4106$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	−0,1101	<0,0001
Rangkorrelationskoeffizient	0,0435	0,0054

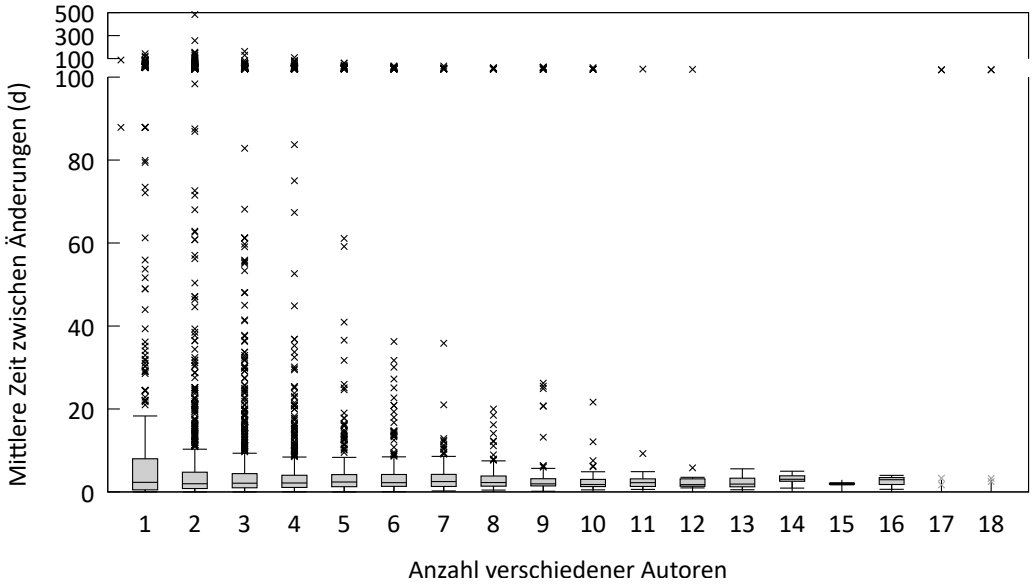

Abbildung 6.10: Boxplots der mittleren Zeit zwischen Änderungen abhängig von der Anzahl verschiedener Autoren.

6.3 Änderungen

Im letzten Abschnitt wurde der Einfluss der Anzahl und des Wechsels von Autoren betrachtet. Im Folgenden wird näher untersucht, in welchen Status es viele Änderungen gibt, was geändert wird und wo die Stories viel Zeit verbringen.

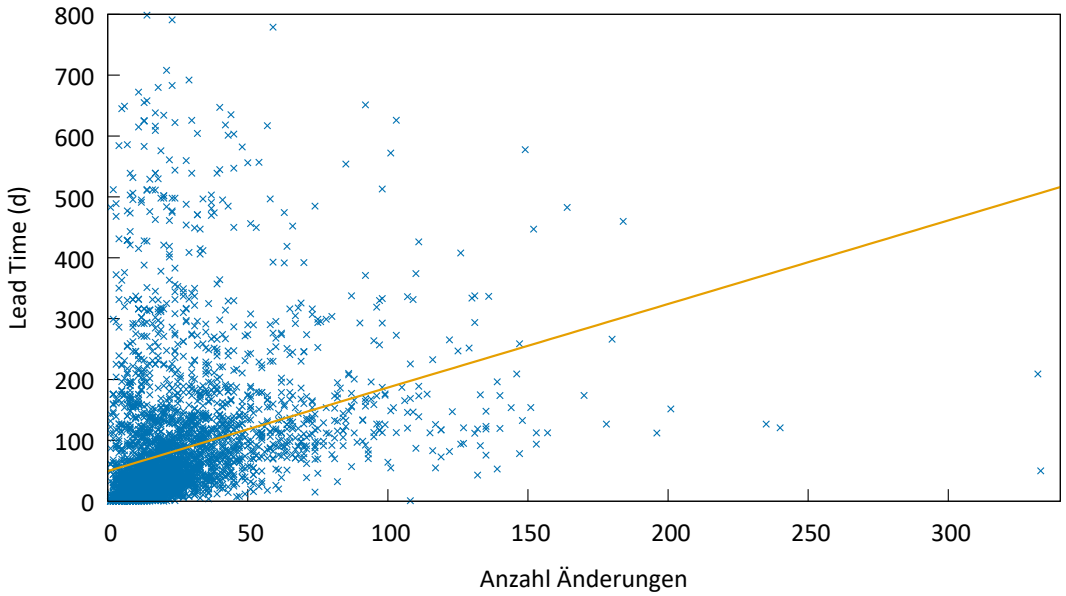

Abbildung 6.11: Scatterplot der Lead Time abhängig von der Anzahl von Änderungen in einer Story. Zusätzlich ist die Regressionsgerade dargestellt.

Um an die letzten Ergebnisse anzuknüpfen, wurde zunächst die Lead Time abhängig von der Anzahl von Änderungen in einer Story untersucht. Für die Anzahl von Änderungen wurden die Anzahl der Einträge im Changelog verwendet. Ein Eintrag kann zwar mehrere Felder ändern, aber die Änderungen werden zur gleichen Zeit durchgeführt.

Tabelle 6.13: Ergebnisse der Korrelationsanalyse für die Lead Time abhängig von der Anzahl von Änderungen in einer Story, $n = 4106$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	0,3084	<0,0001
Rangkorrelationskoeffizient	0,5142	<0,0001

In Abbildung 6.11 ist die Lead Time abhängig von der Anzahl von Änderungen dargestellt. Tabelle 6.13 zeigt die Ergebnisse der dazugehörigen Korrelationsanalyse.

Die Daten zeigen eine schwache lineare und eine mittlere Rangkorrelation, beide sind statistisch signifikant. Die Regressionsgerade ist gegeben durch $LeadTime = 1{,}3958\,\mathrm{d} \cdot \#Änderungen + 49{,}1738\,\mathrm{d}$. Wie im letzten Abschnitt vermutet, gibt es einen deutlicheren Zusammenhang zwischen der Anzahl von Änderungen als der Anzahl von Autoren oder von Wechseln von Autoren.

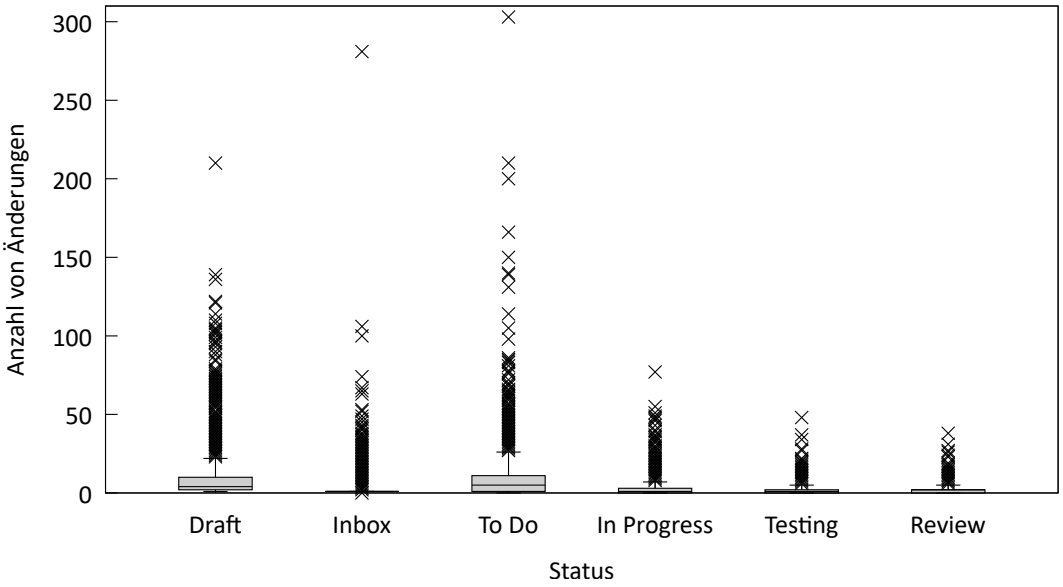

Abbildung 6.12: Boxplots der Anzahl von Änderungen abhängig vom Status, in dem die Änderungen durchgeführt wurden.

Da die Lead Time mit der Anzahl der Änderungen korreliert, ist es interessant zu wissen, in welchen Status diese Änderungen vorgenommen werden. Dazu wurden die Einträge im Changelog gezählt, die in dem Status erstellt wurden. Der Eintrag, in dem der Status geändert wird, wird zum vorherigen Status gezählt.

Abbildung 6.12 stellt die Anzahl der Änderungen abhängig vom Status, in dem sie durchgeführt wurden, dar. In Tabelle 6.14 sind die Ergebnisse der beschreibenden Statistik dazu aufgelistet. Die meisten Änderungen werden in den Status *Draft* und *To Do* vorgenommen. In *Draft* sind viele Änderungen zu erwarten, da darin die Story formuliert wird. Die Änderungen in *To Do* sind unerwartet, da laut der Prozessbeschreibung dieser Status das Backlog darstellt und die Stories hier fertig formuliert sein sollen.

Zusätzlich ist es interessant, ob die Anzahl der Änderungen in den Status mit der Verweildauer der Stories darin übereinstimmen. Dazu wurde aus dem Changelog die Zeit zwischen dem Setzen der Story auf einen Status und dem Verlassen des Status berechnet.

Tabelle 6.14: Beschreibende Statistik für die Anzahl von Änderungen abhängig vom Status, in dem diese Änderungen durchgeführt wurden, $n = 4118$.

