Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Alexander Kaack

Ein interaktives Webinterface zur Unterstützung von Scrum-Prozessen

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Alexander Kaack

Ein interaktives Webinterface zur Unterstützung von Scrum-Prozessen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Gunter Klemke

Eingereicht am: 16. Mai 2014

Alexander Kaack

Thema der Arbeit
Ein interaktives Webinterface zur Unterstützung von Scrum-Prozessen

Stichworte
Agile Softwareentwicklung, verteilte Teams, Scrum, Daily Scrum, CSCW, HCI

Kurzzusammenfassung
Im Rahmen der verteilten agilen Softwareentwicklung stellt sich ein verstreutes Team besonderen Kommunikationsherausforderungen. Bei der Verwendung des Scrum Frameworks ist besonders das tägliche Daily Scrum als Meeting-Ritual zu unterstützen. Innerhalb eines großen deutschen Unternehmens wurden hierzu Untersuchungen durchgeführt, die eine häufige Orientierungslosigkeit ergaben. Zur Reduzierung dieser Problematik wurde ein Taskboard-Prototyp entwickelt, welcher einen synchronen Kommunikationskanal für verteilte Scrum-Teams darstellt. Zur Auswertung folgte eine ausgiebige Evaluierung über Beobachtungen, ein Gruppeninterview, einen Fragebogen und den Interaktionsprotokollen der Anwendung.

Alexander Kaack

Title of the paper
An interactive webinterface for supporting Scrum-processes

Keywords
Agile software development, distributed teams, Scrum, Daily Scrum, CSCW, HCI

Abstract
In the context of distributed Agile software development, dispersed teams are faced with special communication challenges. When using the Scrum framework, the daily Scrum-as-a-ritual meetings must be supported. Studies within a large German enterprise have brought to light a large degree of disorientation. To reduce this, a taskboard prototype was developed which presented a synchronous communication channel for distributed Scrum teams. To study the efficacy, a detailed evaluation was performed through observations, a group interview, a questionnaire and the interaction logs of the application.

Inhaltsverzeichnis

1. Einleitung

Trotz maximaler Produktivität und höchster Effizienz haben europäische Firmen Probleme beim Mithalten auf dem globalen Markt. Ca. 50 Prozent der Arbeitnehmer gehören dabei in die Kategorie *Wissensarbeiter* (engl. *Knowledge Worker*) (Zeitelberger 2012, S.76). Als Wissensarbeiter werden Mitarbeiter bezeichnet, welche für die Anwendung des erworbenen Wissens bezahlt werden (Fischbach und Putzke, 2012). Im Zusammenhang mit dem Wettbewerbsproblemen ist das Management bemüht, die zielgerichtete Steigerung der Produktivität voranzubringen. Informationssysteme wie *Groupware* versuchen die Wissensorganisation zu verbessern und auch der Einsatz von *Social Software* (Blogs, Wikis, etc.) wird hiermit zunehmend interessanter (vgl. Fischbach und Putzke 2012; Schwarzer 2012, S.2).

Die zunehmende Marktdynamik sorgt zudem für weitere Herausforderungen durch eine steigende Planungskomplexität und dem damit verbundenen Aufwand (vgl. Barkalov u. a. 2013, S.6). Innerhalb der Softwareentwicklung haben besonders agile Vorgehensmodelle zur Reduzierung dieser Komplexität beigetragen. Hierbei wird versucht, das Projekt in Teilaufgaben und Zeitabschnitte zu splitten, um so einen kontrollierbaren Abschnitt zu konzipieren.

Agile Vorgehnsmodelle erfreuen sich dabei stetiger Beliebtheit. Gleichzeitig nimmt auch die Verteilung von agilen Teams zu (VersionOne 2013, S.3). Nach einer weltweiten Studie nutzen ca. 88 % der befragten Unternehmen agile Methoden, an der Spitze mit *Scrum* und Scrum ähnlichen Varianten, welche zusammen 78 % ausmachen (VersionOne 2013, S.4). Scrum enthält eine Vielzahl von verschiedenen Meeting-Ritualen (Schwaber, 2011). Ziel dieser Rituale ist eine effiziente und lösungsorientierte Kommunikation zwischen den Teilnehmern (Stray u. a., 2012).

1.1. Problemerfassung

Die notwendige Kommunikation zwischen verteilten Teams stellt ein zentrales Problem dar. Die Herausforderungen besteht dabei besonders bei der Überbrückung der räumlichen Distanz,

fehlenden gemeinsamen Kontexten, Teamkohäsion und zudem die variable Verfügbarkeit der Team-Mitglieder (Paasivaara u. a., 2009). Technische Hilfsmittel wie Telefon, E-Mail, Chats, Screensharing etc. können zur Reduzierung dieser Herausforderungen beitragen. Dieser Ansatz weißt dabei auf Parallelitäten zu Enterprise 2.0 (Mcafee, 2006). Im Rahmen der Zusammenarbeit mit einem großen deutschen Unternehmen ist bei der Analyse der Daily Scrums besonders die regelmäßige Orientierungslosigkeit der Meetingteilnehmer aufgefallen.

Ziel der Arbeit

Ziel dieser Arbeit ist zum einen die Erfassung der Probleme, welche innerhalb eines Daily Scrum bei verstreuten Softwareentwicklungs-Teams auftreten. Insbesondere ist hierbei die Orientierungslosigkeit beobachtet worden. Darauf aufbauend wird die Entwicklung eines Prototypens zur Unterstützung und Minimierung dieses Problems untersucht. Eine Evaluierung soll prüfen, ob die entwickelte Anwendung zur Reduzierung dieser Probleme beitragen konnte. Das Ergebnis dieser Arbeit wird aufzeigen, ob die entwickelte Anwendung zur Unterstützung der zu überbrückenden Distanz zwischen den Teammitgliedern beitragen und die erfasste Orientierungslosigkeit reduzieren kann.

1.2. Struktur der Arbeit

Die Struktur dieser Arbeit unterteilt sich in vier Kapitel.

Zunächst wird im Kapitel Analyse (2) auf das Entwicklungs-Framework *Scrum* (2.1) näher eingegangen, um dann im Kontext hierzu die verteilte agile Softwareentwicklung (2.2) zu erwähnen. Existierende Tools (2.3) wie die Atlassian Suite bieten eine Möglichkeit agile Prozesse zu unterstützen. Durch diese Basis wird danach das *Daily Scrum* (2.4) näher beleuchtet und das Szenario zum Vorgehen (2.4.1) beschrieben sowie die Auswertung (2.4.2) des Ist-Zustands dargestellt. Des weiteren werden Anforderungen (2.5) und eine Spezifikation (2.6) definiert, welche zur Entwicklung einer Lösung notwendig sind. Darauf aufbauend sind vergleichbare Ansätze zu betrachten (2.7) und ein Fazit (2.8) zur Analyse zu stellen.

Im Kapitel Design und Realisierung (3) wird resultierend aus den Ergebnissen des Analyse Kapitels (2) auf den zu entwickelnden Prototypen eingegangen. Hierzu werden im Abschnitt der Systemmodule (3.1) die notwendige Grundmodule erwähnt und daraufhin eine Abgrenzung

(3.2) in Abhängigkeit zur gegebenen Systemumgebung vorgenommen. Zur Beschreibung der zu entwickelnden Anwendung sollen verwendete Entwurfsmuster (3.3) definiert und über Sichten (3.3.2) dargestellt werden. Zur Realisierung des Benutzerinterfaces (3.4) wird ein Entwurf eine Variante der Darstellung und Interaktion bieten. Durch die Möglichkeit von mehreren Clients wird zur Bewältigung der koordinierten Zugriffe (3.5) eine Lösung vorgestellt. Abnahmekriterien und Akzeptanztests (3.6) sollen die Lauffähigkeit der Anwendung garantieren. Um das Kapitel abzuschließen, wird eine technische Evaluierung (3.7) das Kapitel zusammenfassen und zudem mögliche Verbesserungen einer neuen Implementierung oder Erweiterung darstellen.

Zur Evaluierung (4) des Prototypens werden zu Beginn die verwendeten Methoden beschrieben. Hierzu zählen die Beobachtung (4.1), daraufhin das Interview (4.2), der Fragebogen (4.3) und die Interaktionsprotokolle (4.4). Die Auswertung (4.5) stellt die entsprechenden Ergebnisse der beschriebenen Methoden vor, welche am Ende des Kapitels zusammenfassend in dem Fazit (4.6) vorgestellt werden.

Der Schluss (5) greift letztendlich die Inhalte dieser Arbeit auf (5.1) und gibt einen Ausblick (5.2) auf weiterführende Fragestellungen.

1.3. Veröffentlichung

Teile dieser Arbeit werden auf den Wismarer Wirtschaftsinformatik-Tagen 2014 (WIWTA-2014) unter dem Titel „Ein interaktives Taskboard zur Unterstützung von verteilten Scrum-Teams“ veröffentlicht (Kaack u. a., 2014).

Inhalt dieses Beitrags ist die Vorstellung eines interaktiven Taskboard-Prototypens zur synchronen Kommunikation von verteilten Scrum-Teams. Hierzu werden die erfasste Problematik der Orientierungslosigkeit sowie die Ergebnisse der Evaluierung aus der Entwicklungsabteilung präsentiert.

2. Analyse

Dieses Kapitel befasst sich mit der Analyse der Problemstellung, welche in der Einleitung beschrieben wurde. Zunächst wird ein Exkurs zu Scrum-Bestandteilen gegeben und das Vorgehen dieses Entwicklungsmodells erläutert. Weiterführend soll im Kontext der agilen Entwicklung mit Scrum auf das Thema der verteilten Softwareentwicklung eingegangen werden. Das Ticket- und Projektmanagement-System JIRA stellt ein zentrales System für die aktuelle Unterstützung der agilen Softwareentwicklung dar und soll innerhalb der Atlassian Suite näher betrachtet werden. Im Bezug auf eine praxisnahe Problembeschreibung wird ein Daily Scrum einer großen deutschen Firma im Dienstleistungsgewerbe herangezogen und analysiert. Resultierend aus dem Vorgehen der Firma ergeben sich unter der Berücksichtigung des aktuellen Daily Scrum Ablaufes und den verstreuten Teammitgliedern verschiedene Anforderungen an ein neues System zur Unterstützung des Scrum-Prozesses. Aktuelle Lösungen sollen hierzu untersuchen, ob diese eine Option bieten oder eine neue alternative Software gerechtfertigt ist. Das Fazit dieses Kapitels stellt eine Zusammenfassung und einen Ergebnisbericht der Analyse dar.

2.1. Agiles Vorgehen mit Scrum

Scrum (Schwaber, 2011) ist eine agile Vorgehensweise zur Entwicklung von Produkten. Die folgende Beschreibung von Scrum ist aus „The Scrum Guide“(Schwaber, 2011) entstanden. Entwickelt worden ist Scrum von Ken Schwaber und Jeff Sutherland. Dieses Framework eignet sich besonders zur Bewältigung von komplexen Projekten. Es besteht aus Ritualen, Rollen, Artefakten und Regeln. Die Regeln binden und kontrollieren die Abhängigkeitsbeziehungen zwischen diesen Punkten.

Scrum basiert auf einem empirischen Prozessmodell und geht iterativ und inkrementell vor. Durch dieses Vorgehen sollen einfacher Aufwände geschätzt und das Risiko einer fehlerhaften Vorausschau minimiert werden.

Hierzu stehen die drei Säulen *Transparency*, *Inspection* und *Adaptation*.

Transparency (dt. Transparenz) dient zur Aufdeckung der Prozesse. Wichtige Aktionen und Ergebnisse sollen verfolgbar sein. Auch sollen so frühzeitig Probleme erkannt werden können.

Inspection (dt. Inspektion) dient der kontinuierlichen Kontrolle der Prozesse. Nur wenn sich daran gehalten wird, kann ein Problem auch gefunden werden.

Adaptation (dt. Adaptation) stellt die Anpassung der Entwicklung dar. Wenn ein Problem erfasst wird, müssen Anpassungen vorgenommen werden, um dieses zu beseitigen.

Inspection und Adapation stellen die zentralen Aktionen der vier Rituale dar. Zu den Ritualen zählen das Sprint Planning, das Daily Scrum, das Sprint Review und die Sprint Retrospective. Diese werden innerhalb eines jeden Sprints ausgeführt. Ein Sprint stellt einen zeitlich begrenzten Zeitraum dar (z. B. zwei Wochen), welcher am Ende ein fertiges Inkrement des Produkts fordert. Das Sprint Planning wird am Anfang jedes Sprints durchgeführt und dient der Planung des gesamten Sprints. Hier werden Aufgaben aus dem sog. *Backlog* entnommen. Es werden nur so viele Aufgaben entnommen wie für einen Sprint definiert werden. Die Definition wird dabei vom Entwicklerteam beschrieben. Das Daily Scrum ist ein weiteres Meeting, welches täglich stattfindet und auf das im Abschnitt 2.4 noch näher eingegangen wird. Das Sprint Review ist zur Präsentation der Ergebnisse bzw. des erstellten Produktinkrements gedacht und soll das Feedback der *Stakeholder* miteinbeziehen. In der Sprint Retrospective wird der abgeschlossene Sprint rekapituliert und gegebenenfalls werden Anpassungen für den nachfolgenden Sprint vorgenommen.

Das Team selbst besteht aus *Product Owner*, den Entwicklern und dem *Scrum Master* und strebt eine Selbstorganisation und Cross-Funktionalität an. Damit soll erreicht werden, dass das Team unabhängig, flexibel, kreativ und produktiv arbeiten kann. Ein Product Owner stellt die Schnittstelle zwischen Entwicklung und Kunden dar. Die Aufgabe von ihm ist das Aufbauen und Verwalten des Produkt-Backlogs. Dies schließt die Sortierung nach dem Wert (*Business Value*, Geschäftswert) einer Aufgabe ein. Der Scrum Master versucht das Team so zu managen, sodass es sich möglichst selbständig organisieren kann. Zusätzlich beseitigt er auftretende Hindernisse im Entwicklungsprozess und sorgt für die Einhaltung der Scrum-Prinzipien und das stattfinden der darin enthaltenden Rituale.

Artefakte von Scrum sind das Product-Backlog und das Sprint-Backlog. Über das Product-Backlog werden die fachlichen Aufgaben (Anforderungen aus Nutzersicht) in Entwickler-Aufgaben für das Sprint-Backlog transformiert. Beide Backlogs werden abhängig von ihrer

Wichtigkeit sortiert, wobei die Wichtigkeit vom jeweiligen Board und vom geführten Verwalter (Product Owner oder Entwicklerteam) abhängig ist. Im Sprint-Backlog werden die Aufgaben vom Entwicklerteam sortiert und gefüllt. Das Product-Backlog wird wie bereits beschrieben, vom Product Owner geführt.

Die Prinzipien, welche mit Scrum vermittelt werden sollen, sind als Leitlinien zu betrachten. Dabei versteht man unter dem Punkt der Adaptation (Anpassung) neben den Anpassungen in der laufenden Entwicklung ebenfalls die Anpassung des gesamten Scrum-Workflows. Abhängig vom Unternehmen, Projekt, oder Team, ist dieser anzupassen (Eckstein 2012, S.27).

> „Der Weg ist flexibel, das Ziel ist es nicht." (Eckstein 2012, S.27)[1]

2.2. Verteilte agile Softwareentwicklung

Die verteilte Entwicklung von Software lässt sich unterscheiden in verteilte Teams und verstreute Teams (vgl. Eckstein 2012, S.74). Bei verteilten Teams spricht man von Teilteams, die sich jeweils an verschiedenen Standorten befinden, jedoch wird von allen Teilteams zusammen die Software entwickelt. Bei verstreuten Teams wird ein Team oder Teilteam auf verschiedene Standorte verteilt. Es wird also zerrissen. Ein Beispiel hierzu wäre, dass sich drei Entwickler in Berlin, vier Entwickler plus Scrum Master und Product Owner in Hamburg befinden. Die große Herausforderung beider Arten der verteilten Entwicklung ist die Kommunikation. So ist bei Scrum durch die angestrebte Selbstorganisation des Entwicklungsteams eine wesentlich höhere Kollaboration und Interaktion nötigt, um gemeinsam das Ziel eines Sprints zu erreichen (vgl. Pichler 2009, S.15-16). Man kann also sagen, dass sich verteilte agile Teams besonders durch die Bewältigung der räumlichen Distanz auszeichnen. Es stellt sich hierbei jedoch die Frage, ab wann man ein Projekt verteilt nennen kann.

Eine Klassifizierungsmöglichkeit bietet das Verteilungskontinuum nach Simons (Abbildung 2.1). Man sieht in dieser Abbildung, dass man bereits von einer verteilten Situation spricht, sobald das Team über Räume verteilt ist. In so einem Fall ist die Überbrückung der Entfernung zwar trivial, eine Unterstützung ist trotzdem notwendig. Diese notwendige Unterstützung erscheint aber ebenfalls trivial im Gegensatz zu einer stark ausgeprägten Verteilung.

[1] „Be flexible how you get there, but be inflexible in the focus." Martin Fowler, auf der International Conference on eXtreme Programming and Agile Processes in Software Engineering 2001, Sardinien, Italien.

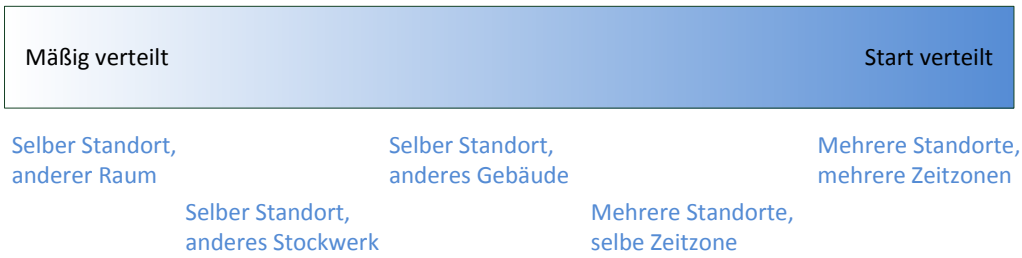

Abbildung 2.1.: Verteiltungskontinuum nach Pichler (2009) S. 127

Wenn man ein Team auch nur „mäßig verteilt“ nennen kann, gilt es, die Kommunikation zu unterstützen. Wenn es einen vertretbaren Kosten- / Zeitrahmen gibt, gilt nach wie vor, dass die direkte Vermittlung von Informationen zu bevorzugen ist. Kann dies nicht erreicht werden, muss ein Wissensverlust bzw. eine Wissensvorbehaltung im Team gegen Null angestrebt werden. Dabei kann es sogar von Vorteil sein, wenn das Team generell verstreut ist und somit immer der Informationsfluss zwischen den Teammitgliedern im Fokus steht. Auch verringert sich hierdurch die Gefahr einer eigenen Projektkultur je Standort (vgl. Eckstein 2012, S.74).

> „Je größer und verteilter ein Projekt wird, desto schwieriger ist es sicherzustellen, dass sich alle Projektmitglieder mit dem Projektziel identifizieren, gemeinsame Normen befolgen und so an einem Strang ziehen.“ (Pichler 2009, S.127)

Bereits bei einem geringen Grad der Verteilung gibt es Praktiken wie zum Beispiel *Collective Code Ownership* (vgl. Eckstein 2012, S.45), womit versucht wird eine gemeinsame Verantwortung für den entwickelten Code zu schaffen. Diese Praktik, welche aus dem *Extreme Programming* kommt, versucht mittels *Code Sharing* es allen Entwicklern zu ermöglichen, Veränderungen am Code des Anderen vorzunehmen (vgl. Hanser 2010, S.44). Dieser Punkt ist allerdings im Verteilungsaspekt mit besonderer Vorsicht zu betrachten und kann nur mittels gemeinsamer Richtlinien und einer geeigneten technischen Infrastruktur zur Unterstützung erfolgen.

2.2.1. Skalierung

Bevor man sich mit der Frage der Skalierung befasst, geht es im ersten Schritt erst einmal um die tatsächliche Größe des Projekts und die daraus resultierende Teamgröße.

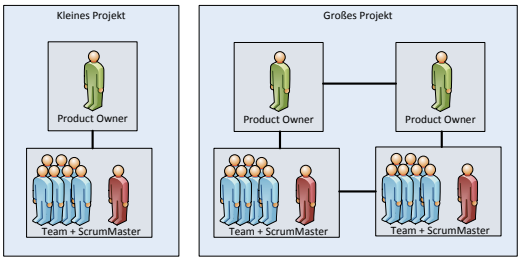

Abbildung 2.2.: Kleines vs. großes Scrum-Projekt nach Pichler (2009) S. 126

In der Regel wird man ab einer bestimmten Größe, wie in Abbildung 2.2, auf einen eigenen Scrum Master sowie Product Owner pro Teilteam nicht verzichten, unabhängig wie sich wiederum das Team auf Standorte splittet. Es ist darauf zu achten, das besonders die Größe je Teilteam von fünf bis neun Personen nicht überschritten wird. Darüber hinaus wird eine Aufteilung empfohlen (vgl. Wirdemann 2011, S.38). Ist nun genau diese Größe überschritten und eine Aufteilung unvermeidlich, so gibt es verschiedene Varianten, wie und nach welchen Kriterien sich ein Team aufsplitten kann. Ein Splitt bedeutet, dass sich wiederum ein Teilteam bildet, welches am Gesamtprojekt mitwirkt.

