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

Unterstützung des User Story Managements mit einer mobilen Applikation

Supporting User Story Management with a mobile Application

Bachelorarbeit

im Studiengang Informatik

von

Oliver Karras

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Prof. Dr. Michael Rohs
Betreuer: Dipl.-Math. Olga Liskin

Hannover, 20. März 2013

Zusammenfassung

Eine verbreitete Technik in der agilen Softwareentwicklung ist die Verwendung von User Stories. In der Praxis werden bei dieser Methode die Funktionalitäten einer Software als kurze Geschichten formuliert und schriftlich auf Karteikarten fixiert. Mit Hilfe dieser Karteikarten, den sogenannten Story Cards, implementieren die Entwickler die Software in iterativen Zyklen.

Ein wesentliches Problem bei dieser Herangehensweise ist das Management der erstellten User Stories. Über die gesamte Dauer des Projektes sammelt sich eine Vielzahl von Karteikarten an, die nur schwer zu überblicken sind. Für eine bessere Übersicht werden die Karteikarten meist an einer Pinnwand strukturiert. Diese Lösung ist aber nur möglich, wenn sich das Entwicklerteam an einem lokalen Ort befindet. Für verteilte Teams ist eine solche Arbeitsweise nur schwer zu realisieren und permanent aufrecht zu erhalten.
Aus diesem Grund gibt es einige Softwareprodukte, die sich mit der Verwaltung von User Stories befassen. Oftmals bieten diese Applikationen nur einen geringeren Funktionsumfang durch den zwar die Verwaltung, aber nicht unbedingt die umfassende Arbeit mit User Stories bezüglich aller agilen Prozessphasen möglich ist.

Diese Arbeit befasst sich mit dieser Problematik. Mittels einer Analyse der Eigenschaften, der Vor- und Nachteile von User Stories und von den zugehörigen Karteikarten sowie der Betrachtung der einzelnen Prozesse der agilen Softwareentwicklung, sollen Anforderungen erhoben und Konzepte entwickelt werden. Durch die Realisierung dieser Konzepte in einer mobilen Applikation wird eine adäquate Überführung der relevanten Aspekte bei der Arbeit mit User Stories und Story Cards in das elektronische System erreicht.
Dabei wurde der Fokus auf den Bereich der mobilen Anwendungen gelegt, da das Format und die Art der Bedienung von Tablet-Geräten einen interessanten Ansatzpunkt für die Umsetzung der Arbeit mit User Stories bieten. Durch die direkte Steuerung des Tablets mit den Fingern ist eine natürlichere Handhabung der elektronischen Story Cards möglich. Daraus resultiert wiederrum eine adäquatere Überführung von den Aktivitäten der einzelnen agilen Prozesse in die Applikation.

Insgesamt kann damit zum einen das Management von User Stories unterstützt werden, indem die Applikation die Verwaltung der erstellten Story Cards ermöglicht. Zum anderen kann mit der entwickelten Android-Applikation ein effizientes Arbeiten bei der agilen Entwicklung eines Projektes mittels User Stories über die gesamte Projektdauer gewährleisten werden.

Abstract

A common technique of agile software development is the use of user stories. In practice of this method, desired software features can be formulated as short stories and written down on index cards. With the help of these cards, the so-called story cards, the developers implement the software in iterative cycles.

A major challenge for this approach is the management of the created user stories. Over the entire duration of the project a large number of index cards accumulates, which is difficult to overlook. To get a better overview of the cards, they are usually structured on a pin board. This solution is only possible, if the development team is co-located. For distributed teams, such a procedure is difficult to achieve and to hold up permanently.
For this reason, there are some software products that deal with the management of user stories. Often these applications only offer certain functionalities for the administrative work, but not necessarily for the extensive work with user stories regarding all agile process phases.

This work deals with this issue. To derive requirements and concepts the characteristics, advantages and disadvantages of user stories and of the related index cards are analyzed. Further, a consideration of the individual processes of agile software development is used. Through the realization of the concepts in a mobile application, the relevant aspects of working with user stories are transferred into an electronic system.
The focus was placed on mobile applications, as the size and type of operation of tablet devices offer an interesting starting point for the implementation of the work with user stories. Since the elements on a tablet can be controlled with the fingers, working with electronic story cards can feel more natural and intuitive. With this, the activities of agile processes can be supported by an application more adequately

Overall, the application supports the management of user stories by enabling the management of created story cards. It also provides efficient work in an agile project using user stories across the entire project lifecycle with the developed Android application.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Dieses Kapitel dient als Einführung in die Thematik dieser Arbeit und führt auf die eigentliche Problematik hin. Nach der Einführung werden das genaue Ziel und die Struktur der Arbeit erörtert, um ein besseres Verständnis für den weiteren Verlauf zu ermöglichen.

1.1 Motivation

Ein wesentlicher Bestandteil für die erfolgreiche Entwicklung von Software liegt in dem Erfassen und Verstehen der Anforderungen, die der Kunde an die Software stellt. Um an diese Anforderungen zu gelangen und sie zu verstehen, ist Kommunikation zwischen Kunden und Entwicklern, sowie zwischen den Entwicklern selbst, unumgänglich [1], [2].

Bei der klassischen Softwareentwicklung entsteht am Anfang eines Projektes eine sehr viele Seiten umfassende sowie zeit- und kostenintensive Anforderungsspezifikation. An dieser wird der Kunde zwar zu Beginn stark beteiligt, jedoch endet diese Zusammenarbeit bereits kurze Zeit nach der Erstellung der Spezifikation. Erst wenn das fertige Produkt für die Abnahme bereit ist, wird die Arbeit mit dem Kunden wieder intensiviert. Es verhält sich aber so, dass kein Kunde von Anfang an weiß und sagen kann, was er genau haben will, da er sich den späteren Soll-Zustand der zu entwickelnden Software nicht vorstellen kann. Der Grund dafür liegt in der Tatsache, dass es sich bei Software um ein immaterielles Produkt handelt, welches mental nur schwer zu erfassen ist. Zusätzlich sind die technischen Lösungsmöglichkeiten, welche den Entwicklern zur Verfügung stehen, für den Kunden oftmals nicht verständlich und zu abstrakt [2].
Desweiteren ist es unwahrscheinlich, dass ein Kunde über die komplette Dauer der Entwicklung seine Meinung und Vorstellung von der Software beibehält [1], [3].

Auf Basis dieses Hintergrundes muss man sich daher folgende Frage stellen:
Wenn es einem Kunden gar nicht möglich ist, am Anfang eines Projektes, konkret zu sagen, was er möchte, warum bindet man ihn dann nicht über die komplette Dauer des Projektes mit in die Entwicklung ein?

Durch die Integration des Kunden, als fester Bestandteil des Teams, ist es möglich seine Vorstellung von der Software durch iterative Schritte weiter zu konkretisieren, um dadurch exaktere Anforderungen zu erheben. In Theorie bedeutet „fester Bestandteil des Teams“, dass der Kunde oder ein Mitarbeiter des Unternehmens des Kunden, der sogenannte On-site-Customer, permanent mit dem Team zusammen arbeitet. In der Praxis ist eine solche Arbeitsweise nicht immer umsetzbar, da der Kunde, beziehungsweise dessen Mitarbeiter, seinen beruflichen Tätigkeiten nachzukommen hat. Als Lösung werden daher oft regelmäßige Treffen in kurzen Abständen abgehalten, in denen dem Kunden die bis zu diesem Zeitpunkt entwickelte Software präsentiert wird, um Feedback zu erhalten.
Bei der Realisierung der Software kann dann schrittweise genau das umgesetzt werden, was der Kunde auch wirklich haben will [1], [3], [4]. Durch dieses Vorgehen wird die Kommunikation gefördert, was zur Folge hat, dass der Kernpunkt, die Erfassung und das Verstehen der Softwareanforderungen, stark an Bedeutung gewinnt.

Dieser Gedankengang ist ein zentraler Bestandteil des Agilen Manifestes [5], bei dem es sich um ein Wertesystem handelt, das als Leitfaden bei der agilen Softwareentwicklung dient. Die Aussagen auf die sich hier bezogen wird, sind „*Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlungen*“ und „*Die effizienteste und effektivste Art, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von*

Angesicht zu Angesicht“ [2]. Diese beiden Prinzipien zeigen, dass bei Anwendung von Agilität der Fokus unter anderem auf die Intensivierung der Kommunikation, sei es zwischen Kunden und Entwicklern oder zwischen den Entwicklern selbst, gelegt wird.
Dieser Gedanke wurde sehr treffend von C. Patel und M. Ramachandran in ihrem Artikel [6] mit folgendem Satz erfasst:

„*Again agile methods are not process oriented, they are people oriented. People are in the heart of the agile process.*“ [6] (S.12)

Ein in diesem Sinne agierendes Konzept sind User Stories. Diese bestehen aus einer kurzen Beschreibung einer Funktionalität der zu entwickelnden Software, wobei angegeben wird, was der Nutzer mit dem System machen kann [1], [3].
Bei einem auf User Stories basierendem Projekt findet keine geradlinige Entwicklung, wie bei einem klassischen auf dem Wasserfallmodell basierenden Projekt statt, sondern es wird in kontinuierlichen Iterationen die Software immer weiter verfeinert und ausgebaut, um so nach und nach die zuerst ungenaue Vorstellung des Kunden zu präzisieren.
Am Ende der Entwicklung liegt ein Produkt vor, das den Wünschen und Erwartungen des Kunden exakt entspricht [1]. Bei dieser agilen Entwicklungsart bilden die User Stories den Kern für die gesamte Arbeit in jeder Iteration. Eine häufige Anwendung ist es die User Stories auf Karteikarten, den sogenannten Story Cards, niederzuschreiben und zentral für alle Teammitglieder an einer Pinnwand anzubringen. Dadurch kann sich jedes Teammitglied, zu jederzeit, einen genauen Überblick über den aktuellen Zustand des Projektes verschaffen [1], [2], [3].

Aus einer Befragung von Gayane Azizyan, Miganoush Katrin Magarian und Mira Kajko-Mattson geht sogar hervor, dass mehr als ein Viertel aller befragten Unternehmen als häufigstes verwendetes Werkzeug die physische Wand und Papier angegeben haben (siehe Abb. 1). Dies verdeutlicht, wie verbreitet diese Methodik der agilen Softwareentwicklung ist.

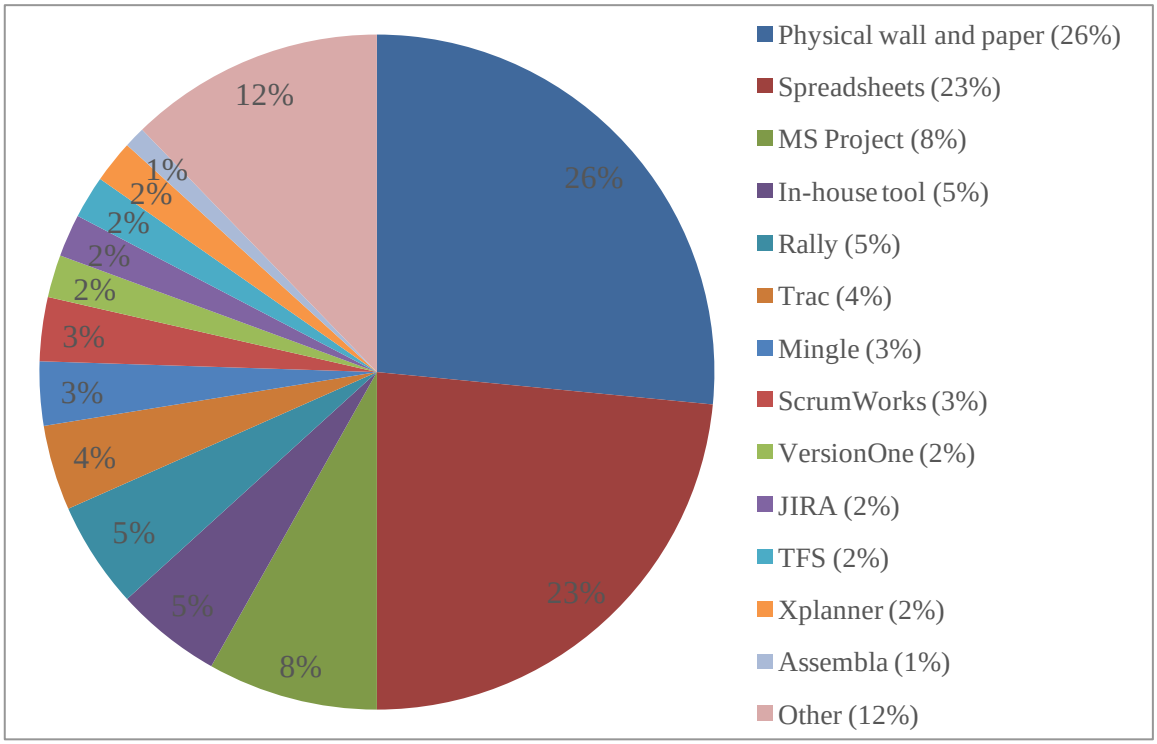


Abb. 1 - Verwendung von agilen Projekt-Management Tools [7]

Als eine Alternative zu diesem Vorgehen werden oft, wie dem obigen Diagramm zu entnehmen ist, elektronische User Story Management-Tools eingesetzt, um die Verwaltung für die Entwickler zu erleichtern.

Dabei nehmen global verteilte Teams eine besondere Stellung ein, da diese gar nicht in der Lage sind, einen zentralen, physischen Ort festzulegen, zu dem alle Teammitglieder jederzeit Zugang haben. Dadurch fällt es ihnen schwer, sich einen Überblick über das Projekt verschaffen zu können, wie es lokalen Teams möglich ist.
Dies stellt ein erhebliches Problem in der agilen Entwicklung mit verteilten Teams dar. Es ist sehr wichtig, dass alle Standorte über eine einheitliche Repräsentation der User Stories verfügen, um zu gewährleisten, dass die Teams sich auf dem gleichen, aktuellen Informationsstand bezüglich des Projektes befinden. Nur so können sie erfolgreich untereinander und mit dem Kunden kommunizieren [2], [8]. Verteilte Teams sind daher meist auf elektronische Lösungen für die Verwaltung ihrer Projekte angewiesen.

Oftmals bieten computerbasierte Tools leider nicht alle Eigenschaften an, wie man sie von der Arbeit mit echten Karteikarten gewohnt ist. So fehlen beispielsweise das Kartenformat, die Beweglichkeit der Karten oder auch die Möglichkeit wie auf einer realen Karteikarte zeichnen zu können [7]. Diese Eigenschaften können die Entwickler aber durchaus bei ihrer Arbeit unterstützen, da sie bestimmte Vorteile mit sich bringen, die im Abschnitt 2.1 genauer betrachtet werden.

Es würde sich sehr anbieten, wenn ein elektronisches Tool für das Management von User Stories gerade über diese Eigenschaften verfügen würde, da die Arbeit mit agilen Projekt-Management-Tools und die Entwicklung mit Karteikarten, sowohl bei lokalen als auch bei verteilten Teams, verbreitet ist [7].

In dem folgenden Diagramm (siehe Abb. 2) aus der gleichen Befragung wie zuvor, ist zu erkennen, dass die verteilten Teams der befragten Unternehmen verstärkt agile Projekt-Management-Tools verwenden und weniger die physische Pinnwand mit Karteikarten einsetzen. Dem gegenüber ist deutlich zu erkennen, dass die befragten lokalen Teams die beiden Techniken etwa gleich oft einsetzen.
Daraus kann man folgern, dass lokale Teams öfter physische und elektronische Werkzeuge zusammen verwenden, als es bei verteilten Teams der Fall ist. Weiterhin zeigt sich, dass die lokalen Teams verstärkt physische Tools nutzen, während globale Teams häufiger elektronische Werkzeuge einsetzen.
Gayane Azizyan, Miganoush Katrin Magarian und Mira Kajko-Mattson kommen zu der ähnlichen Schlussfolgerung, dass die allgemeinen, einfachen, physischen Werkzeuge einen hohen Wert bei lokalen Teams besitzen, während verteilte Teams eher fortgeschrittene Management-Tools vorziehen [7].

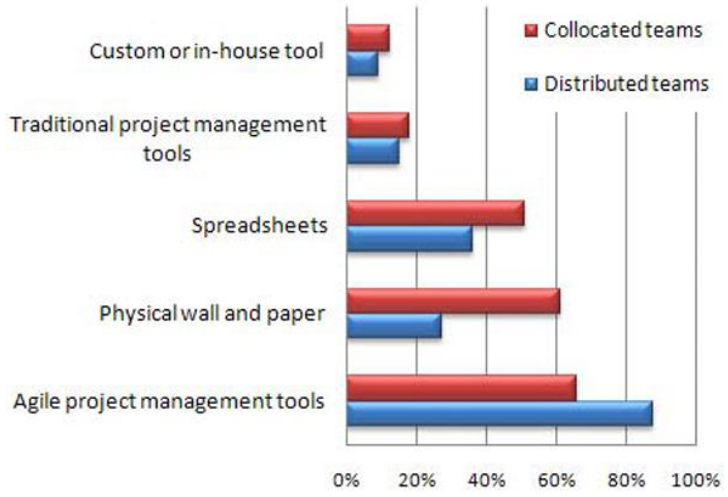


Abb. 2 - Werkzeugkategorien, die in lokalen und verteilten Teams verwendet werden [7]

Dies lässt insgesamt den Schluss zu, dass es praktisch ist, ein elektronisches Werkzeug zu entwickeln, welches die Arbeit mit echten Karteikarten und deren Eigenschaften nachahmt. Ein solches Werkzeug ermöglicht lokalen und verteilten Teams das Durchführen von bekannten Aktivitäten mit realen Karteikarten auf elektronischer Ebene. Weiterhin ist eine Entlastung der Entwickler möglich, indem gewisse Aktivitäten, wie zum Beispiel das Erstellen von Auswertungsdiagrammen, durch das Tool automatisiert werden.
Zum einen haben lokale Teams dann ein zusätzliches Werkzeug zur Hand, das sie in ihrer gewohnten Arbeitsweise direkt unterstützt, und zum anderen ist es verteilten Teams möglich, mit User Stories auf ähnliche Weise zu arbeiten, wie es Teams können, die sich lokal an einem Standort befinden.

Die heutige Technologie der mobilen Endgeräte, speziell das Tablet, bietet sich für die Unterstützung dieser Aktivitäten und Eigenschaften an. Aufgrund ihres Formates und der Interaktionsmöglichkeiten mit den Fingern stellen sie einen geeigneten Ansatzpunkt dar, um mittels einer mobilen Applikation die Arbeit mit User Stories auf eine mit der Realität vergleichbaren Art und Weise zu ermöglichen.

Dabei stellt sich die Frage, warum es sich um eine mobile Applikation handeln soll und nicht beispielsweise um eine Web-Applikation.
Unter dem Begriff der *„mobilen Applikation*“ ist, im Bezug auf diese Arbeit, zuerst einmal zu verstehen, dass es sich um eine Anwendung für mobile Endgeräte, in diesem Fall speziell für das Android Betriebssystem, handeln soll. Dabei ist anzumerken, dass die Applikation nur für Tablets und nicht für Smartphones entwickelt wird, weil sich das größere Display für den geplanten Funktionsinhalt als geeigneter erweist.
Die Integration von mobilen Eigenschaften, wie zum Beispiel GPS-Position, aktueller den Nutzer umgebender Kontext oder die Verwendung von Mikrofon und Kamera, werden in dieser Arbeit nicht mit betrachtet.
Stattdessen wird der Fokus auf die Handhabung der elektronischen Story Cards gelegt. Es soll die Art der Bedienung des Tablets ausgenutzt werden, um durch die Möglichkeit der direkten Steuerung mit den Fingern den Umgang mit den Story Cards in dem elektronischen System auf natürlichere Art und Weise zu realisieren. Zentral geht es um die Überführung der Handhabung von realen Karteikarten auf die virtuellen Story Cards in der Applikation.
Das bedeutet nicht, dass es ausgeschlossen ist, mobile Eigenschaften in der hier zu entwickelnden Applikation mit zu verwenden. Ganz im Gegenteil steckt in der Verwendung dieser Eigenschaften ein großes Potential, um die Unterstützung des User Story Management weiter zu erhöhen. Aber aufgrund der gewählten Fokussierung gilt es zuerst die Basis für die

Arbeit mit Story Cards zu schaffen, bevor man sich mit solch speziellen Features befassen kann.

In dieser Arbeit sollen Konzepte entwickelt werden, die die relevanten Aktivitäten mit Story Cards unterstützen. Mit Hilfe der entworfenen Konzepte wird dann eine solche mobile Applikation für Android-Tablets implementiert.

1.2 Ziel der Arbeit

Der zentrale Bestandteil dieser Arbeit befasst sich mit folgender Problematik:

Dadurch dass in elektronischen Systemen nicht alle Eigenschaften von realen Karteikarten mit User Stories umgesetzt sind, ist es nicht möglich jede Aktivität, wie man sie von echten Story Cards gewohnt ist, bei der Verwendung solcher Systeme durchzuführen.

Zuerst ist zu verstehen was User Stories und Story Cards überhaupt sind. Dabei ist es wichtig auch die Prozesse der agilen Entwicklung, in denen die Story Cards eingesetzt werden, mit zu betrachten. Dies ermöglicht es zu erkennen, was für Vorteile User Stories selbst und deren Verwendung in Form von Story Cards in den verschiedenen Phasen des iterativen Arbeitsablaufes mit sich bringen und welche Eigenschaften dafür gerade relevant sind. Ebenfalls sollen die Nachteile von User Stories betrachtet werden, um für diese vielleicht eine Unterstützung oder sogar möglicherweise Verbesserung durch das elektronische System zu realisieren.

Mit dem so gewonnenen Verständnis von Story Cards und der Erkenntnis der Vor- und Nachteile von User Stories und deren Anwendung gilt es, Anforderungen für ein User Story Managementsystem zu erheben, die in besonderem Maße den Fokus auf die Unterstützung der Arbeitsabläufe mit Story Cards legen.

Ziel dieser Arbeit ist es, durch die Realisierung der entwickelten Konzepte eine Applikation für Tablets zu implementieren, welche die zuvor erarbeiteten Erkenntnisse bezüglich der relevanten Eigenschaften für eine mit der Realität vergleichbaren Arbeitsweise mit Story Cards umsetzt. Weiterhin soll die Anwendung eine gute Bedienbarkeit vorweisen, indem eine einfache und direkte Bedienung, ohne große Umwege über verschachtelte Menüstrukturen, realisiert wird.
Zusätzlich zu der Applikation soll es später einen, am Fachgebiet Software Engineering, entwickelten Webservice geben, der eine webbasierte Infrastruktur für das User Story Management Tool zur Verfügung stellt.

Am Ende der Arbeit gilt es mittels einer Evaluation zu überprüfen, ob die ausgearbeiteten Konzepte und die zugehörigen Anforderungen für die Unterstützung der agilen Arbeitsweise von den Nutzern akzeptiert werden.

Insgesamt besteht die Arbeit aus zwei Teilen:

1.) Es ist ein schriftlicher Bericht über die erarbeiteten Konzepte und das erstellte Werkzeug sowie dessen Überprüfung auf erfolgreiche Umsetzung der Konzepte zu erstellen.

2.) Anhand der schriftlich festgehaltenen Konzepte gilt es eine mobile Android-Applikation für Tablets zu implementieren.

1.3 Struktur der Arbeit

Um ein besseres Verständnis für die Vorgehensweise in dieser Arbeit zu ermöglichen, ist es hilfreich, wenn man den gewählten Aufbau mit den jeweiligen Gründen für die entsprechende Strukturierung kennt. Insgesamt ist die vorliegende Arbeit in sechs Kapitel unterteilt.

Das erste Kapitel dient im Wesentlich zum Einstieg in die Thematik der agilen Softwareentwicklung mit User Stories, mit der sich diese Arbeit befasst. Dabei wird erläutert welche Problematik fokussiert wird und wie diese gelöst werden soll. Weiterhin werden das Ziel und die geplanten Ergebnisse kurz zusammengefasst.

Im zweiten Kapitel werden zuerst die beiden Begriffe *User Story* und *Story Card* definiert, sodass dann die Betrachtung der Eigenschaften, der Vor- und Nachteile erfolgen kann. Zusätzlich wird noch ein Layout für die Story Cards auf Basis der Eigenschaften von User Stories entworfen, dass in der Applikation verwendet wird. Neben den beiden Begriffen werden weiterhin noch kurz die einzelnen Prozesse der agilen Softwareentwicklung zusammengefasst, um ein grundlegendes Verständnis dieser Arbeitsweise sicherzustellen.

Mit dem Wissen aus dem zweiten Kapitel erfolgt in Kapitel 3 die Anforderungsspezifikation. Dazu wurden auf Basis der analysierten Aspekten und unter Zuhilfenahme von Fachliteratur eigenständig Anforderung erhoben und begründet. Daraufhin erfolgt noch eine Auswahl bestimmter Anforderungen, die bei der Implementierung der Applikation fokussiert werden.

Damit die gesamte Idee bezüglich der Verwendung der Android-Applikation im Zusammenspiel mit dem zugehörigen Webservice verständlicher wird, gibt es im vierten Kapitel ein ausführliches Szenario, dass die Anwendung in den einzelnen Phasen der agilen Entwicklung beschreibt. Danach erfolgt die Erläuterung der einzelnen Konzepte für die Realisierung der ausgewählten Anforderungen. Um die Besonderheiten der entwickelten Applikation aufzuzeigen, erfolgt im Abschnitt 4.3 eine Abgrenzung der realisierten Konzepte gegenüber denen andere Softwareanwendungen. Zusätzlich werden noch die technischen Aspekte und Entscheidungen aus der Entwicklungsphase dargelegt.

Ein wichtiger Aspekt dieser Arbeit ist die Evaluation der erstellten Applikation. Diese dient der Feststellung, ob die entwickelten Konzepte zu der adäquaten Überführung von Eigenschaften, Vorteilen und Aktivitäten von User Stories in das elektronische System beitragen und damit das User Story Management unterstützen. Der Aufbau, Ablauf und die Auswertung der Evaluation sind im fünften Kapitel enthalten. Auf Basis der Ergebnisse konnten weitere direkt umsetzbare Maßnahmen zur Verbesserung der aktuellen Applikationsversion durchgeführt werden, die in dem letzten Abschnitt des fünften Kapitels erfasst sind.

Zum Abschluss der Arbeit wird in dem letzten Kapitel zuerst ein Fazit bezüglich der betrachteten Aspekte und ausgearbeiteten Inhalte gezogen. Dabei wird zusammengefasst, welche Vorteile die entwickelten Konzepte in Bezug auf die Thematik der Arbeit und der fokussierten Problematik liefern. Weiterhin wird eine Einschätzung über die Unterstützung des User Story Managements und des effizienten Arbeitens auf Basis der Ergebnisse der Evaluation abgegeben.
In dem letzten Abschnitt des sechsten Kapitels wird abschließend ein Ausblick gegeben, wie die Applikation weiter optimiert werden kann. Dieser Teil greift neben Verbesserungen an der existierenden Implementierung auch noch nicht oder nur teilweise realisierte Anforderungen auf. Dabei wird kurz erläutert, wie und warum der jeweilige Aspekt eine zukünftige Optimierung der bestehenden Applikation darstellt.

2 Grundlagen

Mit diesem Kapitel werden die Voraussetzungen geschaffen, um das Verständnis der weiteren Ausarbeitung in den nachfolgenden Kapiteln zu ermöglichen. So erfolgt zuerst die Erklärung und Definition der Begriffe *User Story* und *Story Card*. Weiterhin wird sich mit dem Aufbau und dem Inhalt von Story Cards befasst, sowie den Eigenschaften, Vor- und Nachteilen von User Stories. Im letzten Abschnitt werden kurz die einzelnen Phasen der agilen Entwicklung zusammengefasst.

2.1 User Stories und Story Cards

Der erste Abschnitt des zweiten Kapitels befasst sich mit den Begriffen *User Story* und *Story Card*. Dabei werden zunächst die beiden Begriffe definiert und danach wird auf deren Eigenschaften, Vorteile und Nachteile sowie deren Aufbau und Inhalt eingegangen.

2.1.1 Definitionen

Zuerst sollen die, im Zusammenhang dieser Arbeit, verwendeten Definitionen für die Begriffe *User Story* und *Story Card* genannt werden, um eindeutig festzulegen, was darunter jeweils zu verstehen ist.
Wie schon im ersten Kapitel erwähnt wurde, handelt es sich bei einer User Story um die Beschreibung einer Funktionalität eines Systems, die für einen Nutzer, einen Käufer oder eine Software relevant ist.

Folgende Definition von User Stories nach Mike Cohn [1] wird in dieser Arbeit verwendet:

„*User Stories setzen sich aus drei Bestandteilen zusammen:*

- *Einer schriftlichen Beschreibung der Story, die zur Planung und als Erinnerung verwendet wird;*
- *Gesprächen über die Story, um die Details der Story herauszuarbeiten;*
- *Tests, die Details vermitteln und dokumentieren und mit denen festgelegt wird, wann eine Story vollständig umgesetzt ist.*“

Während es sich bei einer User Story um die Beschreibung einer Funktionalität des Systems, die sogenannte Story, handelt, ist eine Story Card die in den meisten Fällen verwendete Karteikarte, auf der die Story festgehalten wird [1].

Als Definition für Story Cards wird für diese Arbeit folgendes festgelegt:

„*Eine Story Card ist die visuell wahrnehmbare Repräsentation einer einzelnen User Story in Form von einer Karteikarte.*“[1]

Als nächstes wird auf den Aufbau und den Inhalt von Story Cards eingegangen. Dies impliziert auch die diesbezügliche Betrachtung von User Stories, da diese Teil des Inhaltes von Story Cards sind.

[1] Es handelt sich hierbei um eine eigenständige Definition, die auf Ausführungen von Mike Cohn [1] und Ron Jeffries [18] basiert. Der Grund für die eigene Definition liegt beim Bezug auf der Form der Story Card als Karteikarte, welche im Allgemeinen sonst nicht festgelegt ist, aber in dieser Arbeit eine besondere Rolle spielt.

2.1.2 Format einer Story Card

Zuerst muss das Format einer Story Card geklärt werden, bevor auf den eigentlichen Inhalt eingegangen werden kann. Aus der obigen Definition ist zu entnehmen, dass es sich bei den hier betrachteten Story Cards um Karteikarten handelt, die die User Story enthalten.

Es stellt sich die Frage, wie und warum man die gesamte User Story einer einzelnen Funktionalität auf eine einzige kleine Karteikarte bekommen soll, wenn man diese doch auch zum Beispiel auf einem normalen DIN-A4 Blatt niederschreiben kann, wo man sehr viel mehr Platz hat und somit viel mehr Details festhalten kann.
Genau an dieser Stelle liegt der Grund, weshalb aus der agilen Sicht Karteikarten geeigneter sind. Wie anfangs bereits erwähnt, bildet die Kommunikation zwischen den beteiligten Personen den Kern des Konzeptes der User Stories [2]. Die Story Cards fungieren hierbei als eine Gedanken- und Erinnerungsstütze für den Kunden und die Entwickler. Durch das Format der Karteikarte wird der Platz für eine User Story mit Absicht beschränkt, um dafür zu sorgen, dass die Ausarbeitung der Details einer Funktion erst dann in einem Gespräch erfolgt, wenn die Funktion realisiert werden soll und nicht bereits irgendwann vorher [1], [3].
Es ist aber anzumerken, dass es nicht verboten ist, Details bereits in der Besprechung, also bei der Entstehung einer User Story, festzulegen. Diese Details sollen sich aber nicht in der User Story selbst wiederfinden, sondern als Tests für die Funktionalität auf der Rückseite der Karteikarte vermerkt werden. Sonst können zu umfangreiche Details in einer User Story die Kommunikation zwischen Kunde und Entwickler ersetzen, was auf gar keinen Fall passieren darf [1], [3].

Somit lässt sich zusammenfassend sagen, dass eine Story Card mit dem Format einer Karteikarte das Konzept von User Stories unterstützt [3]. Eine Story Card ist vor allem eine Gedächtnisstützte und dient dazu, die wichtigen Details für die Wiederaufnahme des Gespräches bereitzustellen.

2.1.3 Inhalt von Story Cards

Nach dem Festlegen des Formates von den hier betrachteten Story Cards, wird jetzt erarbeitet, was eine Story Card alles beinhaltet kann.
Hauptbestandteil einer Story Card ist eine User Story. Aber es gibt noch einige weitere wichtige Aspekte, für die es sich anbietet, diese mit auf der Karteikarte festzuhalten.

Doch zuerst einmal zu dem Punkt der User Story. In einer User Story wird eine Funktionalität eines Systems schriftlich fixiert. Es gibt aber keine fest vorgeschriebene Form, wie eine User Story auszusehen hat. Man kann zwischen zwei Arten unterscheiden [9], der informellen oder der formalen. Während es bei der informellen Variante keine feste Struktur für den Text der Story gibt, wird bei der formalen Fassung eine Vorlage, wie beispielsweise die Folgende, für den Aufbau der Story verwendet. [1] (S.100)

„*Ich als (Rolle) möchte (Funktion), damit (Geschäftswert).*“

Diese Vorlage stammt aus dem Extreme Programming und bietet neuen und noch ungeübten Anwendern von User Stories einen guten Leitfaden zum Einstieg. Durch die vorgegebene Struktur werden alle relevanten Punkte, die eine Story beantworten muss, um eine Funktionalität zu erfassen, abgedeckt. Mit *Rolle* wird erfasst für wen die Funktionalität ist. *Funktion* bezieht sich auf die eigentliche Funktionalität mit der sich die User Story befasst. Und *Geschäftswert* stellt das Ziel beziehungsweise den Grund dar, warum es diese Funktion geben soll.

Welche der beiden Arten man am Ende wählt, bleibt einem selbst überlassen. Wichtig ist nur, dass beim Schreiben einer Story sechs Eigenschaften beachtet werden, damit eine gute Story entsteht.

Eine gute Story muss laut Cohn [1]:

- *unabhängig,*
- *verhandelbar,*
- *werthaltig für User oder Kunden,*
- *schätzbar,*
- *klein und*
- *testbar sein*

Im Folgenden wird geklärt, warum die obigen Eigenschaften für die Arbeit mit User Stories von Relevanz sind. Dazu gilt es zu wissen, wie der Ablauf eines Projektes in der agilen Softwareentwicklung festgelegt ist, da die einzelnen Eigenschaften dazu dienen, die Prozesse der Entwicklung zu unterstützen und zu erleichtern.

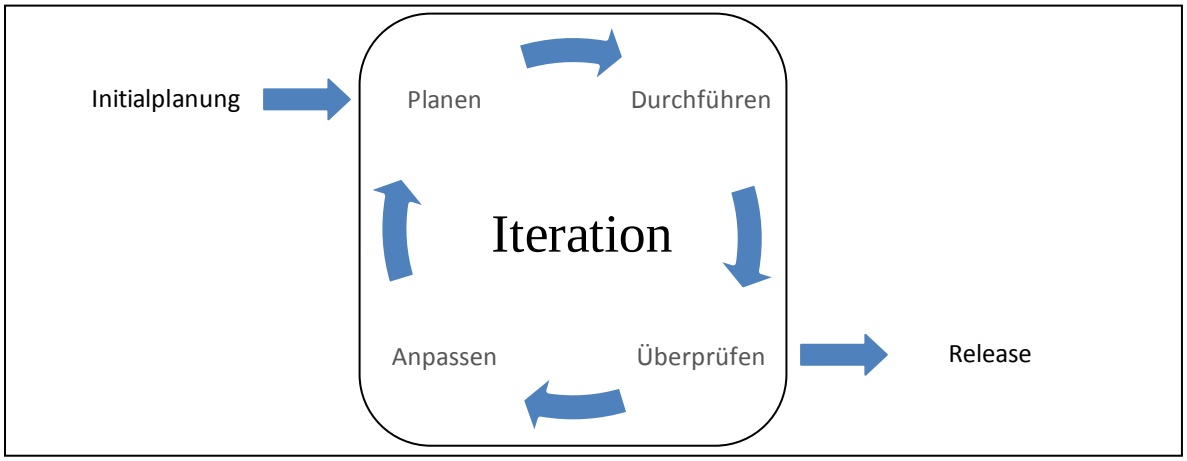


Abb. 3 - Agiler Regelkreis

Wie in der Abbildung 3 zu sehen ist, wird ein iterativer Ansatz verfolgt, in dem ein fester Regelkreis durchlaufen wird [2]. Ein einzelner solcher Zyklus, der aus den Prozessen „*Planen*“, „*Durchführen*“, „*Überprüfen*“ und „*Anpassen*“ besteht, bildet eine Iteration.