Status	Mittelwert	Standardabweichung	Median
Draft	8,8296	13,7755	4
Inbox	2,3221	7,0096	1
To Do	8,9068	13,646	5
In Progress	2,8048	4,7798	1
Testing	1,3162	2,5234	1
Review	1,6039	2,3755	2

Tabelle 6.15: Beschreibende Statistik für die Verweildauer im Status, $n = 4118$.

Status	Mittelwert d	Standardabweichung d	Median d
Draft	15,4193	57,5478	0,0026
Inbox	1,8487	9,4243	≈0
To Do	21,3652	41,865	2,7323
In Progress	6,0725	15,3028	0,0001
Testing	2,3851	11,2048	≈0
Review	34,5571	86,1517	5,7538

In Abbildung 6.13 sind die Verweildauern der Stories in den verschiedenen Status dargestellt. Tabelle 6.15 zeigt die dazugehörigen Ergebnisse der beschreibenden Statistik. Man sieht, dass die Stories relativ viel Zeit in *Draft* und *To Do* verbringen, was mit der Anzahl der Änderungen übereinstimmt. Allerdings ist die Verweildauer im Status *Review* am längsten, wo nur wenige Änderungen durchgeführt werden. Das kann damit zusammenhängen, dass es feste Termine für Reviews gibt und die Stories bis zu so einem Termin in diesem Status bleiben und nicht weiter bearbeitet werden.

Neben der Anzahl der Änderungen ist es auch interessant zu wissen, was geändert wurde. Dazu wurde im Changelog für jede Änderung das Feld ausgelesen, das geändert wurde. Da es sehr viele Felder gibt, die in nur wenigen Stories geändert wurden, werden nur Felder berücksichtigt, die in mindestens 5 % der Stories geändert wurden. Dieser Wert wurde empirisch gewählt, um die Anzahl der Felder mit Mittelwert der Änderungen nahe null zu begrenzen. Dabei wurden die Änderungen zunächst für den gesamten Prozess betrachtet, anschließend nach Status getrennt.

Abbildung 6.14 zeigt Anzahl der Änderungen an Feldern über alle Status, in Abbildungen A.1 bis A.6 sind die Änderungen nach Status getrennt dargestellt. Tabelle 6.16 listet die Ergebnisse der beschreibenden Statistik über alle Status auf, Tabellen A.1 bis A.6 jeweils für

Tabelle 6.16: Beschreibende Statistik für die Anzahl von Änderungen an Feldern der Stories, $n = 4118$.

Feld	Mittelwert	Standardabweichung	Median
Rank	2,6095	3,2827	2
description	4,5508	9,7092	1
assignee	1,3273	1,7073	1
Team Assignment	0,4055	0,5791	0
resolution	1,0736	0,3889	1
status	5,5228	3,0148	6
Link	1,6039	3,5019	0
summary	0,4760	0,9639	0
labels	0,4968	0,9169	0
Epic Link	0,5597	0,6495	0
Sprint	2,0704	2,5035	1
Story Points	0,5478	0,6875	0
Attachment	3,1241	12,7425	0
Fix Version	0,6170	1,5492	0
RemoteIssueLink	0,6008	2,7231	0
DoR fulfilled	0,2865	0,4778	0
Team accepts story	0,1933	0,4070	0
Workflow	0,0872	0,3356	0
Key	0,0597	0,2401	0
priority	0,0634	0,2573	0
project	0,0597	0,2401	0
Version	0,0833	0,2764	0
ProjectImport	0,3281	0,7405	0

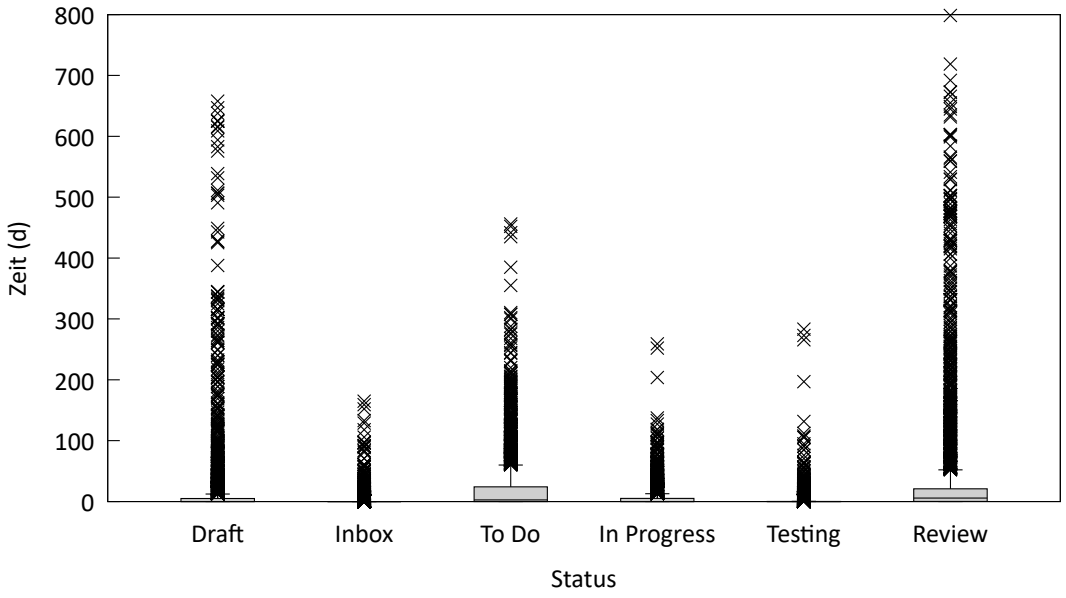

Abbildung 6.13: Boxplots der Verweildauer im Status.

einen Status. Für die Änderungen über alle Status sind die meisten in technischen Feldern. Der Status wird im Median sechs Mal geändert, was aus den sieben Prozessschritten folgt. Die Beschreibung wird im Median einmal geändert, aber aus dem Mittelwert folgt, dass sie auch oft häufiger bearbeitet wird. Die Lösung der Story (resolution) wird wie erwartet in der Regel einmal festgelegt.

Beim Betrachten der Änderungen in den einzelnen Status fällt zunächst auf, dass im Status *Draft* außer dem Status kein Feld in mehr als 5 % der Stories bearbeitet wird. Das kann damit zusammenhängen, dass die Story schon beim Erstellen ausgefüllt wird. In *Inbox* wird bei einem Teil der Stories die Beschreibung verändert, was aus der Prozessbeschreibung zu erwarten war. In *To Do* werden einige Felder bearbeitet, die etwas mit der weiteren Bearbeitung zu tun haben. Das sind z. B. Storypoints, Bearbeiter oder Sprint. Aber es werden auch Felder geändert, die in den vorherigen Status schon fertiggestellt werden sollten, wie Zusammenfassung und Beschreibung. Im Status *In Progress* werden technische Felder bearbeitet, aber teilweise auch die Beschreibung. In *Testing* wird in wenigen Fällen der Sprint oder der Bearbeiter geändert und selten auch die Lösung der Story. Im Status *Review* wird in der Regel die Lösung der Story festgelegt, selten wird dafür ein anderer Bearbeiter zugewiesen oder der Sprint gewechselt.

Eine Story kann auch entgegen des normalen Prozessverlaufs auf einen vorherigen Status zurückgesetzt werden. Um das zu untersuchen, wurden im Changelog die Fälle gezählt, in denen das Status-Feld auf einen vorhergehenden Prozessschritt gesetzt wurde.

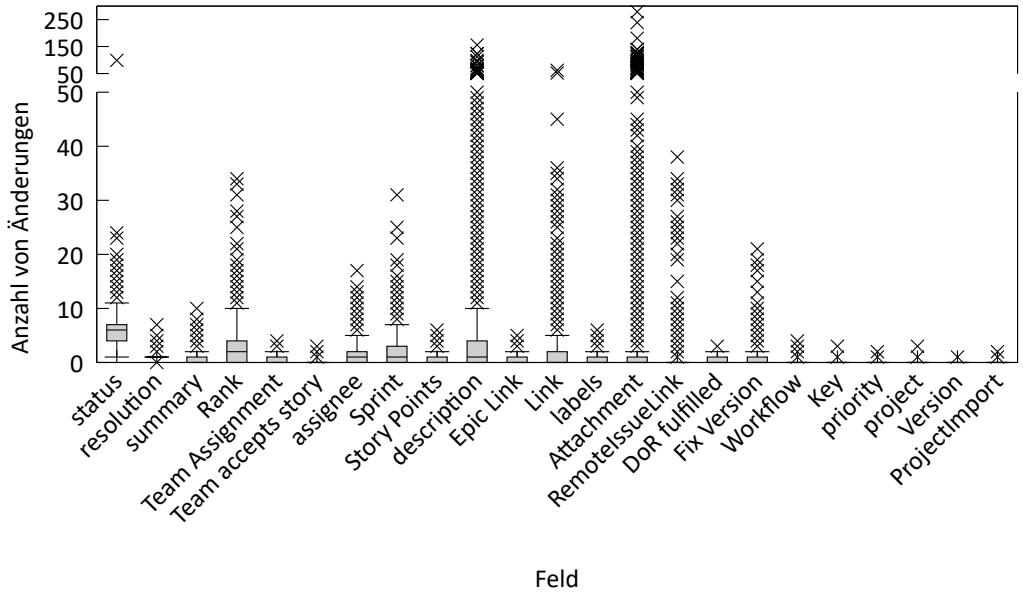

Abbildung 6.14: Boxplots der Anzahl von Änderungen an Feldern der Stories. Es werden nur Felder aufgeführt, die in mindestens 5 % der Stories bearbeitet wurden.