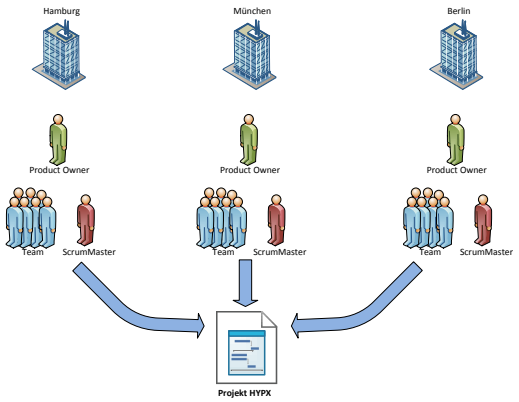

Abbildung 2.3.: Verteiltes Projekt nach Pichler (2009) S. 126

So könnte bei mehreren Standorten eine mögliche Aufteilung wie in Abbildung 2.3 aussehen. Hier wurde zwecks enger Kommunikation in dem Team, dem Entwicklern an jedem Standort ein Scum Master und Product Owner zur Verfügung gestellt. Wie bereits erwähnt, spricht man in diesem Fall von einem verteilten Team. Zur Organisation der Teilteams gilt es eine gemeinsame Basis zu schaffen, damit ein paralleles Arbeiten möglich ist.

Spricht man nun von einem großen Projekt, wo sich ableitend von den Fähigkeiten der Teammitglieder (Entwickler und Product Owner) wiederum unterteilen lässt, so ist es wichtig die Teilteams optimal zu organisieren. Möglichkeiten zur Organisierung bieten die Feature und Komponenten Unterteilungen. Ziel beider Varianten ist es, die Teams möglichst autonom und mit geringer Abhängigkeit untereinander auszustatten (vgl. Pichler 2009, S.136). Dies steht natürlich im Widerspruch zu der bereits erwähnten Variante (Code Sharing aus dem Extreme Programming). Zu beachten ist hier, dass sich diese Praktik wieder in den jeweiligen Teams wiederfinden kann.

2.2.1.1. Komponententeams

Betrachtet wird zunächst die Komponenten-Unterteilung. Das Team wird hierbei zusammen mit jeweils einem Product Owner einer Komponente oder einem Subsystem zugeteilt.

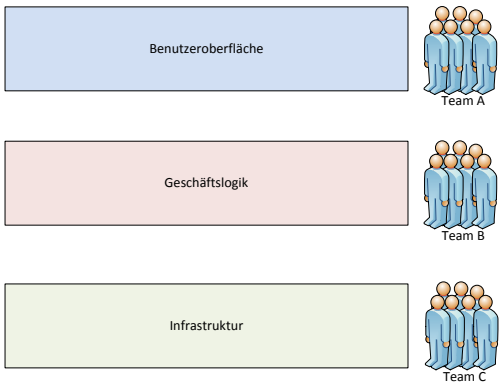

Abbildung 2.4.: Komponententeams nach Pichler (2009) S. 137

In Abbildung 2.4 sieht man, dass ein Team z. B. für die Benutzeroberfläche, eines für die Geschäftslogik und eines für die Infrastruktur verantwortlich ist.

Vorteile dieser Variante sind:

- Konsistenz und Integrität der Softwarearchitektur kann leichter sichergestellt werden.
- Verringerung von Codeduplizierung.

- Unterteilung der Schichten nach Fähigkeiten der Teammitglieder.
- Klarer Verantwortungsbereich mit minimaler Kommunikation zwischen den Komponententeams.

Nachteile der Komponententeams:

- Kein Komponententeam kann autonom ein Inkrement des Sprints fertigen.
- Fokussierung auf Subsysteme und Vernachlässigung der Gesamtstruktur.
- Die Kommunikationsabhängigkeit zwischen den Komponenten steigt gleichermaßen zu der zwischen den Komponententeams.
- Steigender Integrationsaufwand für ein lauffähiges Inkrement, abhängig von der Anzahl der Komponenten.
- Eine komplexe Softwarearchitektur erfordert einen erhöhten Aufwand für Planung und Splitts für die jeweiligen Komponententeams (Anforderungen).
- Die Geschwindigkeit der Entwicklung orientiert sich am langsamsten Team und erschwert die Unterstützung durch die Spezialisierung der einzelnen Mitarbeiter.

Diese Teams lassen sich besonders gut nutzen, wenn die Anzahl der Teams nicht fünf übersteigt und die Architektur der Software etabliert und klar strukturiert ist (vgl. Pichler 2009, S.138).

2.2.1.2. Featureteams

Die Featureteams wiederum arbeiten nach den Anforderungen eines Projekts und agieren somit über alle oder Teilschichten hinweg (Abbildung 2.5). Auch bei dieser Variante steht dem jeweiligen Team wieder ein Product Owner zur Seite.

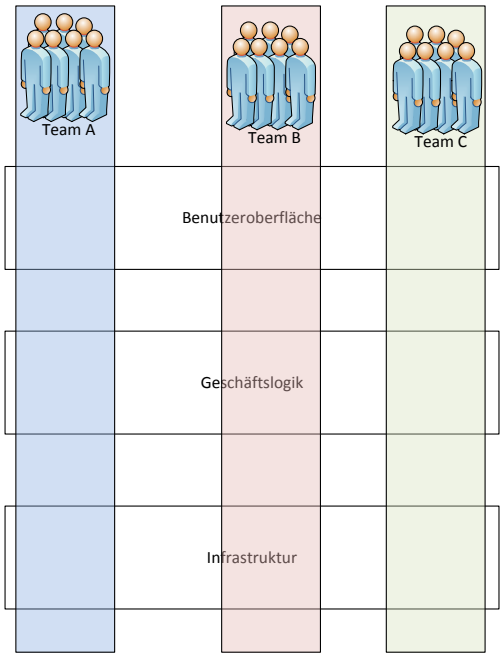

Abbildung 2.5.: Featureteams nach Pichler (2009) S. 139

Hierdurch ergeben sich folgende Vorteile:

- Entwicklung nach Stakeholdersicht.
- Der Aufwand für Anforderungssplitts durch die direkte Verwendung des Backlogs wird minimiert.
- Jedes Team besitzt Spezialisten der jeweiligen Schichten.
- Ein Produktinkrement kann von einem Team umgesetzt werden.
- Integration der Software wird vereinfacht.
- Unterstützung der Teams untereinander ist möglich. Vermeidung von *Bottlenecks*.
- Collective Code Ownership wird verstärkt. Alle Teams arbeiten an allen Komponenten.

Nachteile:

- Integrität und Konsistenz der Gesamtarchitektur wird erschwert.
- Es werden Alleskönner für die jeweiligen Teams gesucht.
- Teambildung dauert länger und Alleskönner sind in der Regel langsamer als Spezialisten.

Diese Variante bietet sich besonders bei einer inkrementellen Entwicklung einer Architektur an (vgl. Pichler 2009, S.140).

2.2.1.3. Mischformen

Nun kann es eine Vielzahl von Mischformen geben, welche je nach Stärke des Unternehmens gewählt wird. Auch je nachdem, ob an einem oder mehreren Standorten entwickelt wird, trifft man unterschiedliche Entscheidung für die jeweilige Struktur der Teams. Man wird somit versuchen, die Nachteile der jeweiligen Variante im Unternehmen durch eine Kombination mit der anderen Variante zu minimieren.

Eine Beispielvariante wäre, dass es drei Teams gibt. Zwei Teams agieren als Featureteams über die Benutzeroberfläche und der Geschäftsschicht. Das dritte Team fungiert als Komponententeam, bezogen auf die Infrastrukturschicht (vgl. Pichler 2009, S.140).

2.2.2. OpenSource

Aufgrund der Projektart und der damit verbunden Größe können unter anderem mehrere Teilteams gebildet werden, welche wiederum verstreut über mehrere Standorte arbeiten (Abbildung 2.6).

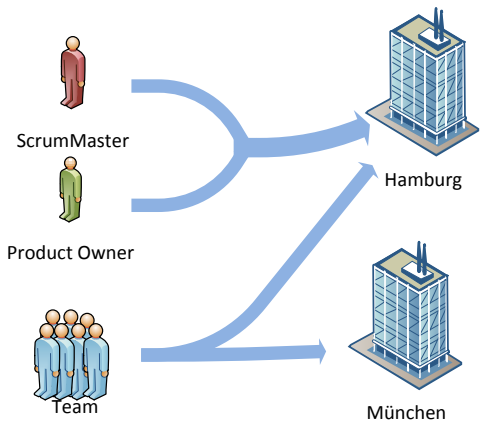

Abbildung 2.6.: Kleines verstreutes Team nach dem praktischen Beispiel

Ein Grund für die Projektart wäre z. B. ein OpenSource Projekt, wo sich das Team und auch Teilteams über Landesgrenzen hinweg splitten. Auch das betrachtete Team eines großen deutschen Unternehmens, welche für die Untersuchungen ausgewählt wurde, ist ein verstreutes Team. Diese Variante der Teamteilung, welche nicht entlang der Teamgrenze führt, ist nach Pichler 2009 (S.154-155) zu vermeiden. Vielmehr sollte man versuchen, wenn es schon nicht möglich ist für jeden Standort einen Product Owner zu stellen, mindestens ein Scrum Master je Standort zur Verfügung zu haben.

Im Gegensatz zur dieser Aussage, dass verstreute Teams gewisse Problematiken beinhalten, erzielen die Open-Source-Projekte Erfolge. Dabei handelt es sich sich um meist verstreute Teams und Teilteams, die gemeinsam ein Projekt entwickeln. Allerdings kann man auch hier beobachten, dass sich zentrale Personen innerhalb der Projekte treffen, um gemeinsam und persönlich über spezielle Themen in Projekten sprechen (vgl. Eckstein 2012, S.80). Direkte Kommunikation ist immer zu bevorzugen und sollte unter der Beachtung des *Return on Investment* (ROI) berücksichtigt werden. Durch die zunehmende Entwicklung bei der Verteilung von Teams (vgl. VersionOne 2013, S.3) werden neuen Lösungen dazu beitragen, die Kosten für die technischen Unterstützungen zu reduzieren.

2.3. Toolunterstützung für agile Prozesse

Im Rahmen der agilen und zugleich verteilten Softwareentwicklung sind eine Vielzahl von Softwarelösungen zur Projektorganisation entstanden. Dabei ist die Atlassian Suite eine der bekanntesten Lösungen.

Atlassian[2] ist ein seit 2002 existierendes Unternehmen, welches mit seiner Suite Bereiche wie Projekt- und Ticket-Verfolgung, Kollaboration, Content Sharing, DVCS-Lösungen und Sicherung von Code-Qualität in Form von miteinander agierenden Softwarelösungen bereitstellt. Insbesondere ist die Software interessant, da diese versuchen, den *Entprise 2.0* Gedanken aufzufassen, aktuelle Entwicklungsmodelle zu integrieren und eine vollständige Integration aller Produkte über das Projektmanagement- und Ticketverfolgungssystem JIRA zu stellen. Zudem wird eine Vielzahl von Addons über den *Atlassian Marketplace* zur Erweiterung bereitgestellt. Bei der Weiterentwicklung der Produkte wird stark auf die Community eingegangen und eine offene Schnittstelle angeboten. Besonders durch die Community und die angebotenen gut dokumentierten Schnittstellen hat sich über den Atlassian Marketplace ein eigener Wirtschaftszweig gebildet.

2.3.1. JIRA

Das Ticketverfolgungs- und Projektmanagement-Tool JIRA stellt das Herz der Atlassian Produkte dar. Tickets stellen Abbildungen der Aufgaben dar. So gut wie jedes andere Produkt aus dem Hause Atlassian, aber auch externe Systeme, lassen sich mit diesem verbinden. Viele andere Produkte wie z. B. ehemals Greenhoper nun JIRA Agile sind als Erweiterung des JIRA-Systems zu betrachten. Das System selber bietet die Möglichkeit, seine Projekte zu planen, zu verfolgen und zu koordinieren. Hierbei steht der Fokus der Enduser bei der Entwicklung von Software.

2.3.2. Greenhopper/Agile

Greenhoper nun JIRA Agile genannt ist eine Erweiterung von JIRA zur Unterstützung und Verfolgung von aktuellen Entwicklungsmodellen. Zur Verfügung stehen hierzu Scrum und

[2] https://www.atlassian.com/ - Abruf am 28.04.2014

Kanban (eine weitere agile Vorgehensweise). Dieses Tool bietet für Scrum die Möglichkeiten, einen Sprint zu planen und zu verfolgen. Hierzu stehen Monitoringmöglichkeiten wie das Taskboard zum Betrachten der Tickets und Charts, wie dem *Burn Down Chart* (Darstellung des Sprint-Fortschritts, offene/geschlossene Aufgaben vom Start bis zum Ende des Sprints). Ebenfalls kann das Taskboard von Entwicklern zum *pullen* (das eigenständige auswählen von Aufgaben) der Aufgaben verwendet werden. Es versucht herkömmliche Standarddarstellungen des Entwicklungsprozesses zu digitalisieren. Hierdurch werden Aufgaben besser verfolgbar und müssen nicht aufwendig von einer Offline-Taskboard-Variante in das Ticketsystem nachgetragen werden, sondern können direkt im JIRA-System bearbeitet werden.

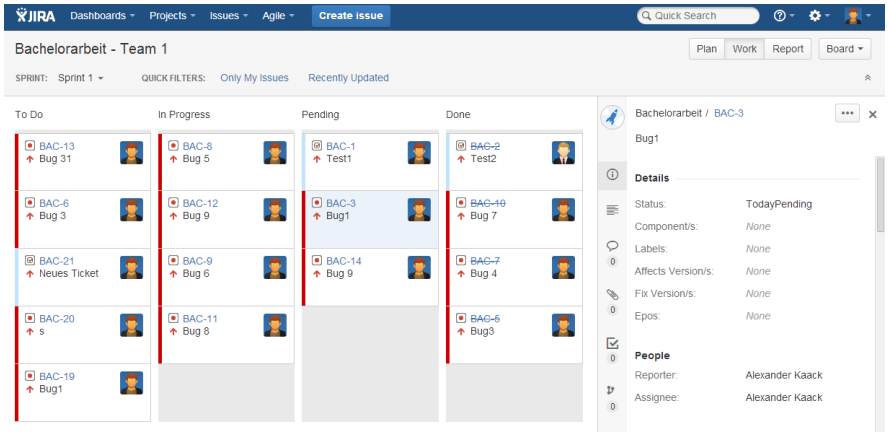

Abbildung 2.7.: JIRA Agile-Board

2.3.3. Schnittstellen

Es werden seitens Atlassian's verschiedene Schnittstellen angeboten. Frühere Versionen stellten XML-RPC[3] und JSON-RPC[4] als Schnittstellen zur Verfügung, welche aber als nicht zukunftsrelevant eingestuft wurden und somit wegfallen. Für die aktuellen und zukünftigen Versionen beschränkt sich die Schnittstelle auf REST[5][6] um Daten von JIRA zu beziehen oder JIRA mit neuen Daten zu versorgen. Weiterhin gibt es die Möglichkeit, direkt Plugins[7] für JIRA und Agile JIRA zu schreiben, welche unmittelbar auf dem System aufbauen.

[3] Extensible Markup Language Remote Procedure Call - Entfernter Prozeduraufruf im XML-Format
[4] JavaScript Object Notation Remote Procedure Call - Entfernter Prozeduraufruf im JSON-Format
[5] Representational State Transfer - Erläuterung im Abschnitt 3.3.1.3
[6] https://developer.atlassian.com/display/DOCS/Developing+a+REST+Service+Plugin - abgerufen am 28.04.2014
[7] https://developer.atlassian.com/display/JIRADEV/Tutorials - abgerufen am 28.04.2014

2.3.4. Alternativen

Alternative zu Atlassian's JIRA gibt es weitere Produkte auf dem Markt, die gleichwertige Funktionen anbieten. Die bekanntesten sind dabei Trac[8], Bugzilla[9] und FogBugz[10]. Im Bezug auf die digitale Unterstützung von agilen Prozessen bzw. Scrum ist die einzige bekannte, vergleichbare Alternative zu JIRA mit JIRA Agile, Agilo for Trac[11]. Dieses baut auf dem Open Source Projekt Trac auf und bietet ebenfalls eine Abdeckung der Funktionen Planen, Koordinieren und Verfolgen von Scrum Prozessen mit der entsprechenden Darstellung über ein Board. Im Falle der im nächsten Abschnitt beschriebenen Praxisumgebung wird JIRA mit JIRA Agile verwendet und als Informationslieferant für die geplante Anwendung bereitgestellt.

2.4. Daily Scrum in der Praxis

Das Daily Scrum ist ein Meeting-Ritual innerhalb eines Scrum Sprints, welches täglich durchgeführt und auf 15 Minuten zeitlich begrenzt ist (*Timeboxed*) (vgl. Wirdemann 2011, S.158). In der Regel ist der Ort und Startzeitpunkt fix. Die Teilnehmer unterscheiden sich allerdings. Grundsätzlich wird innerhalb des Meetings zwischen *Pig* und *Chicken* (Hühner und Schweinen) unterschieden.

> „Dies hat seinen Ursprung in einer Fabel, in der ein Huhn einem Schwein vorschlägt, ein Restaurant namens ‚Eier und Speck'zu eröffnen. Das Schwein lehnt dies mit der Begründung ab, dass das Huhn nur durch die Bereitstellung von Eiern involviert sei, das Schwein jedoch sein eigenes Leben opfern müsste." ... „In der Daily Scrum haben nur die Schweine Rederecht. Hühner, also Interessenvertreter, sind willkommen, dürfen sich aber nicht einmischen." (Pichler 2009, S.104)

Je nach Literatur lässt sich nicht klar definieren, ob der Product Owner den „Schweinen" oder „Hühnern" beiwohnt (vgl. Wirdemann 2011, S.161 und Pichler 2009, S.104) und ob der Scrum Master immer ein „Schwein" bleibt. Deutlicher wird hierbei, dass das Meeting dazu dient, das Team neu zu organisieren, auf den aktuellen Stand zu bringen und Hindernisse zu identifizieren. Die zwei Hauptaufgaben des Scrum Master sind erstens, Hindernisse zu

[8] http://trac.edgewall.org/ - abgerufen am 28.04.2014
[9] http://www.bugzilla.org/ - abgerufen am 28.04.2014
[10] https://www.fogcreek.com/fogbugz/ - abgerufen am 28.04.2014
[11] http://www.agilofortrac.com/ - abgerufen am 28.04.2014

beseitigen und zweitens, dafür zu sorgen, dass die Prinzipien von Scrum eingehalten werden. Durch die angestrebte Selbstorganisation des Teams (vgl. Wirdemann 2011, S.39) hat der Scrum Master somit in späteren Meetings unter Umständen keine wichtigen Informationen, welche die Entwickler direkt voranbringen. Wichtig ist aber, dass der Scrum Master vor allem als Moderator im Daily Scrum agiert. Hierzu gehört die Kontrolle des Ablaufs. Zusätzlich zur zeitlichen Restriktion gehören die 3 Fragen:

1. Was habe ich seit dem letzten Daily Scrum erreicht?
2. Was plane ich bis zum nächsten Daily Scrum?
3. Welche Hindernisse gibt es?

(Wirdemann 2011, S.158)

Auch sollte der Scrum Master drauf achten, dass kein Teilnehmer ins *Story Telling* oder *Problem Solving* verfällt. Story Telling bedeutet, dass man während des Meetings in den Erzählstil verfällt. Aufgaben sollten kurz und präzise beschrieben werden, weitere Informationen können nach dem Meeting ausgetauscht oder aus dem Ticketsystem bezogen werden. Hierdurch kann unter Umständen sonst auch das Problem Solving provoziert werden. Probleme sollten nicht ausgiebig während des Meetings diskutiert werden. Auch hierzu sollten Gespräche nach dem Meeting dienen und auf diese verwiesen werden.

Das Meeting wird auch als Stand-up Meeting bezeichnet, da die Teilnehmer generell im Stehen berichten. Zur Unterstützung wird oft ein *Speech Token* genutzt, welcher z. B. ein Ball oder Stift sein kann und daran erinnert, wer spricht, um Unterbrechungen zu vermeiden (Pichler 2009, S.106).

Um praxisbezogene Erkenntnisse zu gewinnen, wird das Daily Scrum in einem großen deutschen Unternehmen im Dienstleistungsgewerbe als Beispiel herangezogen. Aufgrund von Datenschutzbestimmungen wird hierfür kein Firmenname und keine Personennamen genannt. Indizien zur eindeutigen Identifikation der Firma werden bewusst verschleiert. Essentielle Bestandteile zur Erfassung von Informationen bleiben erhalten.

Die erwähnte Firma ist deutschlandweit mit ca. 320 Softwareentwicklern besetzt und die Untersuchungen beziehen sich dabei auf eine 15 köpfige Entwicklungsabteilung. Am Standort Hamburg sind zum einen ein Scrum Master und zum anderem zwei Product Owner ansässig. Grundsätzlich ist das Team auf zwei Standorte in Deutschland verstreut. Der eine Standort

befindet sich in München, der andere in Hamburg. Das Daily Scrum wird täglich über eine Telefonkonferenz geführt. Für das Sprint Planning, Sprint Review sowie die Sprint Retrospective trifft sich das Team an einem Ort und somit muss ein Teil des Teams reisen. Die Sprintlänge beträgt zwei Wochen. Zur Projektorganisation wird wie im vorherigen Abschnitt erwähnt, das Projektmanagement- und Ticketverfolgungssystem JIRA verwendet, um die Aufgaben zu koordinieren und zu verfolgen.