Während der Planung werden die für eine Iteration zu implementierenden Story Cards ausgewählt. Da jede Funktionalität durch eine separate Story Card erfasst wird und ein Team nur eine bestimmte Menge an Story Cards in einer Iteration bearbeiten kann, ist es wichtig, dass die einzelnen Funktionen weitestgehend *unabhängig* sind. Ansonsten wird das Planen einer Iteration unnötig erschwert.

Die festgelegten Stories stellen dabei keinen Vertrag dar, sondern dienen ausschließlich als Gedächtnisstütze für weitere Gespräche. Folglich ist es wichtig, dass die Stories als *verhandelbar* angesehen werden und nicht als ein bindendes vertragliches Dokument.

Alle aufgeschriebenen Stories für Funktionen müssen einen Geschäftswert aufweisen und damit *werthaltig* sein. Ist dies nicht der Fall, so gibt es keinen Grund diese Funktionen zu implementieren, da sie für den Nutzer oder den Kunden keinen Wert besitzen und von diesen auch nicht priorisiert werden können.

Für die Planung einer Iteration kommt es darauf an, dass die Entwickler den Umfang einer Funktion einschätzen können. Nur so sind sie in der Lage die Story Cards zu ermitteln, die sie in einer Iteration vollständig implementieren können. Zur Steigerung der Genauigkeit einer Planung ist es also nötig, dass die Stories *schätzbar* sind.

Hierbei ist auch zu beachten, dass die Stories eine geeignete Länge haben. Sie müssen klein genug sein, damit das Entwicklerteam sie vernünftig einplanen kann. Ist eine Story zu klein gewählt worden, so verursacht sie mehr Arbeit bei der Einplanung und Schätzung als bei einer direkten Realisierung. Eine zu große Story hingegen, die als Epic bezeichnet wird, verursacht genauso Probleme, weil sie meist aufgrund ihrer Länge mehrere Stories enthält und damit für das Team nicht kalkulierbar ist. Was letztendlich die Bedeutung von *klein* bei einer User Story ist, hängt von dem Entwicklerteam, seinen Fähigkeiten und der verwendeten Technologie ab.

Um die Fertigstellung einer User Story festzustellen, ist die letzte Eigenschaft einer Story unabdingbar. Nur wenn eine Story *testbar* ist, sind die Entwickler in der Lage zu prüfen, ob sie eine Story richtig und vollständig nach den Vorstellungen des Kunden implementiert haben [3].

Nachdem die User Story im Detail betrachtet wurde, gilt es sich mit den restlichen Punkten zu befassen, die sich als potentieller Inhalt für eine Story Card eignen.

Wenn eine neue Story entsteht, ist es praktisch dieser einen Kurztitel zuzuweisen, damit sie in dem späteren Verlauf der Entwicklung leicht identifiziert werden kann. Dies sorgt zum einen für eine gute Übersicht und zum anderen für eine bessere Kommunikation, da alle Beteiligten aus dem Titel sofort wissen, womit sich der Inhalt der Story Card befasst.

Während der Entstehung einer User Story werden immer mehr Details der Funktionalität ausgearbeitet. Wie in Abschnitt 2.1.1 bereits erwähnt, gehören die Details nicht in die Story selbst, sondern sind als Tests beziehungsweise Erwartungen für die spätere Überprüfung der fertigen Implementierung auf der Rückseite der Story Card zu vermerken.

Neben den Details entstehen auch weitere Fragen bezüglich der Implementierung. Es bietet sich an, diese Fragen sowie eine Zusammenfassung des Gespräches von ein bis zwei Sätzen auf der Vorderseite der Story Card zu notieren. Dadurch wird die Wiederaufnahme der Kommunikation bei der Implementierung der Funktion für die Beteiligten erleichtert und es wird sichergestellt, dass die bereits ermittelten Fragen nicht verloren gehen.

Desweiteren können sich auch Beschränkungen für eine Funktionalität aufzeigen, die bei der Umsetzung mit beachtet werden müssen. Wenn sich diese Beschränkungen nicht direkt als Tests formulieren lassen, so ist es dennoch wünschenswert, dass sie zusätzlich fixiert werden, damit sie bei der späteren Testphase mit überprüft werden können.

Wenn die User Stories soweit besprochen wurden, dass mit der Implementierungsphase begonnen werden kann, müssen zuvor noch zwei Zwischenschritte absolviert werden.
Einer der beiden Schritte sieht vor, dass der Kunde mit Hilfe der Entwickler die fertigen Story Cards in dem Sinne priorisiert, dass durch die Realisierung der Stories der Geschäftswert für das Unternehmen des Kunden maximiert wird [2], [3]. Die Priorität für eine Story Card ist sehr wichtig und wird daher auf der Vorderseite der Story Card mit erfasst, weil bei der Planung der Iterationen immer zuerst die wichtigsten Stories umgesetzt werden müssen.

Bei dem anderen Zwischenschritt stehen die Entwickler vor der Aufgabe zu schätzen, wie viel Zeit sie für die Realisierung einer bestimmten Story Card benötigen. Da es sich hierbei auch um eine Information handelt, die für die Planung einer Iteration zwingend notwendig ist, wird sie ebenfalls auf der Vorderseite der Story Card vermerkt.
Die Dauer für die Umsetzung einer Story Card wird als deren Kosten beziehungsweise Aufwand bezeichnet und in der Einheit „*Story Points*“ gemessen. Der Vorteil einer neutralen Einheit liegt darin, dass jedes Team selbst definieren kann, was ein Story Point genau sein soll. Damit ist man nicht an feste Einheiten wie Stunden oder Tage gebunden, sondern kann spezifischer definieren, was bei der Schätzung des Aufwandes einer Story als Ausgangsbasis fungiert. In einer Iteration können die Entwickler nur eine bestimmte Menge an Story Points umsetzen. Dies ist der sogenannte „*Velocity*“ des Teams. Für die Planung von einer Iteration ist es wichtig, dass die Summe der Story Points der ausgewählten Story Cards den Velocity-Wert des Teams nicht überschreitet.
In welcher Reihenfolge die beiden Zwischenschritte, Priorisierung und Schätzung, erfolgen ist jedoch nicht eindeutig festgelegt.

2.1.4 Entwurf eines Layouts für eine Story Card

Auf Basis der zuvor ausgearbeiteten Aspekte ist ein mögliches Layout für den strukturellen Aufbau einer Karteikarte erstellt worden (siehe Abb. 4 und Abb. 5). Diese Schablone bezieht alle der zuvor genannten Punkte für den möglichen Inhalt einer Story Card mit ein.

Zwar rät Mike Cohn von einer Schablone ab, da sonst zu weit vorausgedacht wird und dies unnötigen Aufwand verursachen kann [1], aber es gibt mehrere gute Gründe, warum eine Vorlage für eine Story Card in diesem Fall von Vorteil ist.

Zuerst ist anzumerken, dass man durch die Vorgabe des möglichen Inhaltes einer Story Card unerfahrenen sowie erfahrenen Nutzern eine erleichterte Bedienung und Unterstützung zukommen lassen kann. Desweiteren ist es für die spätere Verwaltung der Daten nötig, dass man die in einer Story Card enthaltenen Informationen vernünftig auslesen kann.
Das Erfassen der Kosten von Story Cards für bestimmte Auswertungsdiagramme, wie zum Beispiel einem Burndown Chart, würde durch ein nutzerspezifisches Design der Karten erschwert werden. Zusätzlich wird die Kommunikation im Team gestört, wenn jedes Teammitglied seine eigene Kartenstruktur definiert [6]. Bei verteilten Teams ist dieser Punkt noch um einiges gravierender, da die Kommunikation zwischen den verschiedenen Gruppen durch unterschiedliche Darstellungen der Story Cards unnötig verkompliziert wird.

Priorität:_____	Titel:_____	Aufwand:_____
User Story:_____ _____ _____ _____ _____ _____ _____		
Anmerkungen:_____ _____ _____		

Abb. 4 - Story Card Vorderseite

Tests / Erwartungen:

Beschränkungen:

Abb. 5 - Story Card Rückseite

2.1.5 Vor- und Nachteile von User Stories

Für die Unterstützung des User Story Managements und um ein effizientes Arbeiten mit Story Cards zu ermöglichen, ist es wichtig die Vor- und Nachteile von User Stories in der Softwareentwicklung zu kennen. Durch das Wissen über die Stärken und Schwächen dieser agilen Methodik können zielgerichteter Anforderungen an die Applikation erhoben werden. Damit kann die Integration der Vorteile in das System sichergestellt werden und für potentielle Nachteile können möglicherweise Verbesserungen entwickelt werden.

Die Ausarbeitung der Vor- und Nachteile wird durch einen Vergleich mit einer anderen, nicht agilen Softwareentwicklungsmethode, erfolgen. Bei der ausgewählten Methode handelt es sich um eine Anforderungsspezifikation für Software nach IEEE 830 [10].
Es gibt zwei Gründe, warum gerade diese Methode ausgewählt worden ist. Zum einen handelt es sich bei einer solchen Anforderungsspezifikation nach IEEE 830 um eine der bekanntesten Methoden zur Erfassung von Anforderungen in der klassischen Softwareentwicklung. Zum anderen lassen sich durch den Vergleich von einer agilen und einer klassischen Softwareentwicklungsmethode sehr gut die Stärken und Schwächen der agilen Herangehensweise ausarbeiten. Desweiteren weisen beide Techniken sehr unterschiedliche Charakteristiken auf und ermöglichen durch einen Gegenüberstellung eine gute Erkenntnis darüber, wodurch sich gerade gewisse Vorteile und Nachteile ergeben.

Bevor es zu dem eigentlichen Vergleich kommt, wird kurz geklärt was eine Anforderungsspezifikation für Software nach IEEE 830 ist.
Es handelt sich dabei um ein Dokument, das sehr detailliert die Anforderungen an ein System schriftlich fixiert. Diese Anforderungen werden zwar mit dem Kunden abgesprochen, aber nur von Anforderungsanalytikern ausformuliert und niedergeschrieben.
Daraus resultiert als Konsequenz ein sehr langes Schriftstück, das einen stark Technik spezifischen Fachjargon aufweist. Aufgrund der Größe, der Komplexität und der Relevanz der Spezifikation ist für deren Ausarbeitung ein großer Teil der gesamten Projektzeit aufzuwenden. Nur so kann gewährleistet werden, dass die Anforderungen des Kunden von den Analytikern korrekt erfasst werden [1], [3].

Im Folgenden werden zuerst die Vorteile von User Stories erörtert und dabei gegenüber einer IEEE 830 Anforderungsspezifikation abgegrenzt, bevor auf die Nachteile genauer eingegangen wird.

Wie aus dem Abschnitt 2.1.1 bekannt ist, ist einer der drei zentralen Bestandteile von einer User Story das Gespräch zur Entwicklung der Story. Somit sind die Mitarbeit des Kunden und die verbale Kommunikation zwingend erforderlich für die erfolgreiche Anwendung von der agilen Methode der User Stories.

Aus diesen Punkten ergeben sich mehrere Vorteile. Im Gegensatz zur Anforderungsspezifikation repräsentieren die kurzen Informationen in einer User Story nur eine Gedächtnisstütze, was die Notwendigkeit der verbalen Kommunikation mit dem Kunden bei der Implementierung verdeutlicht. Der Kunde wird als ein fester Bestandteil des Teams integriert, was für die Entwickler bedeutet, dass ihnen ein permanenter Ansprechpartner bei Fragen oder Unklarheiten zur Seite steht [3].
Durch den Fokus der verbalen Kommunikation minimiert sich die Gefahr von Missverständnissen und Fehlinterpretationen, zu welchen es bei einer IEEE 830 Spezifikation kommen kann, weil das geschriebene Wort entweder unpräzise oder mit mehrfacher Bedeutung behaftet seien kann. Die Entwickler müssen aber aus dem Anforderungsdokument interpretieren, was vom Kunden gewünscht wird, ohne vielleicht jemals mit diesem gesprochen zu haben. Ebenfalls unterscheiden sich die beiden Methoden deutlich in der Übersichtlichkeit und der Verständlichkeit. Während eine Anforderungsspezifikation sehr lang ist und viele Seiten umfasst, sind User Stories kurz und klein, was sich unter anderem aus dem Karteikartenformat und dem Fokus auf der verbalen Kommunikation ergibt.
Desweiteren werden Spezifikationen von Analytikern verfasst, welche meist einen sehr technischen Jargon verwenden [3], der nicht für jeden verständlich ist. Dem gegenüber verfügen User Stories über einen eher auf dem natürlichen Sprachgebrauch basierenden Inhalt, da sie gerade vom Kunden oder zumindest mit diesem zusammen, geschrieben werden [6].
Insgesamt kann durch das Karteikartenformat, in Kombination mit den Gesprächen zwischen Kunde und Entwickler, die Übersichtlichkeit und das Verständnis bei User Stories, im Vergleich zu einer Anforderungsspezifikation, erhöht werden.

Weiterhin ergibt sich aufgrund der Länge der Dokumente eine sehr unterschiedliche Dauer für die Erstellung selbiger.
So nimmt die Ausarbeitung einer klassischen Spezifikation um einiges mehr Zeit in Anspruch, als es User Stories in der agilen Softwareentwicklung tun. Diese können in sehr kurzer Zeit in einer großen Menge erstellt werden. Ein gutes Beispiel dafür sind sogenannte Story-Workshops, die genau dieses Ziel verfolgen und dadurch gut für den initialen Start einer agilen Softwareentwicklung geeignet sind [1].
So ergibt sich als Vorteil eine sehr kurze Entwurfsphase auf die zügig die Entwicklung der Software in Iterationen erfolgen kann.
Durch das Arbeiten in Iterationen und dem Kunden als Teil des Teams, ist es möglich am Ende einer Iteration, in der Überprüfungs- und Anpassungsphase (siehe Abb. 3), mittels Feedback und Analyse die weiteren Iterationen zu justieren, indem neue Story Cards erstellt und alte korrigiert oder verworfen werden.
Die daraus resultierende Flexibilität in dieser Art der Entwicklung ist der wichtigste positive Aspekt von der agilen Methodik der User Stories. Sie bietet den Entwicklern durch die Möglichkeit der schnellen und einfachen Justierung einen entscheidenden Vorteil, welcher sich in der klassischen Softwareentwicklung nicht so einfach realisieren lässt [1], [3].

Aber die Anwendung von Story Cards bringt auch gewisse Nachteile mit sich, die man kennen muss, denn sonst kann es zu fatalen Auswirkungen auf den Erfolg des Projektes kommen.

Da für jede Funktion eine eigene Story Card geschrieben wird, kann man leicht den Überblick über möglicherweise vorhandene Abhängigkeiten verlieren. Diese bringen aber durchaus Komplikationen bei der Planung der einzelnen Iterationen mit sich. Sollten Abhängigkeit existieren, so muss immer sichergestellt werden, dass eine Story Card, die auf einer für eine spätere Iteration geplanten Funktion basiert, nicht vorher implementiert wird. Ansonsten müssen alle bisher erstellten Planungen von Iterationen und die aktuelle Iteration selbst überarbeitet werden, was zusätzlichen Arbeitsaufwand verursacht.

Der Velocity-Wert verkompliziert die Planung weiterhin [2]. Wenn man für eine Iteration mehr Aufwand einplant, als überhaupt möglich ist, ergeben sich direkt Verzögerungen und Probleme für den weiteren Verlauf des Projektes.
Bei einer Anforderungsspezifikation ist die Gefahr von Verzögerungen und damit verbundener Probleme zwar auch vorhanden, wenn mehr eingeplant wird als in der gegebenen Entwicklungszeit geschafft werden kann. Da aber von Anfang an exakter und ausgiebiger geplant und dokumentiert wird, müssten die Risiken für das Eintreten möglicher Verzögerungen und Probleme früh erkannt werden und somit leichter zu korrigieren sein [1].

Eine ausführliche Dokumentation, wie sie bei der klassischen Methode fokussiert wird, tritt bei der Anwendung von User Stories in den Hintergrund. Diese dienen nur als eine Gedächtnisstütze und bieten selbst keine Lösung für das Problem, der vollständigen Dokumentation des Projektes. Um die Rückverfolgbarkeit des Projektes zu gewährleisten, muss eine eigenständige Lösung durch das Team gefunden werden.
Das Fehlen von Dokumentation und der Fokus auf dem Gespräch lässt die Weitergabe von Informationen zwischen den Teammitgliedern bei größeren oder gar verteilten Teams zu einem Problem werden. Der resultierende Aufwand für flexiblen und eher spontanen Informationsaustausch, sei er verbal oder textuell, um alle Mitglieder auf dem gleichen Kenntnisstand zu halten, kann vereinfacht werden, wenn von Projektbeginn an eine definierte Art der Protokollierung relevanter Information erfolgt. Diese muss von allen Beteiligten eingesehen werden können und für sie permanent zur Verfügung stehen [1], [3], [6], [8].

Wie man aus dem obigen Vergleich sehen kann, gibt es gute Gründe für die Anwendung der agilen Methode der User Stories. Nichts desto trotz muss man sich immer im Klaren darüber sein, dass es auch einige Nachteile gibt, für die man eine geeignete Lösung finden muss.

Nach dem abgeschlossenen Vergleich von User Stories mit einer IEEE 830 Anforderungsspezifikation sollen im nachfolgenden Abschnitt die Eigenschaften, Vor- und Nachteile von User Stories beziehungsweise Story Cards noch einmal explizit erfasst werden.

2.1.6 Zusammenfassung der Eigenschaften, Vorteile und Nachteile

Die folgende Zusammenfassung dient zum einen als eine Übersicht, um einen schnellen Überblick über die in den vorherigen Abschnitten erarbeiteten relevanten Aspekte zu ermöglichen. Zum anderen soll dieser Abschnitt später als Verweis für die Ausarbeitung der zu erhebenden Anforderungen und daraus resultierenden Konzepte der mobilen Applikation verwendet werden.

Für eine gute User Story wurden im Anschnitt 2.1.3 die folgenden wichtigen Eigenschaften erarbeitet.

Eine gute Story muss:

- *unabhängig,*
- *verhandelbar,*
- *werthaltig für User oder Kunden,*
- *schätzbar,*
- *klein und*
- *testbar sein*

In dem Vergleich von User Stories mit einer klassischen IEEE 830 Anforderungsspezifikation aus dem vorherigen Abschnitt 2.1.5 resultieren die hier aufgezählten Vor- und Nachteile.

Vorteile von User Stories:

- *Fokussierung der verbaler Kommunikation zwischen den Beteiligten*
- *Integration des Kunden als fester Bestandteil des Teams*
- *Stetiges Feedback durch die Mitarbeit des Kunden*
- *Gute Überschaubarkeit und Handhabung der Story Cards aufgrund des Formates und der permanenten Kommunikation*
- *Gutes Verständnis anhand von natürlicher Sprache und gemeinsamer Ausarbeitung*
- *Einfache und schnell Produktion von Story Cards in großer Menge und damit verbunden ein schneller Start der Iterationen*
- *Sehr hohe Flexibilität durch iteratives Vorgehen mit Anpassung der nachfolgenden Iterationen*

Nachteile von User Stories:

- *Abhängigkeiten zwischen Funktionen sind schwer erkennbar bei einer großen Anzahl Story Cards*
- *Komplexe und aufwändige Planung durch das Arbeiten in Iterationen*
- *Story Cards sind nur Gedächtnisstützen und stellen keine detaillierte Dokumentation dar*
- *Weitergabe und Verteilung von Informationen ist durch den Fokus der verbalen Kommunikation in großen und verteilten Teams kompliziert*

2.2 Prozesse der agilen Softwareentwicklung mit User Stories

Zum Abschluss des zweiten Kapitels und um damit die Grundlagen, die für die Ausarbeitung der Anforderungen für ein User Story Management Tool nötig sind, abzuschließen, gilt es sich die Prozesse der agilen Softwareentwicklung im Detail anzusehen. Durch die kurze Betrachtung der einzelnen Prozesse soll ein besseres Verständnis der agilen Arbeitsweise aufgebaut werden.

Die agile Softwareentwicklung setzt sich aus drei Hauptphasen zusammen, dem Initial-Start, der Iteration und dem Release. Die Iteration und der Release sind dabei zusätzlich noch weiter gegliedert. Eine Iteration besteht, wie zuvor in Abschnitt 2.1.3 erwähnt, aus den Phasen Planung, Durchführung, Überprüfung und Anpassung und ein Release wiederum aus mehreren Iterationen.

Der Initial-Start ist die erste Phase in der agilen Softwareentwicklung und dient zur Planung des Projektes. Dabei gilt es zuerst einmal festzulegen wie lange eine Iteration genau dauern soll. Meist liegt die Dauer einer Iteration zwischen zwei bis vier Wochen. Weiterhin müssen die Entwickler eine erste Schätzung des Velocity-Wertes des Teams abgeben.

Nach dem ersten Sammeln von Story Cards kann mit den Informationen bezüglich der Dauer und des Velocity-Wertes die Planung der ersten Iterationen und damit wiederum verbunden, die Planung des ersten Releases erfolgen.

Eine Iteration besteht wie schon zuvor erwähnt aus vier Teilprozessen.
Die erste Phase, die Planung, dient dazu die entsprechende Iteration im Detail zu organisieren. Dazu gehören insbesondere die Besprechungen des genauen Inhalts der Story Cards, die in der Iteration umgesetzt werden sollen.
In der zweiten Phase, der Durchführung, findet die eigentliche Implementierung der Story Cards statt. Dabei arbeiten Entwickler und Kunde eng miteinander zusammen, um die ausgewählten Funktionalitäten so exakt wie möglich nach den Vorstellungen des Kunden zu realisieren.
In der Phase der Überprüfung gilt es festzustellen, ob die realisierten Story Cards auch wirklich dem entsprechen, was der Kunde haben möchte. Dabei werden zum einen Tests durchgeführt und zum anderen wird die bis dahin entstandene ausführbare Software dem Kunden präsentiert, um von diesem Feedback zu erhalten.
In der letzten Phase einer Iteration, der Anpassung, findet eine Korrektur der weiteren Planung der nächsten Iterationen statt. Dabei werden neu gewonnene Erkenntnisse auf Basis der Testergebnisse, dem Feedback des Kunden und der Analyse der aktuellen Iteration verwendet. Diese ermöglichen eine genauere Schätzung des Velocity-Wertes, das Erstellen neuer Story Cards oder das Überarbeiten beziehungsweise Löschen von vorhandenen Story Cards. Daraus resultiert wiederrum eine Korrektur der Planung der weiteren Iterationen, was auch wieder Einfluss auf die Planung des ersten Release haben kann.

Ein Release ist das Zusammenfassen mehrerer Iterationen. Je nach Länge einer einzelnen Iteration werden meist vier bis acht Iterationen zu einem Release zusammengefasst. Am Ende eines Release erfolgt die Abgabe der fertigen verwendbaren Software an den Kunden. Es kann vorkommen, dass die vom Kunden gewünschte Software am Ende eines Release noch nicht in vollem Umfang fertiggestellt wurde. In dem Fall spricht man von einem sogenannten Zwischenrelease. Hierbei erhält der Kunde aber wie bei einem normalen Release fertige Software, die er in gewissem Umfang bereits nutzen kann, und es beginnt eine neue Release-Phase mit weiteren Iterationen.

3 Anforderungsspezifikation

In diesem Kapitel werden die Anforderungen für die zu entwickelnde Applikation zur Unterstützung des User Story Managements erhoben.
Dabei werden zuerst sämtliche identifizierten Anforderungen für eine solche Anwendung erfasst, ausgearbeitet und strukturiert, um dann eine Auswahl bestimmter Anforderungen zu treffen, die zentral in der Implementierung realisiert werden sollen.

3.1 Anforderungserhebung

Bevor es zu der eigentlichen Erhebung kommt, soll eine kurze Einführung in die nachfolgende Strukturierung der Anforderungen erfolgen, damit die gewählte Aufteilung besser verstanden wird.

Zunächst werden die Anforderungen in zwei Rubriken, die „*allgemeinen Anforderungen*“ und die „*prozessspezifischen Anforderungen*“, unterteilt. Die „*allgemeinen Anforderungen*“ beziehen sich dabei auf generelle Aspekte, die allgemein für eine mobile Applikation, die zur Unterstützung des User Story Managements dienen soll, wichtig sind. Bei den „*prozessspezifischen Anforderungen*“ hingegen werden Aspekte betrachtet, die für eine zufriedenstellende und erfolgreiche Arbeit in den einzelnen Prozessen der agilen Entwicklung wichtig sind. Da jede Anforderung dieser Rubrik in Bezug zu einem bestimmten Prozess der agilen Entwicklung steht, werden die „*prozessspezifischen Anforderungen*“ zusätzlich bezüglich der einzelnen Prozesse gegliedert.
Die Anforderungen der zwei oben genannten Rubriken werden noch einmal in zwei Kategorien aufgeteilt. Die erste Kategorie heißt „*Effizienter Umgang mit Story Cards*“. Alle hier eingeordneten Anforderungen legen den Fokus auf das Ziel einer Arbeitsweise wie mit realen Karteikarten. Dabei ist es wichtig, dass durch die Realisierung der Anforderungen dieser Kategorie die zuvor erarbeiteten Eigenschaften von User Stories und die Vorteile dieser agilen Methodik in das elektronische System überführt werden. Die Nachteile bei der Entwicklung mit User Stories sollen jedoch, wenn möglich, durch eine geeignete Lösung kompensiert werden.
Für eine mobile Applikation zur Unterstützung des User Story Managements sind die Anforderungen dieser Kategorie zwingend erforderlich und dürfen nicht vernachlässigt werden. Daher sind diese Anforderungen als Pflichtanforderungen zu verstehen und besitzen somit eine sehr hohe Priorität bei der Implementierung.
In der zweiten Kategorie „*Guten Prozess vorgeben / ermöglichen*“ werden die Anforderungen einsortiert, die als Ziel die Erfüllung von Bedürfnissen von Nutzern verfolgen [7]. Sie bieten durch ihre Umsetzung zwar Erleichterungen in den Arbeitsprozessen und stellen Hilfen für das erfolgreiche Arbeiten dar. Aber die Realsierung dieser Anforderungen bietet eine eher nutzerspezifische Unterstützung, welche für ein allgemeines, effizientes Arbeiten mit User Stories nur von geringerem Nutzen ist. Daher sind diese Anforderungen, in Bezug auf das Ziel dieser Arbeit, als optional zu betrachten, sodass ihre Umsetzung, in der zu entwickelnden Applikation, eher zweitrangig ist.

Die nun folgenden Anforderungen sind auf Basis der zuvor erarbeiteten Kapitel erhoben und ausgearbeitet worden. Dabei betrachten sie speziell die fokussierte Problematik Aktivitäten mit echten Karteikarten adäquat in das elektronische System zu überführen und das Ziel der Unterstützung des User Story Managements.

3.1.1 Allgemeine Anforderungen an die Applikation:

1. Kategorie: Effizienter Umgang mit Story Cards

[R1] Arbeiten wie mit echten Karteikarten (Zeichnen, Erstellen, Editieren, Löschen)

Aufgrund des Zieles der Unterstützung des User Story Managements ist es wichtig, dass das grundlegende Prinzip der Methodik der User Stories auch geeignet in die mobile Applikation überführt wird. Die Story Card ist dabei das Rahmenwerk, welches eine User Story beinhaltet. Daher soll der Umgang mit diesen in der Applikation möglichst realitätsnahe sein.
Das bedeutet, dass es wichtig ist die Story Cards einfach und schnell erstellen, löschen und editieren zu können [6]. Dies soll durch eine gute Bedienbarkeit, die sich in einer einfachen und direkten Bedienung bezüglich der Menüstrukturen ergibt, unterstützt werden. Für den Nutzer wird somit der Umgang mit der Anwendung erleichtert [7]. Die Integration des Zeichnens auf einer Karteikarte, in das System, bietet dem Nutzer, neben der Möglichkeit Klartext einzutippen, eine zusätzliche intuitive Eingabevariante [7].
Durch die Realisierung dieser Funktionen werden die Vorteile der einfachen und schnellen Produktion von Story Cards in großer Menge und der hohen Flexibilität der agilen Entwicklung in die mobile Anwendung überführt (siehe Abschnitt 2.1.6, Vorteile von User Stories). Die Flexibilität ist die wichtigste Eigenschaft der agilen Softwareentwicklung und wird durch die Leichtigkeit der Erstellung und der Anpassbarkeit der Story Cards in der Applikation gefördert.
Dadurch resultiert insgesamt eine effiziente Unterstützung der Arbeit mit User Stories in den Prozessen der agilen Entwicklung.

[R2] Gute allgemeine Übersicht bezüglich wichtiger Daten mit leichtem Zugriff auf weitere Details bezüglich Story Cards und Pinnwänden

Für die Handhabung der Story Cards und der verschiedenen Projekte ist es wichtig aufgrund des eingeschränkten Platzes auf einem Tablet-Display geeignete Darstellungen für diese zu entwickeln. Da die in Story Cards beziehungsweise Projekten gespeicherten Daten nicht immer von der gleichen Relevanz sind, bietet es sich an für den Nutzer verschiedene Übersichten bereitzustellen.
So ist es wichtig, dass für den ersten Überblick nur absolut notwendige Daten einsehbar sind, um ein effizientes Arbeiten zu ermöglichen. Wenn zusätzliche Informationen für die weitere Arbeit notwendig sind, gilt es, dass auf diese leicht zugegriffen werden kann.
Dies erfordert die Realisierung eines geeigneten Interfaces für die Story Cards und Projekte. Das Interface muss dafür die Überschaubarkeit der Daten fokussieren und wenn nötig, ein einfaches Einsehen aller Daten ermöglichen [7].
Somit kann die Komplexität des Prozesses der Planung in der agilen Entwicklung verringert werden (siehe Abschnitt 2.1.6, Nachteile von User Stories), was für die Arbeit mit User Stories eine Erleichterung darstellt.

[R3] Lokale und globale Projekte in der Applikation anlegen können

Vom Fachgebiet Software Engineering wird ein Webservice zur Verfügung gestellt, der als zentraler Hauptspeicher für Story Cards interagiert. Ein globales Projekt ist stets mit einem solchen aufgesetzten Webservice verbunden. Dabei

fungiert der Webservice als Repository[2] und eine Kopie der Story Cards von dem Webservice auf einem Tablet ist als Working Copy[3] zu interpretieren.
Durch die globale Projektvariante ist es möglich, dass jedes Mitglied eines Entwicklerteams zu jederzeit und an jedem Ort Zugriff auf die elektronische Projektpinnwand mit den Story Cards hat. Dies ist besonders wichtig für verteilte Teams, da sie so, wie ein lokales Team, über einen zentralen Ort für ihre Pinnwand mit einer einheitlichen Repräsentation der Story Cards verfügen (siehe Abschnitt 1.1).
Die globale Projektvariante fördert ebenfalls die Integration des Kunden ins Team, da dieser durch die mobile Applikation stets über den aktuellen Arbeitsverlauf des Projekts informiert ist, selbst wenn er nicht immer bei dem Entwicklerteam Vorort seien kann.
Um die Verwendbarkeit der Applikation weiter zu erhöhen, soll es zusätzlich möglich sein, ein Projekt auch lokal auf dem Tablet zu erstellen. Ein solches lokales Projekt ist unabhängig von einem Webservice und organisiert die komplette Verwaltung der Story Card- und Projektdaten ausschließlich auf dem Tablet.
Dies ermöglicht es einer einzelnen Person die Applikation in vollem Umfang nutzen zu können, ohne auf einen Webservice angewiesen zu sein. Dadurch steht es jedem Nutzer frei, seine eigene Arbeit mit Hilfe der Applikation durchzuführen und zu unterstützen, selbst wenn die Nutzung der Anwendung in einem Projekt nicht vorgesehen ist.

[R4] Sicherstellung der Synchronisation zwischen Web-Service und Applikation

Bei globalen Projekten ist es wichtig, dass der Webservice stets für alle Teammitglieder, seien sie von einem lokalen oder einem globalen Team, die aktuellste Version einer Story Card zur Verfügung stellt. Nur so können alle Beteiligten auf dem gleichen aktuellen Informationsstand sein (siehe Abschnitt 2.1.6, Nachteile von User Stories). Um dies zu gewährleisten, muss die Synchronisation der Daten zwischen Webservice und Tablet sichergestellt sein.

2. Kategorie: Guten Prozess vorgeben / ermöglichen

[R5] Form und Aufbau der elektronischen Story Cards soll bezüglich Inhalt und Format vergleichbar mit echten Karteikarten sein

Um die Methodik der User Stories gut in einer elektronischen Anwendung umzusetzen, kommt es darauf an, dass die Prinzipien dieser Methodik bei der Implementierung beachtet und mit realisiert werden. Das Konzept der User Stories fokussiert die Kommunikation zwischen den am Projekt beteiligten Personen. Die Story Cards dienen nur als Gedankenstütze, um diese Kommunikation zu fördern. Daher gilt es, dass User Stories klein sein müssen (siehe Abschnitt 2.1.6, Eine gute Story muss:).
Um diese Eigenschaft, welche für eine gute User Story wichtig ist, zu unterstützen, soll der zur Verfügung stehende Platz für Aufzeichnungen auf der elektronischen Story Card wie bei einer echten Karteikarte beschränkt sein. Damit wird zusätzlich

[2] Ein Repository ist ein zentralen Datenspeicher eines Versionsmanagementsystems [19].
[3] Eine Working Copy ist eine lokale Arbeitskopie des Inhalts eines Repositories für einen Nutzer [19].

sichergestellt, dass die verbale Kommunikation fokussiert und nicht durch übermäßiges schriftliches Erfassen von Informationen vernachlässigt wird.
Die Arbeit von Teams soll dabei aber nicht durch verschiedene Layout-Varianten der Story Cards erschwert werden, weshalb die Verwendung einer Vorlage für den Aufbau einer Story Card angebracht ist. Dadurch wird eine einheitliche Repräsentation garantiert, was die Überschaubarkeit, die Handhabung und das Verständnis bei der Arbeit mit den Story Cards unterstützt (siehe Abschnitt 2.1.6, Vorteile von User Stories).

[R6] Eine Backlog- und eine Iterationsansicht

In der Applikation soll die virtuelle Pinnwand adäquat der realen Pinnwand nachempfunden sein und diese geeignet replizieren [7]. Da aber ein Tablet im Verhältnis zu einer echten Pinnwand sehr viel kleiner ist, ist es für eine gute Übersicht unpassend die komplette Struktur einer echten Pinnwand auf dem Tablet direkt nachzubilden. Die Konsequenz ist, dass der Nutzer entweder sehr oft die Ansicht korrigieren oder wechseln muss, um einen bestimmten Bereich zu sehen. Und wenn man nur eine vollständige Übersicht anbietet, werden einige Elemente zu klein dargestellt, wodurch sie nur schlecht zu erkennen sind.
Daher bietet es sich an, eine eigene Unterteilung in die zwei relevanten Bereiche Backlog und Iteration zur Verfügung zu stellen und Informationen, wie den Zustand einer Story Card, durch zum Beispiel Farben oder Icons darzustellen [1], [8].
Insgesamt kann so die gleiche Funktionalität wie die einer echten Pinnwand erreicht werden, wobei aber trotz der kleineren Oberfläche des Tablets eine gute Überschaubarkeit und leichte Handhabung angeboten werden kann.