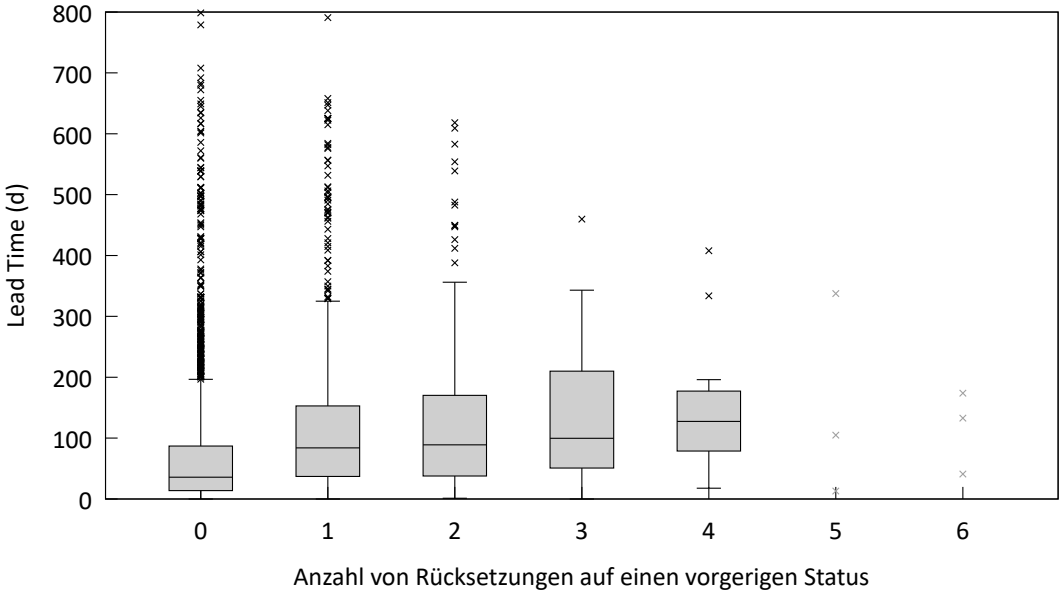

Abbildung 6.15: Boxplots der Lead Time abhängig von der Anzahl der Zurücksetzungen auf einen vorherigen Status.

Tabelle 6.17: Ergebnisse der Korrelationsanalyse für die Lead Time abhängig von der Anzahl der Zurücksetzungen auf einen vorherigen Status, $n = 4116$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	0,2002	<0,0001
Rangkorrelationskoeffizient	0,2654	<0,0001

Die Lead Time abhängig von der Anzahl der Fälle, in denen eine Story auf einen vorherigen Status zurückgesetzt wurde, ist in Abbildung 6.15 dargestellt. Tabelle 6.17 listet die dazugehörigen Ergebnisse der Korrelationsanalyse auf. Es zeigt sich ein schwacher, statistisch signifikanter Zusammenhang. Die lineare Regression ergibt $\mathit{LeadTime} = \#\mathit{Rücksetztungen} \cdot 32{,}5131\,\text{d} + 72{,}8682\,\text{d}$. In Abbildung 6.16 ist außerdem zu sehen, von welchem Status die Stories in welchen vorherigen Status zurückgesetzt werden. Von *Inbox* und *To Do* werden die Stories in der Regel auf *Draft* gesetzt, von *In Progress* auf *To Do*, von *Testing* auf *In Progress* und von *Review* auf *Testing*. Von *Done* werden Stories, die wieder geöffnet werden, in den meisten Fällen auf *To Do* gesetzt.

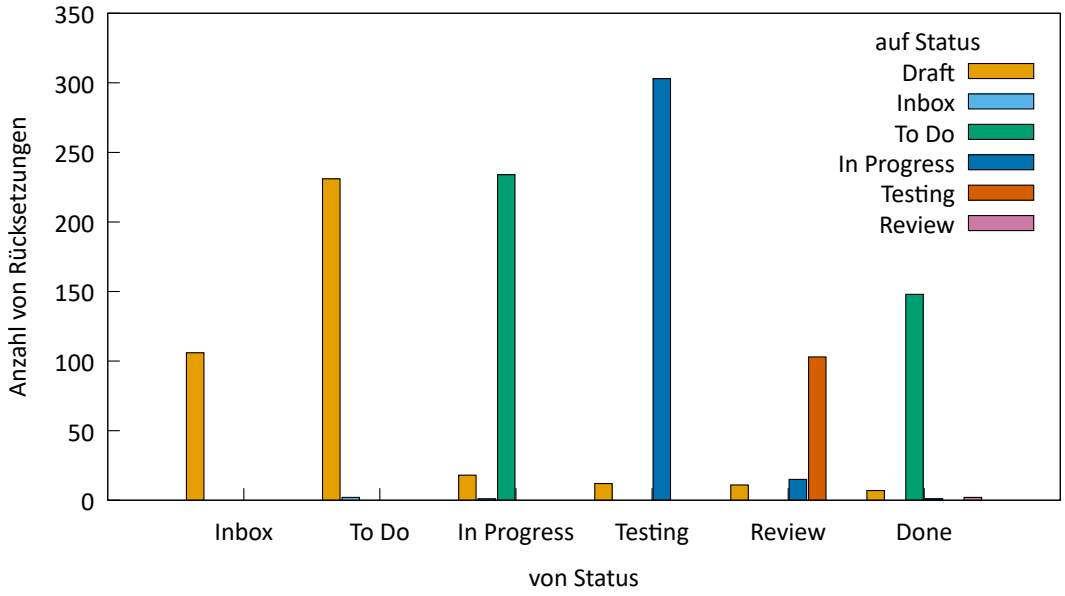

Abbildung 6.16: Anzahl von Rücksetzungen von einem Status auf eine vorherigen Status.

6.4 Weitere Metriken

Neben den bisher betrachteten Metriken wurden noch weitere Faktoren untersucht, die einen Einfluss auf die Lead Time haben könnten. Zunächst wurde die Priorität der Stories betrachtet. Dafür wurde das Priorität-Feld der Story ausgelesen.

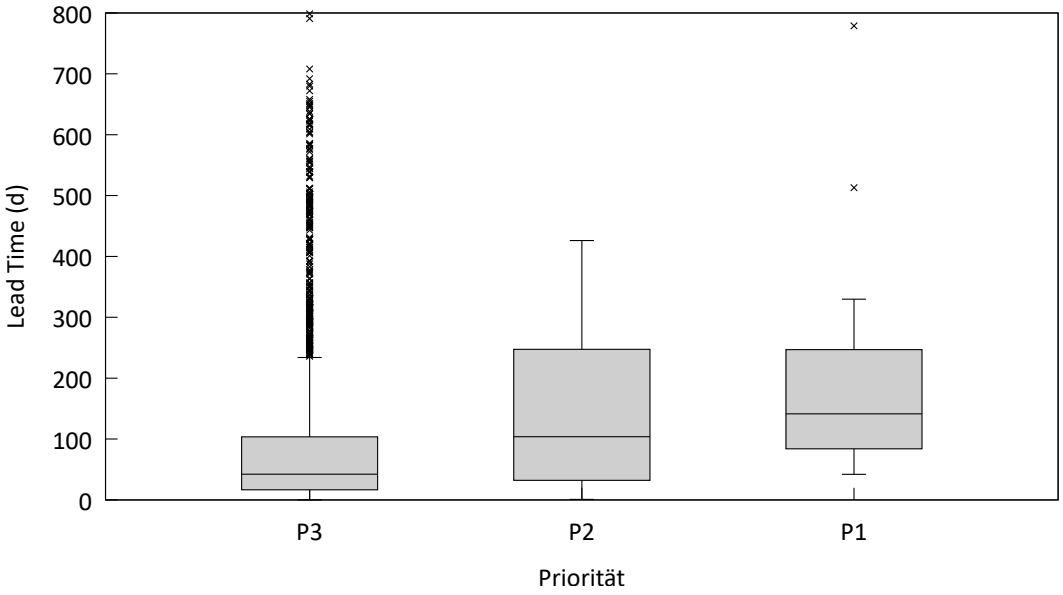

Abbildung 6.17: Boxplots der Lead Time abhängig von der Priorität.

Tabelle 6.18: Beschreibende Statistik für die Lead Time abhängig von der Priorität.

Priorität	n	Mittelwert d	Standardabweichung d	Median d
P3	4055	80,9788	107,2641	42,1794
P2	35	148,8751	128,9082	103,8759
P1	26	192,4558	162,4822	141,4475

In Abbildung 6.17 ist die Lead Time abhängig von der Priorität der Story dargestellt. Tabelle 6.18 zeigt die Ergebnisse der beschreibenden Statistik dazu. Nachdem der Shapiro-Wilk-Test gezeigt hat, dass die Kategorien keiner Normalverteilung folgen, wurden der Kruskal-Wallis-Test und anschließend der Games-Howell-Test angewendet. Der Test zeigt statistisch signifikante Unterschiede zwischen P3 und P2 ($p = 0{,}0103$) und P3 und P1 ($p = 0{,}0049$) aber nicht zwischen P2 und P1 ($p = 0{,}5014$). Daraus kann man schließen,

dass Stories mit der höchsten Priorität schneller umgesetzt werden. Die Priorität wird allerdings nur sehr selten genutzt, weshalb diese Metrik wenig aussagekräftig ist.