2.4.1. Vorgehen

Zur Erfassung des Ist-Zustands werden drei Techniken der Systemanalyse angewendet, um eine möglichst große Bandbreite und Tiefe an Informationen zu erhalten. Die Informationen werden über eine Beobachtung über drei Termine gewonnen. Zusätzlich wird am letzten Termin ein Gruppeninterview durchgeführt, sowie ein Fragebogen ausgehändigt.

	Interview	**Fragebogen**	**Beobachtung**
Gewinnbare Informations-kategorie	Ist-Zustand, Verbesserungen, Soll-Zustand	Ist-Zustand, Verbesserungen	Ist-Zustand
Informations-tiefe	Groß	Mittel	Gering
Informations-breite	Gering	Groß	Gering
Benutzer-beteiligung	Groß	Mittel	Gering
Kosten	Groß	Mittel	Gering

Abbildung 2.8.: Eigenschaften von Techniken der Informationsgewinnung nach Häuslein (2004) S. 62

Wie in Abbildung 2.8 dargestellt ist, weisen unterschiedliche Techniken verschiedene Eigenschaften auf. Um eine möglichst große Informationstiefe zu erreichen, wurde das Interview mit offenen Fragen ausgewählt. Unter der Informationstiefe ist der Grad der Detaillierung gemeint, also was man z. B. an Informationen mit einer Frage erhält.

Um ebenfalls die Informationsbreite im größeren Maße abzudecken, wurde auch ein Fragebogen entwickelt. Unter der Informationsbreite wird die Anzahl der Teilnehmer oder Informationsquellen verstanden.

Die Beobachtungen dienen der ersten Erfassung von Informationen sowie zur Gestaltung der offenen Frage im Gruppeninterview und der Erstellung der Skalarfragen im Fragebogen. Die Beobachtung selbst wird dabei am Standort Hamburg durchgeführt, wie auch das Gruppeninterview. Der Fragebogen wird an alle Meetingteilnehmer am letzten Termin ausgehändigt. Das Gruppeninterview findet dabei bewusst vor der Fragebogenbeantwortung und ausschließlich in Hamburg statt. Hierbei wird jedoch eine spezielle Unterscheidung ausgewählt:

1. Fragebogenauswertung der Teilnehmer, welche am Gruppeninterview teilgenommen haben.

2. Fragebogenauswertung der Teilnehmer, welche *nicht* am Gruppeninterview teilgenommen haben.

Das Ziel dieser Strategie ist es, in Hamburg durch das vorherige Interview ein Bewusstsein für die Befragung mit dem Fragebogen zu schaffen. Im Vergleich zu den Ergebnissen aus München wird ein deutlicheres Ergebnis in Richtung Wandel und Veränderungen der aktuellen Daily Scrum-Durchführung, durch die angestrebte Diskussion im Interview erhofft. Alle verwendeten Hilfsdokumente, welche zur Gewinnung der Informationen entworfen wurden, finden sich im Anhang (A.1) wieder.

2.4.1.1. Beobachtung

Als Methode der Beobachtung wird das sogenannte Multimomentverfahren verwendet. Im Multimomentverfahren werden einzelne Termine zur Beobachtung genutzt. Anschließend wird über eine statistische Auswertung ein Ergebnis abgeleitet (vgl. Heinrich 2007, S.33). Um während der Beobachtung dem Beobachtenden eine Hilfestellung zu geben, wird ein vorgefertigtes Erfassungsformular mit Notizmöglichkeiten angefertigt.

Dieses erfasst:

1. Dauer.

2. Anwesende Rollen und Teilnehmer.

3. Räumlichkeiten.
4. Eingesetzte Medien.
5. Tickets (Erfassung aller genannten Tickets mit der jeweiligen Erwähnungsdauer).
6. Besonderheiten (Ablauf und besondere Beobachtungen).

2.4.1.2. Interview

Das Interview wird, wie bereits erwähnt, als Gruppeninterview geführt und findet ausschließlich in Hamburg nach der Beobachtung statt. Hierzu wurde folgend formuliert:

1. Seht Ihr Störungsfaktoren im Daily Scrum?
2. Was könnte zusätzlich zur Effizienz des Meetings beitragen?
3. Zusätzliche Bemerkungen.

Die erste Frage soll bewirken, dass bereits erfasste Probleme von Teilnehmern genannt werden. Ebenfalls sollen die Anderen im Interview damit konfrontiert werden. Daraus könnte resultieren, dass sie bereits identische Konflikte erkannt haben oder erst jetzt die Probleme sehen können. Über die Frage 2 sollen Lösungsvorschläge aufgezeigt und diskutiert werden. Dabei soll die geplante Anwendung selbst nicht im Fokus stehen. Vermutet wird, dass keine mögliche Softwarelösung genannt wird, sondern sich die Diskussion lediglich auf das Ausstattungsdefizit fokussiert. Durch eine Hilfestellung ohne zusätzliche Hardware soll zudem eine höhere Begeisterung der Anwendung hervorgerufen werden. Dies soll speziell durch die geringfügige Änderung der Ausstattung (nur eine neue Sicht des Boards) und des Ablaufes (der User interagiert mit dem Board) erfolgen. Die letzte Frage soll Platz für weitere Punkte im Zuge der Diskussion aufzeigen. Zur detaillierteren Auswertung soll nach dem Einverständnis aller Teilnehmer das Interview mit einem Diktiergerät aufgenommen werden.

2.4.1.3. Fragebogen

Der Fragebogen wurde, wie bereits im Vorfeld erläutert, auf Basis von Beobachtungen angefertigt und umfasst die folgenden Aussagen:

1. Deine Vorbereitungszeit zum Daily Scrum beträgt:
2. Wenn ein Ticket-Kürzel genannt wird, weißt Du sofort worum es geht.
3. Die eingesetzten Medien: Monitor + Maus und Telefon mit Lautsprecher reichen völlig aus.
4. Das Daily Scrum der verstreuten Teams wird durch die eingesetzten Medien sinnvoll unterstützt.
5. Der aktuelle Ablauf und die Struktur des Daily Scrums ist optimal.

Die erste Aussage gilt der Erfassung der Vorbereitungszeit und soll aufdecken, ob sich eventuell Teilnehmer längere Zeit damit aufhalten müssen, um sich einen Überblick darüber zu beschaffen, woran sie arbeiten, was sie als Nächstes vor haben und was sie davon abhalten könnte. Bei der Aussage 2 bis 5 kann mit „1 trifft voll zu“ , „2 trifft zu“ , „3 trifft eher zu“ , „4 trifft eher nicht zu“ , „5 trifft nicht zu“ oder „6 trifft gar nicht zu“ geantwortet werden. Dieses Auswertungsschema ermöglicht es, immer die Entscheidung für oder gegen einen Punkt zu fällen. Es existiert somit keine neutrale Bewertung. Die Aussagen 2 bis 5 zielen auf direkte Beobachtungen ab und sollen diese bestätigen.

Den Ergebnisse vorweggenommen wurden die Aussagen 2 und 3 im Hinblick auf die entdeckte Orientierungslosigkeit entworfen und zusätzlich soll die Aussage 3 die ungenügende Ausstattung bestätigt. In Bezug auf die Aussage 4, welche nicht nur auf den Beobachtungen beruht, soll neben der Übereinkunft auch eine Vergleichsmöglichkeit für *vor der Einführung der Software* und *nach der Einführung der Software* geboten werden.

Die Ergebnisse der vorgestellten Erfassungsverfahren werden im nächsten Unterkapitel beschrieben.

2.4.2. Auswertung und Ergebnisse der Ist-Analyse

In den nachfolgenden Unterkapiteln werden die Ergebnisse der Beobachtung, des Gruppeninterviews, sowie der Fragebogenauswertungen dargestellt und bewertet. Zusätzlich sollen im Zwischenfazit die Ergebnisse zusammengefasst werden, um somit eine abschließende Bewertung aus den resultierenden Ergebnissen der verwendeten Informationsgewinnungsmaßnahmen zu geben.

2.4.2.1. Beobachtung

Die Beobachtungen fanden am 20.11.2013, 27.11.2013 und 28.11.2013 am Standort Hamburg statt. Die Dauer aller drei Termine lag immer im Rahmen der maximal vorgegebenen 15 Minuten eines typischen Daily Scrums. Die Räumlichkeiten an den jeweiligen Standorten fanden in regulären Büros statt, wobei sich nur Teilnehmer des Meetings im Raum befanden. Die anwesenden Teilnehmer sind der Tabelle 2.1 zu entnehmen.

	20.11.2013	**27.11.2013**	**28.11.2013**
Teilnehmer München	2 Entwickler, beide männlich	1 Entwickler, männlich	2 Entwickler, beide männlich
Teilnehmer Hamburg	2 Entwickler und Product Owner, alle männlich	1 Entwickler und Scrum Master, beide männlich	3 Entwickler und Product Owner, alle männlich

Tabelle 2.1.: Teilnehmer

Eingesetzte Medien waren:

- Maus und Tastatur.
- Monitor mit Sicht auf das JIRA-Taskboard.
- Telefon mit Hörer und aktivierter Lautsprecherfunktion.

Der Hörer des Telefons wurde als Speech Token genutzt. Der Ablauf des Meetings orientierte sich stark an der *normalen* Vorgehensweise eines Daily Scrums und die Fragen (Daily Scrum-Fragen) wurden aufgegriffen. Jedoch gab es immer wieder Teilnehmer, die sich regelmäßig nicht daran gehalten haben.

Dauer	**20.11.2013**	**27.11.2013**	**28.11.2013**
<= 5 Sekunden	2	1	6
> 5 Sekunden	5	3	9

Tabelle 2.2.: Genannte Tickets

Es war zu beobachten (Tabelle 2.2), dass der Abstand zwischen Ticketkürzeln kleiner gleich 5 Sekunden im Schnitt bei 46,67 % lag. Die Vermutung liegt nahe, dass dieser zeitliche Rahmen ungenügend zur Identifizierung der Bedeutung hinter einem Ticketkürzel ist. Nur Teilnehmern,

denen das Tickets bereits eindeutig bekannt ist, können eine Bedeutung ableiten. Auch in der Beobachtung wurde deutlich, dass bei der Nennung eines Tickets alle direkt zum Monitor schauten, um dieses Ticket zu finden. Hierzu wurde ab und zu die Maus genutzt, um schnell noch ein Ticket zu lokalisieren. Allerdings musste sich, je nach Positionierung im Raum, ein hinten stehender Teilnehmer erst nach vorne drängen. Zusätzlich war die Größe des Monitors (19 Zoll) ebenfalls ein Hindernis (Abbildung 2.9). Besonders bei den schnell hintereinander erwähnten Tickets ließ sich eine deutliche Orientierungslosigkeit feststellen.

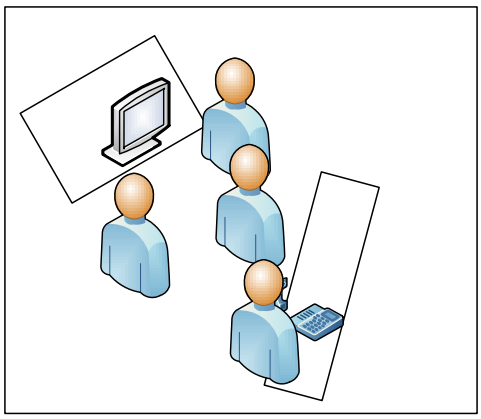

Abbildung 2.9.: Skizze Büro Hamburg

Darüber hinaus fiel auf, dass ein Story Telling und Problem Solving statt fand. Das Story Telling beruhte hierzu oft auf den nicht erfassten Tickets. Eine Abschwächung der Bedeutung lässt sich durch die generelle Zusammensetzung des Teams vornehmen. Das Team koordiniert und bearbeitet Aufgaben aus verschiedenen internen Projekten. Hieraus ergeben sich eine Vielzahl von notwendigen Tätigkeiten außerhalb der *normalen* Software-Entwicklung, welche man aber auch als Ticket erfassen könnte.

2.4.2.2. Gruppeninterview

Das Interview hat folgende Ergebnisse erbracht: Zur ersten Frage „Seht Ihr Störungsfaktoren im Daily Scrum?“ war gut erkennbar, dass die erste Antwort in Richtung, „wir haben keine Störungen“ ging. Nach dieser Aussage wurde direkt eine Diskussion ausgelöst und Faktoren herausgearbeitet. Identifiziert wurden:

1. Grundsätzliches Equipment.
2. Die Sprachqualität über das Telefon.
3. Dass man seine Kollegen in München nicht sieht.
4. Räumlichkeiten sind ungeeignet.
5. Monitor ist zu klein, um die Tickets zu erfassen.

Zur Lösung bzw. auf die Frage 2 „Was könnte zusätzlich zur Effizienz des Meetings beitragen?“ gab es folgende Antworten:

1. Besseres Equipment.
2. Größere Monitore mit Sprach- und Videoübertragung sowie das Board auf einem Blick.
3. Ein Monitor an dem nicht gescrollt werden muss.
4. Die Größe des Monitors sollte circa die Größe eines durchschnittlichen Whiteboards haben.

Als zusätzliche Bemerkung und letzte Frage gab es diese Antworten:

1. Es sollten niemals Kollegen aus dem Meeting gezogen werden.
2. Es fällt manchen schwer, sich an dem Standardablauf und die darin enthaltenen Fragen zu richten.

Durch das geführte Gruppeninterview wurden Beobachtungen der vorherigen Termine bestätigt. Besonders die Problematik der Orientierungslosigkeit auf dem Monitor und die mangelnde Grundausstattung wurden erwähnt. Interessant war auch, dass sich Teammitglieder erst nach dem Erwähnen eines Einzelnen dazu bekannten, dass es Probleme gibt. Dies lässt vermuten, dass diese Befragung vor dem Fragebogen eine Auswirkung auf deren Antworten hat.

2.4.2.3. Fragebogen

Teilgenommen haben insgesamt 7 Teilnehmer. Davon waren 3 Entwickler, einer Product Owner und einer Scrum Master. Die Fragen wurden in Hamburg direkt nach dem Meeting beantwortet

und eingesammelt. Die Teilnehmer in München haben eine digitale Fassung erhalten und reichten die Frage über E-Mail nach. Die Antwort auf die erste Frage „Deine Vorbereitungszeit zum Daily Scrum beträgt?“ hat fünf Minuten niemals überschritten.

Die restlichen Fragen konnten jeweils mit „1 trifft voll zu“ , „2 trifft zu“ , „3 trifft eher zu“ , „4 trifft eher nicht zu“ , „5 trifft nicht zu“ oder „6 trifft gar nicht zu“ beantwortet werden. Im folgenden wird zur jeder Frage ein Median, ein 25 % Anteil, ein 75 % Anteil, der Mittelwert, die minimale Antwort (in Richtung 1) und die maximale Antwort (in Richtung 6) angegeben. Zusätzlich wird unterschieden zwischen allen Rückmeldungen, den Antworten nach dem geführten Gruppeninterview und Antworten ohne Gruppeninterview.

Frage 2			
Wenn ein Ticket-Kürzel genannt wird, weißt Du sofort worum es geht.			
	Alle Rückmeldungen	**Mit Gruppeninterview**	**Ohne Gruppeninterview**
Median	4	4	2
25 % Anteil	2,5	3,75	2
75 % Anteil	4	4	3,5
Mittelwert	3,43	3,75	3
Min	2	3	2
Max	5	4	5

Tabelle 2.3.: Frage 2 aus Fragebogen

Wie in Tabelle 2.3 zu sehen ist, ist aus den Antworten des Gruppeninterviews ersichtlich, dass grundsätzlich nicht immer erkannt wird, was hinter einem Ticket steht. Interessant ist hier auch der Unterschied zwischen den Antworten mit und ohne dem Gruppeninterview. Zwar gab es ohne das Gruppeninterview auch die Antwort „trifft nicht zu“ , trotzdem ergibt das Mittel „trifft zu“. Man könnte hier nun interpretieren, dass sich durch das Gruppeninterview die Antworten doch eher Richtung „trifft eher nicht zu“ bewegt haben und sonst das Ergebnis eher Richtung „trifft eher zu“ ausfallen würde. Dies wird durch die entfachte Diskussion im Gruppeninterview, welche gezeigt hat, dass einige Anfangs anderer Meinung waren und das Gespräch dazu beigetragen hat, dass selbst diejenigen ihre Meinung revidierten, welche Anfangs anderer Meinung waren, zusätzlich bekräftigt.

Frage 3			
Die eingesetzten Medien: Monitor + Maus und Telefon mit Lautsprecher reichen völlig aus.			
	Alle Rückmeldungen	**Mit Gruppeninterview**	**Ohne Gruppeninterview**
Median	4	5	3
25 % Anteil	3,5	4,75	2,5
75 % Anteil	5	5	3,5
Mittelwert	4	4,75	3
Min	2	4	2
Max	5	5	4

Tabelle 2.4.: Frage 3 aus Fragebogen

Die Tabelle 2.4 zeigt, dass dieses Mal der Unterschied zwischen beiden Fragebogengruppen deutlicher ist. Auch hier geht die Gruppe mit dem Interview Richtung *nicht ausreichender Ausstattung* und interessanterweise würden die Teammitglieder ohne das Gruppeninterview die Ausstattung als *genügend* bezeichnen.

Frage 4			
Das Daily Scrum der verstreuten Teams wird durch die eingesetzten Medien sinnvoll unterstützt.			
	Alle Rückmeldungen	**Mit Gruppeninterview**	**Ohne Gruppeninterview**
Median	4	4	3
25 % Anteil	3	3,75	2,5
75 % Anteil	4,5	4,25	4
Mittelwert	3,71	4	3,33
Min	2	3	2
Max	5	5	5

Tabelle 2.5.: Frage 4 aus Fragebogen

Die Tabelle 2.5 zeigt ein ähnliches Bild wie bei Frage 2. Auch hier lässt sich vermuten, dass ein Interview mit allen Befragten zu einem eher negativen Ergebnis geführt hätte. Es ist insgesamt aber eine „trifft eher nicht zu“ Antwort abzuleiten.

Frage 5			
Der aktuelle Ablauf und die Struktur des Daily Scrums ist optimal.			
	Alle Rückmeldungen	**Mit Gruppeninterview**	**Ohne Gruppeninterview**
Median	4	4	3
25 % Anteil	3	3,75	2,5
75 % Anteil	4	4	3,5
Mittelwert	3,43	3,75	3
Min	2	3	2
Max	4	4	4

Tabelle 2.6.: Frage 5 aus Fragebogen

Auch in Tabelle 2.6 ist der Unterschied wieder sehr knapp und mit dem Gruppeninterview eher Richtung „trifft eher nicht zu“ zu deuten.

2.4.2.4. Zwischenfazit

Ein Großteil der Beobachtungen konnte im Gruppeninterview und auch mit dem Fragebogen bestätigt werden. Hierbei war die Bestätigung der ungenügenden Ausstattung zur Überbrückung der räumlichen Distanz als Feedback des Teams am deutlichsten. Es ist zu erkennen, dass die Tickets nicht korrekt am Monitor erfasst werden können und daraus eine Orientierungslosigkeit resultiert. Dieses Problem wurde innerhalb der geführten Debatten im Interview auch erkannt. Besonders das Gruppeninterview hat auch in der Verbindung mit dem Fragebogen gezeigt, dass die Diskussion selber zu Transparenz dieses Problems beigetragen hat. Als nächster Schritt wird die Aufnahme der Anforderungen vorgenommen, welche einen Lösungsansatz zur erfassten Orientierungslosigkeit bieten soll.

2.5. Anforderungen

Die gewonnen Erkenntnisse aus den Beobachtungen, dem Gruppeninterview und dem Fragebogen dienen zur Aufnahme der folgenden Anforderungen. Diese sollen einen Lösungsansatz zur Orientierungslosigkeit und der optimalen Ausnutzung der vorhanden Ausstattung bieten. Zusätzlich wird das vorhandene JIRA-System eine Grundvoraussetzung zur Informationsbeschaffung sein. Hierzu werden die Anforderungen für eine Softwarelösung zur Unterstützung

eines Daily Scrum nachfolgend in Benutzer- und Systemanforderungen unterteilt. Dafür sind sowohl funktionale wie auch nichtfunktionale Anforderungen notwendig. Dabei beziehen sich diese nicht nur auf die Anforderungen des Unternehmens, sondern auch auf visionäre Anforderungen für zusätzliche Einsatzmöglichkeiten.

> „Funktionale Anforderungen: Dies sind Aussagen zu den Diensten, die das System leisten sollte, zur Reaktion des Systems auf bestimmte Eingaben und zum Verhalten des Systems in bestimmten Situationen." (Sommerville 2007, S.152)

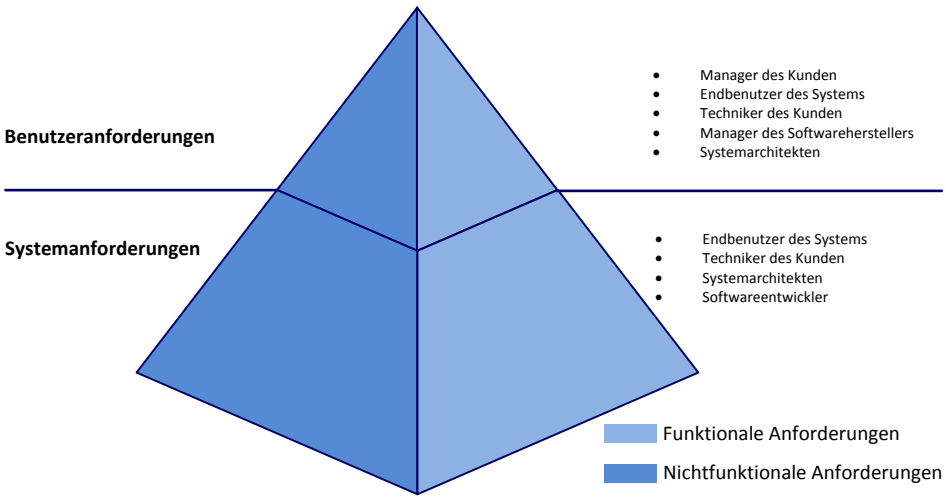

Abbildung 2.10.: Anforderungspyramide nach Schneider und Romano 2012, S.112 und Sommerville 2007, S.152

Nichtfunktionale Anforderungen beschreiben Beschränkungen zur Funktionalität des ganzen Systems (Sommerville 2007 S. 152). Wie in Abbildung 2.10 zu sehen ist, sind jeweils unterschiedliche Interessengruppen den Anforderungsarten zuzuordnen. Zur fachlichen Gliederung werden die nichtfunktionalen und funktionalen Anforderungen zusammengefasst und im Rahmen der Benutzeranforderungen Kategorien zugeordnet.