[R7] Einsehbare Diagramme auf Basis von erstellten Logs

In der agilen Softwareentwicklung mit User Stories sind Auswertungsdiagramme nicht nur wichtig für die Verfolgbarkeit des Projektstatus, sondern auch für die Analyse einer beendeten Iteration, der damit verbundenen Anpassung der im Backlog vorhandenen Story Cards sowie der weiteren Planung.
Wenn diese Diagramme automatisch von der Anwendung generiert und dem Nutzer zur Verfügung gestellt werden, bedeutet dies eine Erleichterung in der Arbeit mit User Stories. Dem Nutzer wird dadurch Arbeit abgenommen, indem man ihm eine geeignete Unterstützung durch die Applikation zukommen lässt [7].

[R8] Möglichkeit der zusätzlichen Dokumentation inklusive erleichtertem Informationsaustausch

Story Cards dienen stellen keine detaillierte Dokumentation dar, weil sie nur wenig Platz für das Fixieren von Informationen bieten. Es gibt aber Fälle, in denen man trotz des Prinzips der agilen Entwicklung, zum Beispiel auf Wunsch des Kunden, eine ausführliche Dokumentation erstellen muss. Zusätzlich wird die Weitergabe und Verteilung von Informationen unter den Teammitgliedern erschwert, da der Fokus auf der verbalen Kommunikation liegt (siehe Abschnitt 2.1.6, Nachteile von User Stories).
Als Folge dessen ist es praktisch, wenn die Applikation Dokumentation und Informationsweitergabe in einem gewissen Maß ermöglicht.
So ist es denkbar, dass die Dokumentation und das Zusammenfassen von Informationen, welche für das gesamte Team wichtig sind, durch eine Art Post-

Bereich realisiert werden. Diese Bereiche sind für alle im Team permanent abrufbar und editierbar, wodurch der Austausch von Informationen erleichtert und ein gewisses Maß an Dokumentation für das Projekt erreicht werden kann.

3.1.2 Prozessspezifische Anforderungen an die Applikation

Prozess: Initialer Start

1. Kategorie: Effizienter Umgang mit Story Cards

[R9] Schnelles und einfaches Erstellen von Story Cards in großer Menge

Für die agile Arbeit mit User Stories ist es notwendig, über eine größere Menge an Story Cards zu verfügen. Das Erstellen dieser Startmenge von Story Cards erfolgt in der Phase des initialen Starts. Dabei werden zuerst nur Ideen gesammelt, ohne jede Story Card im Detail bereits zu besprechen und mit den entsprechenden Daten zu füllen. Die schnelle und einfache Produktion von Story Cards ist einer der Vorteile von User Stories und muss daher auch für eine gelungene Unterstützung des User Story Managements durch die Applikation mit realisiert werden (siehe Abschnitt 2.1.6, Vorteile von User Stories).
Dies lässt sich sehr gut in die Anwendung einbauen, indem fast alle Datenfelder als optionale Eingabebereiche vorgesehen werden und nur für das Titel-Feld, eine Eingabe erwartet wird. Dieses Feld dient dazu die Story Card später identifizieren und ausarbeiten zu können. Zusätzlich muss der Prozess des Erstellens einer Karte einfach durchzuführen sein.

2. Kategorie: Guten Prozess vorgeben / ermöglichen

[R10] Eine Pinnwand pro Projekt mit projektrelevanten Daten anlegen können

Für jedes Projekt müssen bestimmte Informationen bereitgestellt werden, um die agile Arbeit mit User Stories zu ermöglichen. So ist die Festlegung der Dauer einer Iteration und des Velocity-Wertes nötig, um eine Iteration effektive planen zu können [1].

Prozess: Planung

1. Kategorie: Effizienter Umgang mit Story Cards

[R11] Beweglichkeit der Story Cards durch Verschieben zwischen Backlog- und Iterationsansicht ermöglichen

Von der Arbeit mit echten Karteikarten ist man es gewohnt eine Karteikarte leicht mit der Hand von einem Bereich auf der Pinnwand in einen anderen Bereich verschieben zu können.
Diese Handhabung von Karteikarten lässt sich sehr gut auf einem Tablet per Drag & Drop Funktion umsetzen, was zum einen die Besonderheit der Steuerung eines Tablets verwendet und zum anderen die Einfachheit des Planens einer Iteration gut in die Applikation integriert [7].
Insgesamt wird damit eine passende Unterstützung für die Planung einer Iteration durch eine intuitive Bedienung geboten, da die Aktivität des Planens durch eine vergleichbare Handlung, wie mit echten Karteikarten, ermöglicht wird.

2. Kategorie: Guten Prozess vorgeben / ermöglichen

[R12] Passende Unterstützung bei der Planung einer Iteration

Die Planung einer Iteration ist der erste Prozess in dem Regelkreis der agilen Softwareentwicklung auf dem der gesamte weitere Verlauf des Zyklus basiert. Aus diesem Grund ist es wichtig die Iterationsplanung geeignet durch die Applikation zu fördern.
Für eine passende Unterstützung in der Planung sollen für den Nutzer gewisse Aufgaben erleichtert beziehungsweise ihm sogar abgenommen werden [7].
Bei der Auswahl von Story Cards müssen zwei Dinge beachtet werden. Erst einmal ist es wichtig, dass eine Karte über alle relevanten Informationen für die Planung und Bearbeitung verfügt. Dafür muss der Zustand der Karte für den Nutzer ersichtlich sein, welcher leicht durch das System festgestellt und visuell angezeigt werden kann. Desweitern ist es wichtig, dass bei der Zusammenstellung der Karten der Velocity-Wert des Teams nicht überschritten wird. Auch diese Aufgabe kann man der Applikation zuweisen, wodurch die Arbeit des Nutzers erneut vereinfacht wird.
Für den Beginn einer Iteration sind auch noch das Start- und Enddatum wichtig. Da ein Projekt über die Information der Dauer einer Iteration verfügt, kann auch hier dem Nutzer eine Unterstützung angeboten werden, indem er nur den Start festlegt und das System von selbst das Ende vorausberechnet.

[R13] Sortierungen der Story Cards ermöglichen (Priorität, Aufwand, Titel)

In der initialen Phase und im weiteren Verlauf eines Projektes entstehen einige Story Cards. Da sich diese mit der Zeit im Backlog ansammeln, ist es schwer, eine gute Übersicht zu behalten und bestimmte Story Cards schnell und leicht wiederzufinden.
Aus diesem Grund ist es wichtig, dass die Applikation dem Nutzer als Unterstützung die Möglichkeit bietet, die Story Cards nach einem ausgewählten Kriterium zu sortieren, um einen besseren Überblick zu erhalten [4].

Prozess: Durchführung

Hier sind keine Anforderungen zu nennen, die nicht bereits zuvor in den allgemeinen Anforderungen mit erhoben wurden.

Prozess: Überprüfung

1. Kategorie: Effizienter Umgang mit Story Cards

[R14] Testaspekte bezüglich Erfolg / Misserfolg / ungeprüft markieren können

Eine Story Card verfügt auf der Rückseite über Tests beziehungsweise Erwartungen an die entsprechende Funktionalität. Ist ein Entwickler fertig mit der Implementierung und will diese Testkriterien überprüfen, so ist es gut, wenn er die Möglichkeit hat die Testaspekte bezüglich Erfolg, Misserfolg und ungeprüft markieren zu können.
Diese Funktion bietet dem Entwickler eine gewisse Art der zusätzlichen Dokumentation (siehe Abschnitt 2.1.6, Nachteile von User Stories), da er festhalten kann, wie das Testergebnis ausgesehen hat und bezüglich welchem Aspekt er gegebenenfalls nacharbeiten muss.

[R15] Story Cards als erledigt markieren können

Am Ende einer erfolgreichen Überprüfung einer Story Card kann diese als erledigt angesehen werden. Um den Zustand einer Story Card für den Nutzer visuell erkennbar zu machen, ist es wichtig, dass die Story Card bewusst von dem Nutzer in den Zustand erledigt versetzt wird. Zusätzlich verschafft dies dem Nutzer das positive Gefühl etwas fertig bekommen zu haben.
Es ist aber zu bedenken, dass der Nutzer vielleicht versehentlich oder zu früh eine Story Card als erledigt betrachtet und diese Aktion zurücknehmen will, wodurch die Notwendigkeit des Rückgängigmachens geben ist.

2. Kategorie: Guten Prozess vorgeben / ermöglichen

[R16] Logs für erledigte Karten und Iterationen verwalten

Für eine automatische Generierung von Auswertungsdiagrammen durch die Applikation ist es notwendig, dass erledigte Karten und Iterationen jeweils durch ein Log gespeichert werden.
Diese Logs können aber zusätzlich noch als Dokumentation verwendet werden, indem man dem Nutzer ermöglicht, die Daten abzurufen und einzusehen. Dadurch verfügt der Nutzer über die Möglichkeit einen genauen Rückblick auf die erledigte Arbeit in einem Projekt werfen zu können und wird damit in der Überprüfungsphase unterstützt (siehe Abschnitt 2.1.6, Nachteile von User Stories).

Prozess: Anpassung

1. Kategorie: Effizienter Umgang mit Story Cards

[R17] Velocitywert des Teams muss anpassbar sein

Zu Beginn eines Projektes kann nie genau gesagt werden, wie viel Aufwand ein Team in einer Iteration schaffen kann. Die Angabe des Velocity-Wertes beruht meist auf einer Schätzung, die auf Erfahrungen und Vermutungen basiert, wodurch der Wert ungenau ist und damit nicht korrekt sein muss.
Durch die Analyse der einzelnen Iterationen erhält ein Team immer mehr Informationen und kann anhand dieser Daten eine genauere Schätzung für den Velocity-Wert abgeben. Aus diesem Grund muss dieser Wert in den Projektdaten unbedingt anpassbar sein, da so eine bessere Iterationsplanung erfolgen kann [7].
Insgesamt ergibt sich damit ein effizienteres Arbeiten in einem Projekt und der große Vorteil der Flexibilität in der agilen Softwareentwicklung wird durch die Applikation ebenfalls realisiert (siehe Abschnitt 2.1.6, Vorteile von User Stories).

2. Kategorie: Guten Prozess vorgeben / ermöglichen

[R18] Jede Story Card ist immer in einer der beiden Ansicht (Backlog oder Iteration), außer die Story Card wurde explizit gelöscht

Eine erstellte Story Card muss immer aufzufinden sein. Da es zwei Hauptbereiche auf einer Pinnwand, das Backlog und die Iteration, gibt, muss sich eine Story Card auch immer in genau einem dieser Bereiche befinden. Die einzige Möglichkeit um eine Story Card verschwinden zu lassen, ist, wenn sie explizit vom Nutzer gelöscht wird.

Die Gewährleistung des Erhalts der Story Cards ist wichtig, um die Dokumentation durch die Applikation zu unterstützen (siehe Abschnitt 2.1.6, Nachteile von User Stories). Denn der Nutzer ist dann in der Lage, Story Cards von einer bestimmten Iteration zu einem späteren Zeitpunkt noch einmal im Detail betrachten zu können.

Nachdem die ermittelten Anforderungen erhoben sind, ist es aufgrund der Menge an Anforderungen nötig eine Auswahl zu treffen. Nur die nachfolgenden ausgewählten Anforderungen werden im weiteren Verlauf der Arbeit genauer betrachtet, da eine vollständige Implementierung aller Anforderungen den Rahmen einer Bachelorarbeit überschreiten würde.

3.2 Auswahl der zu implementierenden Anforderungen

Für die hier aufgeführten Anforderungen werden in dem 4. Kapitel die einzelnen Konzepte erläutert, die für die Umsetzung dieser Anforderungen in die Applikation entwickelt wurden.
Die Auswahl der Anforderungen erfolgt dabei, mit der Betrachtung der gestellten Problematik Eigenschaften von User Stories und Story Card passend in das elektronische System zu überführen (siehe Abschnitt 1.2). Damit soll das Ziel einer guten Unterstützung des User Story Managements erreicht und Aktivitäten, wie mit realen Karteikarten, ermöglicht werden.
Da die Anwendung speziell für Tablets entwickelt wird, werden als weitere Auswahlkriterien die Handhabung und die Eigenschaften von einem Tablet mit berücksichtig.
Das bedeutet für die ausgewählten Anforderungen, dass sie gerade durch die Realisierung auf einem Tablet eine adäquate Überführung von Eigenschaften und somit auch Aktivitäten mit echten Karteikarten in das elektronische System ermöglichen.
Im Folgenden werden die ausgewählten Anforderungen aufgelistet und es wird der Grund für ihre Auswahl jeweils kurz, in Bezug auf die zuvor genannten Kriterien, erläutert.

Ausgewählte Anforderungen:

[R1] Arbeiten wie mit echten Karteikarten (Zeichnen, Erstellen, Editieren, Löschen)

Das Arbeiten mit Story Cards ist zentraler Bestandteil dieser Arbeit und daher muss diese Anforderung zu denen gehören, auf die genauer eingegangen wird.
Der hierbei interessanteste Aspekt ist die Möglichkeit des Zeichnens auf Story Cards. Dies lässt sich durch die Steuerung eines Tablets mit den Fingern geeignet umsetzen, sodass es sehr nahe an die Arbeit mit echten Karteikarten herankommt. Dies hat zur Folge, dass eine der besonderen Aktivitäten, die mit echten Karteikarten möglich sind, auf eine vergleichbare Art und Weise in der Applikation integriert wird.

[R2] Gute allgemeine Übersicht bezüglich wichtiger Daten mit leichtem Zugriff auf weitere Details bezüglich Story Cards und Pinnwänden

Vergleicht man die Größe eines Tablets von 7 – 10 Zoll mit der Größe einer Pinnwand von etwa 90 x 120 cm, so wird ersichtlich, dass der Informationsgehalt auf einer sehr viel kleineren Fläche für den Nutzer dargestellt werden muss. Aus diesem Grund ist es für die Anwendung wichtig über ein geeignetes Konzept für eine passende Übersicht zu verfügen. Dadurch ist man in der Lage, dem Nutzer leicht die relevanten Informationen zu liefern und ihm aber auch stets die Option bieten zu können auf weitere Details zugreifen zu können.

[R3] Lokale und globale Projekte in der Applikation anlegen können

Um die Verwendbarkeit der Applikation zu erhöhen, soll die Anwendung nicht von der Existenz eines Webservice abhängig sein. Daraus resultiert, dass es sich anbietet, Projekte auch nur lokal auf einem Tablet verwalten zu können.
Dies bietet einem Nutzer die Möglichkeit die Unterstützung durch die Applikation in allen seinen agilen Projekten, die mit User Stories arbeiten, nutzen zu können.

[R6] Eine Backlog- und eine Iterationsansicht

Das Backlog und die Iteration bilden die zwei wichtigsten Bereiche auf einer realen Pinnwand und machen dadurch den Kern der Anwendung aus.
Wie zuvor aber bereits erwähnt wurde, verfügt ein Tablet über weniger Platz als eine reale Pinnwand. Um dem Nutzer aber dennoch eine gute Übersicht zu ermöglichen, ist es wichtig, dass diese beiden zentralen Ansichten geeignet realisiert werden.

[R9] Schnelles und einfaches Erstellen von Story Cards in großer Menge

In der agilen Softwareentwicklung mit User Stories ist eine bekannte Methode für das Sammeln von Story Cards der Story Workshop. Für eine gute Unterstützung bietet es sich an diese Aktivität auch mit in die Software zu integrieren, da sie sich einfach durch die Wahl von einem Pflichtfeld und sonst nur optionalen Eingabefeldern umsetzen lässt. Diese Aktivität steigert den Nutzen der Applikation deutlich in Bezug auf das Ermöglichen von prozessumfassenden Aktivitäten der agilen Softwareentwicklung mit User Stories.

[R11] Beweglichkeit der Story Cards durch Verschieben zwischen Backlog- und Iterationsansicht ermöglichen

Echte Karteikarten verfügen über eine sehr hohe Beweglichkeit. Sie können leicht auf der Pinnwand verschoben werden und bieten damit die Möglichkeit der einfachen Planung einer Iteration.
Dies ist wohl mit die wichtigste Eigenschaft über die echten Karteikarten verfügen. Aus diesem Grund muss eine Applikation, die das Ziel hat das Management von User Story zu unterstützen, diese Eigenschaft für die elektronischen Story Cards adäquat realisieren. Dadurch kann einem Nutzer das Gefühl des vertrauten Umgangs mit echten Karteikarten vermittelt werden, obwohl er mit der Applikation arbeitet.

Es ist anzumerken, dass noch weitere Anforderungen neben den zuvor Ausgewählten umgesetzt werden. Da der Fokus aber auf den ausgewählten Anforderungen liegt, welche eine höhere Relevanz für diese Arbeit besitzen, werden die weiteren umgesetzten Anforderungen nicht ausführlicher erörtert, sondern nur kurz aufgelistet.

Zusätzlich realisierte Anforderungen:

[R5] Form und Aufbau der elektronischen Story Cards soll bezüglich Inhalt und Format vergleichbar mit echten Karteikarten sein

[R7] Einsehbare Diagramme auf Basis von erstellten Logs

[R10] Eine Pinnwand pro Projekt mit projektrelevanten Daten anlegen können

[R12] Passende Unterstützung bei der Planung einer Iteration

[R13] Sortierungen der Story Cards ermöglichen (Priorität, Aufwand, Titel)

[R15] Story Cards als erledigt markieren können

[R16] Logs für erledigte Karten und Iterationen verwalten

[R17] Velocitywert des Teams muss anpassbar sein

[R18] Jede Story Card ist immer in einer der beiden Ansicht (Backlog oder Iteration), außer die Story Card wurde explizit gelöscht

Die einzelnen Konzepte für die Realisierung der obigen sechs ausgewählten Anforderungen werden im folgenden Kapitel 4 im Detail ausgeführt und erläutert.

4 Konzepte der Applikation

Im vierten Kapitel geht es hauptsächlich um die entwickelten Konzepte für die Android Applikation, die im Folgenden **Mobile User Story Tool** (MUST) genannt wird. Diese werden erst genauer erörtert, bevor die Vision der zukünftigen Verwendung von MUST durch eine Szenario-Beschreibung skizziert wird.
Zusätzlich findet im Abschnitt 4.3 eine Abgrenzung von MUST gegenüber anderen User Story Management Systemen statt. Dadurch sollen die Besonderheiten der in MUST realisierten Konzepte aufgezeigt und verdeutlicht werden.
Im letzen Teil dieses Kapitels werden kurz die technischen Aspekte und Entscheidungen, die sich während der Entwicklung der Applikation ergeben haben, zusammengefasst.

4.1 Umsetzung der Anforderungen

Dieser Abschnitt befasst sich mit der Umsetzung der Konzepte für die in Abschnitt 3.2 ausgewählten Anforderungen. Zuerst wird auf das jeweilige Konzept in einem eigenen Abschnitt genauer eingegangen. Dazu werden zusätzlich Screenshots von MUST verwendet, um die Funktionsweise der Konzepte genauer zu erklären. Weiterhin wird aufgezeigt, welche der Anforderungen durch das entsprechende Konzept realisiert werden und was die Ideen und die Gründe für das jeweilige Konzept gewesen sind.

4.1.1 Übersicht

Aufgrund des eingeschränkteren Platzes bei einem Tablet, im Vergleich zu einer echten Pinnwand, ist es wichtig, dass dem Nutzer eine gute Übersicht geboten wird, damit er effizient arbeiten kann und sich leicht zurechtfindet.

Dafür gilt es zu verstehen, welche Ansichten in MUST überhaupt enthalten sind. Insgesamt gibt es acht verschiedene Ansichten in der MUST-Anwendung, die im Anhang A) genauer betrachtet werden können.

Beim Start der Anwendung befindet man sich in der *Projektansicht* (siehe Anhang A), Abb. 28). Diese bietet einen Überblick über alle in der MUST-Applikation enthaltenen Projekte und zeigt für ein ausgewähltes Projekt alle Informationen detailliert an. Von dieser Ansicht aus kann man zu der Pinnwand eines bestimmten Projektes zu gelangen.

Das Konzept der Pinnwand teilt diese erst mal in zwei Bereiche ein. Es gibt das Backlog und die Iteration. Für diese beiden Bereiche wurden drei zentrale Ansichten entwickelt. So verfügt die Applikation über eine *vollständige Backlogansicht* (siehe Anhang A), Abb. 29), einen geteilten Bildschirm, der Backlog und Iteration gemeinsam anzeigt und im Folgenenden als *Splitscreenansicht* bezeichnet wird (siehe Anhang A), Abb. 30), und die *vollständige Iterationsansicht* (siehe Anhang A), Abb. 31).

Der Gedanke hinter den drei Ansichten ist es dem Nutzer den Eindruck zu vermitteln, dass er die ganze Zeit über auf der gleichen Pinnwand arbeitet, welche er aber aus verschiedenen Blickwinkeln betrachten kann (siehe Abb. 6).

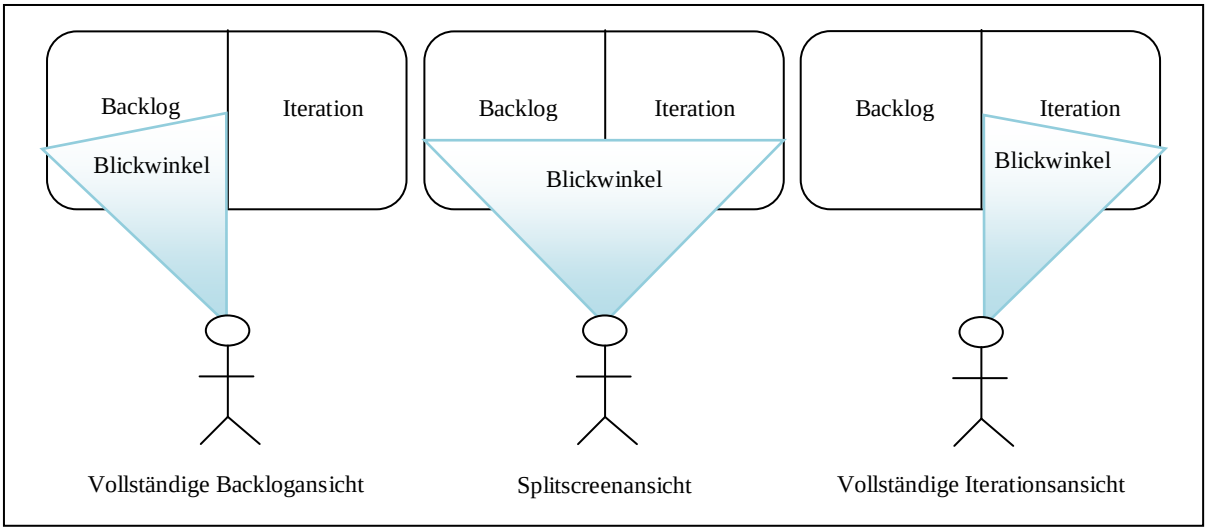


Abb. 6 - Pinnwandansichten und Blickwinkelidee

Durch die Möglichkeit die Ansicht zu verschieben, soll die Bewegung des Kopfes vom Nutzer nachgeahmt werden. Wenn dieser sich zentral vor der Pinnwand befindet, die in Backlog und Iteration aufgeteilt ist, kann er beide Bereiche zur gleichen Zeit sehen (siehe Abb. 6, Splitscreenansicht). Bewegt er nun den Kopf entweder mehr nach links (siehe Abb. 6, Vollständige Backlogansicht) oder mehr nach rechts (siehe Abb. 6, Vollständige Iterationsansicht), so fokussiert er nur einen bestimmten Bereich. Auf eine ähnlich Art und Weise kann man die Ansicht der elektronischen Pinnwand in MUST verändern. Die Besonderheit hierbei ist jedoch, dass nicht der Nutzer seinen Blick verändert, sondern dass die Ansicht der Pinnwand auf dem Tablet und damit der Blickwinkel verschoben werden kann.

Die beiden vollständigen Ansichten sind für ein effizientes Arbeiten in dem jeweiligen Bereich gedacht. Der auf dem Tablet zur Verfügung stehende Platz soll vollständig für eine gute erste Übersicht über die Story Cards im Backlog beziehungsweise in der Iteration genutzt werden. Der Splitscreen hingegen ist besonders für die schnelle Übersicht über Backlog und Iteration zur gleichen Zeit gedacht. Zwar müssen sich die beiden Bereiche den Bildschirm teilen, was zur Folge hat, dass die Übersicht in dem jeweiligen Bereich eingeschränkt wird, aber dafür gewinnt man eine Ansicht, die einen Vergleich der beiden Bereiche ermöglicht.

Diese drei Ansichten enthalten dabei Miniatur Story Cards, welche Teil eines weiteren Konzeptes sind.
Es gibt *zwei verschiedene Varianten einer Story Card*. Zum einen eine *Miniatur Story Card*, welche nur die drei wichtigsten Informationen (Priorität, Aufwand, Titel) enthält, um eine Iteration planen zu können. Zum anderen eine *detaillierte Story Card*, die alle Informationen der Story Card anzeigt, wobei es eine Vorder- und eine Rückseite, wie bei echten Karteikarten, gibt.
Genau für diese detaillierte Story Card gibt es noch zwei weitere Ansichten in der MUST-Applikation. Diese werden ab sofort *Backlogdetailansicht* (siehe Anhang A), Abb. 32) und *Iterationsdetailansicht* (siehe Anhang A), Abb. 33) genannt. Zu diesen gelangt man jeweils, wenn eine der Miniatur Story per Klick selektiert wird. Die beiden Detailansichten dienen zum genauen betrachten einer bestimmten Story Card mit allen Informationen. Weiterhin enthält die entsprechende Detailansicht eine Liste aller Miniatur Story Cards, die im Backlog beziehungsweise in der Iteration enthalten sind. Dadurch ist es dem Nutzer möglich, leicht die im Detail betrachtete Story Card zu wechseln.
Die letzten beiden Ansichten sind die *Zeichenansicht* (siehe Anhang A), Abb. 34) und die *Erstell- / Editieransicht* (siehe Anhang A), Abb. 35). Die Zeichenansicht kann für eine

bestimmte Story Card aufgerufen werden. Sie zeigt die Vorderseite der Story Cards mit allen eingetragenen Informationen und ermöglicht es dem Nutzer eine Zeichnung zu erstellen.
Die Erstell- / Editieransicht wird zweifach verwendet. Zum einen dient sie für das Erstellen von Story Cards und zum anderen für das nachträgliche Editieren einer bereits vorhandenen Story Card. Dafür werden dem Nutzer die Vorder- und Rückseite der Story Card gleichzeitig angezeigt, um hier das Drehen der Story Card zu ersparen, was sonst störend ist, wenn er öfters die Ansicht der Story Card verändern müsste, um einen bestimmten Teil zu editieren.

Durch das Konzept der verschiedenen Übersichten werden die Anforderungen **[R2] Gute allgemeine Übersicht bezüglich wichtiger Daten mit leichtem Zugriff auf weitere Details bezüglich Story Cards und Pinnwänden** und **[R6] Eine Backlog- und eine Iterationsansicht** realisiert. Das Hauptziel dabei ist stets gewesen, dass dem Nutzer eine gute Übersicht geboten wird. Zusätzlich soll beim Nutzer der Eindruck erweckt werden, dass er für jedes Projekt eine einzige digitale Pinnwand hat und diese nur aus verschiedenen Perspektiven betrachtet. Weiterhin wurde darauf geachtet, dass der eingeschränkte Platz, den ein Tablet zur Verfügung stellt, optimal ausgenutzt wird, damit die Arbeit des Nutzers in der jeweiligen Ansicht effizient unterstützt wird. Dazu ist noch einmal anzumerken, dass die drei Pinnwandansichten *vollständige Backlogansicht*, *Splitscreenansicht* und *vollständige Iterationsansicht* vor allem für ein gutes Planen einer Iteration und dem Erstellen von Story Cards gedacht sind. Die beiden Detailansichten, *Backlogdetailansicht* und *Iterationsdetailansicht*, sind wiederum speziell für die Arbeit mit den Story Cards in der Implementierungsphase der agilen Softwareentwicklung entwickelt worden.
In der Planungsphase kommt es dem Nutzer besonders auf einen guten Überblick über die einzelnen Story Cards und deren Informationen bezüglich der Iterationsplanung an. Im Gegensatz dazu arbeitet der Nutzer in der Implementierung für eine längere Zeitspanne stets mit nur einer bestimmten Story Card, weshalb die Detailansicht auch nur genau für eine Story Card aufgerufen werden kann.

4.1.2 Navigation

Durch die Idee der Betrachtung einer Pinnwand aus verschiedenen Blickwinkeln (siehe Abb. 6) und der Möglichkeit Story Cards im Detail zu betrachten, ist es wichtig, dass der Nutzer leicht zwischen den Ansichten wechseln kann. Aus diesem Grund ist eine gute Navigation zwingend erforderlich, da so ein guter Überblick gewährleistet werden kann. Es werden insgesamt zwei Techniken verwendet, um die Navigation umzusetzen.

Zuerst einmal wird die klassische Methode des Selektierens von Objekten per *Klick* genutzt. Diese wird in Bezug auf die Pinnwandansichten vor allem für das Betrachten von Story Cards in den Detailansichten verwendet und um für eine detaillierte Story Card den Editiermodus zu starten.
Weiterhin ist die Selektion per *Klick* auch bei den zwei Tabs *Backlog* und *Iteration* möglich, die in allen Pinnwandansichten enthalten sind. Durch die Auswahl eines Tabs per Klick kann der Nutzer sofort in die jeweilige vollständige Pinnwandansicht gelangen.
Hierbei ist aber anzumerken, dass diese Navigation in drei besonderen Ansichten momentan nicht möglich ist. Bei zwei dieser Ansichten handelt es sich um die *Zeichenansicht* und die *Erstell- / Editieransicht* für eine ausgewählte Story Card. Die Navigationsmöglichkeit ist in diesen beiden Ansichten nicht eingebaut, da in beiden Fällen Daten der Story Card durch den Nutzer verändert werden können. Es wurde sich dafür entschieden, dass die Applikation nicht selbständig speichern soll und die Speicherung von Daten stattdessen nur bewusst durch den Nutzer zu erfolgen hat. Um dabei zu verhindern, dass Eingaben durch einen Wechsel der Ansicht über die Tabs verloren gehen, ist die Navigation in diesen Fällen nicht erwünscht.

Die dritte Ansicht, die nicht über diese Navigationsmöglichkeit verfügt, ist der *Splitscreen*. Hier ist die Selektion eines Tabs momentan für die Auswahl des aktiven Bereiches, der blau hervorgehoben wird, zuständig. Dies ist nötig, um in der Applikation anzugeben, in welchem Bereiche eine neue Story Card hinzugefügt werden soll. Diese Selektion sollte eigentlich durch das Klicken auf die Bereiche unterhalb der Tabs ermöglicht werden. Aufgrund der Verwendung von bestimmten Standard-Android-Elementen in diesen beiden Bereichen, erfolgt eine falsche Interpretation der Klicks. Zur Lösung dieses Problems müssen die verwendeten Standardelemente eigenständig neu definiert werden, sodass das gewünschte Verhalten erreicht wird. Durch die Komplexität der zugehörigen Klassen ist dies aber bisher nicht gelungen. Der Nutzen durch die Neu-Implementierung der Standard-Objekte steht in keinem Verhältnis zu der dazu benötigten Zeit, welche diese aufwändige Arbeit erfordern würde. Daher wird vorerst die zuvor erläuterte Selektion über die Tabs als momentane Lösung des Problems verwendet.

Die zweite Technik, die die Navigation zwischen den Ansichten realisiert, ist die *Gestensteuerung*. Das dafür umgesetzte Konzept basiert auf der Idee der verschiedenen Blickwinkel des Nutzer auf die Pinnwand, welche in Abschnitt 4.1.1 bereits erläutert wurde. Durch die Verwendung von Wischgesten ist der Nutzer in der Lage, den Blickwinkel zu verändern und so zwischen den Hauptansichten der Pinnwand, dem Backlog, dem Splitscreen und der Iteration, zu wechseln (siehe Abb. 6).

In der vollständigen Backlogansicht kann der Nutzer mittels einer horizontalen Wischgeste von rechts nach links, innerhalb des Listenbereiches der Miniatur Story Cards, in die Splitscreenansicht gelangen (siehe Abb. 7, von a) zu b)).

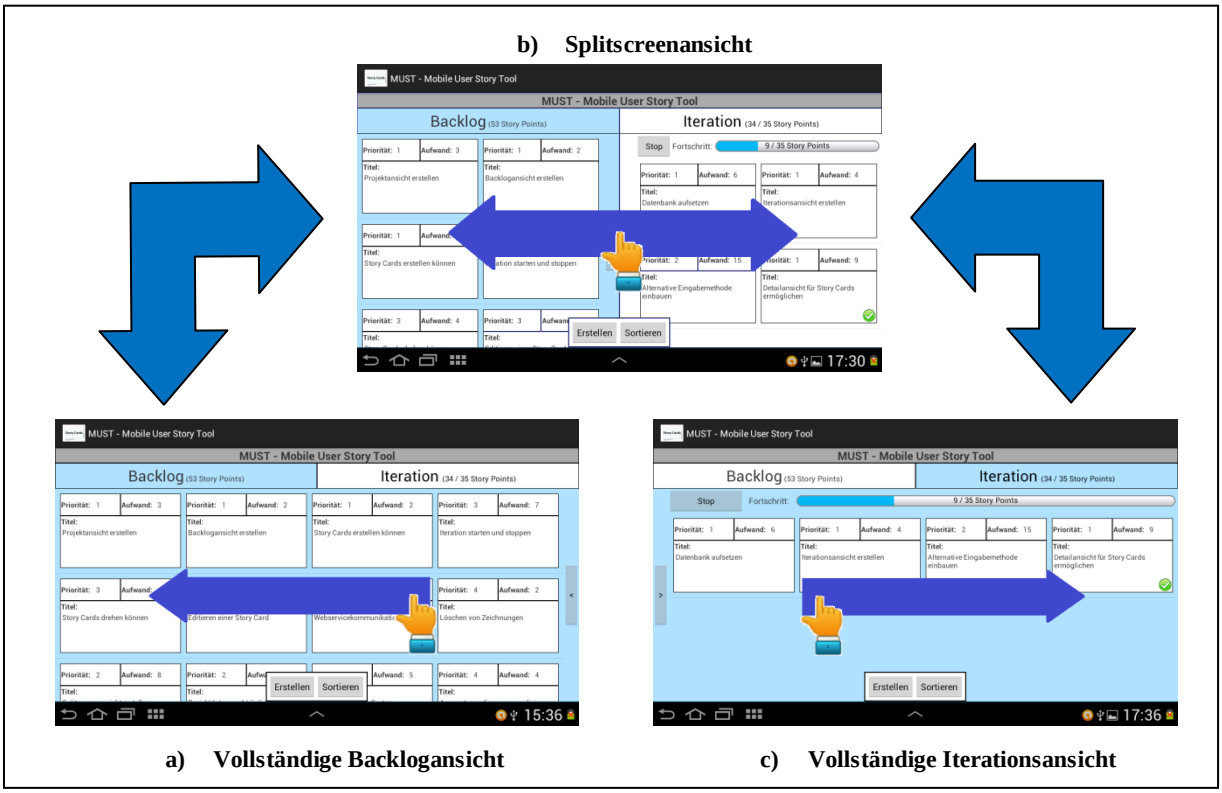


Abb. 7 - Navigation zwischen den Pinnwandansichten mittels Wischgesten

Aus der vollständigen Iterationsansicht besteht auf die gleiche Art die Möglichkeit in den Splitscreen zu wechseln. Nur muss in diesem Fall von links nach rechts gewischt werden (siehe Abb. 7, von c) zu b)).