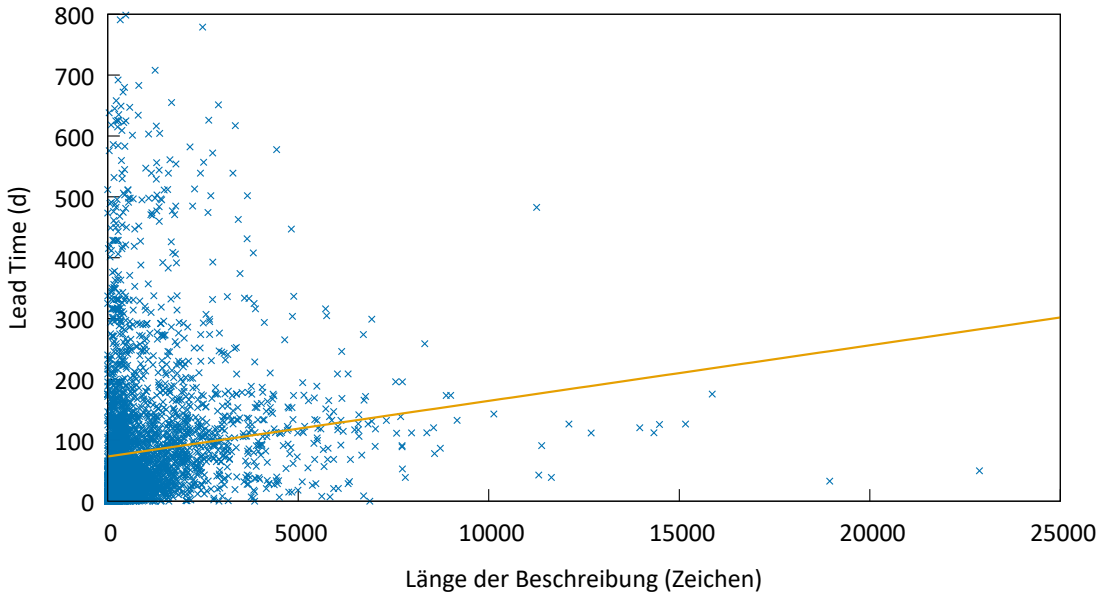

Abbildung 6.18: Scatterplot der Lead Time abhängig von der Länge der Beschreibung. Zusätzlich ist die Regressionsgerade dargestellt.

Die Länge der Beschreibung könnte einen Einfluss auf die Lead Time haben. Zum einen könnten Stories mit längerer Beschreibung aufwendiger sein und damit länger brauchen, zum anderen könnten längerer Beschreibungen detaillierter sein und die Umsetzung erleichtern. Für diese Metrik wurde die Länge des Beschreibungs-Felds ausgelesen. Das Feld kann zusätzlich HTML-Elemente enthalten, die nicht gefiltert wurden.

Tabelle 6.19: Ergebnisse der Korrelationsanalyse für die Lead Time abhängig von der Länge der Beschreibung, $n = 4116$.

Statistik	Wert	p-Wert
Korrelationskoeffizient	0,138	<0,0001
Rangkorrelationskoeffizient	0,3137	<0,0001

Abbildung 6.18 zeigt die Lead Time abhängig von der Länge der Beschreibung. Tabelle 6.19 listet die Ergebnisse der Korrelationsanalyse dazu auf. Der Korrelationskoeffizient zeigt einen sehr schwachen, der Rangkorrelationskoeffizient einen schwachen Zusammenhang, beide statistisch signifikant. Die Regressionsgerade ist gegeben durch

$LeadTime = Länge_{\text{Beschreibung}} \cdot 0{,}0091\,\text{d} + 73{,}5608\,\text{d}$. Eine längere Beschreibung scheint mit einer längeren Lead Time zusammenzuhängen, allerdings ist der Zusammenhang nicht stark genug für eine eindeutige Aussage.

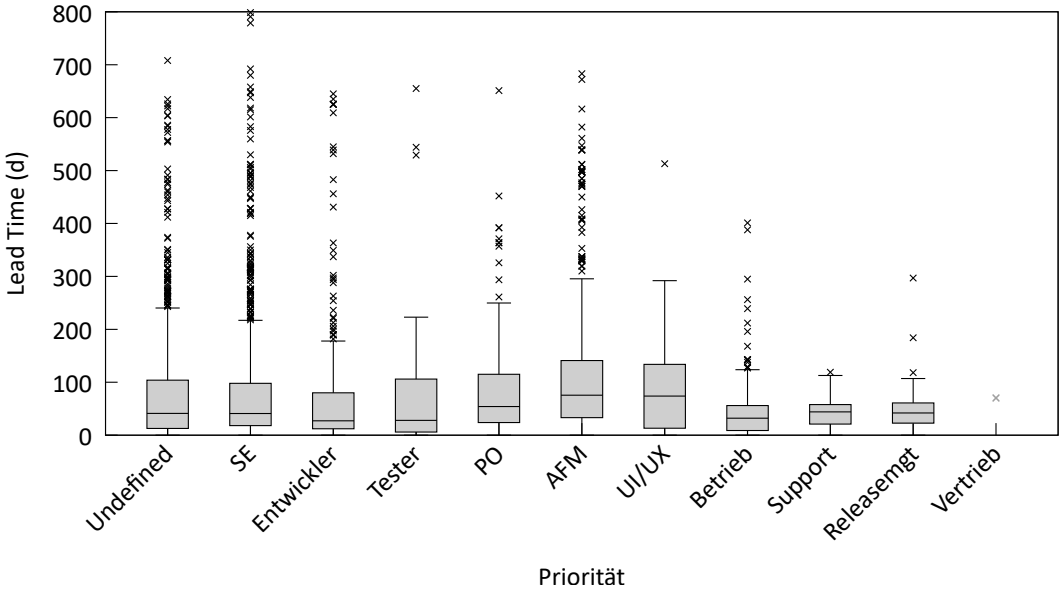

Abbildung 6.19: Boxplots der Lead Time abhängig von der Rolle des Erstellers.

Die Rolle des Erstellers kann die Lead Time dadurch beeinflussen, dass unterschiedliche Arten von Stories erstellt werden. Dazu wurden aus einer CSV-Datei die Rollen von Mitarbeitern ausgelesen und den Erstellern der Story zugeordnete. Erstellern, die nicht in der Datei aufgelistet waren, wurde die Rolle „Undefined“ zugewiesen.

Die Lead Time abhängig von der Rolle des Erstellers der Story ist in Abbildung 6.19 dargestellt. Tabelle 6.20 zeigt die dazugehörigen Ergebnisse der beschreibenden Statistik. Es wurden auch Shapiro-Wilk, Kruskal-Wallis und Games-Howell-Tests angewendet. Statistisch signifikante Unterschiede gibt es unter anderem jeweils zwischen SE (Solution Engineer), PO (Product Owner), Betrieb und AFM (externes Anforderungsmanagement), wobei die Stories vom AFM deutlich länger brauchen.

Der nächste betrachtete Faktor ist die Lösung der Story. Es ist nicht zu erwartet, dass die Lösung einen Einfluss auf die Lead Time hat, aber damit gibt es einen Hinweis darauf, wie lange sogenannter Waste im Repository bleibt. Dafür wurde das Resolution-Feld betrachtet und die die Einträge den Kategorien Abgeschlossen (Done) und Abgebrochen (Cancelled) unterteilt.

Abbildung 6.20 zeigt die Lead Time abhängig von der Lösung der Story. In Tabelle 6.21 sind die Ergebnisse der beschreibenden Statistik dazu aufgelistet. Laut Shapiro-Wilk-Test

Tabelle 6.20: Beschreibende Statistik für die Lead Time abhängig von der Rolle des Erstellers.

Status	n	Mittelwert d	Standardabweichung d	Median d
SE	1273	82,1123	113,4007	41,1009
Undefined	1200	71,9893	99,9150	34,9335
Entwickler	381	71,3224	114,0037	27,2950
Tester	26	111,9334	181,8543	43,5122
PO	331	81,7503	83,3299	54,2323
AFM	615	121,001	122,3123	90,8848
UI/UX	41	99,7412	116,3059	73,9974
Betrieb	179	47,3107	61,2503	32,1409
Support	51	44,5383	31,9809	44,1693
Releasemgt	17	66,9608	75,0982	41,9867
Vertrieb	2	70,2916	0,7183	70,2916

Tabelle 6.21: Beschreibende Statistik für die Lead Time abhängig von der Lösung der Story.

Lösung	n	Mittelwert d	Standardabweichung d	Median d
Done	3123	68,8394	87,3206	40,8116
Cancelled	993	124,4694	149,6291	65,0244

sind beide Kategorien nicht normalverteilt. Der anschließende Mann-Whitney-Test zeigt eine statistisch Signifikanten unterschied ($p < 0{,}0001$). Stories, die abgeschlossen werden, haben eine deutlich kürzere Lead Time als solche, die abgebrochen werden.