2.5.1. Benutzeranforderungen

Benutzeranforderungen stellen Anforderungen auf einem fachlichen Abstraktionslevel dar. Das Verständnis hinter diesen sollte ohne ein technisches *KnowHow* möglich sein. Der Zweck ist dabei der Transfer zum Systembenutzer (vgl. Sommerville 2007, S.159). Wie bereits beschrieben, werden nun die Anforderungen in Gruppen unterteilt. Hierzu sind die Kategorien Integration, Bedienbarkeit und Verteilung definiert worden. Die folgenden Anforderungen sind zum Teil im Zusammenarbeit mit dem Unternehmen und zum anderen über logische Abhängigkeiten entstanden. Zusätzlich wurden mobile Endgeräte als eventuelle Clients mit einbezogen.

2.5.1.1. Integration

Die folgenden Anforderungen stehen im Zusammenhang mit der Integration in eine bestehende Systemlandschaft.

1. Die Anwendung muss bestehende Tickets aus JIRA/JIRA Agile beziehen.
2. Es sollen nur Tickets eines bestimmtes Sprints bzw. aus einem bestimmten Board aus JIRA bezogen werden.
3. Die Anwendung soll im Zuge der Evaluierung einfach installierbar sein.
4. Unberechtigte dürfen keinen Zutritt auf das System haben.

2.5.1.2. Bedienbarkeit

Im Zuge der Bedienbarkeit sind Anforderungen erfasst worden, die auch zum Teil in Verbindung mit dem Thema *Responsive Design* stehen. Bei Responsive Design geht es um die dynamische Darstellung, abhängig vom Endgerät (vgl. Meidl 2014, S.12). Hierzu ist die Auflösung im Zusammenspiel mit der Größe des Displays entscheidend bei der Konzeption der Benutzeroberfläche. Auch das Einbeziehen mehrerer *taktiler Interfaces* (vgl. Stapelkamp 2010, S.224-262) spielte bei den Anforderungen eine Rolle. Neben der Interaktionsmöglichkeit mit Maus und Tastatur wurde ebenfalls die Touchfunktionalität mit aufgegriffen. Zwar ist diese Funktion für das Unternehmen im Laufe der Evaluierung nicht relevant, soll aber in Hinsicht auf spätere Einsatzmöglichkeiten mit untersucht werden. Hierzu wurden zusätzlich spezielle Anforderungen gestellt, die nicht in der Evaluierungsumgebung ausgewertet werden.

1. Der Anwender soll Markierungen auf Tickets setzen können.
2. Tickets sollen *fassbar* sein.
3. Ein Ticket soll in einen neuen Status gesetzt werden können.
4. Die Anwendung soll via Touch bedient werden können.
5. Bedienbar mit unterschiedlichen Displayformaten. Geräte ab 4,7 Zoll mit einer Auflösung von 1280 x 768 Pixeln.
6. Die Anwendung soll auf aktueller Smartphone Hardware bedienbar sein (Android, IOS, WindowsRT).
7. Die Anwendung soll auf Surface Tischen bedienbar sein.
8. Die Anwendung soll eine möglichst intuitive Bedienung ermöglichen.
9. Es soll ermöglicht werden, dass die Anwendung ohne Hilfe eines Handbuches bedient werden kann. Ein Anwender muss aber ein Grundverständnis von Scrum, JIRA und dem jeweiligen Endgerät besitzen.

Relevant für das Unternehmen sind Nummer 1, 8, 9 und zum Teil 5, da eine Darstellung auf den Desktop-Monitoren notwendig ist.

2.5.1.3. Verteilung

Diese Kategorie soll die Anforderungen bezogen auf die Darstellung und den Mehrgewinn im Verteilungsaspekt definieren. Der vorherige Punkt hat bereits bzgl. der Darstellung einige Anforderungen direkt mit abgebildet, welche hier nun nicht mehr aufgegriffen werden.

1. Das System soll über mehre Standorte eine synchrone Darstellung eines ScrumBoards bieten.
2. Durch die Markierung eines Tickets sollen Zusatzinformationen dargestellt werden.
3. Eine Rückmeldung nach ausführen einer Aktion soll kleiner zwei Sekunden sein, wenn die Anwendung im LAN betrieben wird.

4. Das System soll das Verständnis der genannten Tickets verbessern.

5. Der Gesamtüberblick aller Aufgaben innerhalb eines Sprints soll verbessert werden.

6. Das System sollte die Kommunikation des verstreuten Teams verbessern.

2.5.2. Systemanforderungen

Die Systemanforderungen stellen eine Erweiterung der Benutzeranforderungen dar. Zum Entwerfen dieser Anforderungen werden Entwurfsbeschreibungen, grafische Notationen und mathematische Spezifikationen (vgl. Sommerville 2007, S. 162) verwendet. Die Anforderungen des zu entwerfenden Systems bzgl. der Reaktion auf Benutzerinteraktionen und den zu verwendenden Nachbarsystem werden nachfolgend grafisch mit zugehöriger Erläuterung aufgezeigt. Die Abbildung 2.11 soll hierzu dienen.

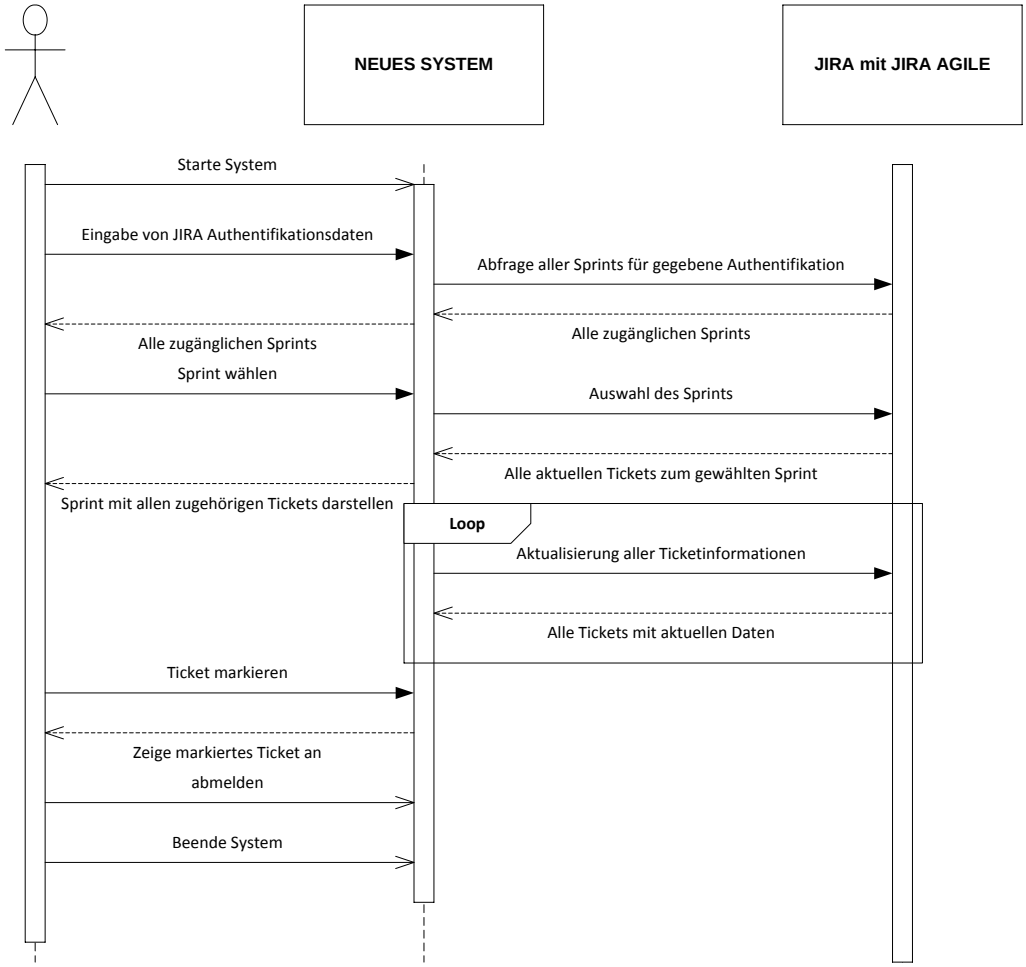

Abbildung 2.11.: Sequenzdiagramm

Das System soll über einen Systemuser gestartet werden können. Mittels einer Eingabe der JIRA-Authentifikation sollen dann alle möglichen Sprints, welche zur eingegebenen Authentifikation passen, für den Enduser zur Auswahl bereitstehen. Zur Kommunikation zwischen dem zu entwerfenden System und JIRA wird immer eine Authentifikation benötigt. Der User wählt somit ein Board aus und das System beschafft sich alle in dem Sprint befindlichen Tickets mit den zugehörigen Informationen. Nun werden regelmäßig alle Informationen zu den jeweiligen Tickets aktualisiert, falls Änderungen bestehen. Der User kann ein Ticket markieren und damit synchron zusätzliche Informationen für alle Clients darstellen. Das bedeutet, dass jeder User

die aktuelle Sicht des Boards sieht. Wird das Board bzw. die Authentifizierung nicht mehr benötigt, so kann der User sich abmelden bzw. das System beenden.

2.5.3. Anforderungen an das Team

Es wird beim Team vorausgesetzt, dass ein gewisser Wissenstand vorhanden ist. Hierzu zählt der Umgang mit JIRA bzw. JIRA Agile, welche die Boarddefinitionen und Tickets über einen kontinuierlichen Informationsimport für die geplante Anwendung bereitstellt. Das Grundverständnis der durch JIRA verwalteten Tickets ist somit eine Grundvoraussetzung für das Verständnis der zu entwickelnden Software.

2.6. Spezifikation

Zur Spezifikation werden des Weiteren Geschäftsprozesse und Anwendungsfälle definiert, um die Semantik der Anwendung näher zu erläutern.

2.6.1. Geschäftsprozesse

Geschäftsprozesse dienen zur Abbildung der relevanten Prozesse, welche durch eine Softwarelösung unterstützt werden sollen. In der Abbildung 2.12 ist ein typischer Ablauf des Daily Scrums der Firma aufgezeigt.

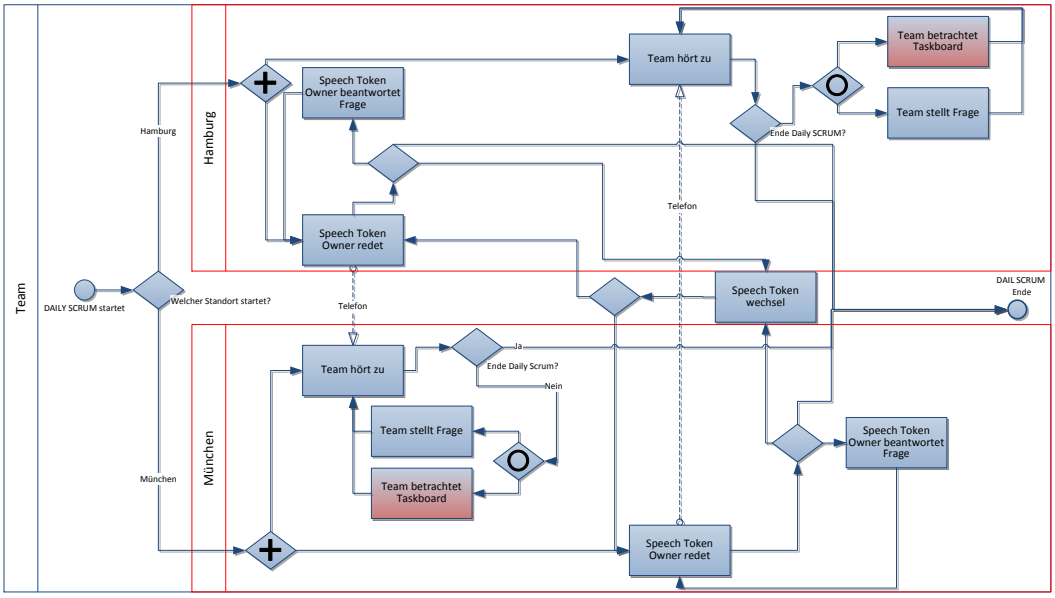

Abbildung 2.12.: BPMN 2.0 OMG (2011)

Das Daily Scrum beginnt an einem der beiden Standorte. An diesem Standort z. B. Hamburg startet der *Speech Token Owner* (Halter des Speech Tokens) und das lokale, wie auch das entfernte Team über das Telefon hören zu. In der Phase des Zuhörens kann das Team das Taskboard in JIRA betrachten oder Fragen stellen. Der Redner kann entweder Fragen beantworten oder weiter berichten. Ist dieser fertig, so wechselt der Redner entweder lokal oder auf den anderen Standort. Ist der letzte Redner bzw. Speech Token Owner fertig, so ist das Meeting beendet. Unterstützt werden soll hierbei die Aktivität „Team betrachtet Taskboard“ , welche in der Abbildung rot markierte wurde.

2.6.2. Anwendungsfälle

Die folgenden Anwendungsfälle oder auch *Use Cases* beschreiben das Verhalten der Anwendung, sowie deren Gewinn für die jeweiligen Akteure.

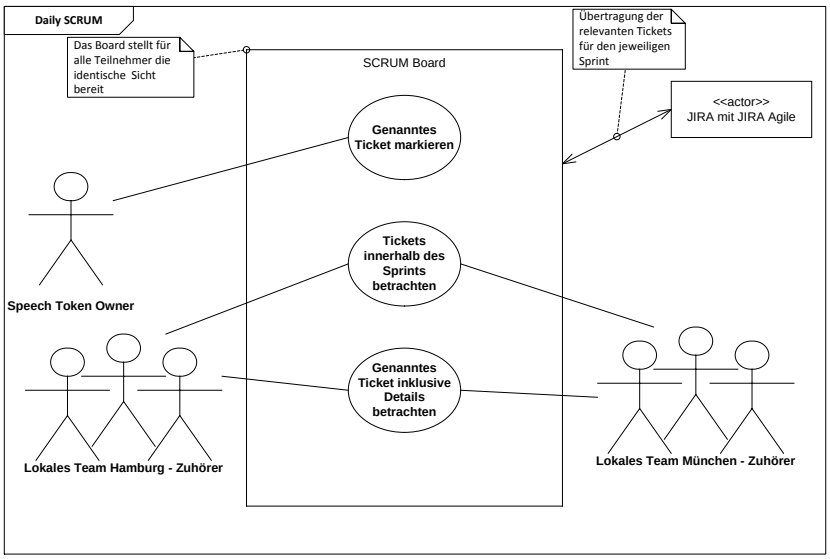

Abbildung 2.13.: Uses Cases Diagramm - Speech Token Owner in Hamburg

Wie in 2.13 zu sehen, soll die Anwendung zum Konsumieren zusätzlicher Informationen für beide Teams innerhalb des Meetings dienen. Hierzu zählen die Informationen zu einem Ticket, welches gerade von einem Redner (Speech Token Owner) erwähnt wird, aber auch die Gesamtübersicht zum aktuellen Sprint. Um die Informationen zu einem aktuell genannten Ticket zu fokussieren, wird eine Markierung des Redners durchführt. Alle Informationen zum Sprint bzw. der darin enthaltenen Tickets werden vom JIRA und JIRA Agile-System bezogen.

2.7. Vergleichbare Lösungsansätze

Die beiden vorherigen Unterkapitel, Anforderungen und Spezifikation, haben zur Beschreibung einer neuen Lösung beigetragen. Die folgenden existierenden Anwendungen werden hierzu untersucht und zum geplanten Prototypen abgegrenzt. Die betrachten Lösungen sollen dabei eine Digitalisierung eines Taskboards für Scrum und Kanban ermöglichen.

SeeNowDo[12] bietet eine typische Taskboard-Ansicht mit einer guten Bandbreite an Funktionen und ebenfalls eine gute Darstellung mit konfigurierbaren Layout und Design. Das Board kann

[12]http://www.seenowdo.com/ - abgerufen am 28.04.2014

von mehreren gleichzeitig betrachtet und bearbeitet werden. Abgesehen von der Performance der Anwendung sind Hauptausschlusskriterien, dass ein Import von Tickets eines externen Systems nur über das *csv*-Format möglich ist, die Anwendung von einen externen Dienstleister betrieben wird und während eines Meetings nicht sichtbar ist, welches Ticket gemeint ist. Es gibt keine direkte Markierungsmöglichkeit für Meetings und stellt somit die identische Problematik wie das Standard-Taskboard von JIRA-Agile dar.

Ein weiteres Tool, welches ein digitales Kanban-Board anbietet, ist projectplace[13]. Auch hier ist es möglich, mit mehreren Teammitgliedern das Board zu pflegen und zu planen. Eine Anbindung an externe Systeme muss über die REST-API entwickelt werden. Für ein verteiltes Meeting ist allerdings auch hier das Problem, dass ein Ticket, welches erwähnt wird, nicht direkt bei anderen Betrachtern sichtbar markiert werden kann. Ebenfalls ist die Anwendung nur online verfügbar.

Die Software der Saxonia Systems AG namens eteoBoard[14] bietet von allen den interessantesten Ansatz, da hier bewusst auf die mögliche Verteilung von Teammitglieder in Meetings gesetzt wird. Ebenfalls ist eine Anbindung an Systeme wie Atlassian's JIRA oder Microsoft's Team Foundation Server möglich. Die Anwendung ist speziell für größere Multitouch Displays entwickelt worden und stellt zur Unterstützung der agilen Software Entwicklung wohl die beste der bereits genannten Anwendungen dar. Allerdings sind hier die Nachteile, dass die Anschaffungskosten je Standort relativ hoch sind. Dies liegt zum einen an der gebundenen Hardware, die mit eingekauft werden muss. Auch sind keinerlei öffentlich zugänglichen wissenschaftlichen Studien hierzu zu finden.

2.8. Fazit

In diesem Kapitel wurde anfangs ein agiles Vorgehen mit Scrum und aufbauend die verteilte agile Softwareentwicklung beschrieben. Hierzu wurde zwischen verteilten und verstreuten Teams unterschieden und auf verschiedene Skalierungsmöglichkeiten für Teams eingegangen. Als Praxisbeispiel eines verstreuten Teams wurde eine große deutsche Firma im Dienstleistungsgewerbe gewählt und eine Ist-Analyse durchgeführt. Die Ergebnisse dieser Analyse haben eine deutliche Orientierungslosigkeit der Teammitglieder aufgezeigt. Dies zeigte sich dadurch,

[13] https://www.projectplace.com/ - abgerufen am 28.04.2014
[14] http://www.eteoboard.de/ - abgerufen am 28.04.2014

dass Teammitglieder nach der Nennung eines Ticket-Kürzels zum Monitor schauten, um dieses zu lokalisieren. Ebenfalls stellte die ungenügende Ausstattung, in Verbindung mit den Räumlichkeiten, ein zusätzliches Hindernis in der Kommunikation dar. Resultierend aus den Erkenntnissen der Ist-Analyse wurden hierzu Anforderungen aufgenommen, welche auch im Bezug auf eine plattformunabhängige Lösung abzielt. So sollen neben den Standard-Desktop-Monitoren auch mobile Endgeräte und große Touch-Oberflächen als Interaktionsmöglichkeiten zur Verfügung stehen. Die Spezifikation wurde daraufhin basierend auf einen BPMN-Modell sowie Anwendungsfällen definiert. Zur Abgrenzung des zu entwickelnden Prototypens wurden vergleichbare Lösungsansätze vorgestellt, welche allerdings zum Großteil keine synchrone Markierungsmöglichkeiten bieten. Eine Lösung hat ein Konzept zur synchronen Unterstützung, allerdings sind hier die Kosten aktuell so hoch, sodass eine Nutzung auf mehre kleine Standorte nicht rentabel erscheint. Die Abhängigkeit an die gebundene Hardware stellt einen zusätzlichen Kritikpunkt dar und bietet somit z. B. nicht die Möglichkeit mobile Endgeräte einzubinden.