Die beiden Wischgesten werden auch von der Splitscreenansicht verwendet, um den Wechsel in die vollständige Backlog- oder Iterationsansicht zu ermöglichen. Durch das Wischen von links nach rechts kommt man zur Backlogansicht und mittels der Wischgeste von rechts nach links erreicht man die Iterationsansicht (siehe Abb. 7, von b) zu a) bzw. c)).

Für einen genauen Überblick des Aufbaus der kompletten Navigationsstruktur sind im Anhang zwei Abbildung zu finden (siehe Anhang B)).

Im Wesentlichen wird durch das Konzept der Navigation die Anforderung **[R2] Gute allgemeine Übersicht bezüglich wichtiger Daten mit leichtem Zugriff auf weitere Details bezüglich Story Cards und Pinnwänden** unterstützt. Durch eine geeignete Navigationsstruktur kann sichergestellt werden, dass der Nutzer bei der Verwendung von MUST stets den Überblick behält, damit er schnell und einfach auf die für ihn relevanten Daten und Ansichten zugreifen kann.
Bei der Entwicklung der Navigation wurde eine grundlegende Hierarchie der Ansichten gewählt. Diese Hierarchie ist wie folgt zu interpretieren. Je tiefer sich der Nutzer in der Hierarchie der Ansichten befindet, desto mehr Details werden im angezeigt. Die entwickelte Hierarchie wird mit Hilfe der Abbildung 8 verdeutlicht.
Insgesamt gibt es zwei Hierarchieebenen, wobei die zweite Ebene zusätzlich in zwei Stufen unterteilt wird.

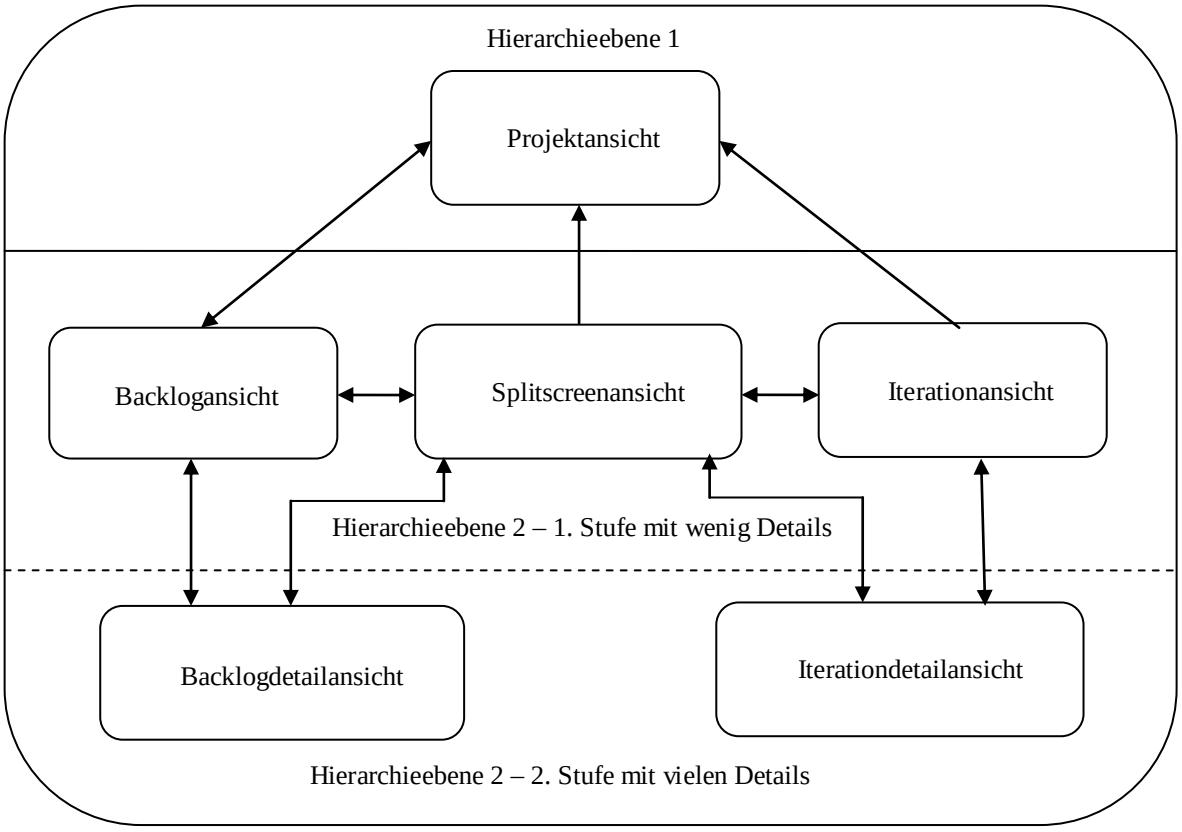


Abb. 8 - Navigationshierachie in MUST

Die erste Ebene enthält nur die Projektansicht von der aus die zweite Ebene, welche alle möglichen Pinnwandansichten enthält, erreicht werden kann. Für den Nutzer bedeutet dies, dass er sich aus der Menge aller Projekte eines aussucht, das er detaillierter betrachten will.
Alle Pinnwandansichten werden noch in zwei Stufen unterteilt. Die erste Stufe enthält die drei Basisansichten vollständiges Backlog, Splitscreen und vollständige Iteration. In diesen wird dem Nutzer ein genauerer Überblick über die gesamte Pinnwand mit allen Story Cards

ermöglicht. Von diesen drei Ansichten aus kann der Nutzer weitere Details der Story Cards des Projektes abrufen. Dies stellt eine weitere Verfeinerung der betrachten Details dar, wodurch der Nutzer zu den Pinnwandansichten der zweiten Stufe gelangt.

Die Verwendung des Zurück-Buttons des Tablets orientiert sich ebenfalls an dieser Hierarchie und ermöglicht das Zurückspringen aus einer Ansicht in die entsprechend vorherige Hierarchiestufe beziehungsweise Ebene.

4.1.3 Grundfunktionalitäten

Im Folgenden wird auf die Grundfunktionalitäten von MUST eingegangen. Diese bestehen aus dem *Erstellen*, *Editieren* und *Löschen* von Story Cards sowie der besonderen Funktion des *Zeichnens*.
Für ein flexibles Arbeiten mit der MUST-Applikation stehen die Grundfunktionalitäten in mehreren Ansichten zur Verfügung. Eine genaue Erklärung der einzelnen Grundfunktionen ist im Anhang C) anhand eines praktischen Beispiels zu finden. Im Folgenden wird erläutert welche Anforderungen durch das Konzept der Grundfunktionalitäten realisiert werden. Dabei wird auch darauf eingegangen wie und warum die Umsetzung auf die entsprechende Art und Weise erfolgte.

Die Grundfunktionen realisieren zuerst einmal die Anforderung **[R1] Arbeiten wie mit echten Karteikarten (Zeichnen, Erstellen, Editieren, Löschen)** und ermöglichen damit das generelle Arbeiten mit den Story Cards.
Die Funktionen *Erstellen* und *Löschen* wurden in dem zentrierten unteren Menü eingebaut, da das Layout der Ansichten von MUST einem Whiteboard nachempfunden ist. Bei einem Whiteboard befindet sich an der gleichen Position eine Ablage für Stifte und Schwamm. Da diese Objekte Analogien für das Erstellen und das Löschen sind, ist der Gedanke hinter der Positionierung der Funktionen, eine geeignete Assoziation bei dem Nutzer hervorzurufen, sodass er das Gefühl hat mit einem Whiteboard beziehungsweise einer Pinnwand zu arbeiten.

Die Funktion des *Editierens* ist enger mit einer bestimmten Story Card verbunden, sodass es passender erschien diese durch das Klicken auf eine Story Card zu realisieren, anstatt durch einen gesonderten Menüpunkt.

Auf die Funktion des *Zeichnens* soll besonders eingegangen werden. Dies hat zwei Gründe. Zum einen stellt das Zeichnen eine alternative Eingabemethode auf einer Story Card dar und zum anderen ist die Implementierung dieser Funktion eine besondere Herausforderung gewesen. Das Ziel für diese Funktionalität war es den Eindruck zu vermitteln, dass man direkt auf einer Story Card zeichnen kann. Dies konnte durch die Verwendung von einer transparenten Zeichenoberfläche und einer komplexeren Verwaltung innerhalb der Klassen, die für das Zeichnen zuständig sind, realisiert werden.
Ein größeres Problem dabei war die standardmäßige Nicht-Transparenz der Zeichenoberfläche. Dadurch war die darunterliegende Ansicht, welche die Story Card mit ihrem Inhalt zeigt, zuerst nicht zu sehen. Wenn man dann die Zeichenoberfläche selbst transparent eingestellt hat, wurden auch die gezeichneten Linien transparent, wodurch die erstellte Zeichnung war nicht mehr angezeigt wurde. Am Ende gelang es aber dennoch eine transparente Zeichenoberfläche mit sichtbaren Zeichnungen zu ermöglichen.
Die Lösung ergibt sich durch eine geschickte jeweilige Formatierung des Zeichenbereichs sowie der erstellten Zeichnung.
Um die zuvor ausgeführten Besonderheiten der Funktionalität des Zeichnens hervorzuheben, ist die Funktion als ein eigenständiger Menüpunkt umgesetzt worden.

Weiterhin wird durch die Grundfunktion des Erstellens und unter Zuhilfenahme des Titels einer Story Card als einziges Pflichtfeld, eine weitere Anforderung realisiert. Dabei handelt es sich um die Anforderung **[R9] Schnelles und einfaches Erstellen von Story Cards in großer Menge**. Indem nur der Titel eingegeben werden muss, wird zuerst einmal sichergestellt, dass die Story Card später wieder identifiziert werden kann. Weiterhin wird die Dauer für das Erstellen einer Story Card so kurz wie möglich gehalten, da nur ein einziges Feld auszufüllen ist. Damit ergibt sich insgesamt eine kurze Erstelldauer mit der Gewährleistung, den Sinn der Story Card wieder ermitteln zu können. Dies hat zur Folge, dass auf einfache Art und Weise sehr schnell Story Cards in großer Menge produziert werden können.

4.1.4 Gestensteuerung

Aufgrund der Entwicklung einer Android-Applikation für Tablets hat es sich angeboten auch die Verwendung von Gesten mit zu überdenken. Aus den Überlegungen entwickelte sich ein Konzept, das die Verwendung von bestimmten Gesten vorsieht, um gewisse Funktionen in MUST zu realisieren. Die Verwendung von Gesten ermöglicht eine Minimierung der benötigten Buttons in den Menüs, was die Menüstruktur vereinfacht.

Insgesamt nutzt die MUST Anwendung für vier Funktionen die Eingabe von Gesten. Neben den bereits in Abschnitt 4.1.2 vorgestellten *Gesten für die Navigation*, werden weiterhin Gesten für das *Drehen* und *Erledigen* einer detaillierten Story Card, in einer der Detailansichten, verwendet.
Zusätzlich gibt es noch als vierte Funktion, das *Drag & Drop*, die für das Verschieben der Story Cards auch eine Geste verwendet. Da es sich bei dem Drag & Drop aber um eine sehr spezielle Funktion mit großer Bedeutung für die MUST-Applikation handelt, wird auf diese im Abschnitt 4.1.5 gesondert eingegangen.
Welche Gesten, neben denen für die Navigation, noch zur Verfügung stehen, wird anhand der nachfolgenden Grafiken erläutert.

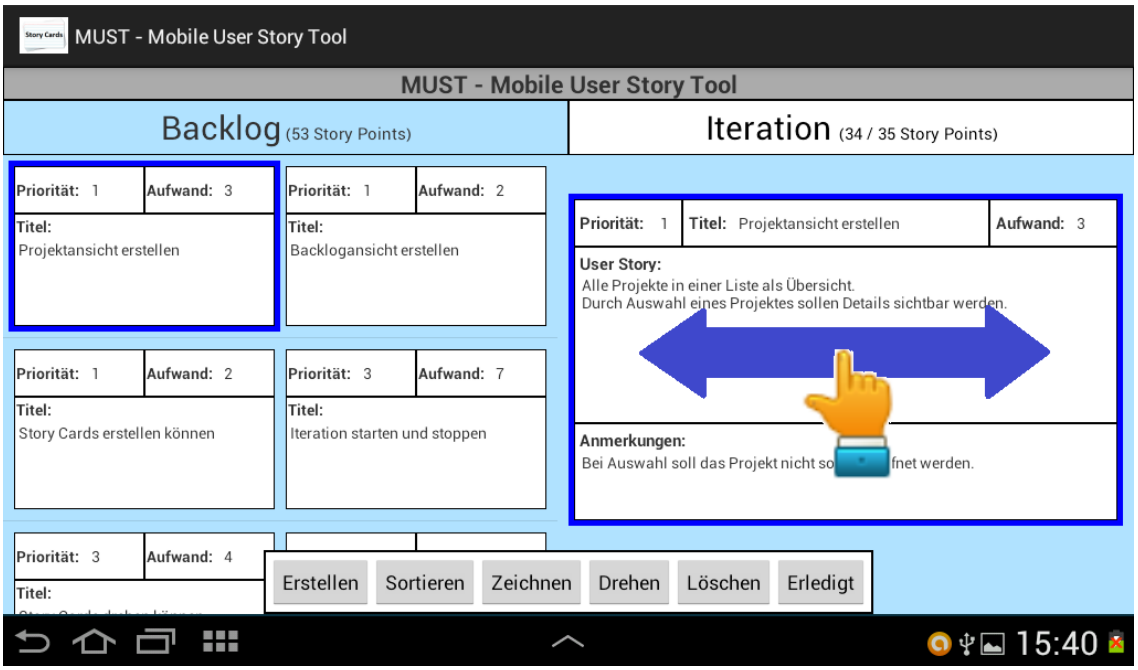


Abb. 9 – Wischgeste zum Drehen der detaillierten Story Card

Als erstes wird die Geste für das *Drehen* einer detaillierten Story Card betrachtet. Dazu kann man auf der detaillierten Story Card horizontal entweder von links nach rechts eine

Wischgeste durchführen oder von rechts nach links (siehe Abb. 9). Durch beide Richtungen wird das Drehen der Story Card ausgelöst, welches durch eine entsprechende Animation dargestellt wird. Am Ende der Drehung wird die entsprechend andere Seite der Story Card angezeigt. Diese Gesten funktionieren auf die gleiche Weise auf beiden Seiten einer detaillierten Story Card. In der gezeigten Grafik ist auf der Vorderseite die Wischgeste durchgeführt worden, sodass als Resultat die Rückseite der Story Card angezeigt wird.
Der Grund für die Wahl dieser Gesten liegt erneut in dem Ziel, dem Nutzer den Eindruck zu vermitteln, dass er mit echten Karteikarten arbeiten würde. Diese würde man auch an einer Seite anfassen und umdrehen. Die Gesten sollen einen ähnlichen Eindruck erwecken. Zwar kann der Nutzer die Story Card nicht in die Hand nehmen, aber es muss für die Drehung stets ein Kontakt zwischen dem Finger des Nutzers und der Story Card hergestellt werden. Somit hat der Nutzer die Story Card zumindest zu einem gewissen Maß in der Hand.

Die zweite Anwendung von Gesten erfolgt ebenfalls bei den detaillierten Story Cards. Eine Story Card kann in genau einem von zwei möglichen Zuständen sein. Entweder ist die Story Card *unerledigt* oder *erledigt*, wobei sie standardmäßig zuerst einmal als unerledigt betrachtet wird. Um eine Story Card zu erledigen, führt man auf der detaillierten Story Card die Wischgeste in Form von einem Haken durch (siehe Abb. 10).

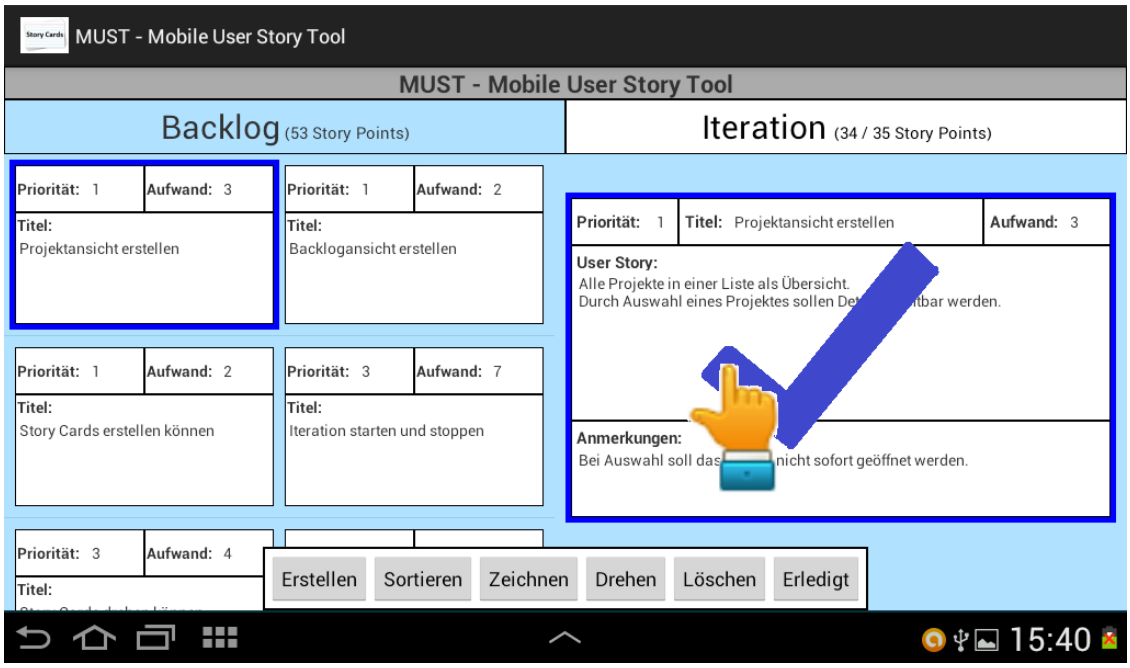


Abb. 10 - Wischgeste zum Erledigen der detaillierten Story Card

Als Ergebnis wird die Story Card auf erledigt gesetzt, was durch das Symbol eines Hakens in einem grünen Kreis in der unteren rechten Ecke sowohl auf der Miniatur Story Card als auch auf der Vorderseite der detaillierten Story Card visuell verdeutlicht wird (siehe Abb. 11). Zusätzlich wird die erledigte Story Card am Ende der Liste der Miniatur Story Cards passend einsortiert, da unerledigte Story Cards wichtiger sind, als bereits erledigte und diese deshalb zuerst zu sehen sein sollen.

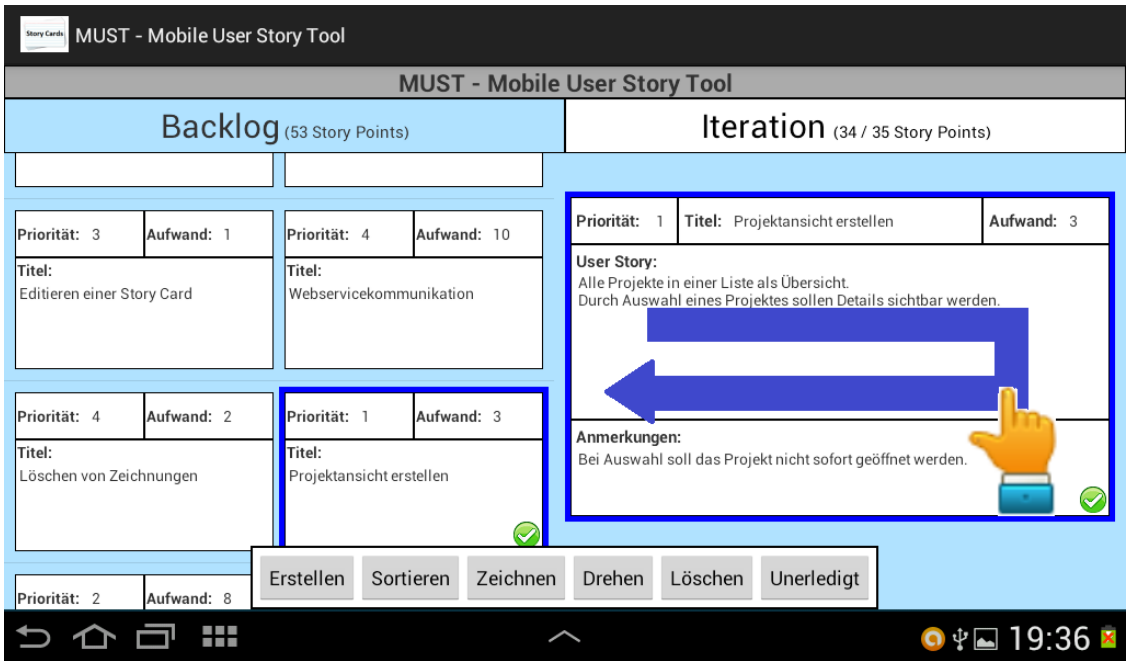


Abb. 11 - Wischgeste, um die detaillierte Story Card auf unerledigt zu setzen

Für den unerledigten Zustand muss auf der detaillierten Story Card eine horizontale Wischgeste von links nach rechts und wieder zurück ausgeführt werden (siehe Abb. 11). Als Reaktion auf die Geste verschwindet das Symbol mit dem Haken auf dem grünen Kreis von den beiden Darstellungen der Story Card. Weiterhin wird die Miniatur Story Card, entsprechend der aktuellen Sortierung, in die Liste wieder passend einsortiert.

Die gewählte Geste entspricht dabei dem Durchstreichen und soll vom Nutzer mental mit dem Negieren des Zustandes der Aufgabe verbunden werden. Es wurde länger überlegt, ob es nicht eine noch geeignetere Geste für den unerledigten Zustand gibt. Aber leider konnte keine bessere Geste gefunden werden, die von der Applikation eindeutig erkannt werden kann. Die Problematik, die sich bei der jetzigen Art der Gestenerkennung aufzeigt, ist die, dass jede Geste in einer Bibliothek vordefiniert sein muss und mit dem einmaligen Aufsetzen des Fingers verbunden ist. Sobald der Finger das Display nicht mehr berührt wird Definition der Geste als abgeschlossen betrachtet und kann nicht weiter ausgebaut werden. Dadurch ist die Konstruktion komplexerer Gesten mit der verwendeten Standarderkennung von Android nach den bisherigen Erkenntnissen nicht möglich.

Wie zu Anfang dieses Abschnitts erwähnt, wird die Gestenerkennung auch noch zum Wechsel zwischen den drei Pinnwandansichten Backlog, Splitscreen und Iteration verwendet. Auf die genau Funktionsweise dieser Gesten ist bereits im Abschnitt 4.1.2 eingegangen worden, da sie Teil der *Navigation* in der MUST-Applikation sind.

Mit der Gestensteuerung werden unter anderem drei der in Abschnitt 3.2 ausgewählten Anforderungen teilweise umgesetzt beziehungsweise unterstützt.
Zuerst einmal wird die Anforderung **[R6] Eine Backlog- und eine Iterationsansicht** realisiert, indem die Idee des Verschiebens der Ansichten der elektronischen Pinnwand darauf basiert, dass es genau diese beiden Bereiche auf einer Pinnwand gibt. Als Konsequenz dessen sollen dem Nutzer aber nicht nur schlicht verschiedene Ansichten der Pinnwand geboten werden, sondern es soll ihm stattdessen vielmehr der Eindruck vermitteltet werden, dass er nur auf einer einzigen Pinnwand arbeitet und dabei den Blickwinkel, aus dem er diese eine

Pinnwand betrachtet, verändern kann. Die vollständige Erläuterung dieser Idee ist bereits in Abschnitt 4.1.1 enthalten.

Weiterhin wird die Anforderung **[R2] Gute allgemeine Übersicht bezüglich wichtiger Daten mit leichtem Zugriff auf weitere Details bezüglich Story Cards und Pinnwänden** in der Applikation integriert. Durch die Konstruktion einer Vorder- und einer Rückseite für Story Cards, wird dem Nutzer der einfach Zugriff auf alle Daten, die eine Story Card enthalten kann, ermöglicht. Man erreicht so erneut das Ziel der adäquaten Überführung von Eigenschaften und Aktivitäten von und mit Karteikarten in das elektronische System.

Aus den vorherigen Ausführungen ist deutlich zu erkennen, dass noch eine dritte Anforderung durch die Gestensteuerung in MUST gefördert wird. Bei der Anforderung handelt es sich um **[R1] Arbeiten wie mit echten Karteikarten (Zeichnen, Erstellen, Editieren, Löschen)**. Eines der Ziele dieser Anforderung ist es, dass die Arbeit mit Story Cards auf eine möglichst realitätsnahe Weise erfolgt (siehe Abschnitt 3.1.1). Die Erfüllung dieses Ziel geht genau aus den beiden obigen Erläuterungen hervor.

4.1.5 Drag & Drop

Die hohe Beweglichkeit echter Karteikarten ist eine besondere Eigenschaft, über die auch die Story Cards in MUST verfügen sollen. Das einfache Handtieren mit Karteikarten ist insbesondere bei der Planung einer Iteration vom Vorteil, da die Karten leicht zwischen dem Backlog- und dem Iterationsbereich auf der Pinnwand verschoben werden können.
Das Überführen dieser Aktivität in die Applikation ist ein weiteres zentrales Ziel gewesen. Dazu ist das Konzept des *Drag & Drop* für die Miniatur Story Cards ausgearbeitet worden. Die *Drag & Drop* Funktion steht in allen Pinnwandansichten, die Miniatur Story Cards enthalten, zur Verfügung.

Der Splitscreen ist eine besondere Ansicht und bietet eine erweiterte Funktionalität bezüglich des Verschiebens von Story Cards im Vergleich zu den anderen Ansichten, die *Drag & Drop* ermöglichen. Aus diesem Grund wird für die nachfolgende Erklärung genau diese Ansicht genutzt.

Durch einen langen Klick auf die Miniatur Story Card wird der Drag Modus gestartet. Als Folge der Aktivierung wird die Story Card transparenter und folgt der Bewegung des Fingers über das Display des Tablets. Sobald sich die Story Card in einem gültigen Bereich befindet, in welchem sie platziert werden kann, wird dieser Bereich, die sogenannte Drop-Zone, durch einen blauen Rahmen hervorgehoben (siehe Abb. 12). Verlässt die Story Card hingegen eine gültige Drop-Zone, so verschwindet der blaue Rahmen. Sollte der Finger vom Display des Tablets genommen werden, während sich die Story Card in einer gültigen Drop-Zone befindet, so wird sie verschoben. Ist kein gültiger Bereich durch die Markierung hervorgehoben, wenn der Finger das Display nicht mehr berührt, dann wird die Story Card an ihre ursprüngliche Position zurückgesetzt.

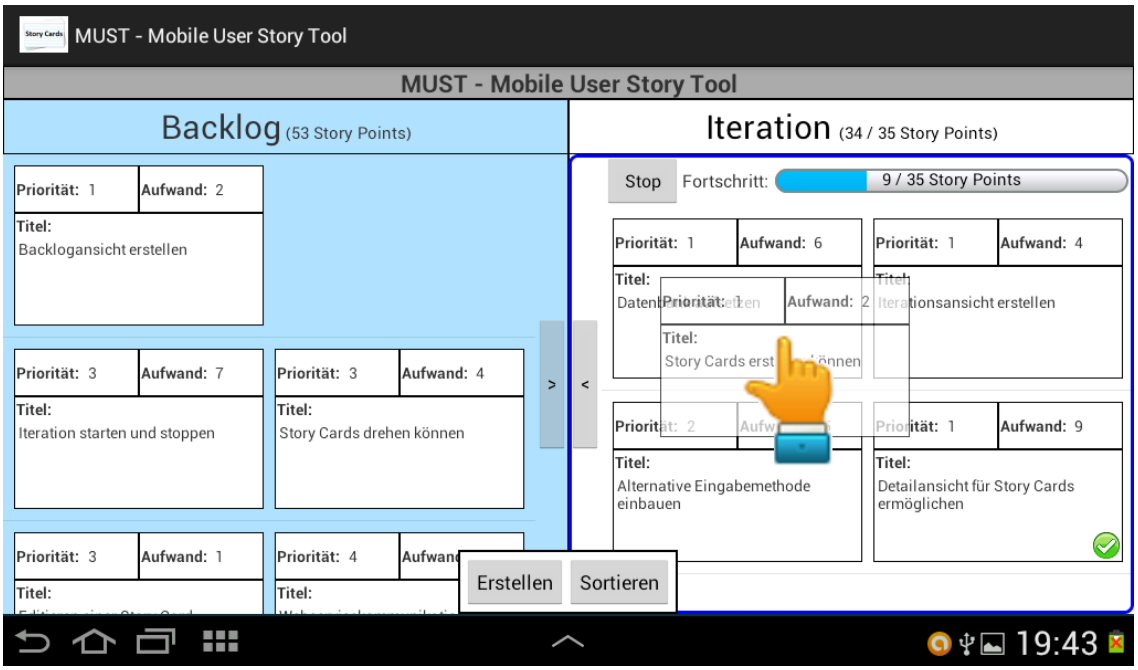


Abb. 12 - Drag & Drop Funktion mit dem Iterationsbereich als Ziel

Dabei ist der Splitscreen eine Besonderheit, weil die Ansicht über vier Drop-Zonen verfügt. Es gibt zum einen zwei Drop-Zonen, die sich über die vollständigen Bereiche des Backlogs und der Iteration erstrecken (siehe Abb. 12). Diese ermöglichen ein unkompliziertes und schnelles Verschieben der Story Cards im Splitscreen. Zum anderen gibt es noch zwei weitere Drop-Zonen, auf dem Backlog- und Iteration-Tab (siehe Abb. 13). Diese Drop-Zonen ermöglichen das gleiche Verschieben von Story Cards, wie die ersten beiden genannten Drop-Zonen. Sie sind aber wichtig, damit der Nutzer auch aus der vollständigen Backlog- und Iterationsansicht, sowie den beiden Detailansichten, Story Cards verschieben kann. Dadurch wird die Aktivität des Verschiebens in allen Pinnwandansichten möglich, was ein effizienteres Arbeiten erlaubt und somit die Bedienbarkeit von MUST steigert.

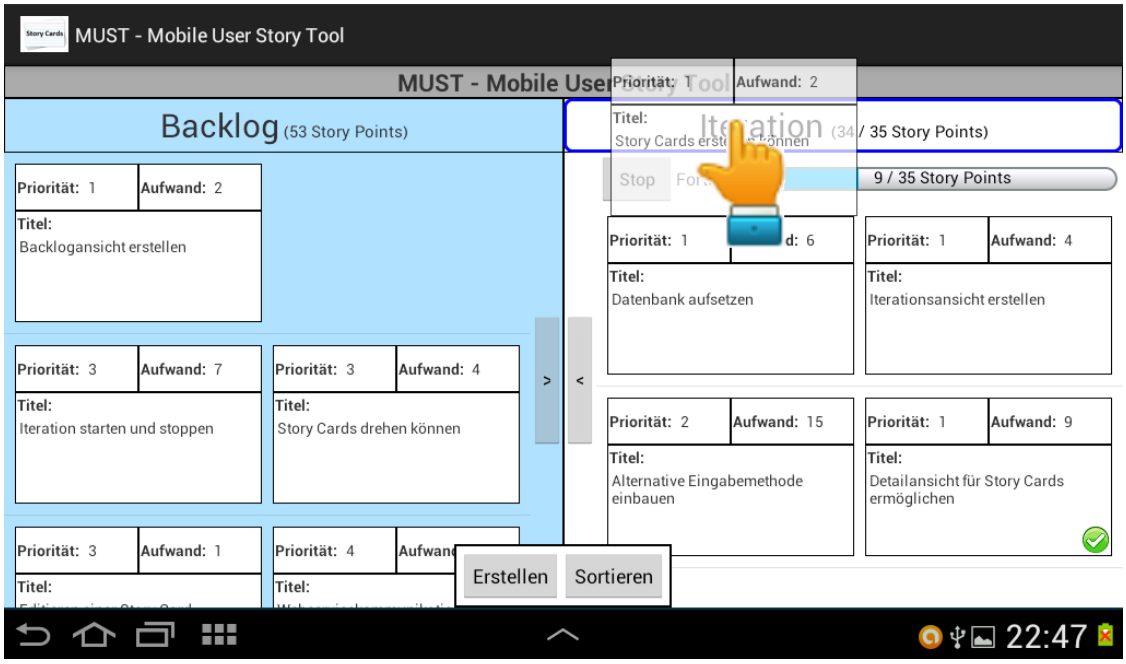


Abb. 13 - Drag & Drop Funktion mit dem Iteration-Tab als Ziel

Die Anforderung **[R11] Beweglichkeit der Story Cards durch Verschieben zwischen Backlog- und Iterationsansicht ermöglichen** ist durch die Ausarbeitung der *Drag & Drop* Funktion in die MUST Anwendung integriert worden.
Dabei wurde darauf geachtet, dass die Bedienung der Funktion dem Umgang mit echten Karteikarten ähnelt. Das Ziel war es, den Ablauf, eine Karteikarte von der Pinnwand zu nehmen und woanders wieder anzubringen, in die Applikation zu überführen. Der Drag startet durch einen langen Klick auf eine Story Card, da dies dem Greifen und Festhalten einer Karteikarte ähnelt. Das Ziehen des Fingers über das Display ahmt die Bewegung einer echten Karteikarte nach. Dabei entspricht die Endposition des Fingers dem Neupositionieren der realen Karteikarte. In der Applikation sind noch automatische Sortierungen enthalten, weshalb die Story Card nicht an der Stelle abgelegt wird, an der sich der Finger zuletzt befunden hat, sondern sie wird in die Listenansicht einsortiert.

4.1.6 Lokales oder globales Projekt

Für eine bessere Verwendbarkeit von MUST ist vorgesehen, dass die Applikation auch mit einem Webservice kommunizieren kann, der als zentraler Speicher für die Story Cards fungiert.

Aufgrund der erhöhten Funktionalität von MUST und der damit verbundenen Komplexität in der Kommunikation mit dem Webservice, ist diese momentan, mit Absprache der Betreuerin, nicht in MUST integriert. Der Grund liegt in den erweiterten Bedienmöglichkeiten von MUST, wodurch der ursprünglich geplante Webservice nicht mehr für die Kommunikation geeignet ist. Es müssen viel mehr Informationen von einem Projekt und den zugehörigen Story Cards verwaltet werden, damit eine korrekte Synchronisation der Daten, auf verschiedenen Tablets, stets gewährleistet werden kann.
Nichts desto trotz wurde die Möglichkeit für die spätere Kommunikation bei der Entwicklung von MUST mit berücksichtigt. Der Nutzer kann die Art eines Projektes manuell in den Projektdetails einstellen. Über einen Toggle-Button ist er jederzeit in der Lage zwischen einem *lokalen* Projekt, dessen Daten nur auf dem Tablet gespeichert werden, und einem *globalen* Projekt, dessen Daten zusätzlich mit einem Webservice synchronisiert werden, zu wechseln (siehe Abb. 14).