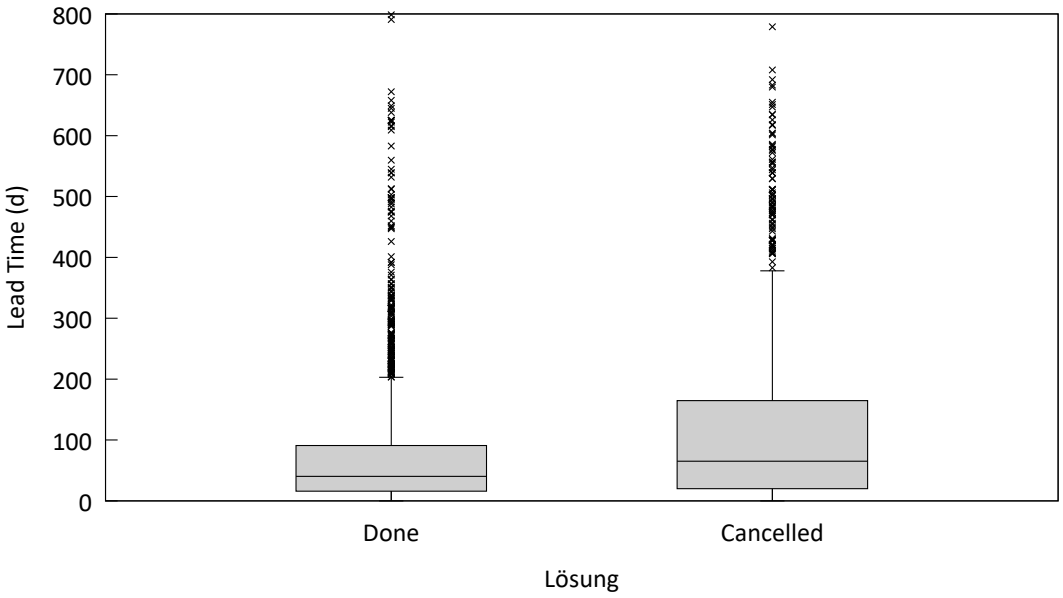

Abbildung 6.20: Boxplots der Lead Time abhängig von der Lösung der Story.

Kapitel 7

Diskussion

Im vorherigen Kapitel wurden 20 Metriken betrachtet. Davon betrafen elf direkte Abhängigkeiten der Lead Time von bestimmten Faktoren. Die restlichen neun haben diese Faktoren näher untersucht. In diesem Kapitel werden die Ergebnisse aus Kapitel 6 diskutiert. Abschließend werden noch Gefährdungen der Validität dieser Arbeit betrachtet.

7.1 Diskussion der Ergebnisse

Die meisten Faktoren, die betrachtet wurden, haben nur eine schwache Korrelation mit der Lead Time. Nur bei der Anzahl unterschiedlicher Autoren, der Anzahl der Fälle eines wiederholten Autors nach einer Unterbrechung und der Anzahl der Änderungen an einer Story zeigte der Rangkorrelationskoeffizient eine mittlere Korrelation. Zusammen mit den Ergebnissen zur Zeit zwischen aufeinanderfolgende Änderungen vom gleichen oder von einem anderen Autor kann man daraus schließen, dass Stories, an denen viele Personen mitarbeiten, länger brauchen. Allerdings sind die vorliegenden Daten nicht ausreichend, um daraus eine Kausalität zu folgern. Der Schluss, dass an Stories, die lange im System sind, mehr Personen mitarbeiten, ist genauso möglich.

Unerwartet war der geringe Einfluss der Storypoints auf die Lead Time bzw. die Zeit der Story im Status *In Progress*. Die Punkte sollen ein Maß für den Aufwand der Story sein und es wurde erwartet, dass der Aufwand mit der benötigten Umsetzungszeit korreliert. Es kann verschiedene Gründe geben, warum das nicht der Fall ist. Außer dass der Aufwand häufig falsch eingeschätzt wird, ist es auch möglich, dass jemand anfängt, die Story zu bearbeiten und die Arbeit aus irgendeinem Grund unterbricht, die Story aber in *In Progress* bleibt. Dann würde die tatsächliche Bearbeitungszeit, und damit der Aufwand, nicht mit der Zeit der Story im Status übereinstimmen.

Der Faktor, der den größten Einfluss auf die Lead Time zu haben scheint, ist die Rolle des Erstellers. Stories die von den Entwicklern selbst erstellt werden, werden auch am schnellsten fertiggestellt. Auch Stories von Betrieb und Support, die wichtig für den Einsatz der Software sind, werden relativ schnell bearbeitet. Langsamer umgesetzt werden Stories, die vom Anforderungsmanagement kommen, das weiter von der Entwicklung entfernt ist und von einem externen Dienstleister durchgeführt wird.

Bei der Untersuchung der Änderungen bzw. Verweildauer fällt auf, dass der tatsächliche Prozess vom Soll-Prozess abweicht. Die meisten Stories überspringen den Status *Inbox*, der für eine Diskussion zwischen Anforderungsmanagement und Technik vorgesehen ist. Stattdessen wird die Beschreibung häufig noch in *To Do* bearbeitet, wo sie schon fertig sein sollte. Auch *Testing* wird häufig übersprungen. Im Gespräch mit Solution Engineers wurde erwähnt, dass sie die Stories teilweise schon testen, während sie in *In Progress* sind, was den *Testing*-Status überflüssig macht.

Was bei allen betrachteten Metriken auffällt, ist die große Zahl an Ausreißern. Während in den meisten Fällen der Median bei wenigen Monaten liegt, gibt es viele Stories, die eine Lead Time von mehr als einem Jahr, bei manchen sogar mehr als zwei Jahren haben. Besonders aufgefallen ist das bei der Verweildauer der Stories in den verschiedenen Status. Der Median für Stories in *Review* ist knapp sechs Tage. Es gibt viele Ausreißer, auch extreme mit mehr als zwei Jahren, aufgrund derer der Mittelwert bei mehr als 34 Tagen liegt. Es befinden sich also im Prinzip fertige Stories für lange Zeit im System. Hier könnte das in dieser Arbeit entwickelte Tool für die Übersicht der Epics helfen, indem es auf inaktive Stories aufmerksam macht.

7.2 Gefährdungen der Validität

Die *Validität* einer Studie bezeichnet die Vertrauenswürdigkeit der Ergebnisse und in welchem Maß sie wahr sind und nicht durch die subjektive Sichtweise der Forscher verzerrt werden. Es ist wichtig, sich vor der Analyse der Daten damit zu befassen, inwieweit die Ergebnisse valide sind. Dabei sollten sie zunächst für die untersuchte Population valide sein. Zusätzlich kann es von Interesse sein, die Ergebnisse für weitere Populationen zu generalisieren [48].

Es gibt vier Aspekte der Validität, die im Folgenden betrachtet werden. Die *interne Validität* beschreibt, inwieweit ein untersuchter Faktor ohne Wissen des Forschers von einem anderen Faktor beeinflusst wird. Die *externe Validität* befasst sich damit, in welchem Ausmaß die Ergebnisse generalisiert werden können und in welchem Ausmaß sie für Personen außerhalb der Studie von Interesse sind. Die *Konstruktvalidität* reflektiert, in welchem Maß die betrachteten Metriken wirklich repräsentieren, was der Forscher untersuchen möchte. Die *Schlussfolgerungsvalidität* ist die Eignung, Schlussfolgerungen über Beziehungen basieren auf statistischen Schlussfolgerungen zu ziehen [48]. Im Folgenden werden die Gefährdungen der Validität für diese Arbeit betrachtet.

Interne Validität Die Objektivität von Messungen kann durch die Interpretation des Forschers beeinflusst werden. Um dieses Risiko zu minimieren, wurden die Ergebnisse regelmäßig mit dem Betreuer diskutiert. Eine weitere Gefahr ist die Korrektheit der Daten. Da die verwendeten Daten bei der täglichen Arbeit anfallen und zusätzlich eine komplette Änderungshistorie existiert, ist das Risiko hier gering. Allerdings gibt es

ein Risiko, dass Vorgänge, die abgeschlossen sind, nicht in den Status *Done* gesetzt werden und dadurch die Ergebnisse verzerrt werden. Dadurch, dass Daten über einen Zeitraum von mehr als zwei Jahren betrachtet werden, wird dieses Risiko als gering eingeschätzt.

Externe Validität Durch den Fokus dieser Studie auf ein Projekt in einem Unternehmen kann sie nur eingeschränkt generalisiert werden. Um einschätzen zu können, in welchem Maß die Ergebnisse auf andere Projekte angewendet werden können, wurde der Kontext beschrieben. In diesem Fall können die Ergebnisse für Projekte gelten, in denen mehrere agile Teams parallel an einer komplexen Software arbeiten.

Konstruktvalidität Die Ergebnisse der Studie können durch die Anwesenheit des Forschers beeinflusst werden. Dieser Einfluss wurde minimiert, indem nur Daten analysiert wurden, die während der üblichen Prozesse anfallen und nicht in den Ablauf eingegriffen wurde. Zusätzliche Risiken ergeben sich dadurch, dass nicht die tatsächliche Lead Time gemessen werden kann, sondern nur die Zeit, die der oben beschriebene Prozess in Jira benötigt. Die Zeit von der Äußerung der Anforderung bis zur Erstellung einer Story in Jira und die Zeit vom Abschluss der Story bis zur tatsächlichen Auslieferung können mit den vorliegenden Daten nicht gemessen werden.

Schlussfolgerungsvalidität Statistische Tests können fehlerhafte Aussagen zur Folge haben, wenn ihre Voraussetzungen verletzt werden. Um dieses Risiko zu minimieren, wurden die betrachteten Daten auf Normalität getestet und entsprechende Test ausgewählt. Das führte dazu, dass in der Regel nichtparametrische Tests verwendet wurden, die allerdings eine geringere Trennschärfe als parametrische Tests haben. Ein weiteres Risiko ergibt sich dadurch, dass die betrachteten Kategorien häufig stark unterschiedliche Populationsgrößen hatten, was besonders bei kleinen Größen die Aussagekraft einschränkt. Die Abhängigkeit von der Implementierung der statistischen Tests stellt ein zusätzliches Risiko dar, da es nur wenige Statistikbibliotheken für JavaScript gibt. Dieses Risiko wurde minimiert, indem für jedes verwendete Maß und jeden verwendeten statistischen Test Testfälle erstellt wurden, deren Sollwerte mit anderen Statistiktools erstellt wurden.