3. Design und Realisierung

Aus den Ergebnissen der Analyse ergibt sich nun das zu entwerfende System. Hierbei sind die notwendigen Markierungsmöglichkeiten zur synchronen Darstellung essentiell, um die beobachtete Orientierungslosigkeit zu reduzieren oder sogar zu verhindern. Dieses Kapitel beschäftigt sich mit dem Aufbau der Anwendung sowie den verwendeten Mustern und deren Erklärung. Zunächst werden Grundmodule definiert und eine Abgrenzung durch bereits existierenden Funktionen vorgenommen. Zur Beschreibung der Anwendung werden Entwurfsmuster für die Architektur und des Designs vorgestellt. Verschiedene Sichten zeigen über Diagramme die Anbindung, die innere Struktur und den Ablauf der Software. Das Benutzerinterface soll schließlich die Interaktionsschnittstelle zum Benutzter definieren. Als besondere Herausforderung im Verteilungsaspekt der Clients sollen Konflikte beschrieben und ein Lösungsansatz gezeigt werden. Zur Abnahme der Anwendung werden beispielhaft die durchgeführten Tests erläutert. Am Ende dieses Kapitels wird eine zusammenfassende, technische Evaluierung dargestellt.

3.1. Systemmodule

Für das zu entwerfende System sind verschiedene Grundmodule nötig, welche getrennt vom Kern der Anwendung zu betrachten sind. Darunter fallen:

- Authentifizierungsmanagement:

  Die Authentifizierung der Anwendung selbst bzw. dem Zugriff auf nur berechtigte Daten aus dem JIRA-System.

- History Management:

  Aktionen, welche eine Auswirkung auf dem bestehenden Sprint haben, wie z. B. das Ändern eines Ticket-Status müssen nachverfolgbar sein.

- Synchronisierungsmodul:

  Der Zugriff von mehreren Anwendern zum Markieren oder Verändern eines Ticket-Status müssen miteinander koordiniert werden.

3.2. Resultierende Abgrenzung

Durch das vorhandene JIRA-System werden Systemmodule zum Teil bereits abgedeckt.

- Authentifizierungsmanagement: JIRA bietet bereits ein User- und Berechtigungsmanagement an, sodass die Zugriffsberechtigung von JIRA selbst übernommen werden kann. Die Authentifizierung am JIRA-System läuft dabei über den Auth-Header eines REST Calls.
- History Management: Jede Aktion, welche eine Änderung eines Tickets verursacht, wird von JIRA selbst geloggt. Somit müssen nur Logs für Aktionen geschrieben werden, welche eine Nachverfolgbarkeit seitens JIRA verhindern oder verschleiern könnte.
- Synchronisierungsmodul: Die komplette Interaktionssynchronisation muss von der Anwendung übernommen werden und über eine Semaphore-Struktur einen inkonsistenten Aggregationszustand vermeiden.

3.3. Architekturentwurf

In den nachfolgenden Unterkapiteln soll der Architekturentwurf der entwickelten Software näher beschrieben werden. Hierbei wird auf die eingesetzten Architekturstile, wie auch die verwendeten Design Patterns, eingegangen.

3.3.1. Architekturstile und Design Patterns

Architekturstile umfassen Muster zur Beschreibung einer Gesamtarchitektur eines Softwaresystems. Dabei werden teils unterschiedliche Muster verwendet, um das Zusammenspiel von Subsystemen untereinander zu zeigen (vgl. Balzert 2011, S.37). Design Patterns oder

auch Entwurfsmuster beschreiben Muster zur Lösung von Entwurfsproblemen. Sie tragen zur Strukturierung der Subsysteme bei und helfen wie auch die Architekturstile bei einer Generalisierung von Problemstellungen. Sie tragen damit zum Verständnis der Software bei und helfen auch bei einer besseren Koordination zwischen den Entwicklern. Die folgenden Stile stellen Hauptmerkmale des entworfenen Systems dar, werden definiert und im Kontext zur Software beschrieben.

3.3.1.1. Model View Controller

Das *model view controller pattern*, kurz *MVC* pattern, ist ähnlich zum observer pattern, auf das in Kapitel 3.3.1.4 auf Seite 45 noch näher eingegangen wird. Grundsätzlich handelt es sich um ein Verhaltensmuster, bei dem das System in drei Subsysteme untergliedert wird (vgl. Balzert 2011, S.62-68). Das model view controller pattern gehört zu den Architekturstilen. Dieses Muster besteht aus drei Teilen:

- model: Diese Komponente beinhaltet die Logik der Anwendung. Dazu zählen zum Beispiel das Persistieren von Daten und die Koordination der Anfragen an das System. Es wird abhängig vom Grad der Abstraktion noch weiter in kleinere Komponenten zerlegt, was auch bei den beiden anderen Subsystemen stattfinden kann.

- view: Hier wird die Darstellungsebene bzw. Benutzeroberfläche implementiert. Darunter fallen die Änderungen der Sicht auf Änderungen im Modell der Anwendung.

- controller: Alle Eingaben des Benutzers fallen unter diese Komponente. Diese Eingaben können z. B. Touch-Events vom Benutzer sein, die Änderungen im Modell hervorrufen.

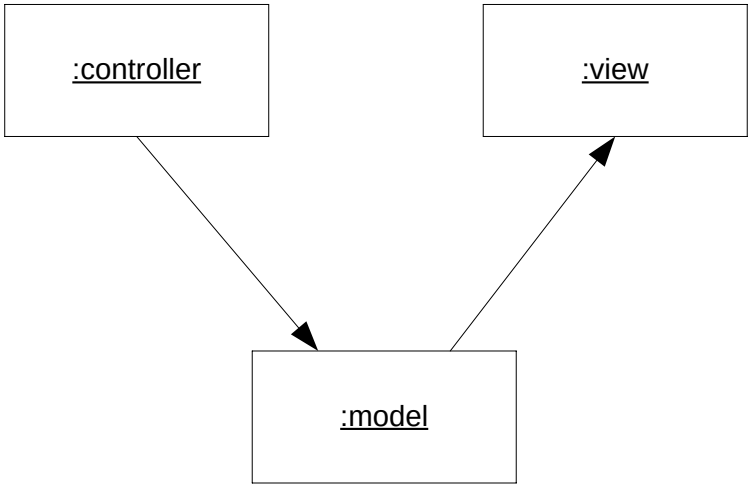

Abbildung 3.1.: MVC Pattern nach Balzert 2011, S.64

Die Grafik 3.1 zeigt ein Beispiel einer MVC Architektur. Der Controller leitet Änderungen an das Modell weiter und das Modell verändert die Darstellungssicht für den Benutzer. Inwieweit Änderungen am View wiederum direkt den Controller beeinflussen oder Anfragen vom View an das Modell beim Initialisieren der Anwendung gestellt werden, sind dabei nicht enthalten.

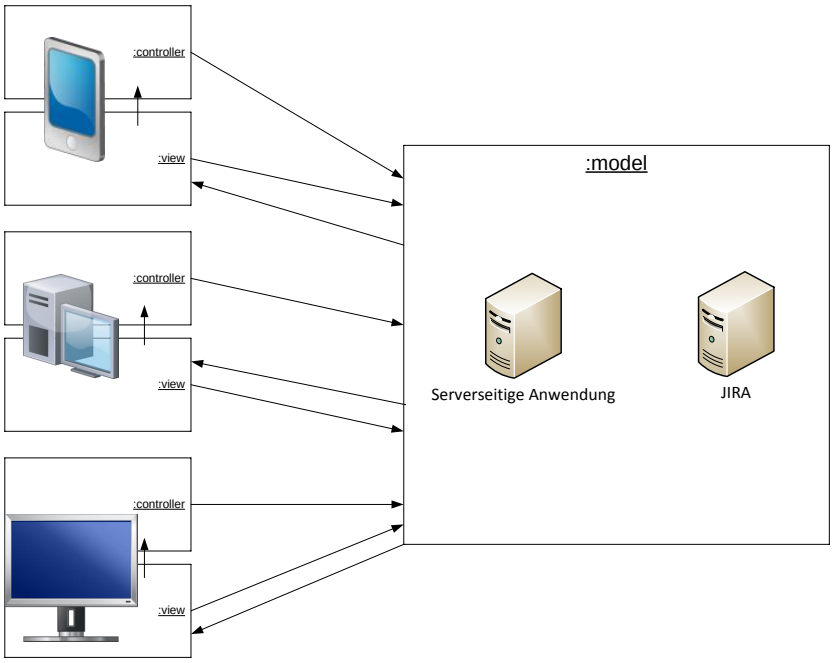

Abbildung 3.2.: MVC Pattern nach Balzert 2011, S.64 auf die Anwendung bezogen

Die entworfene Anwendung unterteilt die Komponenten, wie in der Grafik 3.2 zu sehen ist. View und Controller befinden sich auf den jeweiligen Endgeräten. Das Modell beinhaltet den kompletten Server. Hierzu zählen die serverseitige Anwendung und das JIRA-System. Wird die Anwendung von einem Endgerät aufgerufen, so wird mittels der View-Komponente eine Registrierung am Server vollzogen. Daraufhin wird der View aktualisiert und der Controller angepasst. Die Anpassung folgt nach jeder Änderung im View und daraus resultierenden Einschränkungen in der Eingabe. Wird eine Eingabe getätigt, so werden diese mittels Controller zum Modell gesendet. Das Modell verarbeitet die eingetroffenen Informationen und aktualisiert wiederum den View. Die Aktualisierung beinhaltet alle registrierten Views.

3.3.1.2. Rich Internet Application

Rich Internet Applications, kurz RIAs, sind Webanwendungen, welche dem Anwender das Gefühl einer Funktionfülle vermittelt, welche man sonst nur von Desktop-Anwendungen gewohnt ist (vgl. Deitel und Deitel 2008, S.32). Zu den Features einer solchen Anwendung zählen z. B. Handhabung und Feedback der Applikation nach einer Interaktion, die untypisch für die

Standard-HTML-Umgebungen sind. Unter dem Standard versteht sich hier die *alte* HTML-Welt mit ihren statischen und einfachen Funktionen sowie dazugehörigen Benutzerinterface-Design. Besonders Ajax (Asynchronous JavaScript and XML) hat dazu beitragen, die Funktionalität von Webaplikationen zu erhöhen. In der entworfenen Applikation werden zur Erfüllung der RIA-Eigenschaft HTML5, jQuery und CSS3 (Freeman und Robson 2012, Benedetti und Cranley 2012, Müller 2014) eingesetzt.

Als RIA-Lieferant wird ein Webserver für die Lauffähigkeit der Anwendung benötigt. Dieser Webserver liefert auf Anfrage die benötigten HTML- , CSS- und JavaScript-Files sowie zusätzliche jQuery-Bibliotheken. Der Webserver kann hierbei gleichzeitig die Authentifizierung zur Berechtigung der Ressourcen übernehmen. Dies wurde bereits bei den Systemtests erfolgreich umgesetzt und wird in Kapitel 3.6.1 näher erläutert. Zum letztendlichen Erreichen der *Rich*-Funktionen trägt z. B. die mögliche Touch-Interaktion bei. Interaktionen mit Tickets sind über touchfähige Endgeräte möglich. Weitere Features, welche hierzu beitragen, sind in den Systemtests im Kapitel 3.6.1 beschrieben.

3.3.1.3. REST

Representational State Transfer, kurz REST, beschreibt einen Architekturstil zur Realisierung von Webservices. Das Grundprinzip hinter REST ist die Übermittlung von Befehlen und Anfragen über HTTP zwischen Anwendungen. Dabei werden Methoden wie GET, PUT, POST und DELETE zur Verfügung gestellt, welche analog zu SELECT, INSERT, UPDATE und DELETE in SQL zu sehen sind (vgl. Balzert 2011, S.286-289). Antworten eines REST-konformen Servers können unterschiedlich sein und sind der entsprechenden Dokumentation zu entnehmen. Oft fallen diese allerdings auf JSON (*JavaScript Object Notation*) und XML (*Extensible Markup Language*) zurück.

REST wird zur Kommunikation zwischen der serverseitigen Anwendung selbst und dem JIRA-System verwendet. JIRA bietet diese Schnittstelle unter anderem an, um sämtliche Anfragen und Operationen für Tickets und darüber hinaus auch für Taskboards aus JIRA Agile zu formulieren. Folglich werden von der geschriebenen Software alle Operationen, welche Informationen von JIRA benötigen, in REST Operationen transformiert. JIRA nutzt für die entsprechenden Nachrichteninhalte das JSON-Format.

Hierzu zählen folgende Anfragen:

- Authentifizierung eines Users und Laden von Profilinformationen.
  („/rest/auth/latest/session“)
- Listen aller Boards für eine bestimmte Authentifizierung.
  („/rest/greenhopper/latest/rapidviews/list“)
- Laden aller Board-Informationen, inklusive Tickets.
  („/rest/greenhopper/latest/xboard/work/allData/?rapidViewId=$rapidViewId“)
- Abfrage von Ticket-Details und möglichen Transitionen (Übergänge von einem Status in den anderen).
  („/rest/api/latest/search?jql=issuekey in ($listofIssueKeys)&expand=transitions &fields=id,key,project,summary,issuetype,priority,assignee,fixVersions, comment,reporter,description,status“)
- Ausführen von Transitionen für ein Ticket.
  („/rest/api/latest/issue/$issueId/transitions“ JSON-Message-Inhalt = {transition: $id})

Zur Authentifizierung beim Ausführen von Operationen wurde jeweils der Header angepasst (Authorization:Basic$authdata). Als Beispiel einer Operation wird folgend eine Anfrage an das JIRA-System gestellt, um alle Boards inklusive Tickets zu listen. Die Antwort ist dabei im JSON-Format (JavaScript Object Notation).

Anfrage: „http://MusterURLZuJIRA/rest/greenhopper/latest/rapidviews/list“

Antwort:

```
{
views: [
{
id: 3,
name: "Bachelorarbeit - Team 1",
canEdit: true,
sprintSupportEnabled: true,
filter: {
id: 10100,
name: "Filter for Bachelorarbeit - Team 1",
query: "project = Bachelorarbeit ORDER BY Rang ASC",
```

```
owner: {
userName: "akaack",
displayName: "Alexander Kaack",
renderedLink: " <a class="user-hover" rel="akaack" id="_akaack"
href="/jira/secure/ViewProfile.jspa?name=akaack">Alexander Kaack</a>"
},
canEdit: true,
isOrderedByRank: true,
permissionEntries: [
{
values: [
{
type: "Project",
name: "Bachelorarbeit"
...
]
}
```

3.3.1.4. Observer

Das observer pattern, zu Deutsch Beobachter-Muster, ist wie auch das MVC-Pattern ein Verhaltensmuster, gehört aber zusätzlich den Design Patterns an. Die Idee hinter diesem Muster ist, dass es Beobachter und ein zu beobachtendes Objekt gibt. Beobachter registrieren sich um Änderungen des zu beobachtenden Objekts zu erhalten. Wie ein Beobachter darauf reagiert, ist dabei nicht definiert, sondern nur, dass er die Informationen über getätigte Änderungen erhält (vgl. Balzert 2011, S.54-60).

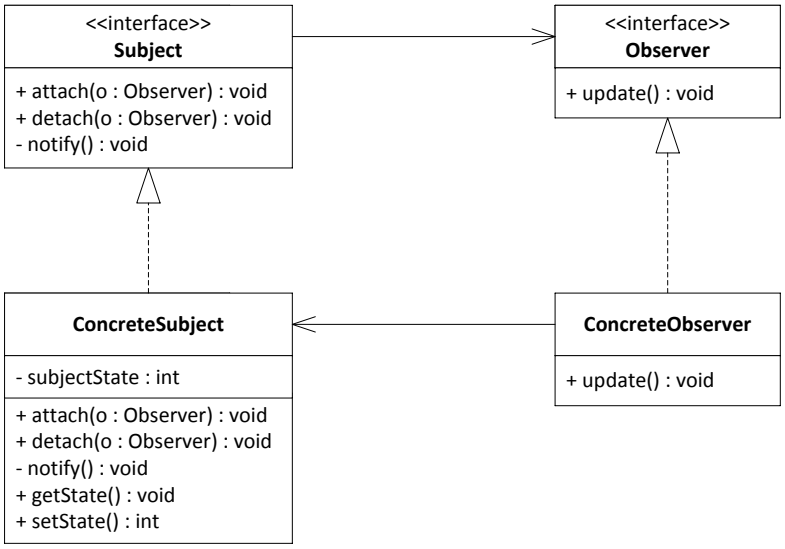

Abbildung 3.3.: Observer pattern nach Balzert 2011, S.60

Die Abbildung 3.3 zeigt dieses Muster als UML-Klassendiagramm. Ein Beobachter (Observer) wird durch einen konkreten Beobachter (ConreteObserver) realisiert, welcher sich durch die Methoden *attach* und *detach* an einem konkreten Subjekt registriert bzw. abmeldet. Dieses Subjekt realisiert das Interface Subject und bietet über die Methode *notify* eine Benachrichtigung aller Beobachter an. Diese Methode nutzt dann *update*, um jeden Beobachter mit den aktuellen Werten zu versorgen. *Update* wiederum nutzt *getState* um den neuen Wert zu erhalten.

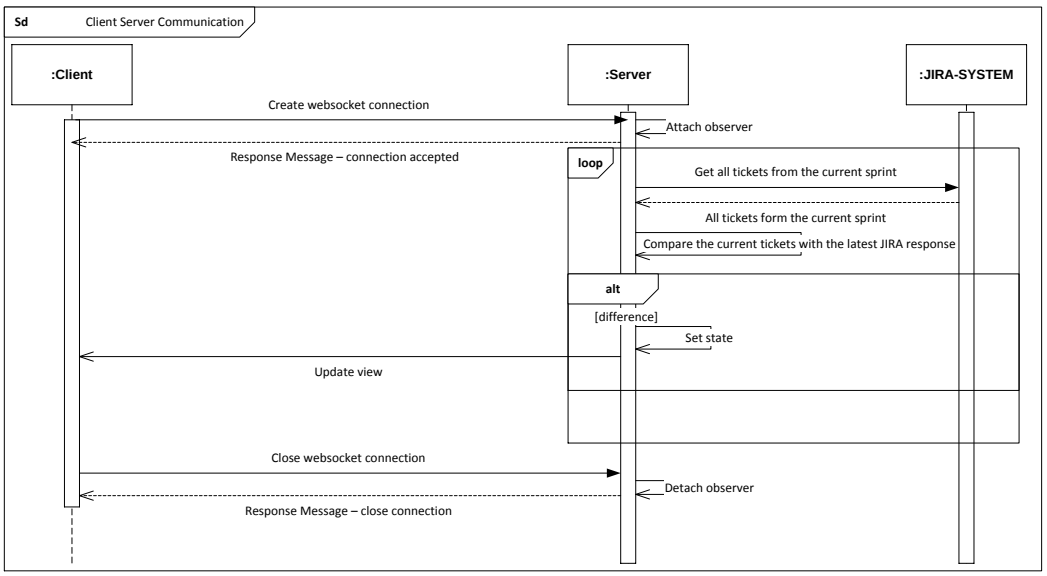

Abbildung 3.4.: Observer pattern - Sequenzdiagramm Client/Server nach Balzert 2011, S.60

Die Abbildung 3.4 zeigt das observer pattern in der realisierten Anwendung. Jeder Client wird durch das Aufbauen einer Websocket-Verbindung beim Server registriert. Der Server hält nun eine Liste aller Clients. Der Server holt sich in einem definierten Zeitabstand regelmäßig alle aktuellen Tickets aus dem JIRA-System und vergleicht diese mit den eigenen gehaltenen Tickets. Ist nun ein Unterschied erkannt, so müssen alle Clients die aktualisierten Tickets erhalten. Zunächst werden auf dem Server die entsprechenden Tickets angepasst und danach eine Aktualisierung an alle Clients versendet. Schließt ein Client die Verbindung, so wird der Client auf dem Server aus der Liste aller aktiven Clients entfernt. Der Client erhält hier noch eine Bestätigung über das Austreten aus der Liste.

3.3.2. Sichten

Die folgenden Sichten sollen einen Gesamtüberblick der Anwendung darstellen, einen detaillierten Blick in die Zusammensetzung und den Ablauf der Software bieten.

3.3.2.1. Verteilungssicht

Die Verteilungssicht, oder auch Verteilungsdiagramm, zeigt die Hardwarearchitektur sowie deren Anbindung an externe Systeme. Zusätzlich werden Artefakte dargestellt, die zur Laufzeit auf der jeweiligen Plattform relevant sind (vgl. Goll 2011, S.471). In der Abbildung 3.5 wird die Anbindung der Endgeräte an die jeweiligen Dienste sichtbar. Zum einen werden über den RIA-Lieferanten (Apache Webserver) die benötigten Ressourcen bezogen, welche sich dann in die Komponenten View und Controller auf dem Client (jeweiligen Endgerät) gliedert. Die Abfrage geschieht dabei via HTTP. Die Clients werden dann mittels einer Websocket-Verbindung an den node.js-Dienst angebunden. Die Serveranwendung wird dabei über das response.js-File gestartet. Die Core-Komponente übernimmt dabei die Abwicklung der Logik. Ein Beispiel hierfür ist die Reaktion auf Anfragen vom Client oder auch Kommunikationsaustausch mit dem JIRA-System.