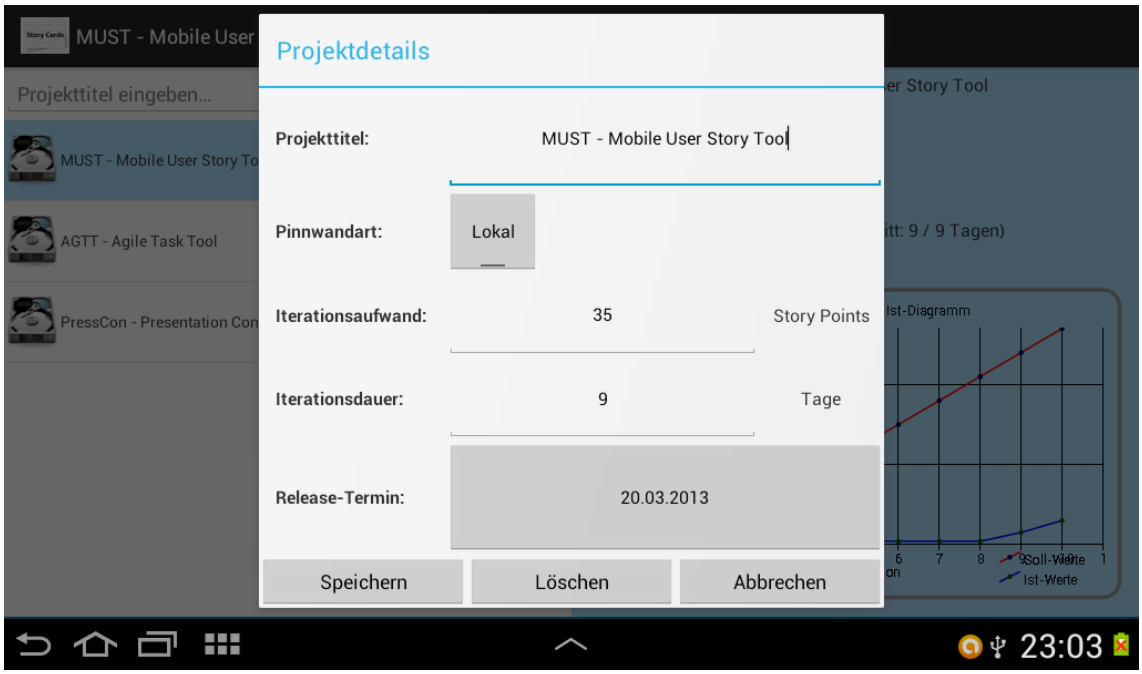


Abb. 14 - Projektdetails mit einer lokalen Pinnwandart

Wenn ein Projekt als *global* eingestellt wird, muss noch der Link, über den der Service erreichbar ist, mit angegeben werden (siehe Abb. 15).

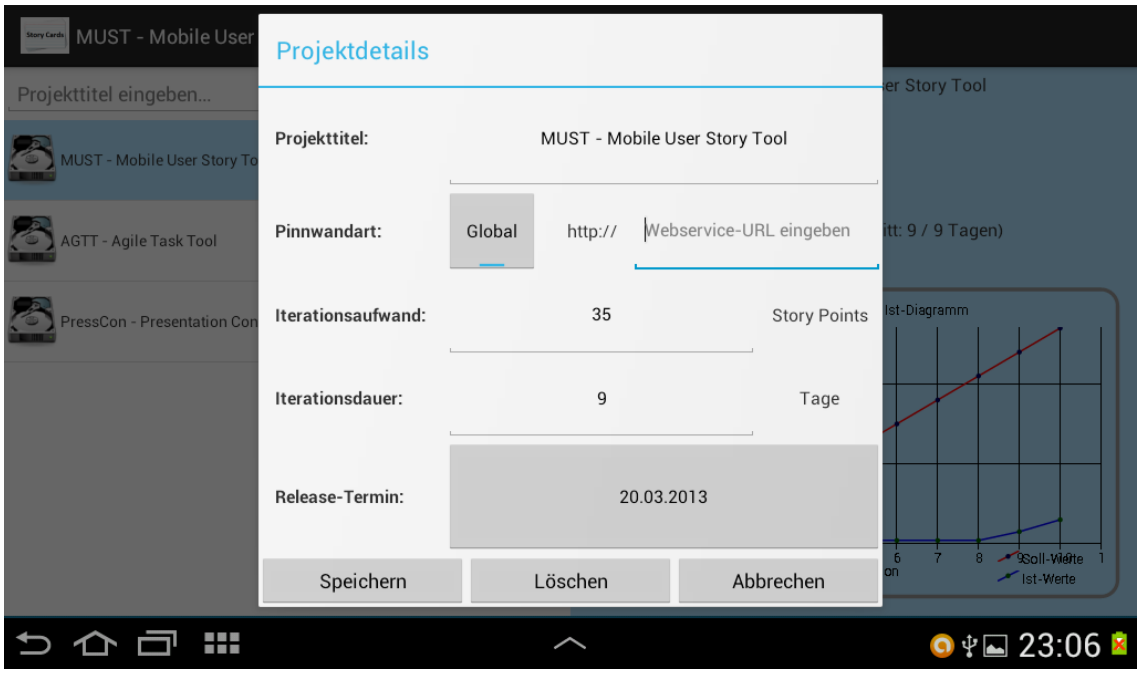


Abb. 15 - Projektdetails mit einer globalen Pinnwandart

Durch das Konzept des Toggle-Buttons für die Einstellung der Projektart wird die Anforderung **[R3] Lokale und globale Projekte in der Applikation anlegen können** realisiert. Dies steigert die Verwendbarkeit der Anwendung deutlich, da die Art eines Projektes zu jedem Zeitpunkt einfach umgestellt werden kann, ohne dass große Veränderungen an einem Projekt nötig sind. Somit ist ein Projekt nicht von einem bestimmten Webservice abhängig, sondern kann auch einfach nur *lokal* weiter verwendet oder mit einem anderen Webservice neu verbunden werden.

Wenn der Webservice zur Verfügung steht, muss die MUST-Applikation für die Kommunikation noch an bestimmten Stellen im Quellcode erweitert werden.
Dabei ist zu beachten, dass ein Tablet die Daten der Story Cards und des Projektes nicht nur an den Webservice sendet, sondern diese auch von dem Webservice abruft.
Die entsprechenden Stellen, an denen die Prüfung der Projektart und die dazugehörigen Kommunikationsbefehle ergänzt werden müssen, sind im Anhang erfasst worden (siehe Anhang D)).

4.2 Szenario der MUST-Applikation

Für eine konkretere Vorstellung der zukünftigen Arbeitsweise mit der MUST-Applikation gilt es, deren Verwendungsgebiet genauer abzustecken Die Anwendung wird dem Sammeln und Verwalten von User Stories dienen. Dabei sollen diese, wie in der Praxis, auf Story Cards notiert werden, welche zum Planen der aktuellen Iteration verwendet werden. Desweiteren werden alle Mitglieder des Teams über einen entsprechenden Webservice, der als zentraler Speicher der Story Cards fungiert, verbunden sein. Dadurch verfügt ein jeder im Team stets über die aktuellste Version der Story Cards.

Nach der nun groben Abgrenzung der Verwendung von MUST, soll der Umgang mit der Applikation in einem Szenario detaillierter dargestellt werden.

Beim initialen Start des Projektes treffen sich die Entwickler mit ihrem Kunden zu einem Story Card Workshop. In diesem sollen so viele Funktionen, der zu entwickelnden Software, wie möglich gesammelt werden. Alle Teilnehmer des Workshops befinden sich in einem Raum, in dem auf einem Smartboard die Oberfläche des leeren Backlogs vom Webservice zu sehen ist. Zusätzlich verfügen alle Teilnehmer über einen Laptop oder ein Tablet, wodurch sie Zugriff auf den Webservice und die elektronische Pinnwand des Projektes haben.

Der Kunde erstellt auf seinem mobilen Gerät im Backlog eine neue Story Card mit einem Titel und speichert diese ab. Die neue Story Card erscheint sofort auf dem Smartboard und auf den Geräten der anderen Teilnehmer. Der Kunde wechselt nun in die Erstell- / Editieransicht der neuen Story Card und erklärt, was er sich unter dem Feature vorstellt. Die relevanten Details der Funktion notiert er dabei im Body der Story Card. Am Ende seiner Ausführung speichert er die Story Card ab, wodurch diese wieder an den Webservice gesendet wird und damit für alle Beteiligten zur Verfügung steht. Nachträglich startet einer der Entwickler den Zeichenmodus für die soeben besprochene Story Card und ergänzt noch ein kleines Bild als spätere Illustration der Funktion. Da der Kunde bei seiner Erläuterung des Features auch schon andere Funktionen mit erwähnt hat, hat ein weiterer Entwickler diese schnell notiert. Dazu legte er jeweils eine neue Story Card an, auf denen er einen kurzen Titel als spätere Erinnerung vermerkte. Diese Story Cards erscheinen ebenfalls in dem Backlog des Webservice und stehen dann auch allen anderen direkt zur Verfügung. Wenn der Kunde nun auf seinem Gerät in die Backlog- oder Backlogdetailansicht wechselt, wird er die neuen Story Cards sofort sehen.

Nachdem der Story Card Workshop beendet ist, führt die Gruppe einige Tage später die Iterationsplanung durch. Dabei findet wieder eine dem Story Card Workshop ähnliche Situation statt, in der jeder Teilnehmende über ein mobiles Gerät, sei es Laptop oder Tablet, verfügt, mit dem er auf den Webservice zugreifen kann. Zusätzlich wird auch auf dem Smartboard wieder der Webservice für ein gemeinsames Bild angezeigt. In den Tagen vor der Planung hat der Kunde, über sein mobiles Gerät, den einzelnen Story Cards bereits eine Priorität zugeordnet, damit die Relevanz der Features für die Entwickler erkennbar wird und sie die Wichtigeren bezüglich ihres Aufwandes zuerst schätzen. Durch das elektronische System ist der Kunde nicht gezwungen vor Ort bei den Entwicklern zu arbeiten, sondern kann von überall aus die Priorisierung vornehmen.
Für eine bessere Übersicht über die Relevanz der einzelnen Story Cards sortieren die Entwickler diese auf ihren Geräten nach der Priorität, wodurch die für den Kunden wichtigsten Features zuerst angezeigt werden und dann in ihrer Relevanz abnehmen. Nachdem jeder der Entwickler die Story Cards auf seinem Gerät sortiert hat, fangen sie mit der ersten an und überlegen sich, wie sie die Funktion realisieren können. Einige Anhaltspunkte dazu notieren sie sich auf der Story Card in dem Bereich „Anmerkungen“. Ein Vorteil hierbei ist, dass jeder Entwickler eine eigene Version der Story Card direkt vor sich hat und die festgehaltenen Anmerkungen leicht einsehen kann.
Letztendlich schätzen sie den Aufwand der Funktion und tragen diesen Wert auf der Story Card ein. Diese Prozedur wiederholen sie für die weiteren Story Cards solange, bis sie in der Summe mindestens etwas mehr als den maximalen Iterationsaufwand des Projektes erreicht haben.

Mit den geschätzten Story Cards kann der Kunde die aktuelle Iteration planen. Dazu wechselt er auf seinem mobilen Gerät in die Splitscreenansicht und wählt als Sortierung für die Story Cards ebenfalls die Kategorie Priorität. Er selektiert die erste Story Card im Backlogbereich des Splitscreens durch einen langen Klick und aktiviert so den Drag & Drop Modus. Nun kann er die Story Card in den Iterationsbereich ziehen, los lassen und sie dadurch in die Iteration verschieben. Die Applikation zeigt ihm dabei in dem Iteration-Tab an, wie viele

Story Points er bereits in die Iteration eingeplant hat und wie groß der Veloctiy-Wert des Teams ist. Wenn er den maximal geplanten Iterationsaufwand überschreitet, wird ihm dies visuell verdeutlicht, indem sich die Schrift des Tabs rot einfärbt. In diesem Fall muss er überlegen, wie er die Iteration genau gestalten will. Entweder wählt er eine Story Card mit der passenden Anzahl an Story Points oder er plant etwas mehr oder weniger, als den festgelegten Iterationsaufwand ein.
Bei der ersten Variante kann sich der Kunde die Story Cards nach Aufwand sortieren lassen, um schneller einen Überblick zu erhalten, ob es überhaupt eine passende Story Card gibt. Wenn es sich im Falle der zweiten Variante nicht um die erste zu planende Iteration handelt, so besteht die Möglichkeit, in dem vom System generierten kumulativen Soll-Ist-Diagramm in der Projektansicht, den bisherigen Arbeitsverlauf des Teams zu begutachten. Anhand der gemessenen Soll- und Ist-Werte kann der Kunde erkennen, wie das Team bisher gearbeitet hat und ob er etwas mehr oder weniger Aufwand einplanen sollte.

Nachdem die aktuelle Iteration vom Kunden fertig geplant wurde, kann die eigentlich Arbeit der Entwickler beginnen. Die Entwickler starten die Iteration und verteilen zuerst die vom Kunden ausgewählten Story Cards untereinander. Dies kann zum Beispiel nach Reihenfolge der Story Cards oder nach der Expertise der Entwickler erfolgen. Jeder Entwickler trägt sich in den Anmerkungen der Story Card ein, die er implementieren wird. Dies dient zur Sicherstellung, dass eine Story Card nicht von mehreren Entwicklern bearbeitet wird.
In der Zwischenzeit überlegt sich der Kunde Abnahmetestfälle für die einzelnen Funktionen, wobei er mit den Story Cards anfängt, die bereits zugewiesen wurden. Diese Information kann er schnell über die Detailansicht ermitteln. Dann setzt er sich mit seinem Tablet zu dem entsprechenden Entwickler und bespricht mit diesem die genauen Abnahmetestfälle. Diese dokumentiert sie direkt über die Editieransicht in der zugehörigen Story Card.
Wenn die Entwickler mit der Implementierung soweit fertig sind, lassen sie die Story Card vom Kunden abnehmen. Dazu benötigen sie die anfangs festgehaltenen Abnahmefälle, die sich auf der Rückseite der detaillierten Story Card befinden. Diese können sie schnell durch die Dreh-Geste auf der Vorderseite der detaillierten Story Card in der Detailansicht abrufen. Nach der erfolgreichen Abnahme durch den Kunden können die Entwickler die Story Card durch die Abhaken-Geste auf erledigt setzen. Sollte dem Kunden später, nach der eigentlichen Abnahme, doch noch etwas zu der Story Card einfallen, ist es kein Problem, die Story Card wieder durch eine Geste zurück auf den Zustand unerledigt zu setzen und die Entwickler nacharbeiten zu lassen.

Zum Abschluss der Iteration geht der Scrum-Master[4], zu den einzelnen Entwicklern und fragt sie, ob sie ihre Story Card noch rechtzeitig fertig bekommen. Falls das nicht der Fall ist, spaltet der Scrum-Master die Story Card in den Teil, der noch fertig wird und den der später noch zu implementieren ist, auf. Der noch zu implementierende Teil geht ins Backlog und muss bei der nächsten Iterationsplanung als neue Story Card mit einbezogen werden. Um bei dem ganzen zuvor beschriebenen Vorgang eine gute Übersicht zu behalten, öffnet der Scrum-Master zuerst die Iterationsansicht und erstellt dort eine neue Story Card in der er den nicht mehr zu schaffenden Teil des Entwicklers dokumentiert. Wenn er die Story Card abgespeichert hat, kann er diese direkt per Drag & Drop in das Backlog verschieben ohne die Iterationsansicht verlassen zu müssen. Nach dem Ende der Iteration kann der Zyklus erneut mit der nächsten Planungsphase beginnen.

[4] Ein Scrum-Master ist der Verantwortliche für die Einführung und Einhaltung der Regeln der agilen Methodik mit User Stories [20].

4.3 Abgrenzung der Konzepte von MUST

Es gibt bereits einige Werkzeuge für die Verwaltung von User Stories. Deshalb stellt sich die Frage, wo die Vorteile der in dieser Arbeit entwickelten Applikation gegenüber bereits existierenden Systemen liegen. In diesem Abschnitt soll durch eine Abgrenzung der Konzepte von MUST gegenüber anderen Werkzeugen verdeutlicht werden, worin sich die MUST-Applikation gerade von anderen Systemen unterscheidet und welche Vorteile die entwickelten Konzepte bieten.

Die Auswahl der Werkzeuge für den Vergleich erfolgte unter Zuhilfenahme einer Auflistung von bewerteten Tools für die Arbeit mit User Stories [11]. Es gab zwei wesentliche Kriterien für die Auswahl. Zuerst einmal sollte es sich um Werkzeuge mit einer sehr hohen Bewertung von Nutzern handeln. Weiterhin wurden die Beschreibungen solcher Werkzeuge studiert, um ihren Funktionsumfang zu ermitteln. Dieser soll geeignet sein, um eine vernünftige Abgrenzung gegenüber der MUST-Applikation zu ermöglichen. Als Resultat dieser Suche wurden die beiden Anwendungen **Planbox Agile Tool** [12] und **Scrum-it** [13] ausgewählt.

Für die Auswahl von **Planbox Agile Tool** gab es zwei wesentliche Gründe. Diese Anwendung gehört zu den am besten bewerteten Produkten und der Autor dieser Arbeit hat Erfahrungen mit diesem Werkzeug.
Der Hauptgrund für die Auswahl von **Scrum-it** war die besondere Hervorhebung der Verwendung von einer Pinnwandansicht und Touch-Technologie, welche für Tablets besonders geeignet sein soll.

Bevor die Konzepte von MUST gegenüber den anderen beiden Tools abgegrenzt werden können, ist es nötig die Konzepte und Features dieser beiden Anwendungen kurz zusammenzufassen.

Bei dem **Planbox Agile Tool** handelt es sich um eine Web-Applikation, die als zentrales Konzept klassische Listenansichten für die verschiedenen Bereiche wie Backlog und Iteration verwendet. Neben diesen Listen gibt es einen zweiten Bereich, in dem entweder verschiedene Messdaten in Diagrammen abgerufen werden können oder in dem die einzelnen Listeneinträge mit all ihren Details dargestellt werden. Die Listeneinträge enthalten eine Art User Story beziehungsweise Task und bieten noch sehr viele weitere Eingabemöglichkeiten. Der Kern von **Planbox Agile Tool** fokussiert das Prinzip der Agilen Entwicklung und beinhaltet dabei aber auch die Methode der User Stories. Die Bedienung dieser Anwendung ist im Wesentlichen auf Maus und Tastatur ausgelegt und bietet per Drag & Drop der Listenelemente, über die einzelnen Tabs, die Möglichkeit, diese zwischen den Bereichen zu verschieben.

Scrum-it ist ebenfalls eine Web-Applikation, die im Kern über drei Ansichten verfügt. Als erstes gibt es eine Ansicht in der Projekte in einer Liste erstellt und bearbeitet werden können. Diesen Projekten können Personen zugeordnet werden. Weiterhin verfügt das Tool über eine Ansicht für Iterationen, den sogenannten Sprints. Diese sind ebenfalls als Listenelemente dargestellt. In den einzelnen Iterationen ist es wiederum möglich User Stories in eine Liste hinzuzufügen. Diese User Stories lassen sich aber nicht von einer Iteration in eine andere verschieben, sondern sind fest an eine Iteration gebunden. Die letzte wichtige Ansicht enthält eine Pinnwanddarstellung. Diese kann für jede Iteration immer nur einzeln aufgerufen werden. In einer Pinnwandansicht ist es möglich, für jede User Story eine beliebige Anzahl an kleinen Karten anzulegen, auf denen kurze Texte geschrieben werden können. Durch das entworfene Layout ist die Anwendung sowohl mit Maus und Tastatur zu bedienen, als auch mit den Fingern direkt über ein Tablet. Die Anwendung basiert stark auf der Methodik der User Stories, aber realisiert die Arbeit mit der Pinnwand anders, als es in der Theorie

vorgesehen ist. Anstatt dass es eine Story Card pro User Story gibt, können viele Story Cards für eine einzelne User Story angelegt werden.

Die folgende Abgrenzung betrachtet die drei Aspekte *Darstellung*, *Beweglichkeit* und *Bedienung*. Anhand dieser Bereiche lassen sich die wesentlichen Unterschiede zwischen MUST und den anderen beiden Anwendungen gut aufzeigen.

Der erste wesentliche Unterschied zwischen den Werkzeugen ist die Art der *Darstellung* der User Stories. In MUST soll dem Nutzer durch die gewählte Repräsentation der Story Cards das Gefühl vermitteln werden, dass er mit echten Karteikarten und einer realen Pinnwand arbeitet. Der Vorteil dieser Herangehensweise ist, dass die Arbeit zum einen an dem in der Literatur schriftlich festgehaltenen Vorgehen orientiert, was den Einstieg für neue Nutzer in der agilen Entwicklung mit User Stories erleichtert. Zum anderen wird der Übergang von der physischen Arbeit mit Karteikarten hin zu der Verwendung des elektronischen Werkzeugs vereinfacht.
Von den beiden ausgewählten Anwendungen hat **Scrum-it** einen ähnlichen Ansatz, da die Struktur der Anwendung an die Idee der Überführung der realen Arbeit mit Karteikarten anknüpft. **Planbox Agile Tool** hingegen bietet nur schlichte Listen für die User Stories in den unterschiedlichen Bereichen. Der Inhalt der Listenelemente ist zwar sehr detailliert, aber die Darstellung ist nicht an die Arbeit mit Karteikarten angelehnt.

Ein weiterer wichtiger Aspekt für ein effizientes Arbeiten mit User Stories ist deren hohe *Beweglichkeit*. Durch Story Cards können schnell und einfach Iterationen geplant und angepasst werden. Die Beweglichkeit wird in MUST durch die Verfügbarkeit der Drag & Drop Funktion in allen Pinnwandansichten umgesetzt.
Die Anwendung **Planbox Agile Tool** bietet die Möglichkeit des Drag & Drop auf etwas eingeschränktere Art, da die Funktion nur an die Tabs gebunden ist, während in MUST auch zwischen Pinnwandbereichen selbst verschoben werden kann.
Bei **Scrum-it** gibt es zwar auch eine Drag & Drop Funktion, die dient aber nicht dem Planen einer Iteration, da die User Stories immer fest an eine Iteration gebunden sind. Stattdessen wird die Beweglichkeit der Karten nur dafür verwendet, um ihren Status in der Bearbeitung auf der Pinnwand darzustellen.

Der letzte Aspekt, auf den noch eingegangen werden soll, ist die Art der *Bedienung* der Anwendungen. MUST fokussiert dabei die Arbeit mit den Fingern, wie es auch in der Praxis mit echten Karteikarten der Fall ist. Der Nutzer soll die Möglichkeit haben, wie in der Realität, mehr mit seinen Finger direkt zu steuern, anstatt nur durch eine Tastatur und Maus Einfluss auf die Anwendung zu haben.
Bei **Planbox Agile Tool** ist dieser Gedanke gar nicht aufgegriffen worden. Durch die Darstellung der Story Cards als Listenelemente liegt der Fokus der Bedienung auf Maus und Tastatur.
Die Anwendung **Scrum-it** versucht sich auch an der realistischeren Umsetzung der Bedienung mit den Fingern, in dem Touch-Technologie eingesetzt wird. Diese Technologie wird aber nur sehr geringfügig in der Pinnwandansicht genutzt.
MUST hingegen setzt sogar ganz besonders auf die Steuerung mit Gesten für verschiedene Funktionen, um so die Bedienung der Applikation durch die direkte Verwendung der Finger weiter zu erhöhen.

Mit der Gegenüberstellung der drei Aspekte *Darstellung*, *Beweglichkeit* und *Bedienung*, wird deutlich, worin der wesentliche Unterschied zwischen der MUST Applikation und den anderen Anwendungen besteht.

Die bisherigen Applikationen für das Management von User Stories fokussieren das Prinzip der agilen Entwicklung, aber orientieren sich sehr wenig an der realen Arbeit mit User Stories. Das bedeutet für viele Nutzer, die mittels eines elektronischen Systems ihre User Stories verwalten wollen, dass sie oft auf bestimmte Eigenschaften von Karteikarten, sowie gewisse Aktivitäten bei der Arbeit mit diesen, verzichten müssen.
MUST hingegen ist genau für solche Bedürfnisse ausgelegt. Die entwickelten Konzepte fokussieren insbesondere die realitätsnahe Arbeit mit User Stories, sodass die Nutzer in der Lage sind mit dem elektronischen System auf eine Art und Weise zu arbeiten, die ihnen das Gefühl vermittelt, als würden sie mit echten Karteikarten arbeiten. Weiterhin wurde, neben der Methode der User Stories, auch die agile Entwicklung, mit ihren einzelnen Prozessen, bei der Entwicklung berücksichtig und in die Applikation integriert.

Durch die besondere Herangehensweise der Ermittlung von Eigenschaften und Aktivitäten der Karteikarten, welche für die Arbeit und Verwaltung von User Stories von großer Bedeutung sind, konnten zielgerichtet Konzepte entwickelt werden, die ein effizientes Arbeiten und Verwalten von User Stories in der MUST-Applikation ermöglichen und unterstützen (siehe Abschnitt 4.1).

4.4 Technische Aspekte und Entscheidungen

Im folgenden Abschnitt werden kurz technische Aspekte und Entscheidungen ausgeführt, die sich während der Entwicklung und Implementierung von MUST ergeben haben.

Grundlegend fand die Entwicklung von MUST mit der Entwicklungsumgebung **Eclipse Juno** statt, wobei die Standard **Android SDK** verwendet worden ist.
Zusätzlich wurde die externe Bibliothek **Androidplot** [14] genutzt, um das Zeichnen des kumulativen Soll-Ist-Diagramms in der Projektansicht zu ermöglichen. Die **Android SDK** stellt keine Methoden und Klassen für die Erstellung von Graphen zur Verfügung. Aus diesem Grund wurde auf eine externe Bibliothek zurückgegriffen, die speziell für diesen Zweck entwickelt worden ist.
Die Realisierung der Wischgesten wird durch die Verwendung von GestureOverlayViews ermöglicht, welche Standard-Android-Elemente sind. Diese benötigen eine Gesten-Bibliothek, welche mit dem Gesture-Builder, der aus den Android Samples stammt, definiert werden konnte

Für das Testen der einzelnen Funktionen von MUST ist ein Samsung Galaxy Tab 2 mit 7 Zoll Display und der Android-Version 4.0.4 verwendet worden. Auf den Emulator von Android wurde nicht zurück gegriffen, da dieser zum einen bei der Verwendung der Gestensteuerung nicht das gleiche Gefühl bei der Bedienung ermöglicht, wie es mit einem echten Tablet der Fall ist. Zum anderen startet der sehr langsam, was die Arbeit mit ihm umständlich und aufwendig gemacht hätte.

Ein wesentliches Ziel bei der Entwicklung von MUST war die Unterstützung einer möglichst großen Menge an Tablet-Geräten, um so die Verwendbarkeit der Applikation zu erhöhen. Aus diesem Grund sind alle Android-Versionen für Tablets in der Entwicklung berücksichtigt worden. Damit kann MUST auf allen Geräten verwendet werden, deren Android-Version zwischen Android 3.0 und dem aktuellen Android 4.2 liegt.
Das hat zur Folge, dass momentan insgesamt 43.9% aller Geräte (siehe Abb. 16), die auf Google Play Anfang Februar zugegriffen haben, die MUST Applikation verwenden könnten. In Abbildung 16 wird aber nicht genauer zwischen Smartphone- und Tablet-Geräten unterschieden, sodass der genaue Anteil an Tablet-Geräten, für die MUST explizit entwickelt worden ist, nicht angegeben werden kann. Es konnte auch keine andere Quelle ermittelt werden, die genauer angibt wie viele Tablet-Geräte in den 43,9% enthalten sind.

Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	2.2%
2.2	Froyo	8	8.1%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	45.4%
3.1	Honeycomb	12	0.3%
3.2		13	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.0%
4.1	Jelly Bean	16	12.2%
4.2		17	1.4%

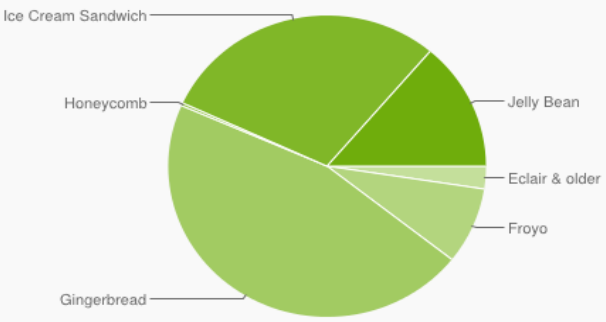


Abb. 16 - Aktuelle Android Distribution (Stand 04.02.2013) [15]

5 Evaluation der Applikation

Dieses Kapitel befasst sich mit der Evaluation der MUST-Applikation, die im Rahmen dieser Arbeit durchgeführt wurde. Neben der Beschreibung des Ablaufes der Evaluation sollen die gemessenen Daten präsentiert werden. Durch die Interpretation und Auswertung dieser Messdaten gilt es Erkenntnisse zu erhalten, mit denen Aussagen und Maßnahmen bezüglich des Zieles der Evaluation getroffen werden können.

5.1 Ziel und Aufbau der Evaluation

Dieser erste Abschnitt dient zum besseren Verständnis des Ziels dieser Evaluation und ihrem gesamten Aufbau.

Das Ziel der Evaluation war die Überprüfung der Akzeptanz und des Verständnisses von bestimmten umgesetzten Konzepten zur adäquaten Überführung von Aktivitäten und Eigenschaften der Arbeit mit echten Karteikarten in das elektronische System, durch die Probanden. Dabei wurde der Fokus speziell auf die Aspekte Navigation und Gestensteuerung gelegt.
Als Metrik wurde die Bearbeitungszeit der Probanden, gemessen in Sekunden, gewählt. Aus den gemessenen Zeiten wurden weitere Werte, wie Median und arithmetischer Mittelwert berechnet. Diese Werte können durch ihre Interpretation aufzeigen, ob die Probanden die entsprechenden Konzepte ermitteln, verstehen und wiederholt anwenden konnten. Bei hohen Werten bezüglich Median und Mittelwert handelt es sich um einen deutlichen Hinweis, dass es bei dem Verständnis und der Anwendung der realisierten Konzepte Schwierigkeiten gab.

Für dieses Ziel wurde als Evaluationstechnik eine Art *Think-Aloud Studie* verwendet. Der Proband wurde gebeten während der Arbeit mit MUST seine Gedanken laut zu äußern. Dadurch sollte zum einen das von ihm aus dem Systembild abgeleitete mentale Modell für den Prüfer ersichtlich werden und zum anderen konnte der Prüfer schneller erkennen, ob der Proband Probleme bei einer Aufgabe hat.
Es war davon auszugehen, dass die Probanden noch nie agile Softwareentwicklung mit User Stories in der Praxis durchgeführt hatten. Deshalb konnte nicht erwartet werden, dass sie über das nötige Vorwissen verfügen, wie auf diese Weise gearbeitet wird. Aus diesem Grund war es den Probanden erlaubt dem Prüfer Fragen zu stellen, wenn sie nicht weiter wussten. Dabei wurde nicht genau auf die in der Applikation realisierten Konzepte eingegangen, sondern es wurde vielmehr versucht mittels der Erklärung der praktischen Arbeit mit User Stories die Probanden zu unterstützen.

Insgesamt setzte sich ein Evaluationsdurchgang mit einem Probanden aus zwei Teilen zusammen. Im ersten Teil sollte der Proband versuchen fünf Aufgaben (siehe Anhang E)) zu lösen. Nach der Abarbeitung der Aufgaben füllte jeder Proband einen Bewertungsbogen (siehe Anhang F)) aus, in dem gezielt nach dem subjektiven Eindruck des Probanden bezüglich bestimmter Aspekte gefragt wurde.
Der genaue Ablauf sah so aus, dass den Probanden nach der Begrüßung kurz die Planung des Evaluationsdurchganges erläutert wurde. Um die Dauer des Tests zu verkürzen und damit sich die Probanden mehr auf die Verwendung der Applikation konzentrierten, wurde ihnen zusätzlich sehr kurz mit einer groben Übersichtsgrafik erklärt, welche Ansichten es gibt und wie diese im Zusammenhang stehen. Dabei wurde jedoch nicht genauer auf die einzelnen Bedienmöglichkeiten eingegangen, sondern es wurde nur erwähnt, dass es Gestensteuerung und Selektion durch Klicken gibt.
Nach der Einführung fing der Proband an alleine die einzelnen Aufgaben nacheinander abzuarbeiten. Wenn er bei einer Aufgabe nicht weiter wusste, so konnte er vom Prüfer Hilfe

beziehen. Für jede Aufgabe wurde die Bearbeitungszeit gemessen. Sobald der Proband anfing zu arbeiten, startete die Messung, und wenn er meinte die Aufgabe beendet zu haben, wurde die Zeit gestoppt.
Nach dem Abschluss des ersten Teils der Evaluation wurde dem Proband der Bewertungsbogen vorgelegt. Dieser diente zur Feststellung des Eindrucks und der Akzeptanz des Probanden bezüglich bestimmter Aspekte. Dabei wurden Likert-Skalen mit vier anstatt fünf Stufen verwendet, um bei jeder Antwort eine genauere Tendenz des Eindrucks des Probanden zu bekommen.

5.2 Evaluationsauswertung

Für die Auswertung der gemessenen und berechneten Daten ist es nötig, dass diese zuerst geeignet aufbereitet und präsentiert werden. Dadurch kann ein besserer Bezug auf die Beobachtungen und Schlussfolgerung bezüglich des Zieles der Evaluation hergestellt werden. Im Folgenden werden die Daten daher zuerst in Tabellen und Grafiken dargestellt, um dann anhand dieser Abbildungen die eigentlichen Beobachtungen und die damit verbundenen Schlussfolgerungen genauer zu erläutern.

5.2.1 Beobachtungen und Schlussfolgerungen

Die nachfolgenden Tabellen und Abbildungen wurden anhand der ermittelten Daten, welche auf der gemessenen Bearbeitungszeit der Probanden in Sekunden basieren (siehe Anhang G)), erstellt und stellen diese für einen geeigneten Vergleich dar.

	Aufgabe 1	**Aufgabe 2**	**Aufgabe 3**	**Aufgabe 4**	**Aufgabe 5**
Fokussierter Aspekt	Navigation	Gesten	Navigation	Navigation & Gesten	Gesten
Summe (s)	131	377	241	64	262
Mittelwert (s)	33	94	60	16	66
Median (s)	31	95	58	14	64
Max (s)	47	116	77	32	98
Min (s)	23	71	49	5	37
Erwartete Dauer (s)	30	120	105	30	75

Abb. 17 - Auswertungsstatistik der Bearbeitungszeiten

Die obige Tabelle (siehe Abb. 17) zeigt eine Statistik der gemessenen Bearbeitungszeiten der einzelnen Aufgaben. Dabei wurden insbesondere die Summe aller Bearbeitungszeit pro Aufgabe, der Mittelwert und der Median berechnet. Weiterhin sind die maximale und die minimale gemessene Bearbeitungszeit jeder Aufgabe angegeben.
Vor der Durchführung der Evaluation wurde geschätzt, wie lange jede Aufgabe maximal in ihrer Bearbeitung durch einen Probanden dauern sollte. Die jeweilige erwartete Dauer ist ebenfalls in dieser Tabelle enthalten.

Einige der in dieser Tabelle enthaltenen Daten werden in dem nachfolgenden Balken-Diagramm für einen geeigneten visuellen Vergleich dargestellt. Damit wird die Ausführung und Erläuterung der festgestellten Beobachtungen und der daraus abgeleiteten Schlussfolgerungen verständlicher.

Das Diagramm (siehe Abb. 18) stellt die maximale, durchschnittliche, minimale und erwartete Bearbeitungszeit der einzelnen Aufgaben einander gegenüber.