Kapitel 8

Zusammenfassung und Ausblick

In diesem Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst. Anschließend gibt es einen Ausblick auf weitere Arbeiten, die aufbauend auf den Ergebnissen durchgeführt werden können.

8.1 Zusammenfassung

In dieser Arbeit hatte das Ziel, Einflussfaktoren auf die Lead Time in der Softwareentwicklung zu finden. Dazu wurde eine Webanwendung entwickelt, mit der Jira-Daten anhand verschiedener Metriken analysiert werden können. Diese Anwendung wurde genutzt, um die Daten des Projekts ConnectFleet von VWN zu analysieren. Außerdem wurde ein Tool für die Anwendung entwickelt, mit dem der Fortschritt von Epics dargestellt werden kann. Dieses Tool soll die Entwickler dabei unterstützen, zu erkennen, ob die Epics zu Beginn der Umsetzung ausreichend Stories enthalten. Zusätzlich kann es helfen, auf inaktive Stories aufmerksam zu machen.

Die Analyse der Jira-Daten hat keine eindeutigen Einflussfaktoren auf die Lead Time gefunden. Den größten Einfluss hatte die Rolle des Erstellers, wobei die Lead Time größer war, je weiter der Ersteller von der Entwicklung entfernt war. Weitere Faktoren, die Korrelationen mit der Lead Time zeigen, sind die Anzahl der Autoren und die Anzahl der Änderungen einer Story. Es konnten aus den vorliegenden Daten allerdings keine kausalen Zusammenhänge ermittelt werden. Besonders der geringe Einfluss der Storypoints, die eine große Bedeutung in der agilen Entwicklung haben, war überraschend.

8.2 Ausblick

Ein möglicher Ansatz für weitere Forschung ist es, die Anwendung zu nutzen, um andere Projekte zu analysieren, sowohl bei VWN als auch in anderen Unternehmen, die Jira nutzen. Damit kann untersucht werden, ob die beobachteten Zusammenhänge auf dieses Projekt beschränkt sind, oder ob sich in andern Umgebungen ähnliche Muster zeigen. Dadurch könnte auch verglichen werden, ob verschiedene Prozesse besser zur Untersuchung der Lead Time geeignete Daten in Jira produzieren.

Auch eine weitere Analyse der vorliegenden Daten ist sicher auch sinnvoll. Möglicherweise liefern komplexere Datenanalysemethoden weitere Hinweise auf Einflussfaktoren, die sich aus der Kombination verschiedener Metriken ergeben. Aus eine Analyse der Ausreißer kann weitere Informationen liefern, die zu einer Erweiterung des Tools führen können, um sie zukünftig zu vermeiden.

Laut Forsgren et al. [12] sind Metriken aus verschiedenen Bereichen nötig, um die Produktivität von Entwicklern und damit die Lead Time einschätzen zu können. Daher wäre es sinnvoll, die vorhandenen Daten, die sich hauptsächlich für Aktivitätsmetriken eignen, um weitere Metriken zu erweitern. Beispielsweise kann die Kommunikation zwischen den beteiligten Personen untersucht und mit den anderen Metriken kombiniert werden.

Anhang A

Zusätzliche Abbildungen und Tabellen

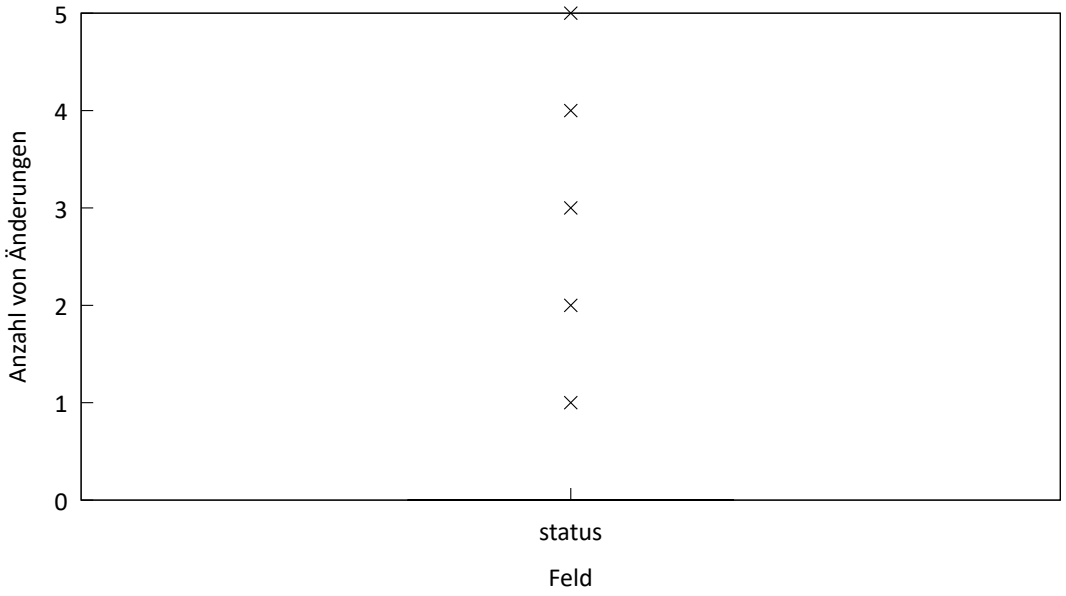

Abbildung A.1: Boxplots der Anzahl von Änderungen an Feldern der Stories im Status *Draft*. Es werden nur Felder aufgeführt, die in mindestens 5 % der Stories bearbeitet wurden.

Tabelle A.1: Beschreibende Statistik für die Anzahl von Änderungen an Feldern der Stories im Status *Draft*, $n = 4118$.

Feld	Mittelwert	Standardabweichung	Median
status	0,1141	0,4029	0

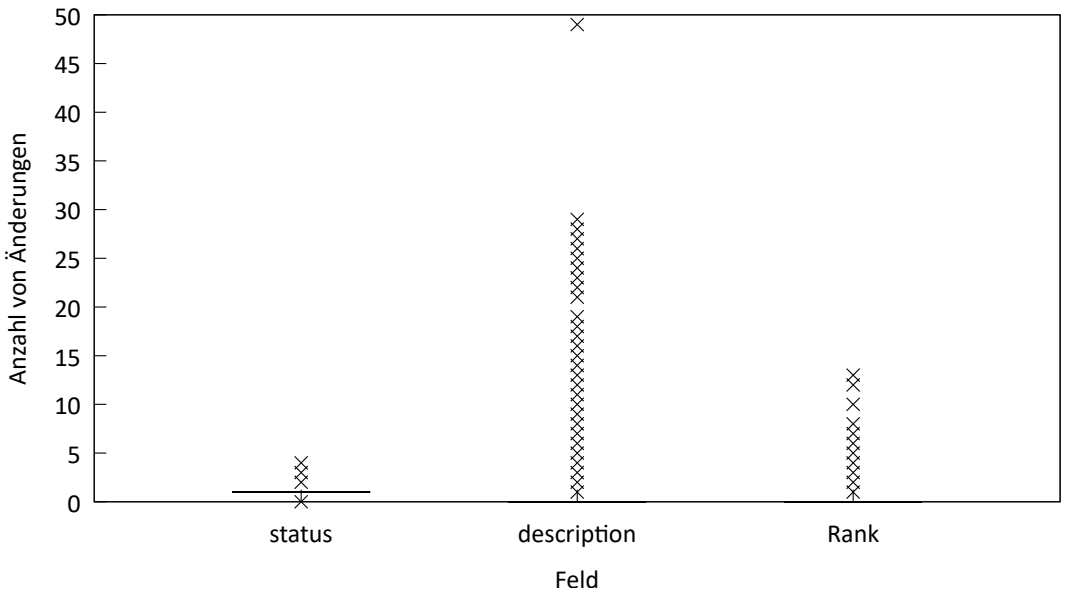

Abbildung A.2: Boxplots der Anzahl von Änderungen an Feldern der Stories im Status *Inbox*. Es werden nur Felder aufgeführt, die in mindestens 5 % der Stories bearbeitet wurden.

Tabelle A.2: Beschreibende Statistik für die Anzahl von Änderungen an Feldern der Stories im Status *Inbox*, $n = 4118$.

Feld	Mittelwert	Standardabweichung	Median
status	0,9218	0,4583	1
description	0,4490	2,2268	0
Rank	0,1382	0,6946	0

Tabelle A.3: Beschreibende Statistik für die Anzahl von Änderungen an Feldern der Stories im Status *To Do*, $n = 4118$.

Feld	Mittelwert	Standardabweichung	Median
status	1,2258	0,7922	1
resolution	0,1408	0,3500	0
assignee	0,5962	0,8043	0
Sprint	1,0937	1,7536	0
Rank	1,5617	2,4373	0
Team Assignment	0,2047	0,4469	0
labels	0,1608	0,4669	0
Epic Link	0,0935	0,3295	0
Link	0,3973	1,4063	0
summary	0,1355	0,4784	0
description	1,0056	3,0644	0
Story Points	0,3504	0,5767	0
DoR fulfilled	0,1838	0,3985	0
Team accepts story	0,1314	0,3478	0
RemoteIssueLink	0,2093	1,4133	0
Attachment	1,0425	7,5980	0
Fix Version	0,1952	1,1345	0

Tabelle A.4: Beschreibende Statistik für die Anzahl von Änderungen an Feldern der Stories im Status *In Progress*, $n = 4118$.