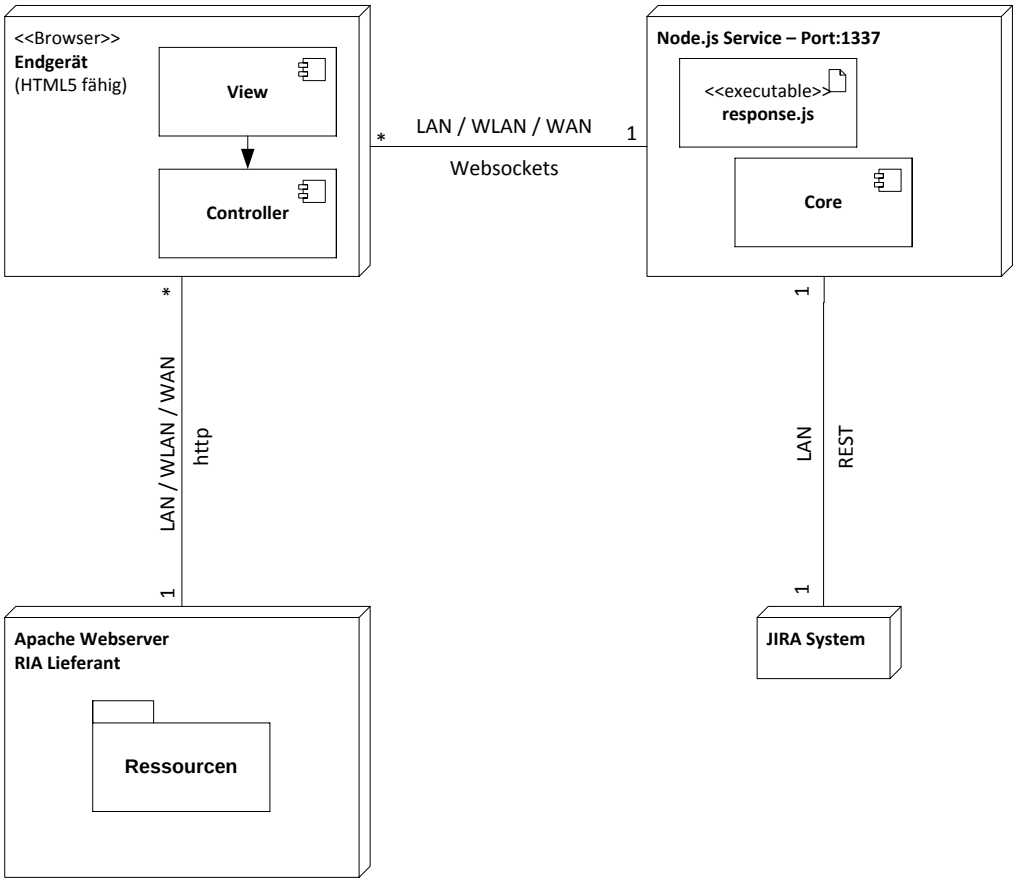

Abbildung 3.5.: Verteilungssicht der Software

3.3.2.2. Komponentendiagramm

Die Grafik 3.6 zeigt den inneren Aufbau der Software. Generell wird die Anwendung in drei Komponenten zerteilt. Die View-, Controller- und Modell-Komponente. Die View-Komponente ist über eine Websocketverbindung an der Modell-Komponente angebunden, welche auf die View-Komponente zugreifen kann. Auch kann diese Komponente auf die Controller-Komponente zugreifen. Die View-Komponente unterteilt sich wiederum in eine kleinere View-Komponente und eine View-Websocket-Adapter-Komponente. Die Controller-Komponente kann auf die Modell-Komponente über eine Websocketverbindung zugreifen und gliedert sich in eine kleinere Controller-Komponente und eine Controller-Websocket-Adapter-Komponente.

Die Modell-Komponente enthält außer der eigenen Modell-Websocket-Adapter-Komponente, welche sich in die View-Websocket-Adapter-Komponente und Controller-Websocket-Adapter-Komponente gliedert, noch die Core-Komponente und als logische externe Komponente das JIRA-System. Das JIRA-System ist dabei kein Bestandteil der Software. Es ist allerdings im logischen Aspekt Teil der Modell-Komponente.

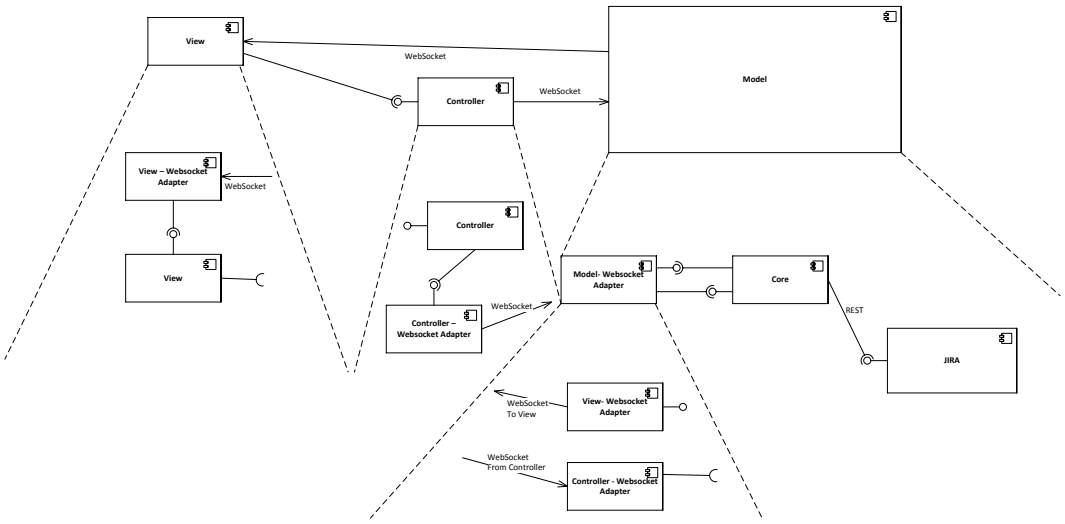

Abbildung 3.6.: Komponentendiagramm der Software

3.3.2.3. Laufzeitsicht

Die Laufzeitsicht, oder auch Sequenzdiagramm genannt, wird in Abbildung 3.7 abgebildet und zeigt den Ablauf zwischen den Komponenten sowie die Aktionen vom User, welche Events auslösen. Jede Kommunikation zwischen der Modell- und der View-Komponente steht dabei für jeden einzelnen Client, welcher an der Session teilnimmt, mit Ausnahme beim Schließen des Browsers.

Zunächst wird über das „:frontend“ die Webapplikation gestartet, welches wiederum die Komponenten View und Controller startet. Die Modell-Komponente wurde bereits über den Server gestartet, wie auch das JIRA-System. Gibt der User Authentifizierungsinformationen ein, so wird die Eingabe über die Controller-Komponente an die Modell-Komponente weitergeleitet. Diese Komponente holt sich nun alle Sprints für den eingegeben User und Zusatzinformationen

von diesem über REST (Kapitel 3.3.1.3) vom JIRA-System. Diese Informationen werden dann als Admin-Informationen an die View-Komponente übertragen.

Der Client wählt nun einen Sprint über die Administrationsoberfläche und die Controller-Komponente stellt diese Anforderung an die Modell-Komponente. Diese holt sich daraufhin den Sprint und anschließend alle Detailinformationen zu den enthaltenden Tickets, wieder via REST vom JIRA-System. Diese Informationen werden als Board zur View-Komponente übermittelt, welche das Board für die Oberfläche konstruiert.

Nun wird in einem Zyklus regelmäßig überprüft, ob sich die Tickets oder das Board verändert haben. Ist dies der Fall, so wird ein neues Board an die View-Komponente übertragen und das Board neu zusammengesetzt. Filtert der Enduser nach einem Bezeichner für ein Ticketkürzel oder Bearbeiter, so wird jedes Zeichen einzeln übertragen und in der Modell-Komponente auf das bestehende Board angewendet. Die aktualisierte Board Version wird dann an die View-Komponente eines jeden Clients übermittelt.

Wird ein Ticket markiert, leitet die Controller-Komponente diese Aktion an die Modell-Komponente weiter, welche die Markierung an alle Clients verteilt. Trifft eine Markierungsnachricht an der View-Komponente ein, wird der Fokus auf das Ticket gelenkt und die Markierung gesetzt.

Im Fall, dass ein User eine Authentifizierung im Administrationsbereich abmeldet, stellt die Controller-Komponente die Anforderung an die Modell-Komponente zur Aktualisierung auf. Die aktualisierte Fassung überträgt die Komponente dann an die View-Komponente.

Die Session und das Board werden komplett auf den Startzustand zurückgesetzt, sobald die Session geleert wird. Dies betrifft ebenfalls den Admin-Bereich. Dieses Event wird über die Controller-Komponente wieder an die Modell-Komponente geleitet und dann an alle Clients übermittelt. Die Clients leeren daraufhin die Bereiche über die View-Komponente.

Schließt ein User den Browser, so werden alle Komponenten nacheinander (View, Controller und „:frontend“) beendet. Dabei wird die Modell-Komponente darüber informiert und kann feststellen, wer sich abgemeldet hat. Die Abmeldung kann allerdings auch über einen Timeout vom Server geschehen und der Client veranlasst dann bei der Rückmeldung automatisch eine neue Anmeldung am Server.

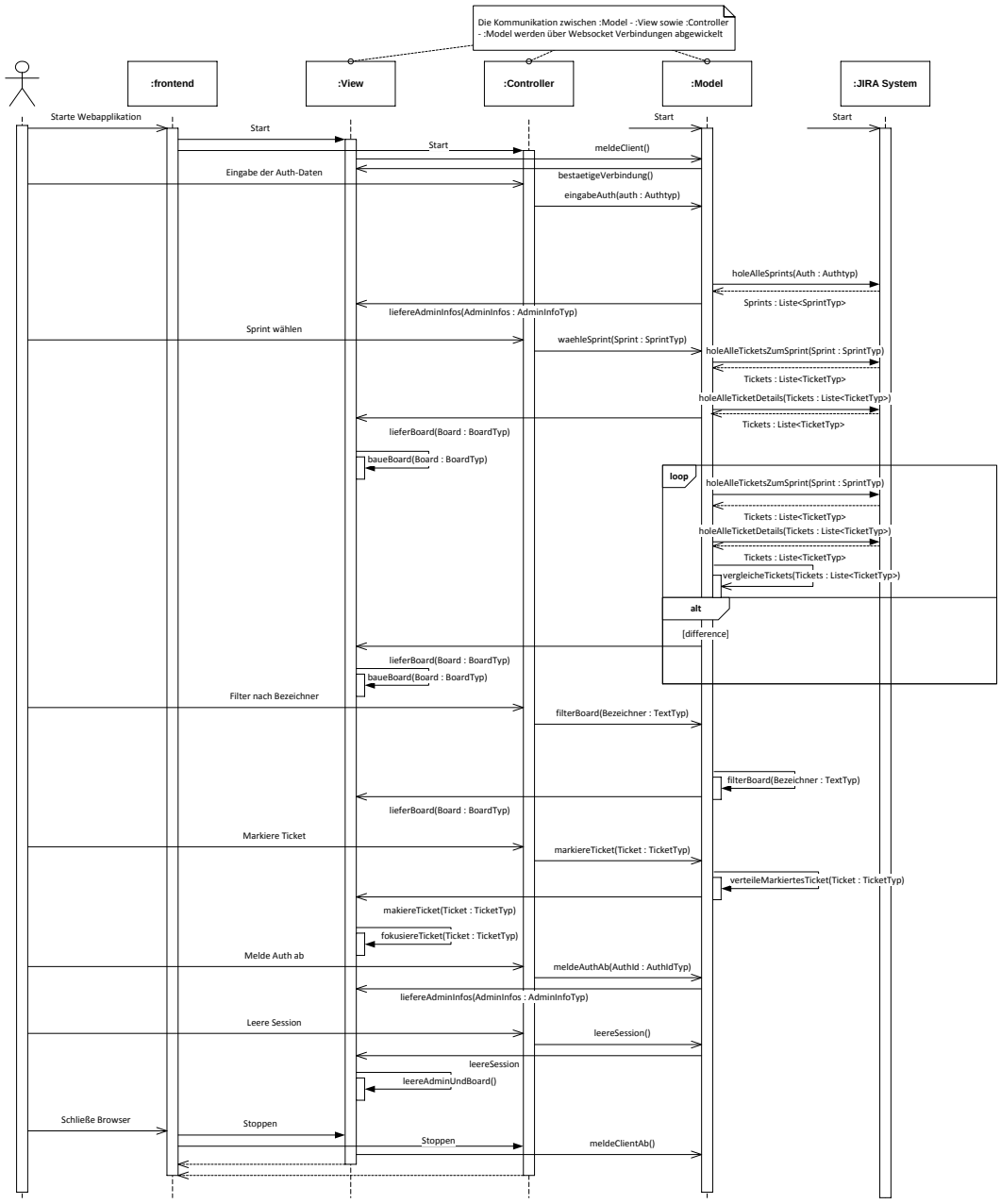

Abbildung 3.7.: Sequenzdiagramm der Software

3.4. GUI Design

Das Design der Benutzerschnittstelle hat sich im Laufe der Entwicklung verändert, es existieren dabei zwei unterschiedliche Ansätze. Der erste Ansatz beschäftigt sich mit der Darstellung auf größeren Displays und beinhaltet sowohl den Administrationsbereich, wie auch den Interaktionsbereich für das Meeting selbst. In einer zweiten Variante wurde der Administrationsbereich auf eine separate, eigene Seite ausgelagert, um für das Meeting selbst die Fläche optimal zu nutzen. Der Erfassungsbereich konnte somit speziell auf kleinere Displays vergrößert werden.

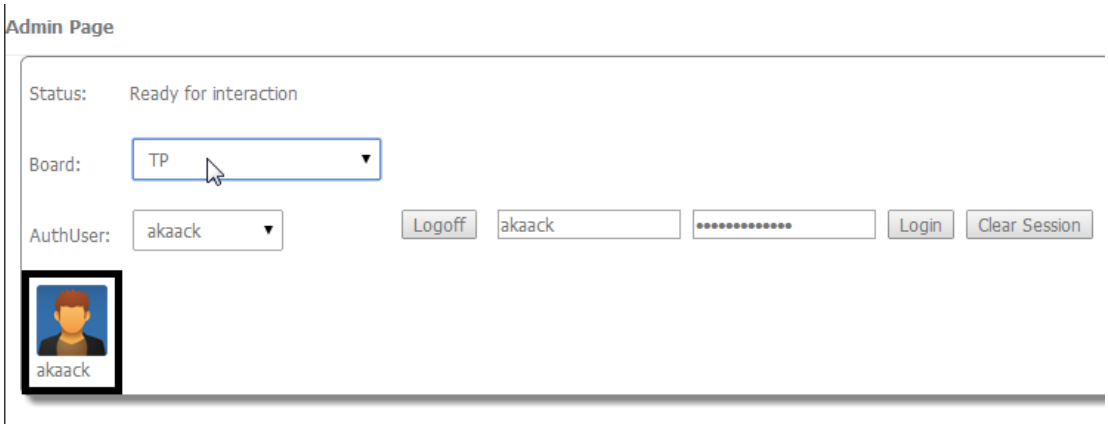

Abbildung 3.8.: Administrationsbereich

Die Abbildung 3.8 zeigt den erwähnten Administrationsbereich. Dieser Bereich ermöglicht es den Anwendern Authentifizierungsinformationen für das JIRA-System einzugeben und damit relevante Boards für diesen User zu wählen. Die Boards werden dann aus dem JIRA-System extrahiert und aggregiert auf dem Interaktionsbereich dargestellt. Zusätzlich zur *DropDown*-Auswahl stehen auch die entsprechenden Avatare aus JIRA bereit. Es ist möglich zwischen verschiedenen *AuthUsern* zu wechseln, welche dann auch wieder auf andere Boards zugreifen können. Es ist auch möglich, einzelne User wieder abzumelden oder auch die ganze Session zurückzusetzen. Das Zurücksetzen der Session hätte zur Folge, dass die Anwendung auf den Startzustand zurückgesetzt werden würde.

Die Interaktionsfläche, welche in Abbildung 3.9 zu sehen ist, ist auf die häufigsten Interaktionen und essentiellen Informationen hingehend optimiert.

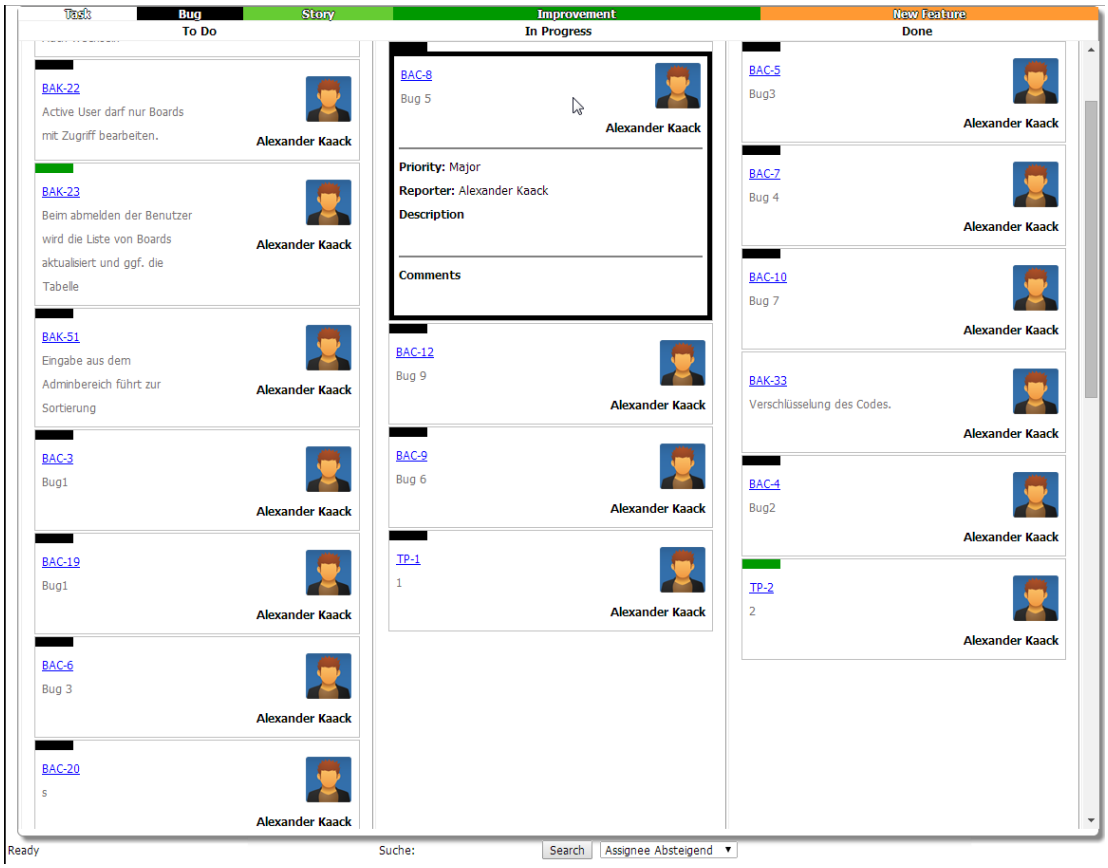

Abbildung 3.9.: Interaktionsbereich

Jedem Ticket wird abhängig von seinem Typ (z. B. Task, Story oder Improvement) eine Farbe zugeteilt. Diese Farben sind identisch zu den Farben in JIRA, solange die Farbe nicht doppelt vorkommt. Zur Orientierung aller vorhandenen Ticket-Typen ist am oberen Rand eine Legende mit allen Typen und deren verknüpften Farben hinterlegt. Wie auch das Board in JIRA sind die Tickets ihrem Status zugeteilt (im Beispiel entweder „To Do“ , „In Progress“ oder „Done“). Die Tickets sind im unmarkierten Zustand so gestaltet, sodass diese mit Informationen zum Ticketkürzel, der Zusammenfassung, der Bearbeiter mit seinem Avatar und der bereits erwähnten Farbe zum Typ des Tickets versehen sind. Im Fall, dass ein Ticket markiert wird, wird es abhängig vom Betrachter farbig umrahmt und mit zusätzlichen Informationen versehen. Ist ein Betrachter derjenige, welcher die Markierung gesetzt hat, erhält das Ticket eine schwarze Umrandung. Andere Betrachter erhalten einen roten Rahmen, was eine Sperrung des Interaktionsbereiches zur Folge hat (hierauf wird in Kapitel 3.5 noch weiter eingegangen). Gleichzeitig fokussiert sich der Bildschirm aller Beobachter auf dieses Ticket. Die Zusatzinformationen

enthalten die Priorität, den Reporter, die Beschreibung und die Kommentare des Tickets. Um die Suche nach dem Ticket zu vereinfachen, wird ein Suchfeld am unteren Rand angeboten. Im Standardfall wird der Fokus der Tastatur automatisch auf dieses Feld gelegt, sodass jede Taste sofort in ein Suchkommando übersetzt wird. Die Suche läuft über alle enthaltenen Bearbeiter und Ticketkürzel. Zusätzlich können alle Tickets noch auf- und absteigend nach ID oder Bearbeiter sortiert werden.