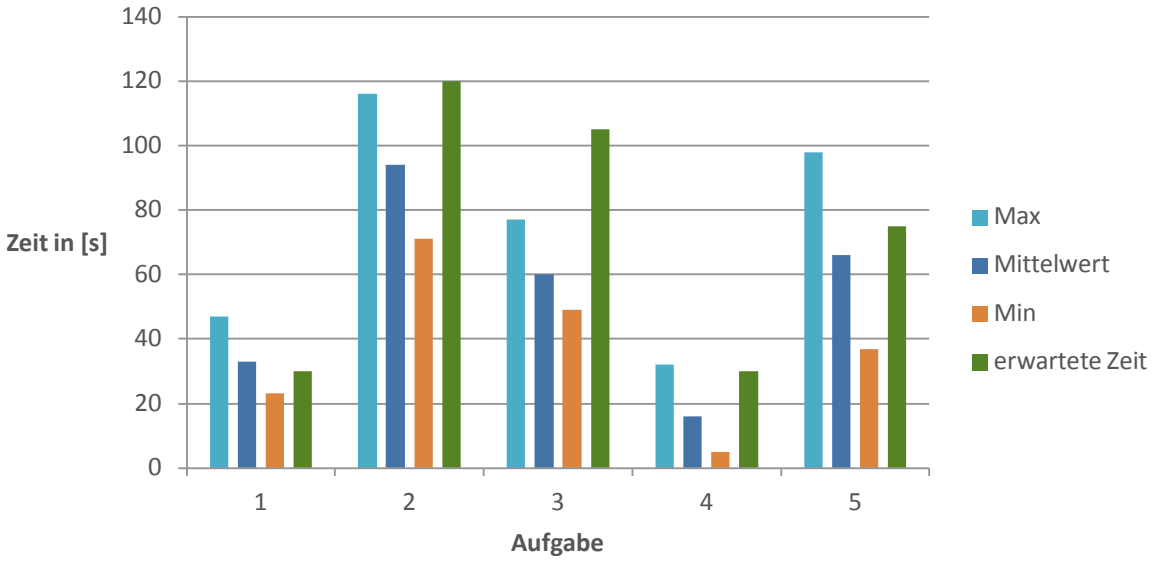


Abb. 18 - Vergleich bestimmter Bearbeitungszeiten

Die letzte Tabelle (siehe Abb. 19) enthält die ausgewerteten Antworten der Bewertungsbögen. Es werden pro Aspekt die Anzahl der in den Likert-Skalen angegebenen Antworten präsentiert. Dabei ist anzumerken, dass es aufgrund des Funktionsumfanges von MUST möglich war, einige Aufgaben auf verschiedene Arten zu lösen. Abhängig von dem Lösungsweg, den der Proband gewählt hat, konnte es passieren, dass bestimmte Funktionen von MUST nie verwendet wurden. Da sich die letzten vier Aspekte aber auf bestimmte Funktionen beziehen, wurde zusätzlich für den Fall, dass ein Proband eine Funktion gar nicht genutzt hat, die Option „nicht verwendet“ bereitgestellt.

Aspekt	**Bewertung der Probanden**				
	sehr schlecht	**schlecht**	**gut**	**sehr gut**	**nicht verwendet**
Gesamteindruck	0	0	3	1	-
Übersichtlichkeit	0	0	3	1	-
Navigation durch Ansichten	0	0	3	1	-
Geste: Drehen	0	0	2	2	0
Geste: Erledigen	0	0	2	2	0
Drag & Drop: Splitscreen	0	0	2	1	1
Drag & Drop: andere Ansichten	0	0	1	2	1

Abb. 19 - Auswertungsstatistik der Bewertungsbögen

Auswertung des ersten Teils der Evaluation:

Betrachtet man das Diagramm in Abbildung 18 und vergleicht die berechneten Mittelwerte mit der erwarteten Bearbeitungszeit der jeweiligen Aufgaben, so sieht man, dass der Mittelwert der Bearbeitungszeit nach der ersten Aufgabe stets unter der erwarteten Dauer liegt. Der in etwa gleich hohe Mittelwert gegenüber der erwarteten Bearbeitungsdauer bei der ersten Aufgabe lässt sich dadurch erklären, dass sich die Probanden zuerst mit der Applikation vertraut machen mussten. Nachdem sie einen Überblick über die Applikation hatten konnten sie sich besser orientieren und waren sogar in allen weiteren Aufgaben schneller als erwartet.

Bei den Aufgaben 2 und 5 ist der Mittelwert nur sehr knapp unterhalb der erwarteten Bearbeitungszeit, während bei den Aufgaben 3 und 4 die Probanden im Mittel nur etwa halb so lange gebraucht haben, wie es erwartet wurde. Für ein besseres Verständnis dieses Verlaufes muss man sich darüber klar werden, welcher Aspekt bei den jeweiligen Aufgaben fokussiert wurde.
Die beiden Aufgaben 2 und 5 waren auf die Verwendung der Gestensteuerung ausgerichtet. Dabei mussten die Probanden in beiden Aufgaben jeweils eine neue nicht bekannte Geste verwendet. Für das Ermitteln und Verstehen der jeweiligen Geste haben die Probanden länger gebraucht, wodurch eine erhöhte Bearbeitungsdauer resultiert.
Die anderen beiden Aufgaben hingegen waren auf die Navigation zwischen und in den Ansichten ausgelegt. Betrachtet man dazu noch die erste Aufgabe, welche ebenfalls für den Aspekt der Navigation gedacht war, so zeigt sich, dass die Probanden mit jeder weiteren Aufgabe bezüglich der Navigation immer schneller wurden. Weiterhin bestand bei den Aufgaben 3 und 4 für die Probanden die Möglichkeit die in Aufgabe 2 erlernte Geste wiederzuverwenden. Die Anwendung der bereits bekannten Geste ist für alle Probanden kein Problem gewesen.

Insgesamt lassen sich zwei wesentliche Beobachtungen festhalten.

1. Beobachtung:
   Bei der Verwendung von neuen Gesten kam es zu einer deutlichen Erhöhung der Bearbeitungszeit für die jeweilige Aufgabe. Die Probanden haben länger gebraucht, um die Gesten zu ermitteln und anzuwenden.

2. Beobachtung:
   Bei der Navigation zwischen und innerhalb der Ansichten ist eine Verbesserung in der Dauer der Bearbeitungszeiten zu erkennen. Nachdem die Probanden sich einen ersten Überblick verschafft hatten, konnten sie immer zügiger und sehr viel schneller als erwartet zwischen den Ansichten navigieren.

Aus diesen beiden Beobachtungen lassen sich zwei wesentliche Schlussfolgerungen ziehen. Aus der ersten Beobachtung wird deutlich, dass es den Probanden schwer gefallen ist neue, für sie unbekannte Gesten, zu ermitteln und anzuwenden. Dies lässt den Schluss zu, dass es sich bei der Gestensteuerung um eine Bedienmöglichkeit handelt, die für Anfänger, die das erste Mal mit der Applikation arbeiten, nicht geeignet ist. Bei den Probanden zeigte sich eine deutliche Hemmschwelle Gesten von selbst auszuprobieren und zu testen, vermutlich aus Angst vor möglichen Fehlern, die sie hätten machen können.
Die Steuerung der Applikation mit Gesten ist also nicht für Anfänger geeignet und stellt eher eine Expertenbedienung dar. Aus diesem Grund ist es nötig, dass ein passendes Hilfssystem in die MUST-Applikation zu integrieren durch das die Nutzer die Gesten erlenen und abrufen können. Beispielhaft kann dies durch ein Hilfemenü realisiert werden, das über einen Button in jeder Ansicht aktiviert werden kann. Dadurch können dann, in einer transparenten Oberfläche, Grafiken und kurze Anleitungen angezeigt werden, die dem Nutzer die möglichen Gestensteuerungen erläutern [16].

Für die zweite Beobachtung kann man zu dem Schluss gelangen, dass die Probanden mit den einzelnen Pinnwandansichten, der Idee der Veränderung des Blickwinkels auf die Bereiche der Pinnwand und den zwei Detailstufen der Story Cards gut zurecht gekommen sind.
Die Probanden mussten sich aber zuerst mit der Anwendung vertraut machen und sich einen Überblick über den Aufbau der Applikation verschaffen. Nach einer sehr kurzen Eingewöhnungsphase gelang es allen Probanden schnell ein gutes mentales Modell der Applikation bezüglich ihres Aufbaus und den jeweiligen Optionen aufzubauen, wodurch sie

immer zügiger zwischen den Ansichten navigieren konnten. Dadurch resultierte eine im Mittel deutlich kürze Bearbeitungszeit der Probanden als eigentlich erwartet.

Auswertung des zweiten Teils der Evaluation:

Aus den Kommentaren in den Bewertungsbögen (siehe Anhang H)) und dem direkten verbalen Feedback der Probanden konnten ebenfalls Beobachtungen gezogen werden, die die Schlussfolgerungen des ersten Teils bestätigen. Wie in der Tabelle in Abbildung 19 zu sehen ist, haben die Probanden durchgehend einen guten bis sehr guten Eindruck von der Applikation mit den realisierten Konzepten.
Die Navigation und das Konzept hinter dem Aufbau der elektronischen Pinnwand wurden durchgehend als gut wahrgenommen. Einer der Teilnehmer äußerte schriftlich sogar seine Zufriedenheit mit dem geschaffenen Systembild, da es ihm dem Aufbau eines guten mentalen Modells ermöglichte.
Was bei den Bewertungen aber besonders auffällt, ist die hohe positive Resonanz bei den verschiedenen Gesten. Obwohl die Ergebnisse des ersten Teils der Evaluation zeigen, dass die Probanden mit der Gestensteuerung ihre Schwierigkeiten hatten, zeigt sich in den Bewertungen selbst, dass jeder zweite Proband die Gestensteuerung sogar als sehr gut bewertete. Zwar kommentierte jeder Proband entweder verbal oder schriftlich, dass es eher schwierig war die Gesten zu ermitteln oder dass sie sie ohne Nachfragen wohl nicht erkannt hätten, aber keiner bewertete die Bedienung selbst als schlecht.
Dies lässt die Schlussfolgerung zu, dass die Problematik bezüglich des Erlernens der Gesten von den Probanden als nicht so kritisch angesehen wird, wie man es anhand der deutlich erhöhten Bearbeitungszeit (siehe Abb. 18) hätte vermuten können. Nichts desto trotz kann man den Bewertungen, schriftlichen Kommentaren und verbalen Feedback deutlich entnehmen, dass die Nutzer ein Hilfssystem brauchen und sich wünschen.
Bezüglich der Navigation lässt sich festhalten, dass der hierarchische Aufbau der Ansichten mit den zwei Stufen des Detaillevels der Story Card gut von den Probanden akzeptiert und verstanden wurde.

Als Gesamtresultat für die Evaluation der MUST-Applikation lässt sich festhalten, dass die entwickelten Konzepte für die Überführung von Aktivitäten mit Karteikarten in das elektronische System grundsätzlich gut von den Probanden angenommen, verstanden und angewendet werden konnten. Die beiden fokussierten Aspekte Navigation und Gestensteuerung haben gutes Feedback von den Probanden erhalten.
Für die Navigation gilt dabei, dass sich der Nutzer selbst zuerst einen Überblick verschaffen muss, bevor er sich schnell und zügig in der Applikation zu Recht findet. Aber dies ist wohl bei den meisten neuen Anwendungen Fall.
Die Gestensteuerung ist für alle Probanden bei der ersten Verwendung von neuen Gesten problematisch gewesen. Es hat sich deutlich gezeigt, dass diese nur als Standardbedienung geeignet sind, wenn das System zusätzlich über die Möglichkeit verfügt den Nutzern die Gesten und deren Verwendung während der Benutzung der Applikation beizubringen. Zusätzlich ist es gut, wenn diese Informationen, zum Beispiel im Falle des Vergessens einer Geste, erneut abgerufen werden können.

Während der Durchführungen der Evaluation mit den Probanden konnte eine sehr wichtige Beobachtung bezüglich der Performance der Applikation gemacht werden, die in Abschnitt 6.2 als Ausblick für die Optimierung von MUST festgehalten wurde.

Aus den Ergebnissen der Evaluation konnten zusätzlich noch einige kleine Verbesserungen für das Design der MUST- Applikation entwickelt werden, die im Abschnitt 5.3 ausführlicher behandelt werden.

5.2.2 Diskussion der Ergebnisse

Die Beobachtungen und Schlussfolgerungen aus dem vorherigen Abschnitt 5.2.1 basieren alle auf den in der Evaluation erfassten Messdaten und den daraus berechneten Werten. Somit ist die Qualität der Ergebnisse von mehreren Faktoren abhängig, welche vielleicht nicht genug berücksichtigt werden konnten. Dadurch kann es bei der Interpretation der ermittelten Daten zu Fehler gekommen sein, die die Ergebnisse in gewisser Weise beeinflusst haben.

Im Wesentlichen gibt es drei relevante Faktoren für potentielle Fehlerquellen bei der Auswertung der Messdaten.

1. Anzahl der Probanden:
   Leider gab es nur 4 Probanden, die Zeit gefunden hatten, um an der Evaluation teilzunehmen. Dies ist nur eine sehr kleine Gruppe an Probanden, wodurch die gemessenen Daten nicht so repräsentativ sind, wie sie es bei einer größeren Gruppe gewesen wären. Dadurch ist es möglich, dass die berechneten Mittelwerte der einzelnen Aufgaben noch zu ungenau sind, um für eine repräsentative Interpretation geeignet zu sein.

2. Voreingenommener Prüfer und Auswerter:
   Bei dem Prüfer und Auswerter der Messdaten handelt es sich um ein und dieselbe Person. Es ist nicht auszuschließen, dass es bei der Auswertung der Messdaten zu einer unbeabsichtigten Fehlinterpretation gekommen ist. Dieses unbewusste Verhalten kann durchaus dazu geführt haben, dass die festgestellten Beobachtungen und die damit verbundenen Schlussfolgerungen zu stark aus nur einer bestimmten Perspektive erfolgten.

3. Problematische Zeitmessung:
   Aufgrund der Tatsache, dass der Prüfer während der Evaluationsdurchgänge mehrere Rollen zu gleichen Zeit ausfüllen musste, konnte die exakte Erfassung der Bearbeitungszeit nicht immer gewährleistet werden. Problematisch wurde es wenn mehrere Ereignisse, wie beispielsweise die Beendigung einer Aufgabe mit dem damit verbundenen Messen der Bearbeitungsdauer und das Erfassen einer Beobachtung bezüglich der Handhabung des Probanden mit Applikation, fast zeitgleich eintraten.

Trotz der drei potentiellen Faktoren für Fehlerquellen in der Evaluation, lassen sich dennoch sehr interessante Ergebnisse vorweisen. Es zeigt sich in den zuvor ausgeführten Beobachtungen und Schlussfolgerungen die deutliche Tendenz, dass die entwickelten Konzepte für die Überführung von Aktivitäten mit echten Karteikarten in die Applikation gut von den Nutzern verstanden, akzeptiert und verwendet werden können.

5.3 Maßnahmen für die MUST-Applikation

Wie am Ende des Abschnitts 5.2.1 bereits erwähnt wurde, konnten, aus dem Resultat der Evaluation, Maßnahmen zur Verbesserungen des Designs von MUST entwickelt werden. Diese werden im Folgenden erläutert und anhand von Screenshots, vor und nach der Anwendung der jeweiligen Maßnahme, verdeutlicht.

1. Maßnahme: Verbesserung der Iterationsplanung

   Der Backlog-Tab zeigt nun an, wie viele Story Points insgesamt im Backlogbereich enthalten sind, um die aktuelle Größe des Projektes wiederzugeben. Der Iteration-Tab hingegen stellt das Verhältnis der momentan in der Iteration eingeplanten Story Points

zu dem Velocity-Wert des Teams dar. Wenn der eingeplante Aufwand den Velocity-Wert überschreitet, wird die Schrift des Tabs rot eingefärbt.
Die Splitscreenansicht war für das Planen einer Iteration vorgesehen und enthielt daher diese Informationen. In der Evaluation hat aber sich gezeigt, dass einige Probanden mittels der Funktion des Drag & Drop auch über andere Ansichten die Iterationsplanung durchführten. Sie mussten öfters in die Iterationsansicht wechseln, um zu erfahren, wie viele Story Points bereits eingeplant waren. Für die Probanden verkomplizierte dies die Arbeit mit MUST. Die Verschiebung dieser Information in den Iteration-Tab stellt eine deutliche Verbesserung der Iterationsplanung dar, da es dem Nutzer nun möglich ist, aus jeder Pinnwandansicht heraus, eine Iteration planen zu können.

MUST-Oberfläche vor der Maßnahme:

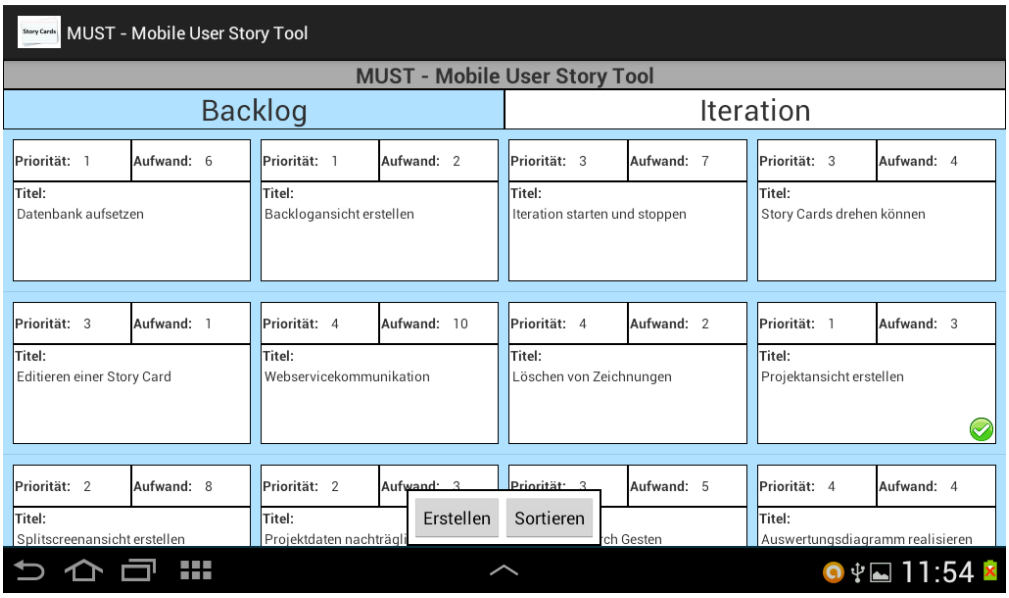


Abb. 20 - Tabs von MUST vor der Maßnahme

MUST-Oberfläche nach der Maßnahme:

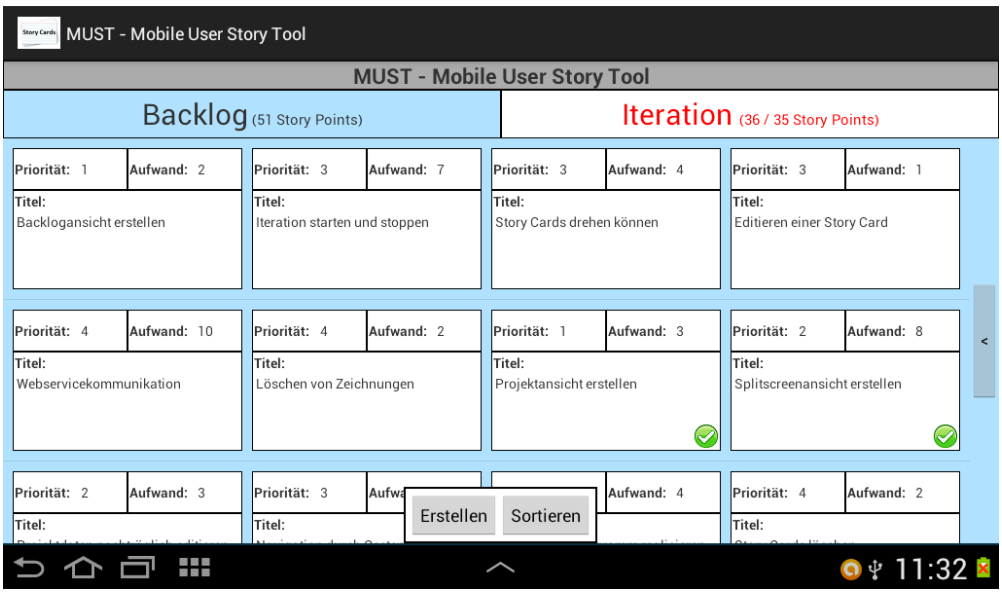


Abb. 21 - Tabs von MUST nach der Maßnahme

2. Maßnahme: Design der TextProgressBar überarbeitet

Die selbstgeschriebene TextProgressBar wurde durch das verwendete Standard-Theme als dünner Strich dargestellt, wodurch der angezeigte Text wie durchgestrichen wirkte. Für eine bessere Wahrnehmung des Fortschrittsbalkens ist ein eigener Style definiert worden. Es wurde darauf geachtet, dass sich das neue Aussehen der TextProgressBar in das restliche Design der Applikation integriert. Zusätzlich wurde der Zweck des Fortschrittsbalkens geändert. Anstatt nur den zeitlichen Verlauf einer Iteration anzuzeigen, wird jetzt die Menge der erledigten Story Points einer Iteration dargestellt, da diese Information ist für den Nutzer während der Iteration relevanter ist.

MUST-Oberfläche vor der Maßnahme:

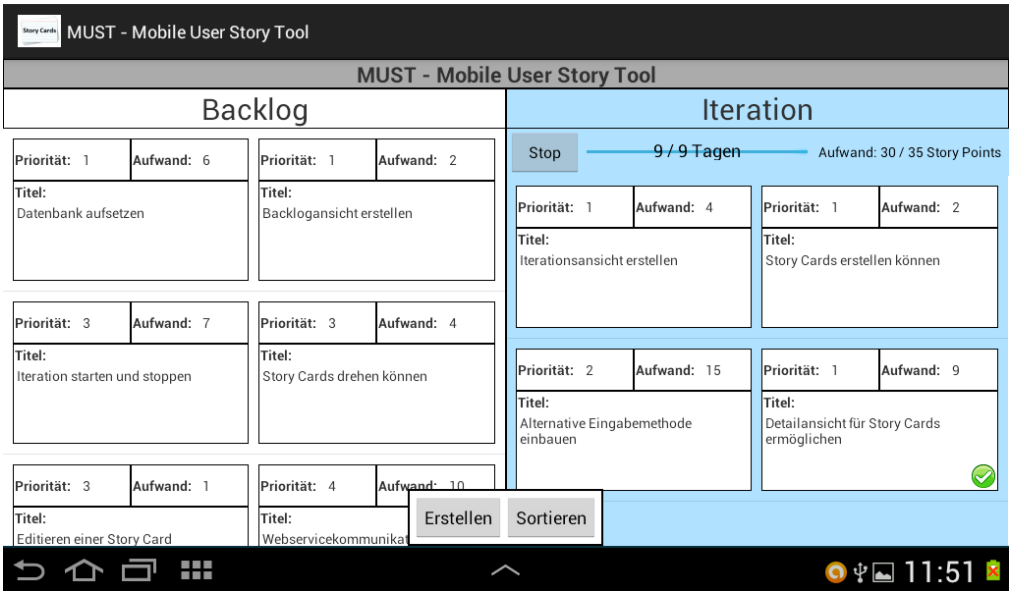


Abb. 22 - Design des Fortschrittsbalkens vor der Maßnahme

MUST-Oberfläche nach der Maßnahme:

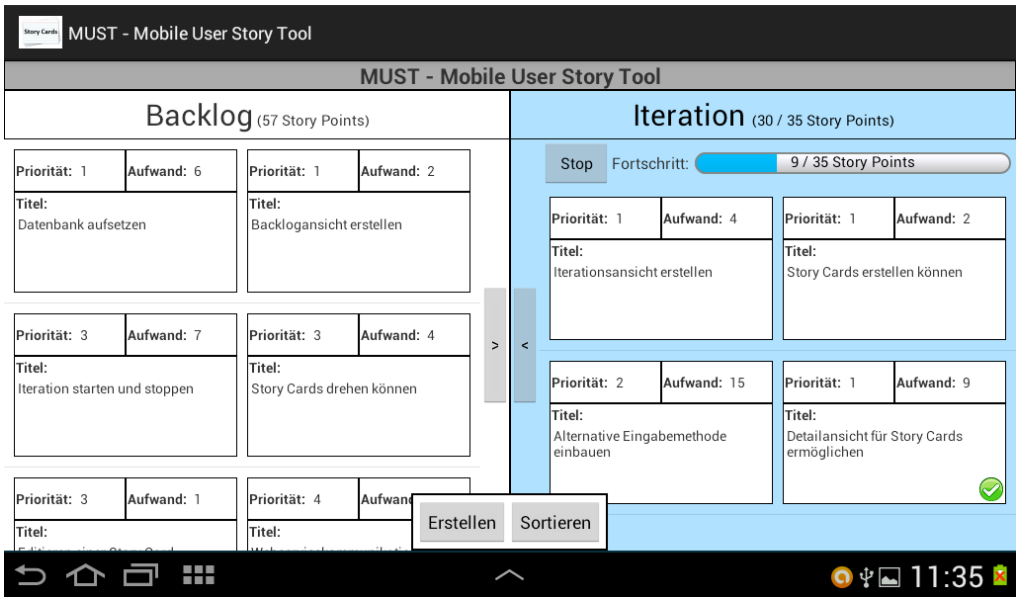


Abb. 23 - Design des Fortschrittsbalkens nach der Maßnahme

3. Maßnahme: Ersatz für das spätere Hilfssystem bezüglich Gestensteuerung

   Die Erlernbarkeit der Gesten geht über den für die Entwicklung abgesteckten Bereich hinaus, weshalb nur eine temporäre Lösung angeboten werden kann, um sicherzustellen, dass ein Nutzer alle Funktionen der Applikation erreichen kann.
   Alle Gesten wurden zusätzlich durch Buttons in die Applikation integriert. Durch zwei Buttons kann zwischen den drei Pinnwandansicht gewechselt werden (siehe Abb. 24 und Abb. 25). Für die Gesten zum Drehen und Erledigen einer Story Card gibt es im Menü der Detailansichten zwei weitere Buttons (siehe Abb. 26 und Abb. 27).
   Diese Zwischenlösung bietet keinen adäquaten Ersatz für das Hilfssystem, denn durch die Buttons kann Nutzer die Gestensteuerung nicht erlernen. Sobald dieses System in die Anwendung integriert wurde, muss diese Maßnahme rückgängig gemacht werden.

   MUST-Oberfläche vor der Maßnahme:

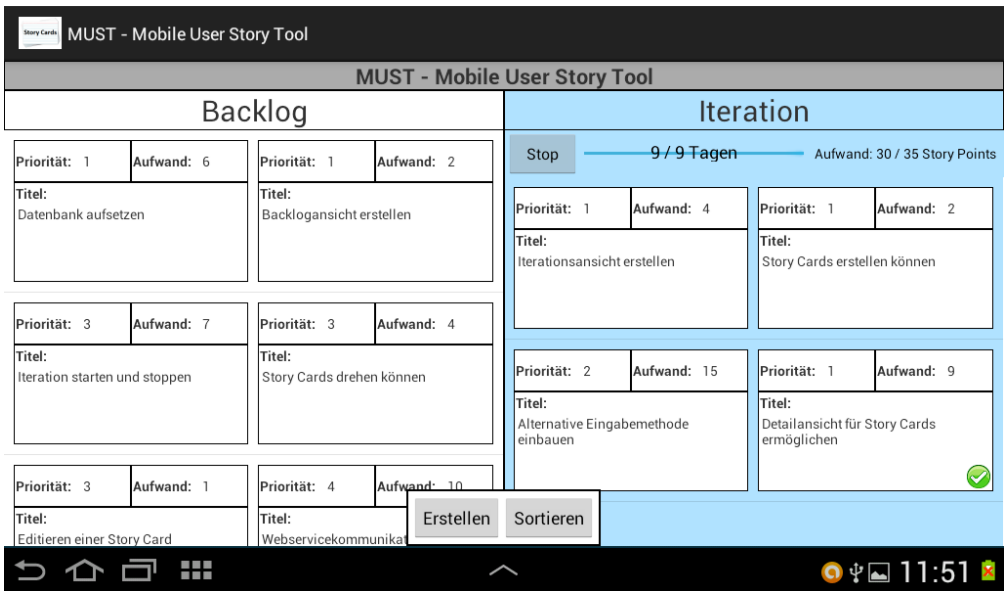


Abb. 24 - Oberfläche von MUST vor der Maßnahme für das Wechseln der Ansichten

MUST-Oberfläche nach der Maßnahme:

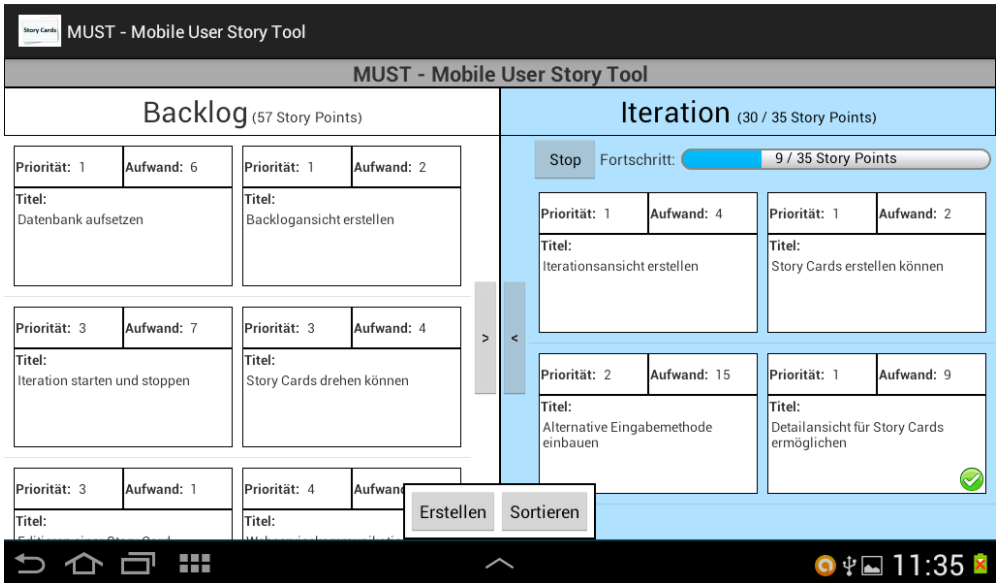


Abb. 25 - Oberfläche von MUST nach der Maßnahme für das Wechseln der Ansichten

MUST-Oberfläche vor der Maßnahme:

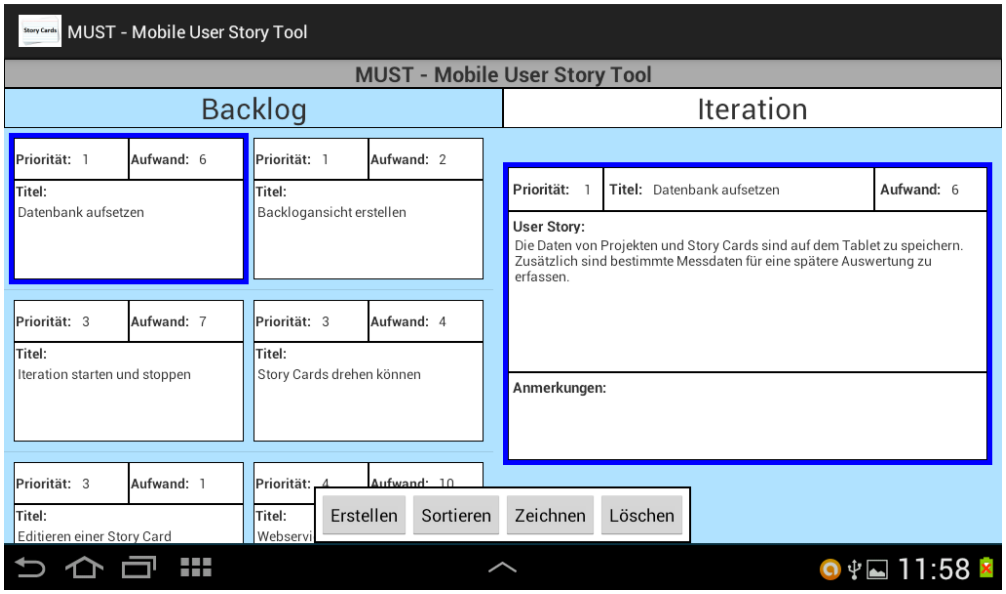


Abb. 26 - Oberfläche von MUST vor der Maßnahme für das Drehen und Erledigen

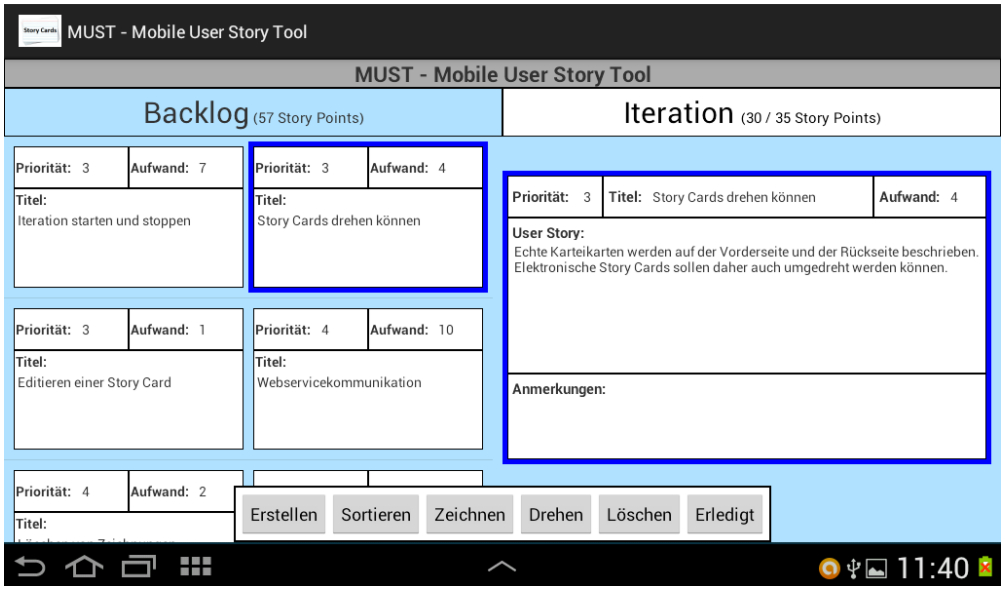


Abb. 27 - Oberfläche von MUST nach der Maßnahme für das Drehen und Erledigen

6 Fazit und Ausblick

In diesem letzten Kapitel wird zuerst die gesamte Arbeit zu dem Thema „*Unterstützung des User Story Management mit einer mobilen Applikation*“ noch einmal kurz resümiert, um dann ein abschließendes Urteil bezüglich des Erfolges der ausgearbeiteten Konzepte festzuhalten. Da noch nicht alle erhobenen Anforderungen und Ideen in der Android-Applikation realisiert wurden, wird im letzten Teil dieser Arbeit ein Ausblick auf mögliche zukünftige Aspekte der MUST-Applikation gegeben. Diese befassen sich mit der Verbesserung und dem Ausbau der aktuellen MUST-Version.