Feld	Mittelwert	Standardabweichung	Median
description	0,2300	1,0627	0
resolution	0,1195	0,3259	0
status	0,8987	0,7126	1
assignee	0,2079	0,6377	0
Attachment	0,2618	2,1247	0
Link	0,1515	0,7767	0
Rank	0,1863	0,7780	0
Sprint	0,3842	1,0436	0
Fix Version	0,0942	0,4622	0

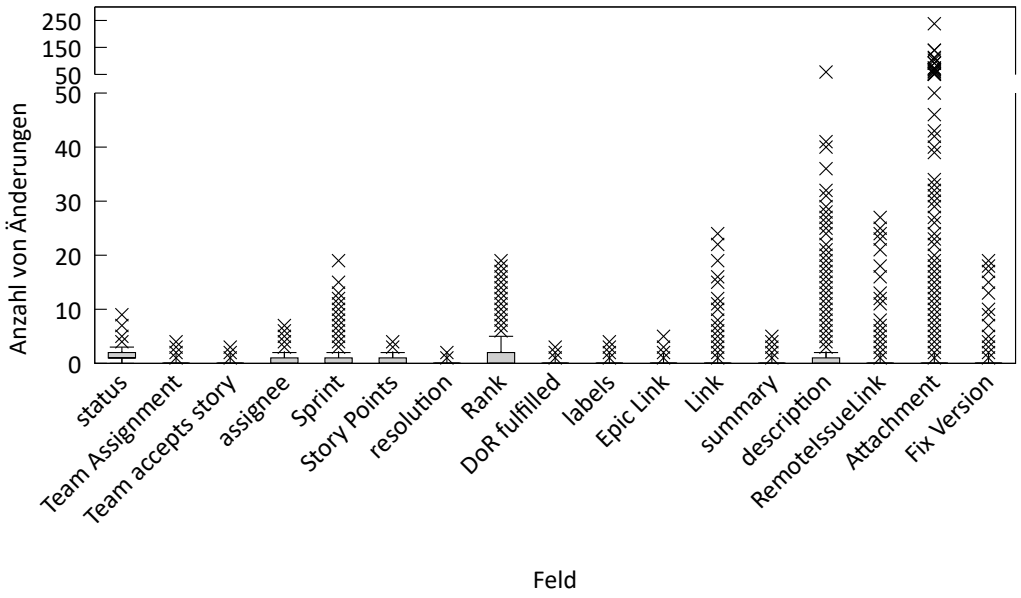

Abbildung A.3: Boxplots der Anzahl von Änderungen an Feldern der Stories im Status *To Do*. Es werden nur Felder aufgeführt, die in mindestens 5 % der Stories bearbeitet wurden.

Tabelle A.5: Beschreibende Statistik für die Anzahl von Änderungen an Feldern der Stories im Status *Testing*, $n = 4118$.

Feld	Mittelwert	Standardabweichung	Median
status	0,6833	0,7195	1
resolution	0,0996	0,3003	0
Sprint	0,1166	0,5641	0
assignee	0,1302	0,4914	0

Tabelle A.6: Beschreibende Statistik für die Anzahl von Änderungen an Feldern der Stories im Status *Review*, $n = 4118$.

Feld	Mittelwert	Standardabweichung	Median
status	0,5670	0,5800	1
resolution	0,5350	0,5309	1
Sprint	0,1229	0,5055	0
assignee	0,1127	0,4198	0

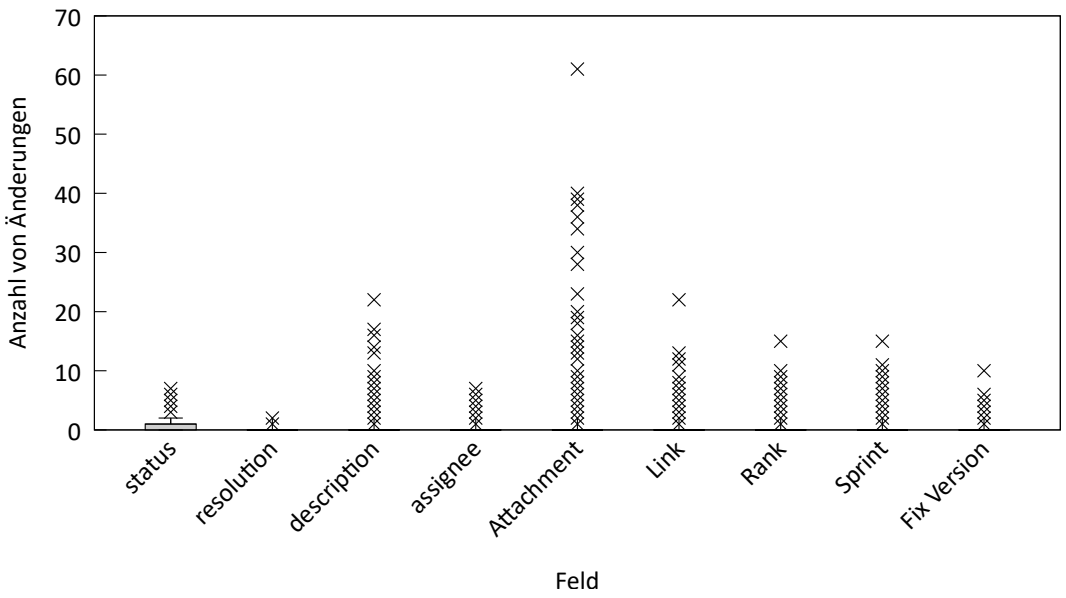

Abbildung A.4: Boxplots der Anzahl von Änderungen an Feldern der Stories im Status *In Progress*. Es werden nur Felder aufgeführt, die in mindestens 5 % der Stories bearbeitet wurden.

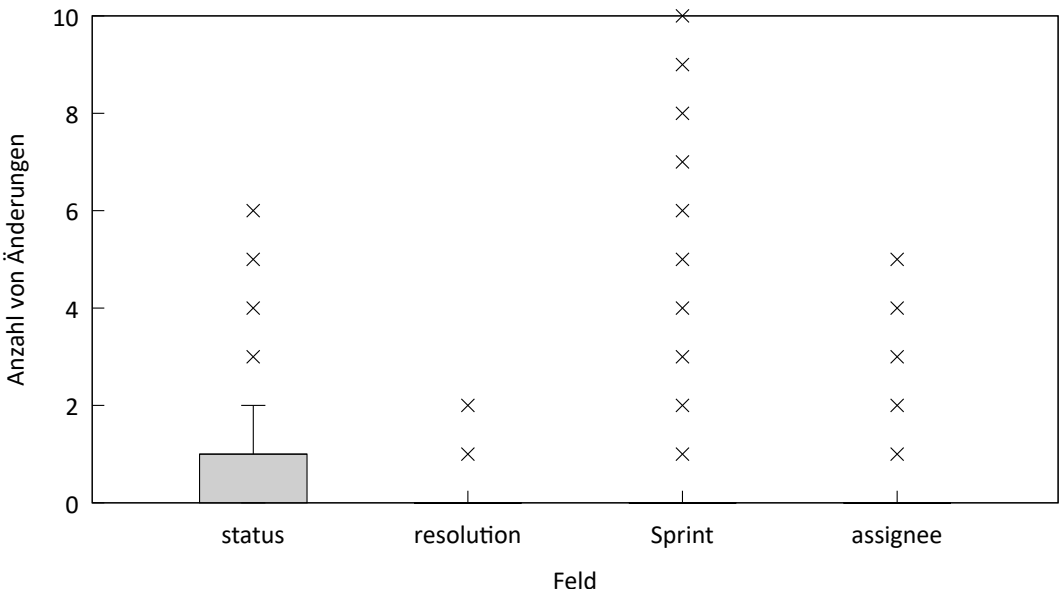

Abbildung A.5: Boxplots der Anzahl von Änderungen an Feldern der Stories im Status *Testing*. Es werden nur Felder aufgeführt, die in mindestens 5 % der Stories bearbeitet wurden.