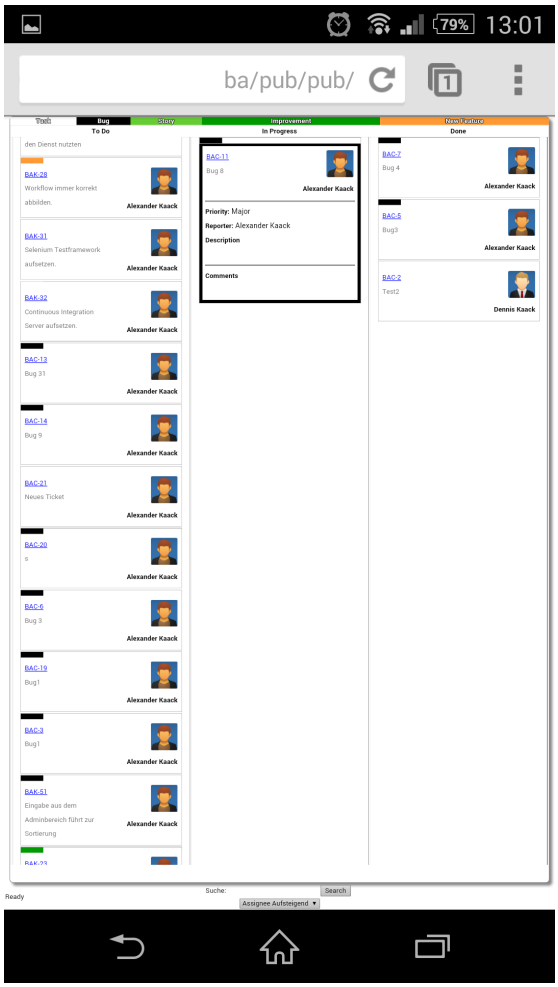

Abbildung 3.10.: Beispielansicht auf einem Android-Smartphone mit Chrome 34.0.1847.114

Durch die Umsetzung der Weboberfläche mit HTML5 ist wie bereits erwähnt, die Nutzung mit mobilen Endgeräten möglich. Die Abbildung 3.10 zeigt die Darstellung auf einem Smartphone.

Auch hier sind sämtliche Interaktionen via touch-Eingabe möglich. Die einzige Voraussetzung ist ein HTML5-fähiger Browser auf dem entsprechenden Smartphone oder Tablet.

3.5. Konflikte bei verteilten Interaktionen

Die Applikation bietet den Anwendern neben dem einfachen Konsumieren von Informationen auch die Möglichkeit Aktionen auszuführen. Hierzu zählen zum Beispiel die Markierung von Tickets, aber auch das Anmelden von weiteren *AuthUsern*. Durch den verteilten Zugriff zur Anwendung muss somit eine Koordination erfolgen, um einen inkonsistenten Zustand zu verhindern. Dies kann durch einen gegenseitigen Ausschluss realisiert werden. Hierfür wird ein zentraler Algorithmus gewählt.

Lösen die Clients jeweils Events aus, so treffen diese beim Server ein, welcher nun diese Events prüfen muss. Grundsätzlich wird auf dem Server jede Anfrage in ein Erzeuger-Verbraucher-Problem umgewandelt. Mittels eines Semaphores wird eine Anfrage dann zugelassen oder nicht. Der Mutex ist ein Semaphore, welcher genau einem oder keinem den Zugriff auf eine Ressource gewährt oder verweigert.

In der realisierten Anwendung wird hierfür eine Variable genutzt, welche bei einer Aktion einen Sperrzustand setzt und nach dem Ausführen dieser Aktion den Sperrzustand in einen offenen Zustand überführt. Trifft nun fast parallel eine Aktion von zwei verschiedenen Clients ein, so wird nur die erste Aktion übernommen und die zweite verworfen. Gleichzeitig ist die Anwendung clientseitig so entworfen, dass eine Markierung auch eine Sperrung auf dem Client bewirkt. Diese Sperrung geschieht allerdings nicht bei dem Auslöser selbst, sondern nur bei den anderen Teilnehmern. Dies soll verhindern, dass innerhalb der Markierungsaktion z. B. das Board gewechselt werden kann oder andere Markierungsaktionen möglich sind. Der Auslöser hat allerdings die Möglichkeit eine Folgeaktion auszuführen oder die Markierung aufzuheben. Damit hierdurch kein *Deadlock* entstehen kann, im Fall das der Auslöser einfach die Session verlässt oder Ähnliches, ist ein zeitliches Limit hinter die Sperrung gekoppelt. Die Stoppuhr wird nach jeder Folge-Markierung erneut auf 0 zurückgesetzt. Wird das Limit nun trotzdem erreicht, so wird eine Endsperrungsnachricht an alle Clients vom Server versendet.

3.6. Abnahmekriterien und Akzeptanztests

Zur Validierung der Software wurden zum einen Systemtests als auch Abnahmetests in der Evaluierungsumgebung durchgeführt.
Die Systemtests dienen zur Sicherstellung der Systemfunktionalität. Dieses wird durch die Integration aller Anwendungskomponenten erreicht. Auch sollen die entwickelten Tests die funktionalen und nicht funktionalen Anforderungen abdecken und somit deren Erfüllung bestätigen (vgl. Sommerville 2007, S.110).
Die Abnahmetests bilden die Tests seitens der Firma, in dem die Software zu Evaluierungszwecken eingesetzt werden soll. Hier werden speziell Tests durchgeführt, die den realen Daten im Anwendungsszenario entsprechen.

3.6.1. Systemtests

Die Testfälle sind tabellarisch aufgeführt und wurden in einer speziellen Testkonfiguration ausgeführt. Dieses Setup wird als Vorbedingung für jeden Test definiert.

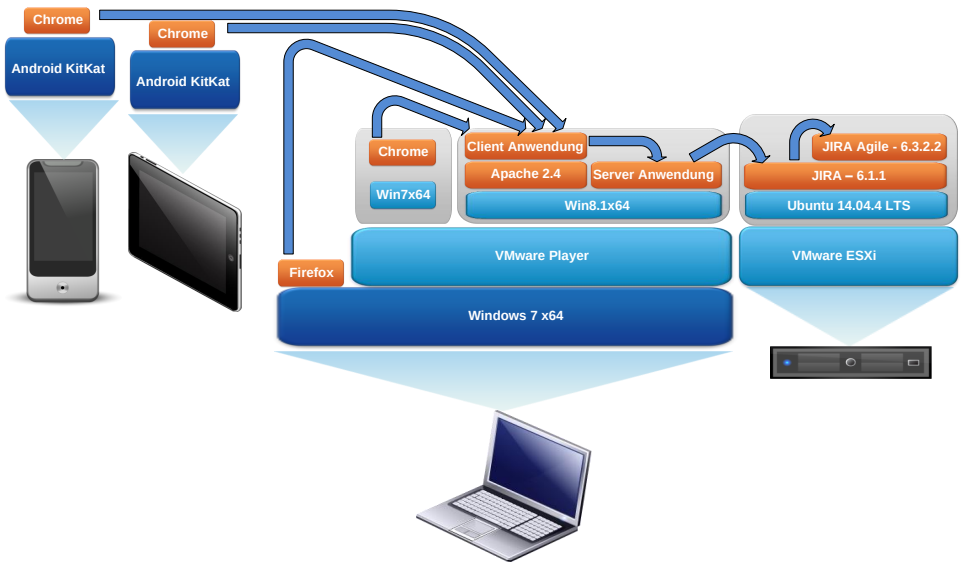

Abbildung 3.11.: Testsetup für Systemtests - Überblick der Testkonfiguration

Die Abbildung 3.11 zeigt einen Gesamtüberblick über die Testsystemkonfiguration. Von links aus sind zwei mobile Endgeräte mit jeweils Chrome-Browsern aktiv. Eins davon auf einem Smartphone, das andere auf einem Tablet. Ein Notebook mit Windows 7 x64 hat eine aktive Verbindung als Client über einen Firefox-Browser. Zusätzlich läuft ein VMware Player mit zwei virtuellen Instanzen. In der einen Instanz läuft ein Windows 7 x64 mit einem Chrome-Browser als Client. Die andere Instanz läuft unter Windows 8.1 x64 und stellt den Webserver als RIA-Lieferanten (Kapitel 3.3.1.2) mit der Clientanwendung sowie die serverseitige Anwendung bereit. Die serverseitige Anwendung stellt dann mittels REST-Schnittstelle (Kapitel 3.3.1.3) eine Verbindung zur JIRA-Anwendung, welche wiederum mit dem Addon JIRA Agile bestückt ist. JIRA läuft dabei auf einem Ubuntu Server, der mittels VMware ESXi virtualisiert wurde.

Die folgenden Tests stellen funktionale Systemtests dar. Sie werden im Blackbox-Test-Verfahren, ohne Kenntnisse über die interne Systemstruktur, durchgeführt (vgl. Franz 2007, S.28). Die Basis für den Entwurf dieser Testfälle stellt die Anwendungsfallbasierte Testfallermittlung (vgl. Franz 2007, S.46-48). Hierfür sind die Anwendungsfälle aus Kapitel 2.6.2 herangezogen worden. Auch wurden Tests über die Standardfälle in der Evaluierungsumgebung hinaus konstruiert, um eine Lauffähigkeit auf mobilen Endgeräten zu gewährleisten. Diese wurden bereits in den Benutzeranforderungen definiert und abgegrenzt. Die Abbildung 3.12 zeigt hierbei die Authentifizierung eines Smartphones am Apache Webserver um die Client-Anwendung zu beziehen. Diese Authentifizierung war für alle Clients notwendig und sollte eine einfache Absicherung darstellen. Diese Absicherung ist notwendig und sollte auch im produktiven System, wenn keine andere Sicherstellung der Zugriffs gewährleistet werden kann, vorhanden sein. In Verbindung mit JIRA kann der Webserver z. B. als SSO-Lösung (Single Sign On) betrieben werden und würde somit das Berechtigungsmanagement für mehrere Applikationen übernehmen können.

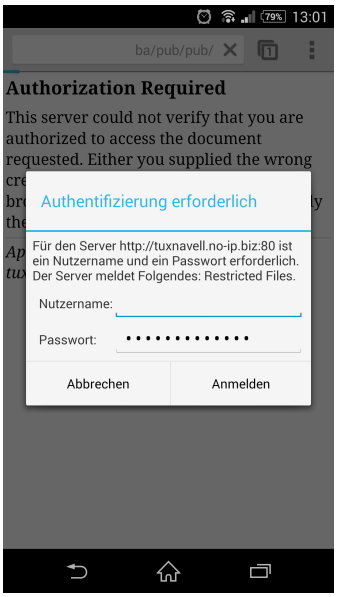

Abbildung 3.12.: Beispielansicht zur Authentifizierung auf einem Smartphone

Als Beispiel der durchzuführenden Tests werden drei Testfälle näher beschrieben. Die restlichen Testfälle finden sich im Anhang dieser Arbeit (A.2) und beinhalten auch Tests zur Änderung des Ticketstatus.

Testnummer	2
Name	Setzen von farblichen Markierungen.
Beschreibung	Ein Client markiert via Mausklick ein Ticket. Das Ticket sollte eine rote Umrandung bei allen Clients besitzen. Die Ausnahme ist eine schwarze Umrandung beim Client, welcher die Markierung gesetzt hat.
Fokus	sychrone Kennzeichnung des Tickets.
Vorbedingung	Alle Clients müssen bereits die Seite des Interaktionsbereiches offen haben und der Testfall 1. sollte erfolgreich abgeschlossen sein.

Tabelle 3.1.: Systemtests - 2. Testfall

Die Tabelle 3.1 definiert den Testfall zur synchronen Darstellung von Markierungen. Diese stellen die zentrale Funktion der Anwendung für das Daily Scrum und bieten jedem Client ein schnelleres Auffinden des genannten Tickets.

Testnummer	3
Name	Fokus des Tickets.
Beschreibung	Ein Client markiert via Mausklick ein Ticket. Der Fokus sollte nun bei allen Clients auf diesem Ticket liegen.
Fokus	sychrone Fokussierung.
Vorbedingung	Alle Clients müssen bereits die Seite des Interaktionsbereiches offen haben und der Testfall 1. sollte erfolgreich abgeschlossen sein.

Tabelle 3.2.: Systemtests - 3. Testfall

Mittels des Testfalls in Tabelle 3.2 wird das Auffinden des Tickets zusätzlich unterstützt. Bei einer Vielzahl von Tickets kann der aktuelle Fokus vom markierten Ticket entfernt sein und man müsste wieder das Board durchsuchen. Dies soll durch die Funktion, welche mit diesem Testfall abgedeckt wird, verhindert werden.

Testnummer	4
Name	Sortieren nach Bearbeiter
Beschreibung	Ein User gibt Buchstabe für Buchstabe den Namen eines Bearbeiters an. Nach jedem Zeichen sollte die Auswahl automatisch angepasst werden.
Fokus	sychrone Darstellung
Vorbedingung	Offene Session und ausgewähltes Board.

Tabelle 3.3.: Systemtests - 4. Testfall

Um dem User beim Auffinden seiner Tickets behilflich zu sein, wird durch das Sicherstellen der Suchfunktion, mittels direkter Eingabe des Bearbeiters, diese Möglichkeit gegeben. Die Tabelle 3.3 stellt diesen Testfall dar.

3.6.2. Abnahmetests

Die Abnahmetests laufen in der Produktivumgebung des Dienstleistungsunternehmens. Die Abbildung 3.13 soll einen Überblick darüber verschaffen. Dabei stehen zwei Clients, der eine in Hamburg, der andere in München, zur Verfügung. Die Anwendung selbst läuft mit dem Apache auf einem eigenen Server, welcher JIRA auf einem weiteren Server für den Informationsaustausch kontaktiert.

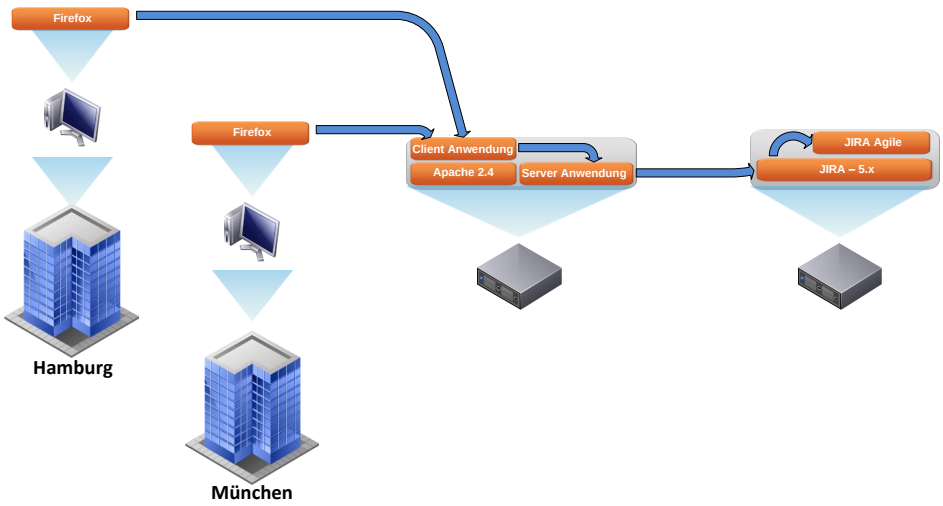

Abbildung 3.13.: Produktivumgebung

Getestet wurden Tests aus dem Systemtestbereich, allerdings ohne die dort beschriebene Systemumgebung als Vorbedingung. Speziell sind dies die Testfälle 1-8, 11 und 13. Auf die Testfälle zur Änderung der Status eines Tickets, wie auch die Nutzung mobiler Endgeräte, wurde verzichtet.

3.7. Technische Evaluierung

Dieses Kapitel hat zu Beginn relevante Systemmodule der zu entwerfenden Software und darauf aufbauend eine Abgrenzung in Verbindung mit dem angebundenen JIRA-System gezeigt. Hierbei wurde deutlich, dass bereits ein Großteil durch JIRA abgedeckt wird und der Hauptimplementationsaufwand, in Bezug auf die Systemmodule, beim Synchronisierungsmodul liegt. Zur Beschreibung des Systems haben sich die verwendeten Architekturstile und Design Patterns besonders geeignet. Das Unterkapitel GUI-Design beschrieb resultierend aus den Anforderungen eine Möglichkeit zur Gestaltung der Oberflächen für die Clients. Die Kombination von HTML5, CSS3 und jQuery hat hierbei alle benötigten Funktionen zur Implementierung einer Lösung bereitgestellt. Es wurden außerdem jQuery Bibliotheken hinzugefügt, um den Aufwand zur Realisierung eines neuen Programmcodes zu verringern. Die Server-Seite wurde mittels *node.js* (Chrome's JavaScript runtime)[1] entwickelt und mit geeigneten npm-

[1] http://nodejs.org/ - abgerufen am 30.04.2014

Packages[2] erweitert. Diese stellten insbesondere Funktionshilfen für die Interaktionsprotokolle, die REST-Schnittstelle und die Websockets dar. Die möglichen Konflikte bei der Interaktion mit verteilten Teilnehmern wurden erläutert und konnten über die gewählten Technologien bewältigt werden. System- und Abnahmetests haben die Funktionsfähigkeit des entwickelten Systems bestätigt.

Als abschließendes Resümee des Designs würde in einer Erweiterung der Anwendung das Design weiter angepasst und hierzu die bereitgestellte Bibliothek von JIRA[3] verwendet werden. Des Weiteren könnte eine *Overlay*-Funktion (CSS Funktion zum Abdunkeln des Hintergrunds) den Fokus verstärken. Zur Realisierung der Serverseite war node.js ideal für einen schnellen Prototypen, allerdings, bezogen auf den Aspekt der internen Architektur, würde bei einer Neuimplementierung wieder *Java* als Entwicklungssprache eingesetzt werden. Node.js ist an einigen Stellen zur Realisierung von synchronen REST-Calls und weiteren Umsetzungen zu Adaptern recht umständlich. Hier sind bzgl. Java vor allem gut dokumentierte Hilfen und bereits erlangtes Wissen vorhanden.

[2] https://www.npmjs.org/ - abgerufen am 30.04.2014
[3] https://docs.atlassian.com/aui/latest/ - abgerufen am 30.04.2014

4. Evaluierung

In diesem Kapitel soll die Evaluierung der entworfenen Software dokumentiert werden. Zunächst wird beschrieben, wie die Erhebung der Daten stattgefunden hat. Dabei wird als Erstes auf die Beobachtung eingegangen gefolgt von einem Interview, einen Fragebogen und der Auswertung der Interaktionsprotokolle seitens der Anwendung.

4.1. Beobachtung

Auch dieses Mal wurde wie bei der Erfassung des Ist-Zustandes das sogenannte Multimomentverfahren eingesetzt. Insgesamt gab es zwei Beobachtungstermine, der erste diente dabei der generellen Erfassung vom Verwendungszustand. Hieraus wurden Fragen für das Gruppeninterview und den Fragebogen konstruiert. Der zweite Termin fand am selben Tag wie das Interview und das Aushändigen des Fragebogens statt. Es sollten hierdurch nochmals die Punkte der vorherigen Beobachtung bestätigt werden.

Als Hilfestellung zur Erfassung wurde auch dieses Mal ein Bogen genutzt. Dieser war identisch zum Erfassungsbogen zur Erfassung des Ist-Zustandes aufgebaut.

4.2. Interview

Das Interview wurde als Gruppeninterview geführt. Der Leitfaden des Gruppeninterviews war wie folgt aufgebaut:

Erfasst wurden zunächst Rahmeninformationen:

1. Dauer
2. Teilnehmer
3. Räumlichkeiten der Befragung
4. Besonderheiten der Befragung

Zu den Besonderheiten zählten, dass das Gespräch als Ton aufgezeichnet wird, wenn die Teilnehmer hierzu ihr Einverständnis geben.

Die Fragen wurden in drei Fragen plus ein Bereich für zusätzliche Bemerkungen unterteilt.

Die drei Fragen waren:

Frage 1. Was hat Euch an der Anwendung gefallen?

Frage 2. Was hat Euch bei der Anwendung gefehlt bzw. gestört?

Frage 3. Was hat Euch davon abgehalten, die Software zu nutzen?

4.3. Fragebogen

Die Fragen gliederten sich in die folgenden sechs:

1. Die Anwendung trägt zum Verständnis des genannten Ticket-Kürzels bei.
2. Die Software gibt mir innerhalb des Meetings einen besseren Gesamtüberblick über den aktuellen Sprint.
3. Die zur Verfügung stehende Ausstattung (Räumlichkeiten, Monitor, Tastatur, Maus und Telefon mit Lautsprecher) stellt kein Hindernis bei der Verwendung der Software dar.
4. Das Daily Scrum der verstreuten Teams wird durch die eingesetzte Software sinnvoll unterstützt.
5. Der aktuelle Ablauf und die Struktur des Daily Scrums ist optimal.

6. Die Funktionsweise der Software ist intuitiv.

Der Fragebogen wurde absichtlich so entworfen, dass ein besserer Vergleich zum Zeitpunkt vor der Einführung und nach Einführung der Software möglich ist. Hierzu wurde die erste Frage zur Frage 2 im Fragebogen der Ist-Analyse erstellt. Sie zielt zum einen auf die potenzielle Verbesserung durch die Software und zum anderen auf die Erfüllung der Anforderung ab. Die zweite Frage wurde im Hinblick auf die Beobachtung konstruiert. Über Frage drei soll herausgefunden werden, ob die bereits in der Ist-Analyse herausgefundene Problematik der Ausstattung eine optimale Nutzung der Software behindert. Sie steht in Verbindung mit der Frage 3 im Fragebogen der Ist-Analyse. Mit der Frage 4 wird eine Verknüpfung zur Frage 4 des Ist-Bogens hergestellt und soll eine Verbesserung hervorheben und eine weitere Bestätigung der Software ergeben. Frage 5 stellt wieder einen direkten Vergleich zwischen vorher und nachher mit der Frage 5 aus der Ist-Analyse. Die letzte Frage soll zeigen, ob die Software die Anforderung „Die Anwendung soll eine möglichst intuitive Bedienung ermöglichen." decken kann.