6.1 Fazit

Das Ziel dieser Arbeit ist es gewesen Konzepte für eine Android-Applikation für Tablets zu entwickeln, durch die das User Story Management unterstützt wird. Dafür wurden die relevanten Eigenschaften von echten Karteikarten in Bezug auf die Anwendung der Methode der User Stories ausgearbeitet. Dazu gehörte ebenfalls die Betrachtung der Vor- und Nachteile von User Stories sowie der einzelnen Prozesse in der agilen Softwareentwicklung.
Auf Basis der gesammelten Erkenntnisse konnten Anforderungen an die Applikation erhoben werden, für die wiederrum entsprechende Konzepte zur adäquaten Überführung der Eigenschaften, Aktivitäten und Vorteile von und mit User Stories beziehungsweise Story Cards in die MUST-Anwendung ausgearbeitet werden konnten.
Mit Hilfe der Integration dieser Konzepte und Aspekte in die Applikation und dem geplanten Webservice vom Fachgebiet Softwareengineering, ist es möglich, die in der Praxis durchgeführte physische und ortsgebundene Arbeit der agilen Softwareentwicklung mit User Stories auf eine elektronische und globale Ebene zu überführen.
Dadurch wird jedem Entwicklerteam, sei es lokal oder global organisiert, ein Werkzeug zur Verwaltung und Entwicklung geboten, das ihnen die Arbeit mit User Stories auf eine realitätsnahe Weise erlaubt.
Die Unterstützung für die Entwickler ergibt sich bezüglich zweier Aspekte. Zum einen wurde bei den entwickelten Konzepten darauf geachtet, dass die wichtigen Aktivitäten bei der Entwicklung mit User Stories durch die MUST-Applikation ermöglicht werden. Dazu gehören unter anderem das Planen einer Iteration, das Anpassen der Projektdaten aber auch die Möglichkeit des Zeichnens auf den Story Cards. Zum anderen bietet die Applikation eine Unterstützung für die Entwickler, indem sie bestimmte Aufgaben, wie das Erstellen des kumulativen Soll-Ist-Diagrammes, für den Nutzer automatisch ausführt. Weiterhin werden an einigen Stellen wichtige Daten visuell dargestellt, wie beispielsweise der Fortschrittsbalken des erledigten Aufwandes der aktuellen Iteration.

Die Ergebnisse der durchgeführten Evaluation im Abschnitt 5.2 weisen die deutliche Tendenz auf, dass die realisierten Konzepte gut von den Probanden verstanden, angenommen und angewendet werden konnten. Dies lässt den Schluss zu, dass die Konzepte zur Überführung der Eigenschaften und Aktivitäten mit User Stories, welche in der MUST-Applikation realisiert wurden, von den Nutzer akzeptiert werden und somit ein effizientes Arbeiten mit den elektronischen Story Cards ermöglichen.
Dabei ist besonders zu erwähnen, dass die Konzepte des Drag & Drop und der Gestensteuerung, welche speziell die Bedienung des Tablets mit den Fingern ausnutzen, gut von den Nutzern akzeptiert wurden. Dieser Aspekt der Bedienung war ein wesentlicher Grund, weshalb es sich bei der Software gerade um eine mobile Applikation handeln soll (siehe Abschnitt 1.1). Allerdings muss man im Hinterkopf behalten, dass es gewisse Fehlerquellen für die Ergebnisse der Evaluation gibt, die im Abschnitt 5.2.2 ausführlich erörtert wurden.

Desweiteren zeigten sich Schwierigkeiten bei den Probanden, wenn sie zum ersten Mal Gesten verwendeten (siehe Abschnitt 5.2.1). In der Evaluation konnte ebenfalls ein Performanceproblem ermittelt werden auf das im Abschnitt 6.2 genauer eingegangen wird.
Leider ist die im Szenario (siehe Abschnitt 4.2) beschriebene Kommunikation zwischen MUST und dem Webservice noch nicht möglich, da der realisierte Funktionsumfang von MUST eine komplexere Datenverwaltung erfordert, über welche der bisher geplante Webservice aber noch nicht verfügt. Aus diesem Grund konnten die benötigten Befehle für die Kommunikation auch noch nicht in MUST integriert und getestet werden. Dennoch wurden im Anhang dieser Arbeit alle Stellen im Quellcode der MUST-Applikation erfasst, an denen die Kommunikation ergänzt werden muss.

Insgesamt lässt sich festhalten, dass die entwickelten Konzepte das User Story Management auf einer mit der Realität vergleichbaren Art und Weise ermöglichen und dabei den Nutzer in vieler Hinsicht unterstützen.
Als Konsequenz der Realisierung der Konzepte ist die Android-Applikation „*Mobile User Story Tool“* für Tablets entstanden, mit der die verschiedenen Aktivitäten der agilen Softwareentwicklung mit User Stories effizienter durchgeführt werden können.

6.2 Ausblick

Der letzte Abschnitt dieser Arbeit befasst sich mit möglichen Aspekten, wie die aktuelle MUST-Version optimiert und erweitert werden kann. Dazu gehört neben der Verbesserung der bestehenden Probleme, bezüglich des Erlernens der Gesten und Performance, auch die Betrachtung weiterer nicht realisierter Anforderungen für ein effizienteres Arbeiten mit der Applikation.

Wie genau das bestehende Problem, für das Erlernen der Gesten, gelöst werden kann, wurde bereits in dem Abschnitt 5.2.1 erläutert. Daher wird hier nicht noch einmal im Detail auf dieses Hilfssystem eingegangen.

Im Folgenden wird das im Abschnitt 5.2.1 erwähnte Performance-Problem betrachtet. Während der Arbeit der Probanden mit der MUST-Applikation zeigte sich eine mit zunehmender Verwendungszeit abnehmende Performance in Bezug auf die Aktualisierungen der Oberfläche nach Eingaben. Das verwendete Tablet war mit einem Laptop verbunden, welcher alle Systemausgaben erfasste, um etwaige Fehlermeldungen anzuzeigen.
Mit Hilfe der erfassten Ausgaben konnte ermittelt werden, dass durch den häufigeren Wechsel der Ansichten, der Heap der Applikation immer stärker beschrieben wurde.
Der Grund für dieses Verhalten konnte durch eine Recherche im Internet ermittelt werden. Die Darstellung der Pinnwandansichten beruht auf einem bestimmten Android-Element, dem sogenannten ViewFlipper. Dieser ist verwendet worden, da er den animierten Austausch von gewissen Teilen einer Ansicht erlaubt. Diese Eigenschaft wurde für das Verschieben der Ansichten sowie bei dem Drehen der detaillierten Story Card genutzt.
Bei der Auswahl des Android-Elementes ist nicht bekannt gewesen, dass der ViewFlipper beim Verändern seiner enthaltenen Ansicht bestimmte referenzierte Daten in den Heap der Applikation schreibt [17]. Durch einen häufigen Wechsel der Ansichten resultiert eine starke Belastung des Heaps, was sich wiederum deutlich auf die Performance der MUST-Applikation auswirkt.
Die weitere Recherche zeigte auf, dass der ViewFlipper durch einen sogenannten ViewPager, welcher keine Daten im Heap abspeichert, auszutauschen ist.
Da der ViewFlipper aber zentral in Darstellung der Pinnwandansichten integriert ist, muss der Großteil der bestehenden Implementierung umstrukturiert und überarbeitet werden, was eine intensive Überarbeitung der kompletten Pinnwandansichten mit sich führt. Aus diesem Grund

kann die Korrektur dieses Problems nur ein Ausblick für die weitere Entwicklung von MUST seien. Durch die Optimierung der Performance kann ein flüssigerer Arbeitsverlauf für den Nutzer sichergestellt werden kann.

Im Folgenden werden noch verschiedene Aspekte betrachtet, durch die entweder bereits umgesetzte Konzepte weiter verbessert oder noch nicht umgesetzte Anforderungen an die Applikation realisiert werden können.

Bezüglich der Anforderungen **[R7] Einsehbare Diagramme auf Basis von erstellten Logs** und **[R16] Logs für erledigte Karten und Iterationen verwalten** gibt es noch deutliches Verbesserungspotential. Momentan verfügt MUST nur über ein nicht einsehbares Log der beendeten Iterationen, durch das das kumulative Soll-Ist Diagramm erstellt wird.
Da ein Log im Allgemeinen eine gewisse Art der Dokumentation darstellt, soll durch eine Analyse aller zur Verfügung stehenden Informationen ein genauer Entwurf für das Design des Logs erstellt werden. Zusätzlich ist es für die Nutzer von Vorteil, wenn sie das Log auch im Detail betrachten können, um ihre Arbeit selbst besser analysieren und planen zu können. Durch den detaillierten Entwurf des Logs ist es auch möglich den Nutzern weitere Auswertungsdiagramme durch die Anwendung generieren zu lassen. Dies stellt eine deutliche Verbesserung der Unterstützung des Nutzers durch die Applikation dar.

Die Anforderungen **[R8] Möglichkeit der zusätzlichen Dokumentation inklusive erleichtertem Informationsaustausch** und **[R14] Testaspekte bezüglich Erfolg / Misserfolg / ungeprüft markieren können** sind beide noch gar nicht in der momentanen Version von MUST berücksichtigt worden. Die Gemeinsamkeit dieser beiden Anforderungen besteht in dem Ziel der besseren Dokumentation des Projektes.
Bei der Anforderung **[R8]** geht es darum, dass die Dokumentation des Projektes und der Austausch von wichtigen Informationen auch über die Applikation ermöglicht werden. Eine mögliche Realisierung dieser Anforderung ist eine Art von Notiz-Log. In diesem Log können die Nutzer wichtige Entscheidungen und Notizen festhalten, die, in Kombination mit dem Log der Messdaten, eine gewisse Dokumentation jeder Iteration zu verfügen stellen, mit der bei Bedarf eine ausführlichere Dokumentation erstellen werden kann.
In der Anforderung **[R14]** wird auch ein Teil der Dokumentation fokussiert. Dabei geht es speziell um das Erfassen von den Abnahmeergebnissen der Testaspekte einer Funktionalität, die auf der Rückseite einer Story Cards festgehalten sind. Damit die Beteiligten besser erkennen können, ob ein Testaspekt geprüft wurde und bestanden ist oder nicht, bietet es sich an, dass die erstellten Testaspekte entsprechend markiert werden können.
Beide Anforderungen tragen zu der Unterstützung der Arbeit und Verwaltung von Story Cards bei, indem sie eine Basis für die Ausarbeitung einer detaillierten Dokumentation bilden.

Die aber wohl wichtigste noch nicht realisierte Anforderung ist **[R4] Sicherstellung der Synchronisation zwischen Web-Service und Applikation**. Wie im vorherigen Fazit (siehe Abschnitt 6.1) bereits erläutert wurde, konnte die im Szenario (siehe Abschnitt 4.2) beschriebene Kommunikation zwischen dem Webservice und MUST noch nicht in der Applikation ermöglicht werden. Die Integration dieser Anforderung macht aber einen wesentlichen Nutzen und Vorteil der Applikation aus, weil ein effizienter Einsatz von MUST durch ein gesamtes Team nur dann erfolgen kann, wenn die Applikation mit einem zentralen und global erreichbaren Speicher für die Story Cards kommuniziert. Sobald der Webservice existiert, stellt die Anpassung von MUST für die Kommunikation mit diesem keine große Herausforderung mehr dar, weil bereits alle nötigen Stellen im Quellcode für den Einbau der entsprechenden Befehle ermittelt wurden. Damit steht der Realisierung der Vision der Verwendung von MUST, im Zusammenspiel mit dem Webservice, kein großes Hindernis im Weg.

Anhang

A)Ansichten von MUST

Dieser Teil des Anhanges enthält beispielhafte Grafiken für die Ansichten in MUST. Der genaue Inhalt ist dabei nicht relevant, stattdessen dienen die Grafiken vielmehr einer Übersicht über die Strukturen der verschiedenen Ansichten.

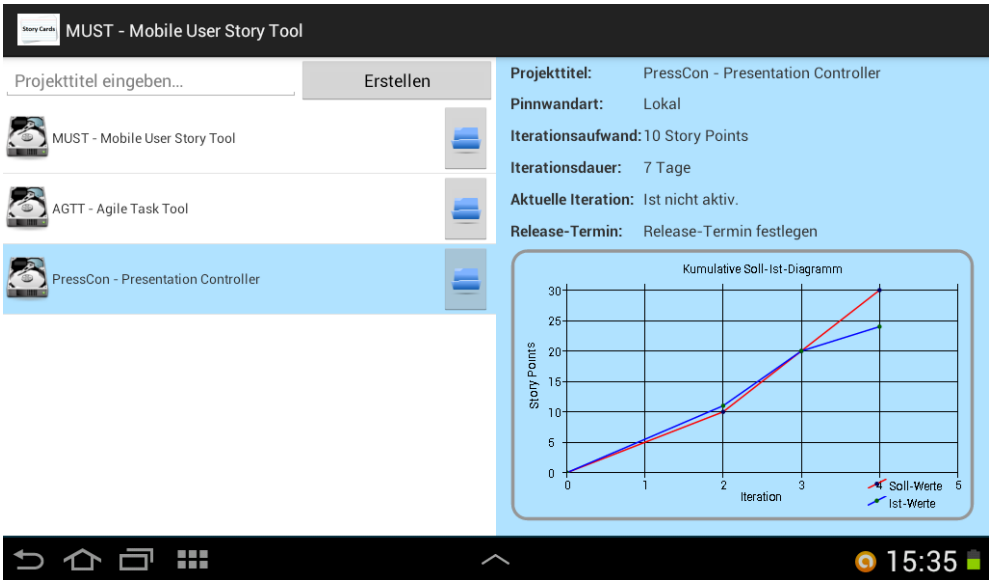


Abb. 28 - Projektansicht von MUST

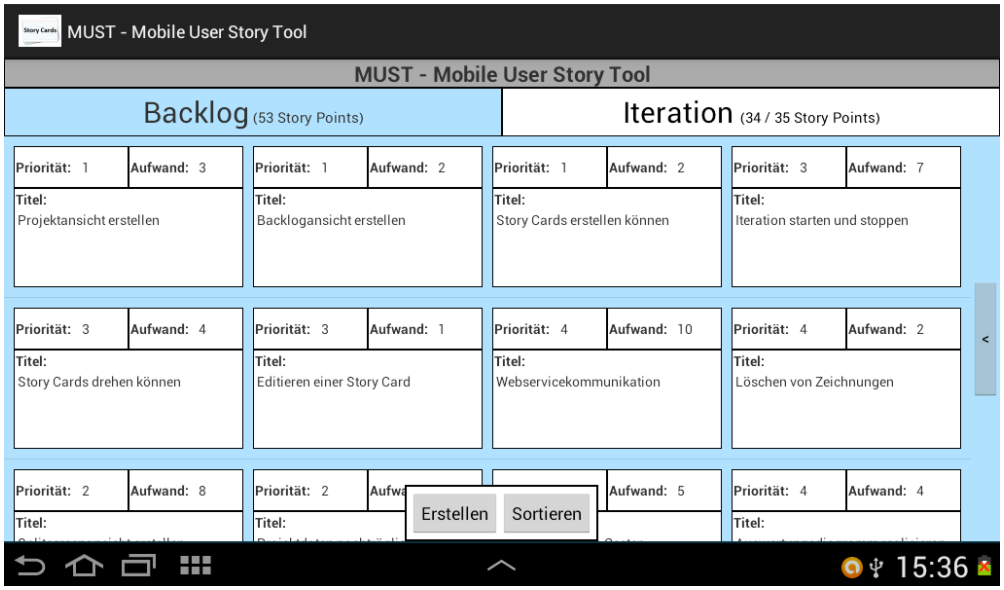


Abb. 29 - Vollständige Backlogansicht von MUST

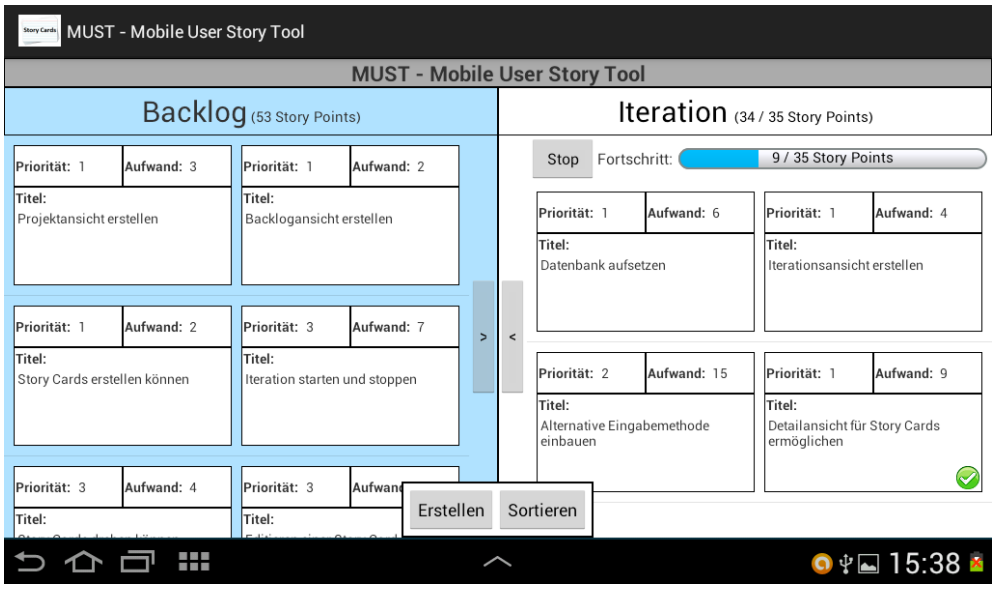


Abb. 30 - Splitscreenansicht von MUST

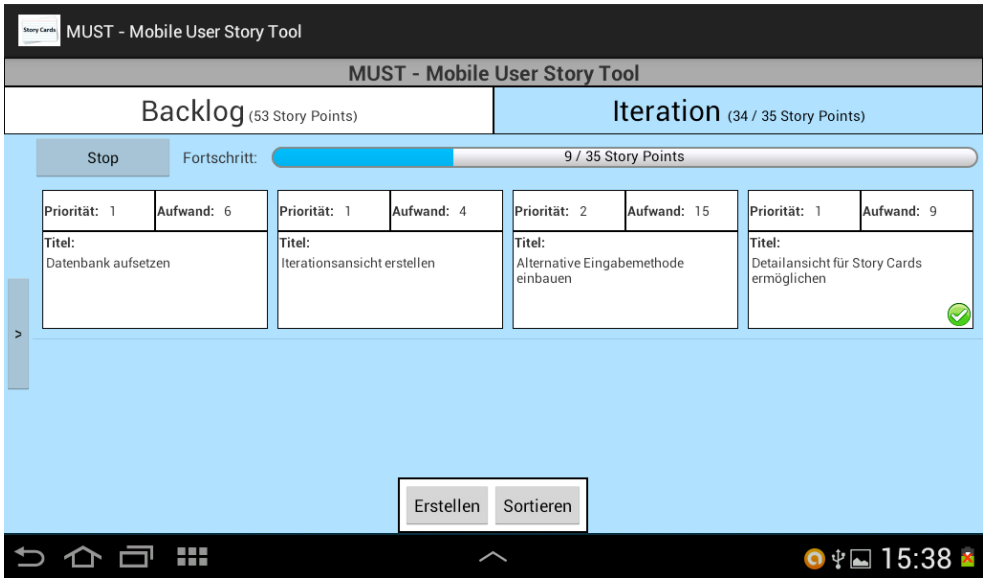


Abb. 31 - Vollständige Iterationsansicht von MUST

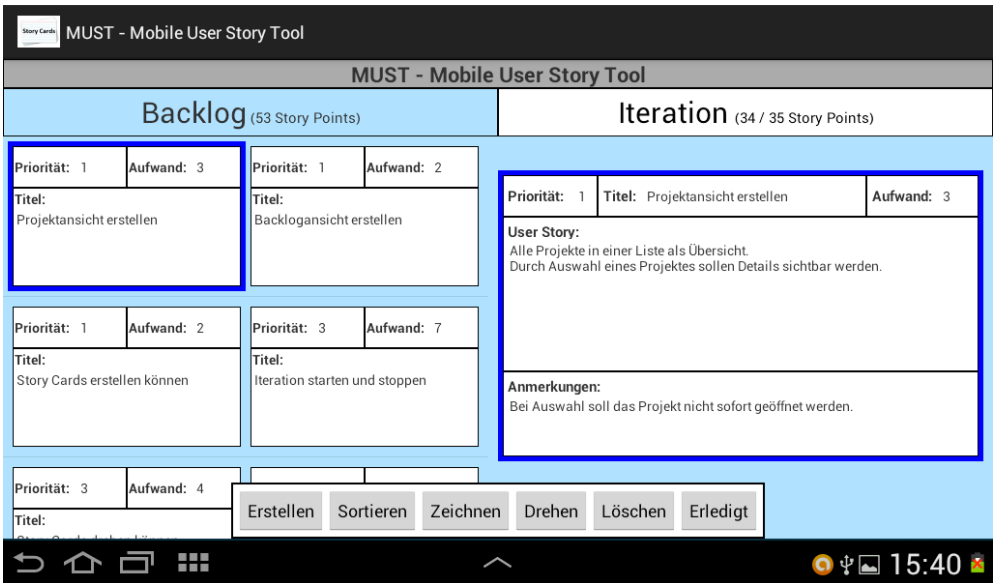


Abb. 32 - Backlogdetailansicht von MUST

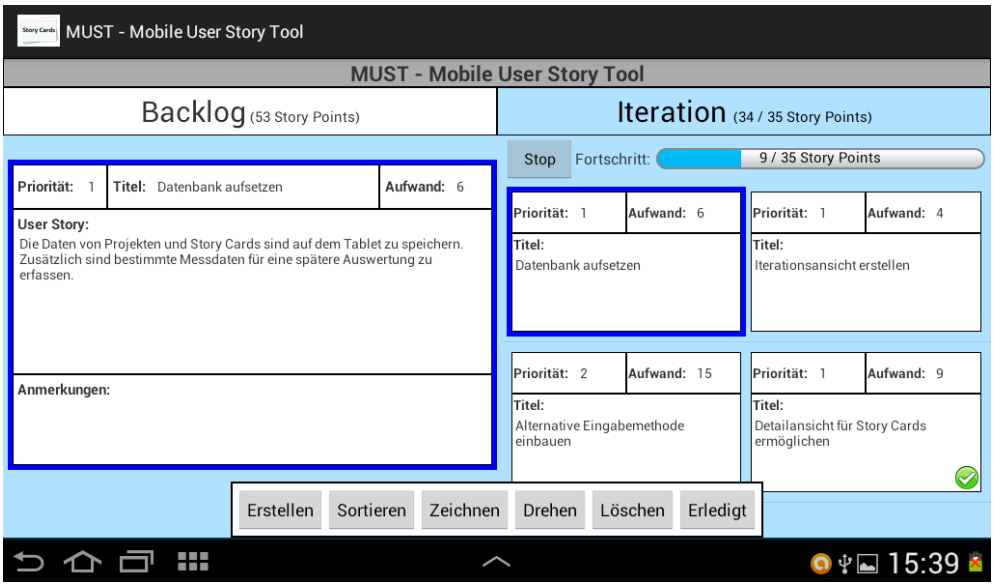


Abb. 33 - Iterationdetailansicht von MUST

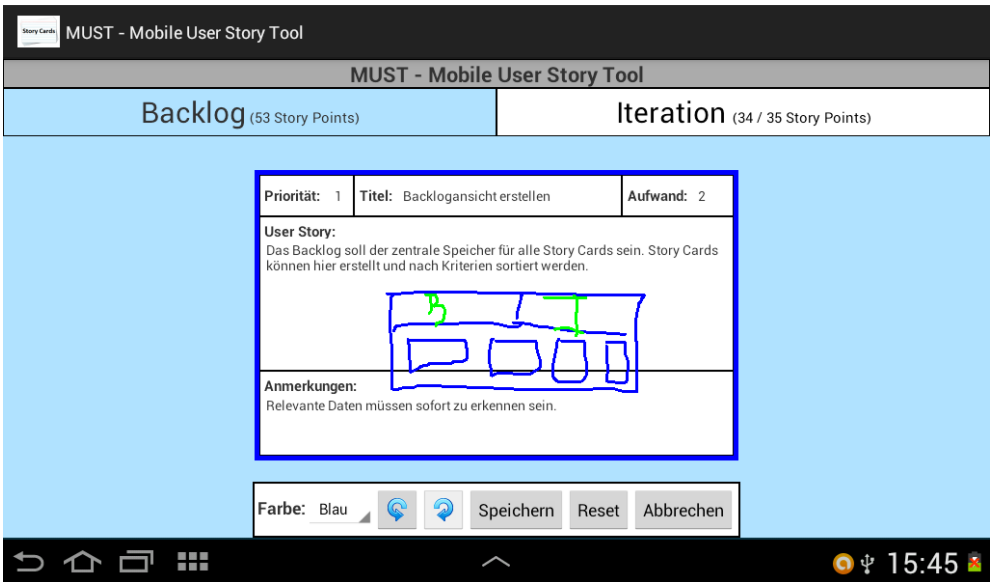


Abb. 34 - Zeichenansicht von MUST

Abb. 35 - Erstell- / Editieransicht von MUST

B)Navigationsstrukturen in MUST

Die beiden nachfolgenden Abbildungen beschreiben die Navigationsstruktur in der entwickelten Applikation. Die Rückkehrreihenfolge (Abb. 36) zeigt dabei an, in welche Ansicht man aus einer bestimmten heraus zurückkehren kann. Und die Aufruf-Reihenfolge (Abb. 37) gibt an, welche Ansicht von einer bestimmten aus gestartet werden kann.

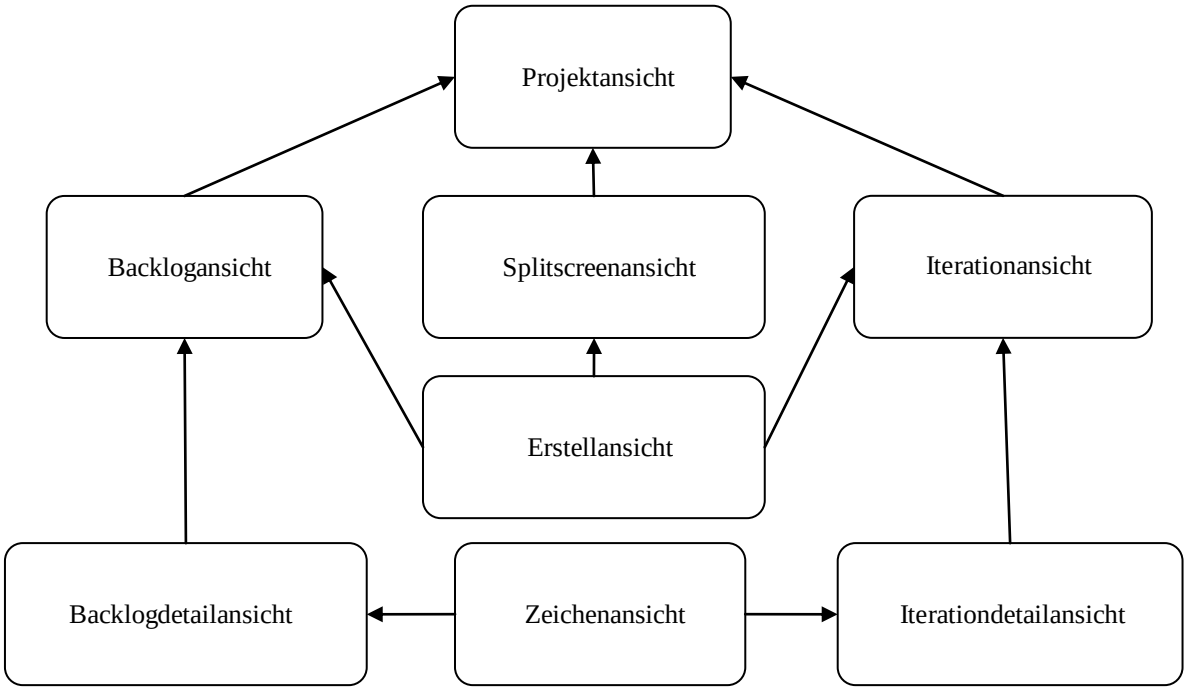


Abb. 36 - Rückkehr-Reihenfolge der MUST-Ansichten

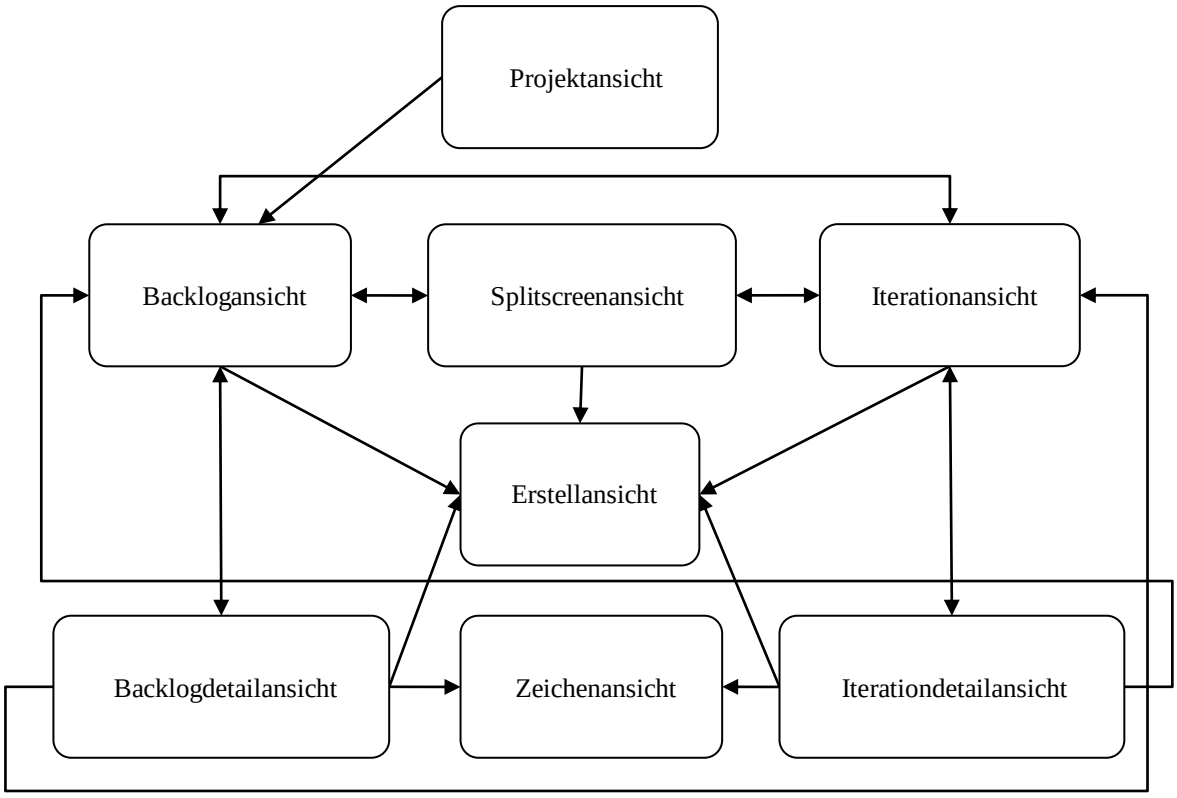


Abb. 37 - Aufruf-Reihenfolge der MUST-Ansichten

C)Beispiel für die Anwendung der Grundfunktionalitäten

Für das Beispielprojekt „AGTT – Agile Task Tool“ wird eine neue Story Card bezüglich der Funktionalität des Erstellens von Aufgabenlisten benötigt. Diese Story Card wird dabei den Zyklus vom Erstellen über das Editieren und Zeichnen bis hin zum Löschen durchlaufen. Die Bedienung der einzelnen Funktionen ist in allen Ansichten, in denen sie jeweils zur Verfügung stehen, identisch aufgebaut, dadurch ist das gewählte Beispiel unabhängig von der gewählten Startansicht.

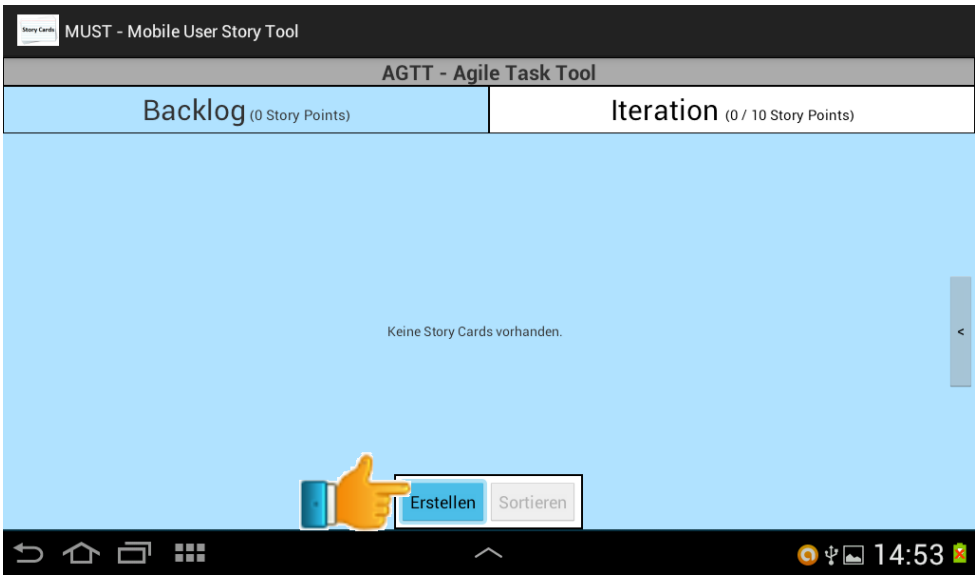


Abb. 38 - Starten des Erstellens der Story Card

Wie in Abbildung 38 zu sehen ist, befindet man sich in diesem Beispiel in der vollständigen Backlogansicht der MUST-Applikation des Projektes „AGTT – Agile Task Tool“. Zum Erstellen einer neuen Story Card wird auf den Button „Erstellen“ im Menü am unteren Bildrand geklickt, wodurch sich die Ansicht für das Erstellen einer Story Card öffnet. Hier ist können nun die weiteren Daten bezüglich der neuen Funktionalität in der Story Card eingetragen werden.

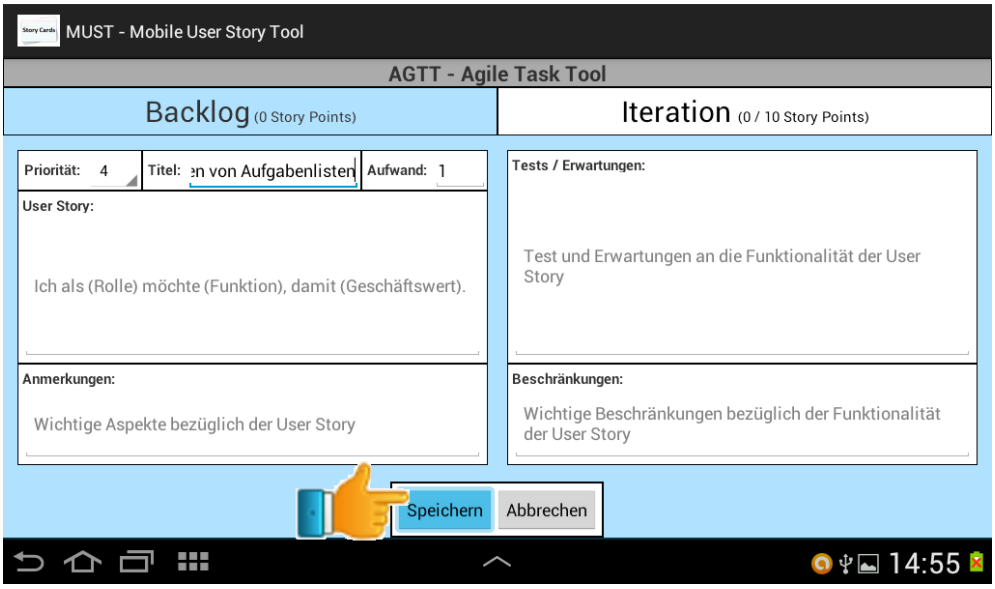


Abb. 39 - Speichern der erstellten Story Card

Es ist anzumerken, dass jede Story Card als Standardwerte die niedrigste Priorität 4 und den Mindestaufwand von 1 Story Point hat. Der Grund dafür liegt darin, dass jede Story Card immer eine Priorität haben soll auch mindestens einen Aufwand von 1 Story Point verursacht. Nur der Titel der Story Card ist als Pflichtfeld angelegt, damit die Story Card später identifiziert werden kann. Alle anderen Datenfelder sind optional und können leer bleiben. Nachdem der Titel „Erstellen von Aufgabenlisten“ eingetragen ist (siehe Abb. 39) wird auf den Button „Speichern“ gedrückt und man gelangt zurück in die Backlogansicht, welche jetzt die neu erstellte Story Card enthält (siehe Abb. 40).