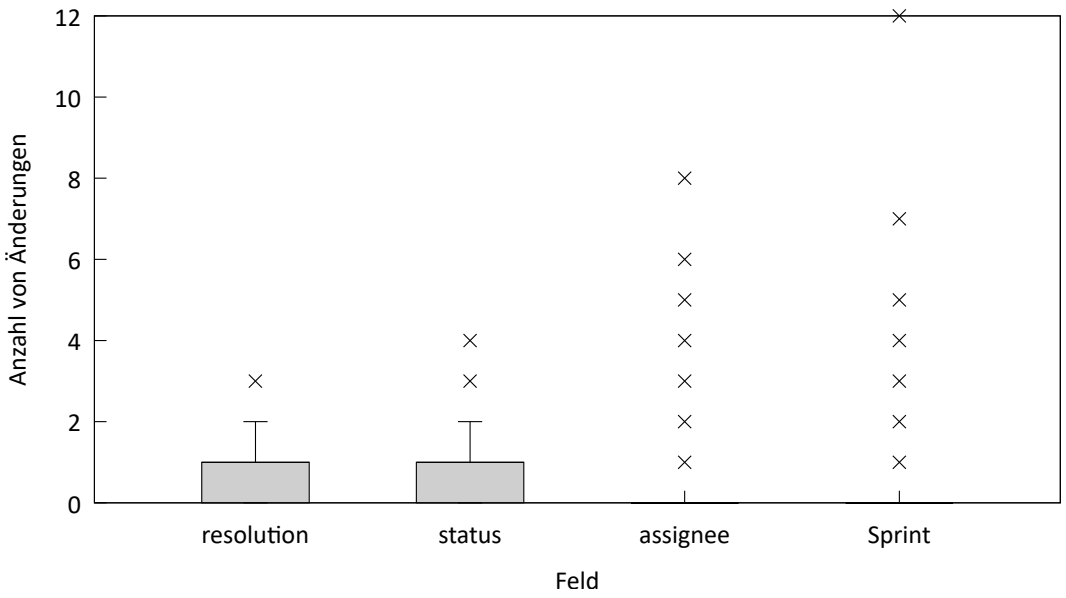

Abbildung A.6: Boxplots der Anzahl von Änderungen an Feldern der Stories im Status *Review*. Es werden nur Felder aufgeführt, die in mindestens 5 % der Stories bearbeitet wurden.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

VWN Volkswagen Nutzfahrzeuge

REST Representational State Transfer

API Application Programming Interface

HTTP Hypertext Transfer Protocol

URI Uniform Resource Identifier

JSON JavaScript Object Notation

HTML Hypertext Markup Language

JQL Jira Query Language

Literatur

[1] Kent Beck. „Extreme programming: A humanistic discipline of software development“. In: *Fundamental Approaches to Software Engineering*. Hrsg. von Egidio Astesiano. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, S. 1–6. ISBN: 978-3-540-69723-7.

[2] Kent Beck et al. *Manifesto for Agile Software Development*. 2001. URL: http://agilemanifesto.org (besucht am 16. 02. 2021).

[3] Güngör Budak. *Mann Whitney U Test (Wilcoxon Rank-Sum Test) Javascript Implementation*. URL: https://www.gungorbudak.com/blog/2016/01/16/mann-whitney-u-test-wilcoxon-rank-sum-test-javascript/ (besucht am 17. 03. 2021).

[4] *Chaos Report 2015*. The Standish Group International, Inc., 2015. URL: https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf.

[5] „Chi-square Distribution“. In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, S. 70–72. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_54.

[6] „Correlation Coefficient“. In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, S. 115–119. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_83.

[7] *CouchDB Website*. The Apache Software Foundation. URL: https://couchdb.apache.org (besucht am 01. 03. 2021).

[8] Wolfgang Dröschel und Manuela Wiemers. *Das V-Modell 97: der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz*. Walter de Gruyter GmbH & Co KG, 2015.

[9] *Express Website*. OpenJS Foundation. URL: https://expressjs.com (besucht am 24. 02. 2021).

[10] Roy T Fielding. *Architectural styles and the design of network-based software architectures*. Bd. 7. University of California, Irvine Irvine, 2000.

[11] „Fisher Distribution“. In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, S. 199–200. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_143.

[12] Nicole Forsgren et al. „The SPACE of Developer Productivity: There's More to It than You Think." In: *Queue* 19.1 (Feb. 2021), S. 20–48. ISSN: 1542-7730. DOI: 10.1145/3454122.3454124.

[13] Paul A. Games und John F. Howell. „Pairwise Multiple Comparison Procedures with Unequal N's and/or Variances: A Monte Carlo Study". In: *Journal of Educational Statistics* 1.2 (1976), S. 113–125. DOI: 10.3102/10769986001002113.

[14] Philipp Hohl et al. „Real-Life Challenges on Agile Software Product Lines in Automotive". In: *Product-Focused Software Process Improvement*. Hrsg. von Michael Felderer et al. Cham: Springer International Publishing, 2017, S. 28–36. ISBN: 978-3-319-69926-4.

[15] *Jira Produktwebsite*. Atlassian. URL: https://www.atlassian.com/de/software/jira (besucht am 16. 02. 2021).

[16] „Kruskal-Wallis Table". In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, S. 287–288. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_215.

[17] „Kruskal-Wallis Test". In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, S. 288–290. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_216.

[18] Franz Kudravzev. „Analyse des Anforderungsflusses im Automotivebereich durch Jira-Daten". Bachelorarbeit. Leibniz Universität Hannover, 2020.

[19] Robert Lagerström et al. „Identifying factors affecting software development cost and productivity". In: *Software Quality Journal* 20.2 (2012), S. 395–417.

[20] Tom MacWright. *Simple Statistics Website*. URL: https://simplestatistics.org (besucht am 25. 02. 2021).

[21] *Material Design Website*. Google. URL: https://material.io (besucht am 24. 02. 2021).

[22] *Material-UI Website*. Material-UI. URL: https://material-ui.com (besucht am 24. 02. 2021).

[23] Roy C. Milton. „An Extended Table of Critical Values for the Mann-Whitney (Wilcoxon) Two-Sample Statistic". In: *Journal of the American Statistical Association* 59.307 (1964), S. 925–934. DOI: 10.1080/01621459.1964.10480740.

[24] *Nano GitHub Repository*. The Apache Software Foundation. URL: https://github.com/apache/couchdb-nano (besucht am 01. 03. 2021).

[25] *Node.js Website*. OpenJS Foundation. URL: https://nodejs.org (besucht am 23. 02. 2021).

[26] Trevor Norris. *jStat GitHub Repository*. URL: https://github.com/jstat/jstat (besucht am 01. 03. 2021).

[27] *npm Website*. npm, Inc. URL: https://npmjs.com (besucht am 23. 02. 2021).

[28] M. Ortu et al. „Measuring and Understanding the Effectiveness of JIRA Developers Communities“. In: *2015 IEEE/ACM 6th International Workshop on Emerging Trends in Software Metrics*. 2015, S. 3–10. DOI: 10.1109/WETSoM.2015.10.

[29] Kai Petersen. „An Empirical Study of Lead-Times in Incremental and Agile Software Development“. In: *New Modeling Concepts for Today's Software Processes*. Hrsg. von Jürgen Münch, Ye Yang und Wilhelm Schäfer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 345–356. ISBN: 978-3-642-14347-2.

[30] Pieter Provoost. *Jerzy GitHub Repository*. URL: https://github.com/pieterprovoost/jerzy (besucht am 01. 03. 2021).

[31] M. Mahibbur Rahman und Z. Govindarajulu. „A modification of the test of Shapiro and Wilk for normality“. In: *Journal of Applied Statistics* 24.2 (1997), S. 219–236. DOI: 10.1080/02664769723828.

[32] *React Router Website*. React Training. URL: https://reactrouter.com (besucht am 24. 02. 2021).

[33] *React Website*. Facebook, Inc. URL: https://reactjs.org (besucht am 24. 02. 2021).

[34] *Recharts Website*. Recharts Group. URL: https://recharts.org/ (besucht am 01. 03. 2021).

[35] *Recoil Website*. Facebook, Inc. URL: https://recoiljs.org (besucht am 24. 02. 2021).

[36] Winston W Royce. „Managing the development of large software systems: concepts and techniques“. In: *Proceedings of the 9th international conference on Software Engineering*. 1987, S. 328–338.

[37] T. Savor et al. „Continuous Deployment at Facebook and OANDA“. In: *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. 2016, S. 21–30.

[38] Ken Schwaber. „SCRUM Development Process“. In: *Business Object Design and Implementation*. Hrsg. von Jeff Sutherland et al. London: Springer London, 1997, S. 117–134. ISBN: 978-1-4471-0947-1.

[39] S. S. Shapiro und M. B. Wilk. „An Analysis of Variance Test for Normality (Complete Samples)“. In: *Biometrika* 52.3/4 (1965), S. 591–611. ISSN: 00063444. URL: http://www.jstor.org/stable/2333709.

[40] *stdlib GitHub Repository*. The Stdlib Authors. URL: https://github.com/stdlib-js/stdlib (besucht am 01. 03. 2021).

[41] Michael R. Stoline. „Tables of the Studentized Augmented Range and Applications to Problems of Multiple Comparison“. In: *Journal of the American Statistical Association* 73.363 (1978), S. 656–660. DOI: 10.1080/01621459.1978.10480073.

[42] Student. „The Probable Error of a Mean“. In: *Biometrika* 6.1 (1908), S. 1–25. ISSN: 00063444. URL: http://www.jstor.org/stable/2331554.

[43] Stefan Tilkov et al. *REST und HTTP: Entwicklung und Integration nach dem Architekturstil des Web*. dpunkt. verlag, 2015.

[44] John W. Tukey. „Comparing Individual Means in the Analysis of Variance“. In: *Biometrics* 5.2 (1949), S. 99–114. ISSN: 0006341X, 15410420. URL: http://www.jstor.org/stable/3001913.

[45] A. Valdez et al. „Sentiment Analysis in Jira Software Repositories“. In: *2020 8th International Conference in Software Engineering Research and Innovation (CONISOFT)*. 2020, S. 254–259. DOI: 10.1109/CONISOFT50191.2020.00043.

[46] *visx Website*. Airbnb, Inc. URL: https://airbnb.io/visx (besucht am 01. 03. 2021).

[47] Stefan Wagner und Melanie Ruhe. *A Systematic Review of Productivity Factors in Software Development*. 2018. arXiv: 1801.06475 [cs.SE].

[48] Claes Wohlin et al. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[49] Matt Zabriskie. *Axios GitHub Repository*. URL: https://github.com/axios/axios (besucht am 24. 02. 2021).