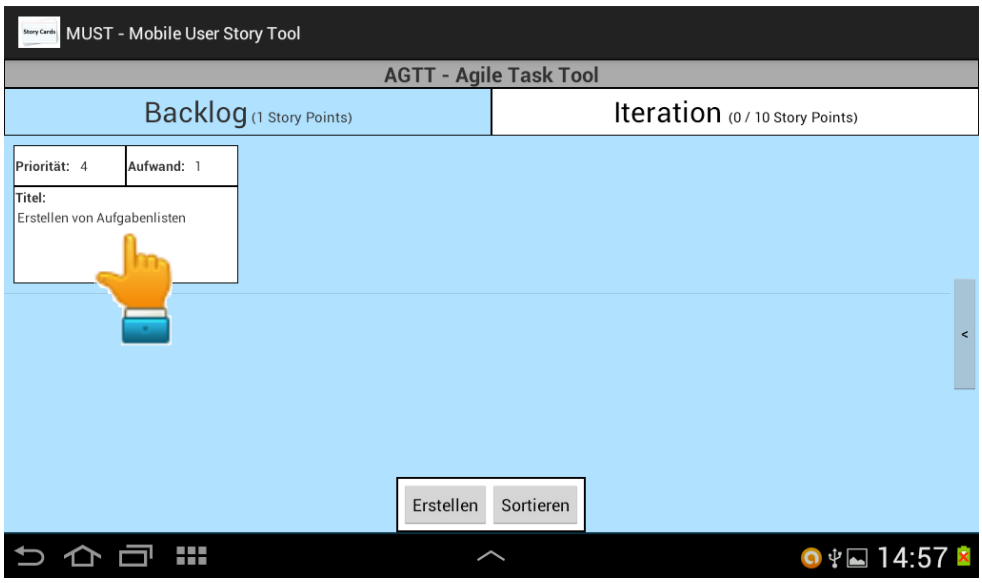


Abb. 40 - Öffnen der Backlogdetailansicht

Nun soll die soeben angelegte Story Card nachträglich editiert werden. Dazu wird auf die Story Card geklickt, um in die Backlogdetailansicht zu gelangen (siehe Abb. 40). Hier muss nochmal auf die große detaillierte Story Card rechts geklickt werden (siehe Abb. 41), wodurch man wieder in die Erstellen-Ansicht gelangt, welche auch für das Editieren genutzt wird.

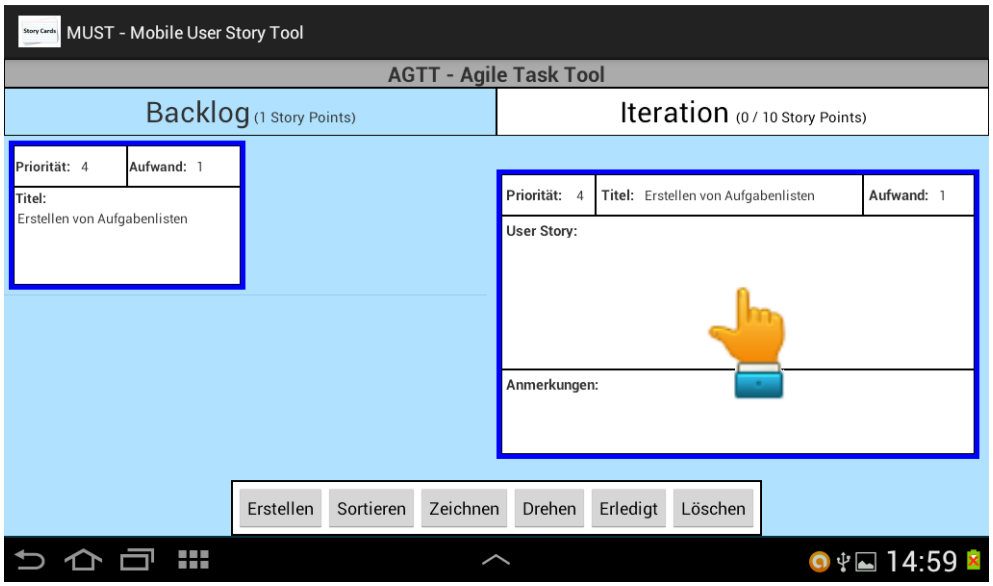


Abb. 41 - Editieren der Story Card starten

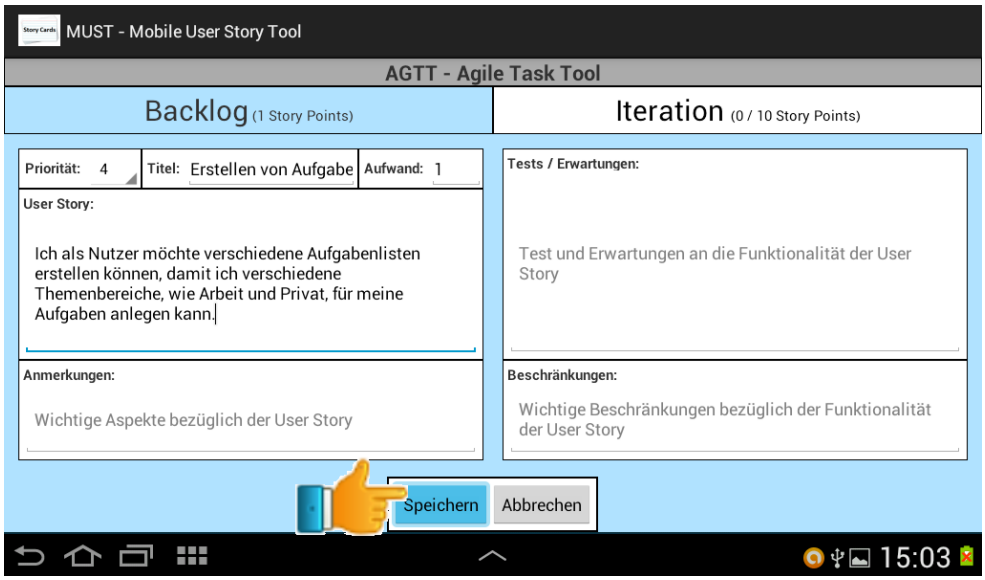


Abb. 42 - Speichern der editierten Story Card

Hier wird als nächstes die User Story „Ich als Nutzer möchte verschiedene Aufgabenlisten erstellen können, damit ich verschiedene Themenbereiche, wie Arbeit und Privat, für meine Aufgaben anlegen kann.“ in das entsprechende Feld eingetragen und die Story Card wird gespeichert (siehe Abb. 42). Dadurch gelangt man zurück in die Backlogdetailansicht, die die bearbeitete Story Card mit der eingefügten User Story im Detail anzeigt (siehe Abb. 43).
Als nächstes soll eine Zeichnung zu der Story Card hinzugefügt werden. Dazu wird der Button „Zeichnen“ gedrückt, was die Zeichenansicht für die ausgewählte Story Card öffnet.

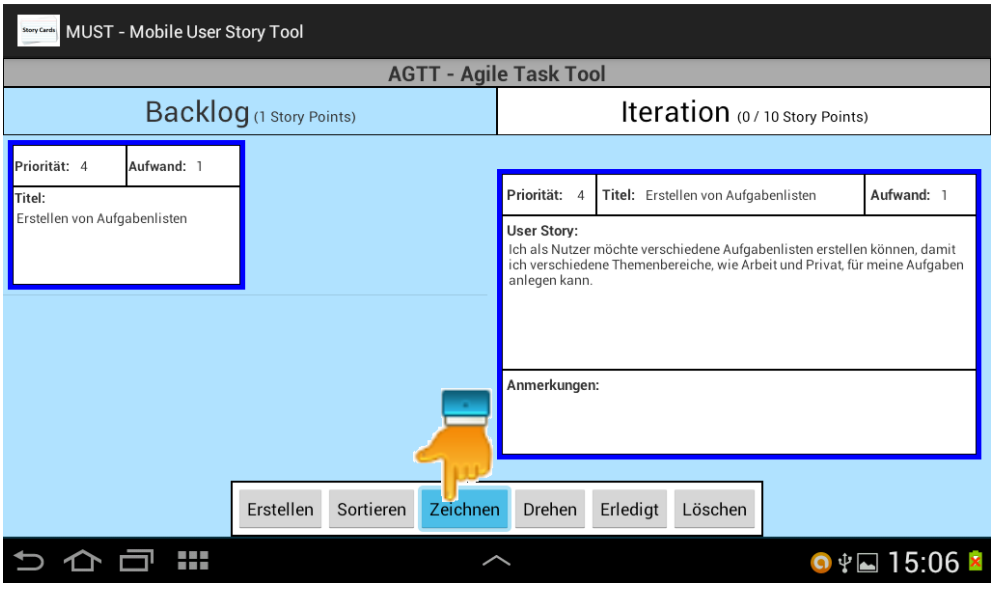


Abb. 43 - Starten des Zeichnens auf der Story Card

In dieser Ansicht ist es möglich auf der Vorderseite der Story Card eine Zeichnung zu erstellen und diese zu speichern (siehe Abb. 44). Danach verlässt man die Zeichenansicht durch den Zurück-Button des Tablets oder den Button „Abbrechen“ und gelangt wieder in die Backlogdetailansicht, in der die detaillierte Story Card mit der angefertigten Zeichnung dargestellt wird (siehe Abb. 45).

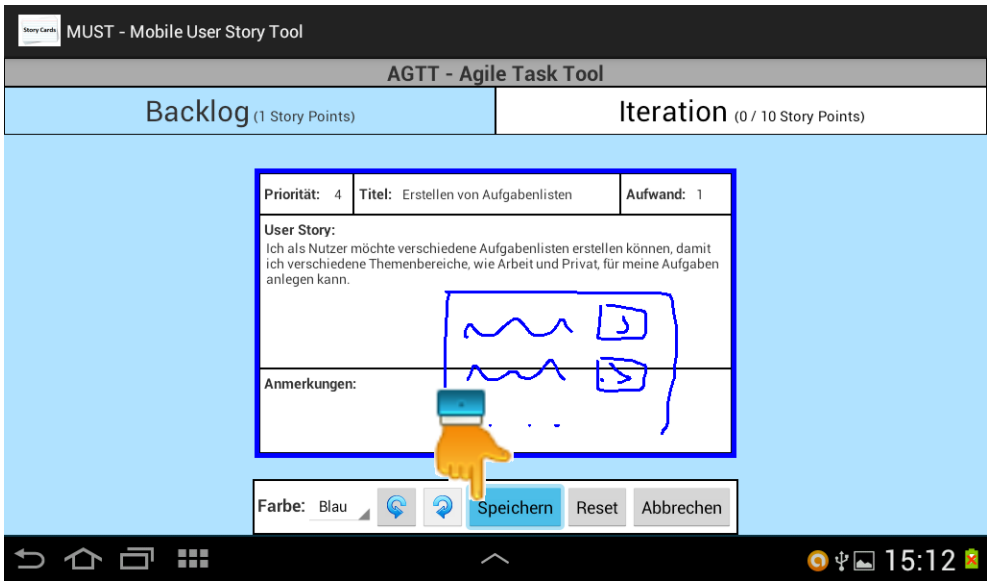


Abb. 44 - Speichern der erstellten Zeichnung

Als letztes startet durch den Klick auf den Button „Löschen“ ein Dialog, der den Nutzer zuerst fragt, was er löschen möchte. Es besteht die Möglichkeit entweder nur die vorhandene Zeichnung der in der Backlogdetailansicht ausgewählten Story Card zu löschen. Oder es kann die gesamte Story Card einschließlich einer vorhandenen Zeichnung gelöscht werden.

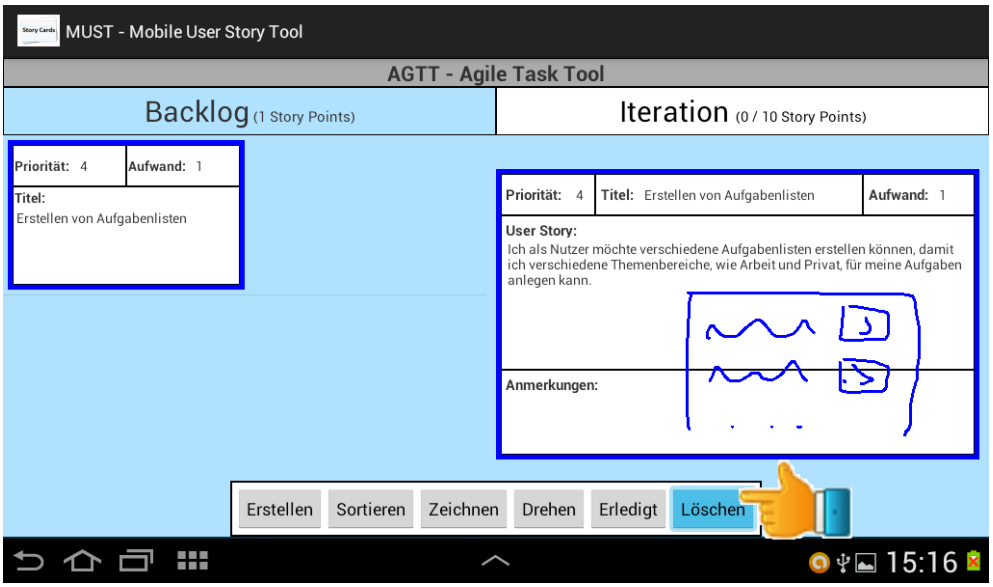


Abb. 45 - Starten des Löschens der Story Card

In diesem Beispiel soll die gesamte Story Card gelöscht werden, weshalb der zweite Punkt im Dialog gewählt wird. Als Folge dessen startet ein weiterer Dialog, der den Nutzer noch einmal explizit fragt, ob er die Story Card wirklich löschen will (siehe Abb. 46). Dies ist eine Vorsichtmaßnahme, da gelöschte Story Cards und Zeichnungen nicht wiederhergestellt werden können.
Mit der Bestätigung durch den Klick auf „Ja“ wird die Story Card und die Zeichnung gelöscht und man gelangt aufgrund des Spezialfalles, dass die Backlogdetailansicht nach dem Löschen leer ist und somit keine Story Card mehr im Detail anzeigen kann, zurück in die leere vollständige Backlogansicht des Projektes „AGTT – Agile Task Tool“ (siehe Abb. 38).

Abb. 46 - Bestätigung des Löschens der Story Card

D)Quellcodestellen für Webservice Kommunikation

Nachfolgend sind alle Zeilen in den verschiedenen Java-Klassen aufgeführt, in denen die nötige Kommunikation mit dem Webservice ergänzt werden muss.

ProjectActivity.java:
Zeile 120: Abrufen der Projektdaten vom Webservice
Anmerkung:
Wenn mehrere Personen versuchen ein Projekt oder eine Story Card zur gleichen Zeit zu editieren, besteht die Gefahr von Inkonsistenz in den Daten. Eine einfache Lösung, Um dies zu verhindern, ist das entsprechende Objekt beim Start des Editiermodus zu blockieren, sodass nur eine Person zu einem Zeitpunkt die Daten des Objektes manipulieren kann. Beim Verlassen des Editiermodus kann das Objekt dann wieder für alle Nutzer zum Editieren freigegeben werden.

ProjectDetailDialog.java:
Zeile 203: Senden der editierten Projektdaten an den Webservice

DrawingActivity.java:
Zeile 324: Senden der erstellten Zeichnung an den Webservice
Anmerkung: Es ist momentan nicht spezifiziert, ob der Webservice auch die Zeichnungen verwalten soll. Und es muss dazu noch geklärt werden, ob die Zeichnungen nachträglich editiert werden können. Die aktuelle MUST Version verfügt nicht über diese Option.

DragListener.java:
Zeile 72: Senden des neuen Bereiches der verschobenen Story Card an den Webservice
Zeile 87: Senden des neuen Bereiches der verschobenen Story Card an den Webservice
Zeile 102: Senden des neuen Bereiches der verschobenen Story Card an den Webservice
Zeile 117: Senden des neuen Bereiches der verschobenen Story Card an den Webservice
Zeile 134: Senden des neuen Bereiches der verschobenen Story Card an den Webservice

BoardActivity.java:
Zeile 293: Abrufen der Story Card Daten vom Webservice
Zeile 730: Senden des Iterationsstarts an den Webservice
Zeile 750: Senden der Iterationsbeendigung an den Webservice
Zeile 965: Senden des Iterationsstarts an den Webservice
Zeile 985: Senden der Iterationsbeendigung an den Webservice
Zeile 1195: Senden der neu erstellten oder editierten Story Card Daten an den Webservice
Anmerkung: siehe Anmerkung ProjectActivity.java

Zeile 1457: Senden der Zustandsveränderung der Story Card an den Webservice
Zeile 1510: Informiert Webservice Zeichnung der Story Card zu löschen
Zeile 1584: Informiert Webservice Zeichnung und Story Card zu löschen
Zeile 1766: Senden der Zustandsveränderung der Story Card an den Webservice
Zeile 1776: Senden der Zustandsveränderung der Story Card an den Webservice
Zeile 2106: Senden der Zustandsveränderung der Story Card an den Webservice
Zeile 2159: Informiert Webservice Zeichnung der Story Card zu löschen
Zeile 2233: Informiert Webservice Zeichnung und Story Card zu löschen
Zeile 2416: Senden der Zustandsveränderung der Story Card an den Webservice
Zeile 2426: Senden der Zustandsveränderung der Story Card an den Webservice
Zeile 2491: Senden des Iterationsstarts an den Webservice
Zeile 2512: Senden der Iterationsbeendigung an den Webservice

E)Evaluation: Aufgabenstellung

Aufgabenstellung

1. Aufgabe
   Wie viele Story Cards enthält das Projekt insgesamt?

   Anmerkungen:

2. Aufgabe
   Planen Sie eine Iteration, in dem Sie der Iteration Story Cards aus dem Backlog zuweisen. Dabei sollen Sie so nahe wie möglich an den maximalen Iterationsaufwand (oben rechts in der Iteration) kommen, diesen aber nicht überschreiten.

   Anmerkungen:

3. Aufgabe
   In welchem Bereich (Backlog oder Iteration) befindet sich die Story Card mit der Anmerkung „Bearbeitet von: Oliver Karras“?

   Anmerkungen:

4. Aufgabe
   Bringen Sie die Story Card aus Aufgabe 3 in den jeweils anderen Bereich. Ist die Story Card im Backlog, so soll sie in die Iteration. Und ist sie in der Iteration, so soll die Story Card ins Backlog.

   Anmerkungen:

5. Aufgabe

Finden Sie eine Story Card, die sowohl einen Test / eine Erwartungen als auch eine Beschränkung aufweist und setzen Sie diese Story Card auf erledigt.

Hinweis 1: In der Praxis wird mit Karteikarten gearbeitet, die sowohl auf der Vorder- als auch auf der Rückseite beschrieben werden. Dieses Prinzip ist auch in MUST mittels Gestensteuerung umgesetzt worden.

Hinweis 2: Die Veränderung des Zustandes einer Story Card erfolgt ebenfalls durch Gestensteuerung. Für den Zustand „erledigt“, müssen Sie die Story Card abhaken.

Anmerkungen:

F)Evaluation: Bewertungsbogen

Bewertungsbogen

1. Zu den Aufgaben:

Hatten sie bei den Aufgaben Schwierigkeiten?
Kreuzen Sie die entsprechende Antwort an und begeben Sie an, warum Sie Schwierigkeiten mit einer Aufgabe hatten.

Aufgabe	Nein	Ja	Anmerkung
1			
2			
3			
4			
5			

2. Bewerten Sie die folgenden Aspekte:

Setzen Sie dazu bei jeder Frage nur genau ein Kreuz und geben Sie eine kurze Begründung für ihre Antwort an.

Gesamteindruck:

Wie wirkt die gesamte Applikation auf Sie?

sehr schlecht	schlecht	gut	sehr gut

Begründung:

Übersichtlichkeit:

Wie gut haben Sie sich in der Anwendung zurechtgefunden?

sehr schlecht	Schlecht	Gut	sehr gut

Begründung:

Bedienung:

1) Navigation durch die Ansichten

Wie gut kam Sie mit der Navigation durch die verschiedenen Ansichten zurecht?

sehr schlecht	schlecht	gut	sehr gut

Begründung:

2) Gesten zum Drehen einer Story Card

Wie gut fanden Sie die Gestensteuerung zum Drehen einer Story Card?

sehr schlecht	schlecht	gut	sehr gut	Nicht verwendet

Begründung:

3) Geste für das Erledigen einer Story Card

Wie gut fanden Sie die Gestensteuerung für das Erledigen einer Story Card?

sehr schlecht	schlecht	gut	sehr gut	Nicht verwendet

Begründung:

4) Drag & Drop im Splitscreen

Wie gut kam Sie mit der Drag & Drop Funktion im Splitscreen (geteilter Bildschirm) zurecht?

sehr schlecht	schlecht	gut	sehr gut	Nicht verwendet

Begründung:

5) Drag & Drop in anderen Ansichten

Wie gut kam Sie mit der Drag & Drop Funktion in den anderen Ansichten zurecht?

sehr schlecht	schlecht	gut	sehr gut	Nicht verwendet

Begründung:

6) Sonstiges

Wenn es einen Aspekt der Applikation gibt, der Ihnen selbst besonders aufgefallen ist (sei es positiv oder negativ), geben Sie diesen hier an und bewerten Sie ihn wie die anderen zuvor.

Aspekt: _____

Wie gut hat Ihnen, der von Ihnen genannte Aspekt gefallen? / Wie gut kam Sie mit diesem Aspekt zurecht?

sehr schlecht	schlecht	gut	sehr gut

Begründung:

G)Evaluation: Gemessene Bearbeitungszeiten

Die folgende Tabelle enthält die gemessenen Bearbeitungszeiten der einzelnen Probanden für jede einzelne Aufgabe in Sekunden.

Aufgabe	1	2	3	4	5
Fokussierter Aspekt	Navigation	Gesten	Navigation	Navigation & Gesten	Gesten
Proband 1	36s	104s	63s	5s	45s
Proband 2	25s	71s	49s	13s	37s
Proband 3	47s	86s	77s	14s	82s
Proband 4	23s	116s	52s	32s	98s

H)Kommentare der Probanden aus den Bewertungsbögen

Die folgenden Kommentare sind jeweils nur Auszüge aus den Bewertungsbögen aller Probanden und sollen einen Überblick über das erhaltene Feedback geben.

Feedback bezüglich der einzelnen Aufgaben:

2	~~muss~~	X	brauchte einen Hinweis, dass man die SC per Drag & Drop ziehen kann.

5	X		Man muss von der Gestensteuerung zum drehen der Karte wissen

5		X	recht lange gesucht, Umdreh-Geste nur auf Detailansicht und gleich mit Fensterwechsel-Geste → verwirrt

Feedback zu dem Aspekt *„Gesamteindruck“*:

Gesamteindruck:

Wie wirkt die gesamte Applikation auf Sie?

sehr schlecht	schlecht	gut	sehr gut
			X

Begründung:

Intuitiv bedienbar, das mentale Modell stimmte durchgehend mit dem System überein.

Feedback zu dem Aspekt *„Übersichtlichkeit“*:

Übersichtlichkeit:

Wie gut haben Sie sich in der Anwendung zurechtgefunden?

sehr schlecht	Schlecht	Gut	sehr gut
		X	

Begründung: kleinere „Probleme“ bei Erstnutzung, aber nach sehr kurzer Eingewöhnung sehr gut

Übersichtlichkeit:

Wie gut haben Sie sich in der Anwendung zurechtgefunden?

sehr schlecht	Schlecht	Gut	sehr gut
			X

Begründung:

Die Anzahl der verschiedenen Ansichten ist gut überschaubar. ⊛

Feedback zu dem Aspekt *„Navigation durch die Ansichten“*:

1) Navigation durch die Ansichten

Wie gut kam Sie mit der Navigation durch die verschiedenen Ansichten zurecht?

sehr schlecht	schlecht	gut	sehr gut
			X

Begründung:

siehe ⊛

1) Navigation durch die Ansichten

Wie gut kam Sie mit der Navigation durch die verschiedenen Ansichten zurecht?

sehr schlecht	schlecht	gut	sehr gut
		X	

Begründung: Teilweise versehentlich Sicht gewechselt (Gesten für andere Dinge ausprobieren ⅋ in Aufgabe 5), anfangs Unsicher, wie Sichten angeordnet sind, aber schnell eingewöhnt

Feedback zu dem Aspekt *„Geste zum Drehen einer Story Card“*:

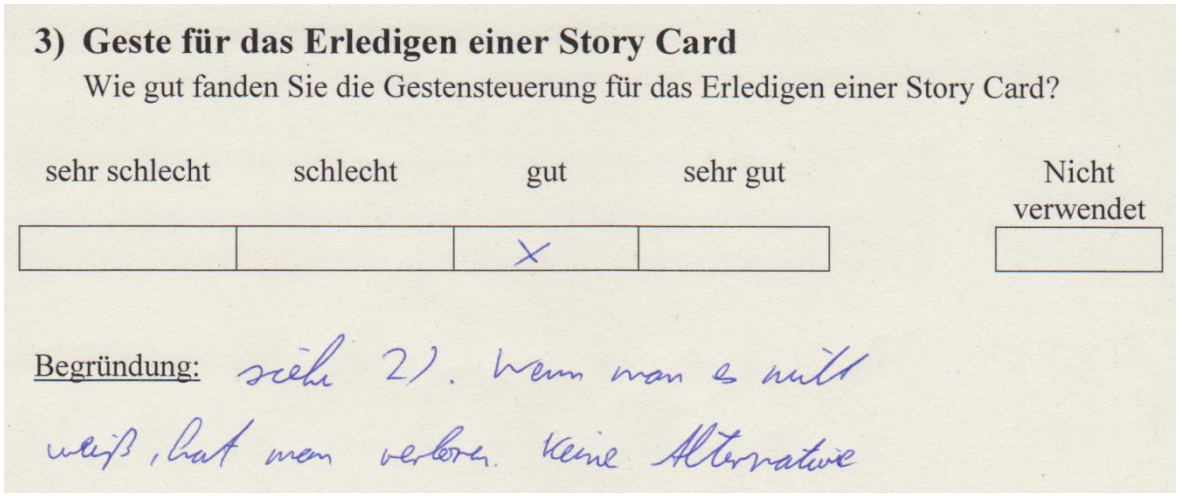
2) Gesten zum Drehen einer Story Card

Wie gut fanden Sie die Gestensteuerung zum Drehen einer Story Card?

sehr schlecht	schlecht	gut	sehr gut	Nicht verwendet
		X		

Begründung: Aber man muss davon wissen. Es gibt keine Alternative

Feedback zu dem Aspekt *„Geste zum Erledigen einer Story Card“*:



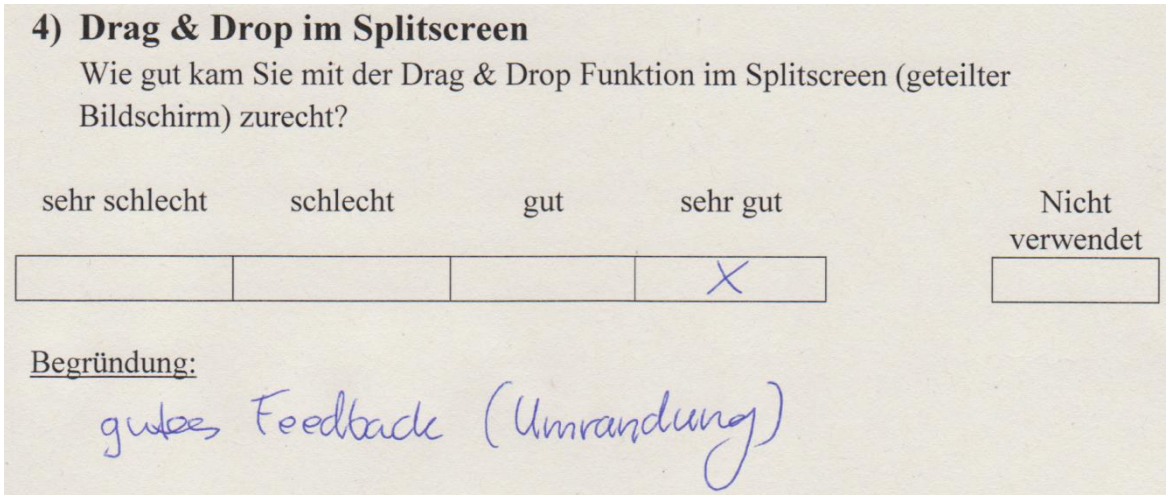
3) Geste für das Erledigen einer Story Card

Wie gut fanden Sie die Gestensteuerung für das Erledigen einer Story Card?

sehr schlecht	schlecht	gut	sehr gut	Nicht verwendet
		X		

Begründung: siehe 2). Wenn man es nicht weiß, hat man verloren. Keine Alternative

Feedback zu dem Aspekt *„Drag & Drop im Splitscreen“*:



4) Drag & Drop im Splitscreen

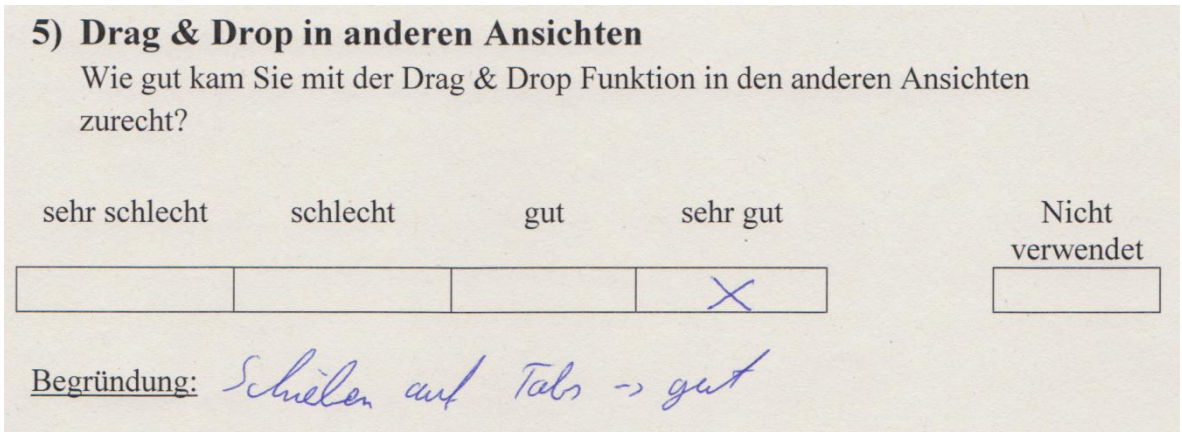
Wie gut kam Sie mit der Drag & Drop Funktion im Splitscreen (geteilter Bildschirm) zurecht?

sehr schlecht	schlecht	gut	sehr gut	Nicht verwendet
			X	

Begründung:

gutes Feedback (Umrandung)

Feedback zu dem Aspekt *„Drag & Drop in anderen Ansichten“*:



5) Drag & Drop in anderen Ansichten

Wie gut kam Sie mit der Drag & Drop Funktion in den anderen Ansichten zurecht?

sehr schlecht	schlecht	gut	sehr gut	Nicht verwendet
			X	

Begründung: Schieben auf Tabs -> gut

Literaturverzeichnis

[1] M. Cohn, User Stories für die agile Software-Entwicklung mit Scrum, XP u.a., Heidelberg, München, Landsberg, Frechen, Hamburg: mitp, eine Marke der Verlagsgruppe Hüthig Jehle Rehm GmbH, 2010.

[2] J. Eckstein, Agile Softwareentwicklung mit verteilten Teams, Heidelberg: dpunkt.verlag GmbH, 2009.

[3] I. G. Stamelos and P. Sfetsos, Agile Software Development Quality Assurance, Hershey, London: Idea Group Inc., 2007.

[4] Q. Ktata und G. Lévesque, „Designing and Implementing a Measurement Program for Scrum Teams: What do agile developers really need and want?," 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1822341. [Zugriff am 2 Januar 2013].

[5] „Agile Alliance," [Online]. Available: http://agilealliance.org. [Zugriff am 17 Dezember 2012].

[6] C. Patel und M. Ramachandran, „Story Card Baesd Agile Software Development," April 2009. [Online]. Available: http://www.sersc.org/journals/IJHIT/vol2_no2_2009/12.pdf. [Zugriff am 2 Januar 2013].

[7] G. Azizyan, M. K. Magarian und M. Kajko-Mattson, „Survey of Agile Tool Usage and Needs," in s *Agile Conference*, Salt Lake City, 2011.

[8] P. E. McMahon, „Extending Agile Methods: A Distributed Project And Organizational Improvement Perspective," in s *Systems & Software Technology Conference*, Salt Lake City, 2005.

[9] S. W. Ambler, „Agile Modeling," [Online]. Available: http://www.agilemodeling.com/artifacts/userStory.htm#InitialFormal. [Zugriff am 17 Dezember 2012].

[10] IEEE Standards Association, „830-1998 - IEEE Recommended Practice for Software Requirements Specifications," 1998. [Online]. Available: http://standards.ieee.org/findstds/standard/830-1998.html. [Zugriff am 20 Dezember 2012].

[11] userStories, „userStories.com," [Online]. Available: http://www.userstories.com/products. [Zugriff am 5 März 2013].

[12] Planbox, „Planbox," [Online]. Available: http://www.planbox.com/. [Zugriff am 15 März 2013].

[13] ti8m technology inovation & management, „ti8m technology inovation & management," BSgroup Technology Innovation AG, [Online]. Available: http://www.ti8m.ch/scrumit. [Zugriff am 15 März 2013].

[14] Androidplot, „Androidplot," [Online]. Available: http://androidplot.com/. [Zugriff am 15 März 2013].

[15] Android Developers, „Android Developers," [Online]. Available: http://developer.android.com/about/dashboards/index.html. [Zugriff am 2013 Februar 24].

[16] Mercury Intermedia, „slideshare - Present yourself," 28 Februar 2011. [Online]. Available: http://de.slideshare.net/mercmedia/mercury-intermedia-nashville-ux-presentation. [Zugriff am 03 März 2013].

[17] Username: pankai@gmail.com, „android - Android - An Open Handset Alliance Project," 26 Februar 2010. [Online]. Available: http://code.google.com/p/android/issues/detail?id=6885. [Zugriff am 03 März 2013].

[18] R. Jeffries, „Essential XP: Card Conversation, and Confirmation," *XP Magazine*, 30 August 2001.

[19] Prof. Dr. Kurt Schneider, *Grundlagen der Softwaretechnik (SWT)*, Hannover, 2010.

[20] K. Schwaber und A. Armstrong, „scrum.org Improving the Profession ofSoftware Development," 2009. [Online]. Available: http://www.scrum.org/Resources/What-is-Scrum. [Zugriff am 15 März 2013].

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig 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, 30. August 2012

_____

Oliver Karras