4.4. Interaktionsprotokolle

Die Interaktionsprotokolle waren ein aktiver Bestandteil der laufenden Software und wurden mittels des NPM Package winston[1] unterstützend integriert. Die Erfassung lief dabei rein Serverseitig. Verfolgt wurden dabei sämtliche Interaktionen, wie Anmelden von Sessions, Eingabe von Authentifizierungen, der Austausch zwischen Anwendung und JIRA, das Auslösen von Events wie Markieren von Nachrichten und viele mehr. Es wurde aber darauf geachtet, keine sensiblen Daten, wie direkte Authentifizierungsinformationen, darzustellen. Es handelte sich nur um die zeitliche direkte Erfassung von Interaktionen. Zur Gruppierung der Daten wurden die Kategorien *info*, *rest*, *websocket* und *error* festgelegt. Die Kategorie „info" hat allgemeine Informationen dargestellt, wie starten des Servers oder logische Events, wie starten der Verteilung von Markierungsnachrichten. Über die Kategorie „rest" wurden die Aktionen dokumentiert, die die Kommunikation zwischen JIRA und der Serverapplikation betreffen. Über der Kategorie „websocket" wurde die Kommunikation zwischen Client und Server dokumentiert. Zur Erfassung der Fehlermeldungen gab es zusätzlich die Kategorie „error".

[1] https://www.npmjs.org/package/winston - abgerufen am 27.04.2014

4.5. Auswertung

Die folgenden Unterkapitel geben die Ergebnisse der Evaluierungsschritte wieder. Jeder Punkt wird dabei als einzelnes Ergebnis interpretiert. In Kapitel 4.6 wird dann eine zusammenfassende Darstellung der Erkenntnisse aufgezeigt.

4.5.1. Beobachtung

Die Beobachtungen der beiden Termine wurden in Hamburg durchgeführt. Die Dauer der Meetings hat auch dieses Mal nicht die einzuhaltenden 15 Minuten überschritten. Die Beteiligung der Teilnehmer war am 16. Februar 2014 sehr gering, es waren nur vier Teilnehmer anwesend, davon fielen zwei Entwickler plus Product Owner auf Hamburg und ein Entwickler auf München. Am 18. März 2014 waren weitaus mehr Personen anwesend. Hier waren zwei Entwickler in München, einer davon im Home-Office und drei Entwickler plus Product Owner und Scrum Master in Hamburg am Meeting beteiligt. Die Räumlichkeiten der beiden Termine waren mit Ausnahme des Entwicklers im Home-Office in den regulären Räumen, wie bereits in der Ist-Analyse beschrieben wurde. Auch hier wurde besonders die Fehlpositionierung des Monitors und der Teilnehmer im Büro deutlich. Diese sind mit der aktuell gegebenen Ausstattung und dem Raum nicht zu optimieren. Im Gegensatz zu Ist-Analyse, war kein JIRA-Taskboard offen, sondern die entwickelte Anwendung. Die restlichen Medien waren identisch. Dies traf auf beide Termine zu. Der Ablauf orientierte sich wie bisher am Standard. Die Ticketerwähnung war an beiden Terminen extrem gering, besonders bei der ersten Beobachtung. Es wurde wieder deutlich, dass ein Großteil der Fülle an Diskussionsthemen außerhalb der Tickets statt fand. Allgemein bewegte sich ein Meetingteilnehmer, wenn er dran war, zum Telefonhörer und bediente parallel die Maus, um seine Tickets zu markieren. Andere lokale Teilnehmer beobachteten diese Aktion und lokalisierten die Tickets direkt am Monitor, was allerdings durch die Größe des Monitors und den Räumlichkeiten nicht ideal war. Es schien so, als ob im Gegensatz zu vorher eine Verbesserung der Orientierung stattfand. Beim zweiten Termin fiel besonders die sichtbare Assoziation eines Tickets durch einen Mausklick auf. Hier wurde zunächst nur eine aktive Aufgabe beschrieben ohne ein direktes Ticketkürzel zu nennen und erst durch den Klick wurde die Verbindung hergestellt. Auch das Einbeziehen eines Entwicklers im Home-Office stellte kein Problem dar.

4.5.2. Interview

Das Gruppeninterview fand auch dieses Mal wieder in Hamburg, ausschließlich mit den dort ansässigen Teilnehmern und direkt nach dem regulären Meeting, statt. Die Dauer des Interviews beschränkte sich auf sechs Minuten. Insgesamt nahmen fünf Personen teil, wovon drei Entwickler, einer Product Owner und einer Scrum Master waren. Zur besseren Nachbereitung und zum Erfassen aller relevanten Aussagen wurde das Gespräch im Einverständnis aller Anwesenden aufgezeichnet.

Zur Frage 1 „Was hat Euch an der Anwendung gefallen?“ ergab das Feedback die folgenden relevanten Aussagen:

Antwort 1. Es wurde strukturiert sichtbar, woran man gerade arbeitet.

Antwort 2. Man ruft sich in Gedanken, woran man gerade arbeitet.

Antwort 3. Man erkennt, welche Tickets man noch offen hat.

Antwort 4. Es wird sichtbar, wenn man nicht an einem Ticket arbeitet.

Antwort 5. Es fällt im Meeting auf, wenn Aufgaben nicht als Ticket erfasst wurden.

Antwort 6. Es wird bemerkt, wenn keine Tickets bearbeitet werden.

Antwort 7. Das Meeting ist fokussierter. Man versucht noch besser sich am Standardablauf zu orientieren.

Antwort 8. Das Eingrenzen / Filtern auf Namen.

Antwort 9. Die Anwendung läuft sehr flüssig.

Antwort 10. Der Gesamtüberblick über den Sprint.

Das Feedback zur ersten Frage war sehr deutlich und positiv. Die zweite Frage „Was hat Euch bei der Anwendung gefehlt bzw. gestört?“ ergab keine gravierenden Aussagen.

Antwort 1. Man könnte den Wechsel von Markierungen über die Tastatur regeln.

Antwort 2. Das Design könnte man an den JIRA-Style[2] anpassen.

[2] https://docs.atlassian.com/aui/latest/ - abgerufen am 28.04.2014

Antwort 3. Die Aktualisierung der Tickets könnte noch verbessert werden.

Hauptsächlich ergab diese Frage einen *Feature-Request* und zwei *Improvements*, welche in einer zukünftigen Version einfach zu realisieren sind.

Die Frage 3 „Was hat Euch davon abgehalten die Software zu nutzen?“ ergab folgende Erkenntnisse:

Antwort 1. Die Größe des Monitors. Ein größerer mit Touch-Funktionalität könnte hier behilflich sein.

Antwort 2. Maus und Tastatur als Eingabegeräte stören.

Antwort 3. Die Räumlichkeiten und besonders der Aufwand zur Interaktion mit Maus und Tastatur stören. Jeder muss sich zur Interaktion zum Monitor drängen.

Antwort 4. Vergessen das Board zu öffnen.

Zusätzliche Bemerkungen ergaben sich im Gruppeninterview nicht, allerdings im folgenden Unterkapitel 4.5.3.

4.5.3. Fragebogen

Der Fragebogen wurde teilweise direkt nach dem Interview oder digital ausgefüllt. Wie im Unterkapitel 4.5.2 bereits erwähnt wurde, gab es zusätzliche Bemerkungen an den Fragen selbst.

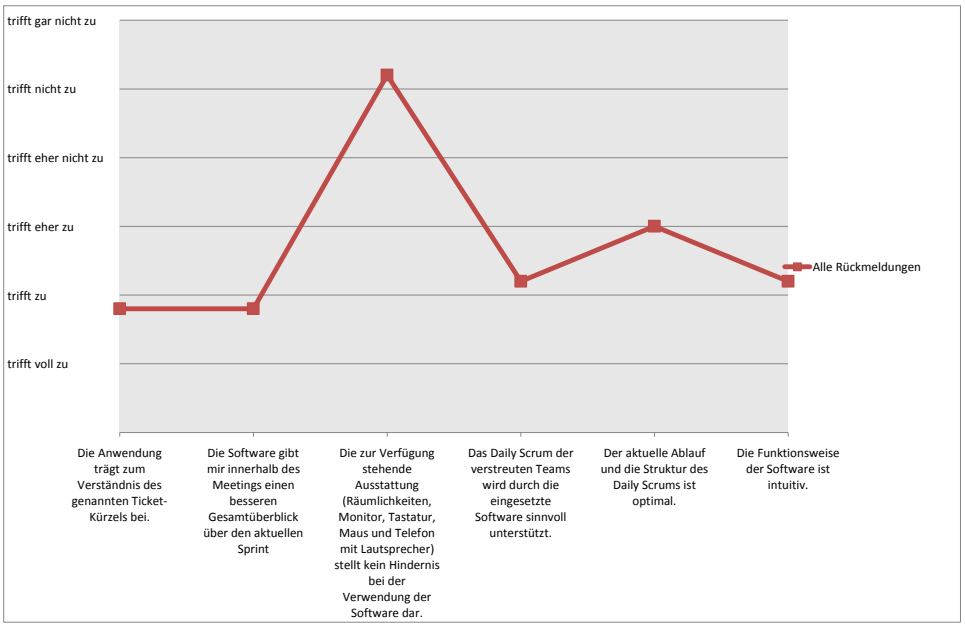

Abbildung 4.1.: Grafische Darstellung der Ergebnisse im Mittel

Die Beantwortung der Fragen im Mittelwert aller Antworten ist im Diagramm bzw. der Abbildung 4.1 dargestellt. Mit der Frage 1 wurde bestätigt, dass die Anwendung zum Verständnis der genannten Ticket-Kürzel beiträgt. Hier gab es den schriftlichen Zusatz, dass es nur nicht „voll zutrifft“ , „weil man parallel zuhören und lesen muss, was sich nicht vermeiden lässt“. Wie auch im Gruppeninterview bereits erwähnt, wurde bei der Frage 2 nochmals gezeigt, dass ein besserer Gesamtüberblick über den Sprint gegeben ist. Dies wurde mittels eines zusätzlichen Kommentars eines Antwortenden „besonders durch die Filtermöglichkeit“ begründet. Auch die optimale Ausnutzung der Anwendung selbst wird wie erwartet durch die Ausstattung (Räumlichkeiten, Monitor, Maus, etc.) verhindert. Hier gab es den Zusatz auf einem Fragenbogen, „größerer Monitor, schnurlose Eingabe wäre schön“. Der generelle Ablauf des Daily Scrums wird eher mit „optimal“ bewertet. Die Anforderung, eine intuitive Software zu entwerfen, wurde erfüllt bzw. wurde bestätigt.

4.5.4. Interaktionsprotokolle

Die Interaktionsprotokolle ergaben abzüglich Feiertagen, Wochenenden und nicht stattfinden der Daily Scrums (z. B. Sprint Planning-Tag) eine Nutzung von ca. 75 %. Dabei lag die Anzahl der Verbindungen, die mit der Anwendung hergestellt wurden, im Durchschnitt bei sieben pro Tag. Die Anzahl der Markierungen bewegte sich von *minimal keinem* bis *maximal 14* und lag im Schnitt bei fünf Markierungen pro Meeting. Interessant war, dass man den Versuch unternahm, eine Interaktion auszuführen, die nicht komplett an das bestehende JIRA-Board angepasst wurde. Es handelt sich dabei um die Möglichkeit, ein Ticket in einen anderen Zustand zu setzten (von Status „open“ in „in progress“). Diese Aktion wurde im Testsetup erfolgreich mit allen Endgeräten (auch via Touch) getestet, allerdings aufgrund des komplizierten Filterausdruckes der Produktions-Umgebung (aktiv genutzte Arbeitsumgebung) ist sie nicht zu 100 Prozent freigegeben worden. Es wurden neben diesen beiden Aktionen alle verfügbaren Funktionen genutzt. Nur die aufsteigende und absteigende Sortierung der Bearbeiter oder Ticket-ID’s wurde nicht regelmäßig verwendet.

4.6. Fazit

Die erste Beobachtung, welche zum Entwerfen der Fragen zum Interview und Fragebogen verwendet wurde, konnte die erwarteten Ergebnisse hervorrufen. Beobachtungen wurden bestätigt und untermauert. Dies zeigte sich besonders im Interview, aber auch in der Auswertung der Fragebögen.

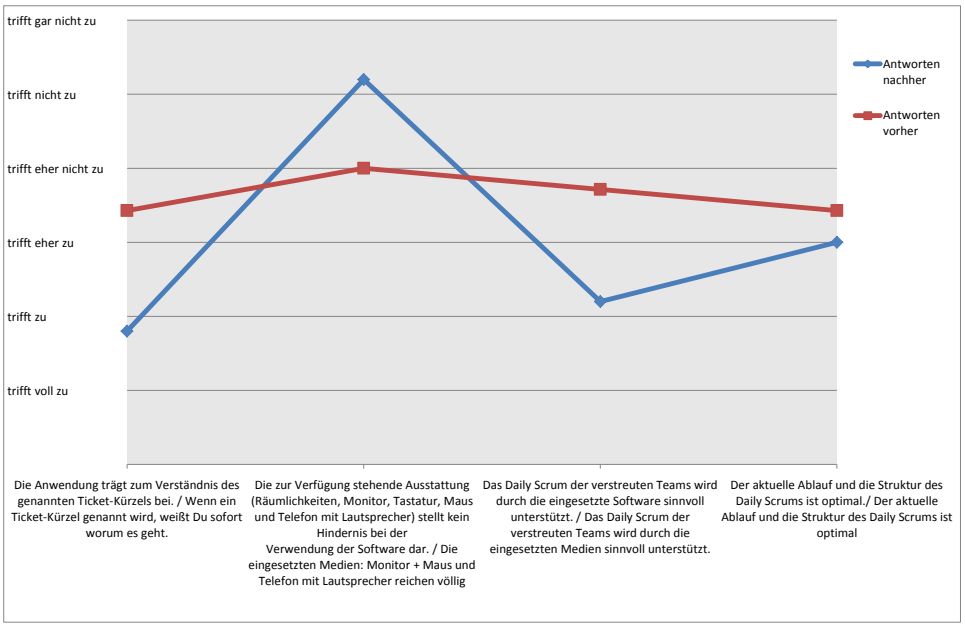

Abbildung 4.2.: Grafische Darstellung der Ergebnisse im Mittel und Vergleich

Die Abbildung 4.2 zeigt den Vergleich der Antworten aus der Evaluierung und der Antworten der Ist-Analyse. Die Fragen wurden im Diagramm im Format Evaluierungsfrage / Ist-Analyse-Frage dargestellt. Die Frage 1 zeigt eine deutliche Verbesserung der Orientierung bzw. des Verständnisses eines Ticketkürzels im Vergleich zur vorherigen Situation. Man kann somit ableiten, dass die Orientierungslosigkeit zu mindestens reduziert wurde. Dies wurde auch wie beschrieben in der Beobachtung und dem Gruppeninterview bestätigt. Über die Frage 2 kann man eine ungenügende Ausstattung und auch eine Behinderung beim Einsetzen neuer Software zur Unterstützung des Meetings interpretieren. Innerhalb des Interviews und der Beobachtung war dies auch deutlich zu erkennen. Die Untersuchung hat somit auch dazu beigetragen, dass dieser Punkt bewusster wahrgenommen wird. Allerdings zeigt Frage 3 die bessere Unterstützung der vorhandenen Ausrüstung mit der evaluierten Software. Die Nutzung der vorhandenen Mittel wurde somit verbessert, obwohl sich ebenfalls ein Hindernis bei der Verwendung darstellt. Ebenfalls zeigt sich eine Optimierung des aktuellen Meetings mittels der entworfenen Anwendung über die Frage 4. Im Gruppeninterview wurde hierzu erwähnt,

dass das Meeting fokussierter abläuft und besonders Aufgaben sichtbar wurden, die nicht als Ticket erfasst wurden.

Trotz der teils geringen Anzahl von Tickets ist die Evaluierung mittels der verwendeten Evaluierungsmaßnahmen eine positive Aussage bzgl. der Verwendung zu treffen. Die Software wurde innerhalb der drei Monate regelmäßig verwendet und zeigt Potential. Eine bessere Ausstattung bzw. eine Veränderung des Ablaufes könnte hier neue Erkenntnisse bringen. Auch die vollständige Integration der Funktion „Ticketstatus wechseln“ für die entsprechende Evaluierungsumgebung könnte das Board für weitere Einsatzmöglichkeiten bereitstellen. Ebenfalls wäre eine weitere Funktion zu implementieren, mit der man Tickets einem Bearbeiter zuweisen könnte. Mit diesem Funktionsumfang könnten ggf. weitere Meetings im Verteilungs-Aspekt unterstützt werden.

Die Umsetzung der Client-Seite durch HTML5 bietet zusätzlich zu Standard-Desktops die Variante weiterer Endgeräte wie Smartphones einzubinden. Daraus könnten weitere Szenarien entstehen, wie das Einbinden von Mitarbeitern auf Dienstreisen, welche mittels Smartphone in ein Meeting mit einbezogen werden können. Auch die Verwendung der Endgeräte als reines Interaktionsmedium in einem weiteren Meeting-Szenario wäre eine Möglichkeit. Ein Meeting, welches über einen Beamer eine große Darstellungsfläche bietet, könnte über die Endgeräte gesteuert werden. Teammitglieder hätten somit jeweils eine eigene Ausführungsmöglichkeit zur Markierung von Tickets. Ein Interaktionsleiter, welcher sonst alle Aktionen auf Zuruf ausgeführt hätte, wäre nicht nötig. Weiterhin ist zu untersuchen, ob große Touchscreens in Verbindung mit der entwickelten Anwendung die Evaluierungsumgebung hinsichtlich der technischen Ausstattung verbessern könnten.

5. Schluss

Dieses Kapitel soll zunächst eine Zusammenfassung der gesamten Arbeit darstellen und eine abschließende Bewertung geben. Der Abschnitt Ausblick (5.2) soll zudem weiterführende Fragestellungen aufzeigen, welche sich im Zusammenhang mit dieser Arbeit ergeben haben.

5.1. Zusammenfassung

Inhalt dieser Arbeit war es, einen Prototypen zu entwickeln, welcher eine interaktive Unterstützung von Scrum Prozessen im Kontext der verteilten Softwareentwicklung ermöglicht. Hierzu wurde speziell das Daily Scrum-Ritual als eines der wichtigsten Kommunikationsmeetings gewählt.

Zur Bewältigung dieser Aufgabe wurde eine Analyse eines Daily Scrums (2.4) in einem großen deutschen Unternehmen im Dienstleistungsgewerbe durchgeführt. Die Beobachtungen (2.4.2.1), das Gruppeninterview (2.4.2.2) und der Fragebogen (2.4.2.3) haben eine deutliche Orientierungslosigkeit bei der Erfassung der Aufgabe hinter einem Ticketkürzel gezeigt sowie eine mangelnde Ausstattung der Meetingstandorte für die vorhandene Gruppengröße. Der Bedarf einer Unterstützung wurde somit bestätigt. Basierend auf diesem Praxisszenario haben Anforderungen (2.5) und eine resultierende Spezifikation (2.6) eine zu entwerfende Lösung skizziert. Vergleichbare Ansätze (2.7) haben gezeigt, dass eine neue Lösung außerhalb der existierenden Anwendungen benötigt wird.

Aus den Erkenntnissen der Analyse (2) wurde dann ein Entwurf für eine entsprechende Anwendung definiert (3). Benötigte Grundmodule (3.1) und Abgrenzungen (3.2) durch das gegebene JIRA-Systemen zeigten bereits vorhandenen und noch zu implementierende Anforderungen. Zum Verständnis der Anwendung haben Entwurfsmuster (3.3) im Abschnitt 3.3 in Verbindung mit den entwickelten Sichten (3.3.2) beigetragen. Das Erscheinungsbild der Anwendung zum

Client hat im Abschnitt GUI Design (3.4) eine ausführliche Darstellungsvariante beschrieben sowie Interaktionsmöglichkeiten aufgezeigt. Auftretende Konflikte beim verteiltem Zugriff (3.5) wurden mittels eines Mutex-Ansatzes gelöst und die Funktionsfähigkeit über Testfälle (3.6) sichergestellt.

Die Evaluierung (4) wurde im Vorgehen beschrieben und durchgeführt. Auch hier haben zur Gewinnung der Ergebnisse Beobachtungen (4.1), Interview (4.2), Fragebogen (4.3) und Interaktionsprotokolle (4.4) beigesteuert. Das Resultat aus der Evaluierung hat die Reduzierung der erfassten Orientierungslosigkeit gezeigt. Das positive Feedback hat besonders das Fokussieren der Tickets, das Sortieren nach Bearbeiter und den daraus verbesserten Überblick des Sprints gelobt. Der Fragebogen (4.5.3) hat den besseren Überblick bestätigt, wie auch das bessere Verständnis hinter einem Ticket, die sinnvolle Unterstützung des Meetings und die intuitive Funktionsweise der Software. Über die Interaktionsprotokolle (4.5.4) war eine regelmäßige Nutzung der Software zu erkennen.

Der Abschluss dieses Kapitels mit dem Fazit (4.6) hat eine erfolgreiche Lösung nochmals zusammengefasst, bestätigt und einen Vergleich der Fragebögen zum Zeitpunkt der Ist-Analyse (2.4.2.3) und der Evaluierung (4.5.3) der Anwendung hergestellt. Der Vergleich hat zudem eine wahrgenommene Verbesserung der Orientierung zum Ticket, eine optimierte Nutzung der vorhandenen Soft- und Hardware-Ressourcen und Steigerung der Effizienz beim Ablauf des Daily Scrums aufgezeigt.

5.2. Ausblick

Die Ergebnisse dieser Arbeit haben gezeigt, dass die Unterstützung zur Überbrückung der vorliegenden Distanz zwischen verstreuten Teams über eine Softwarelösung möglich ist. Allerdings wurde deutlich, dass die vorliegende technische Ausstattung ein Hindernis darstellt. Als Anregung aus dem Feedback könnten *Handsfree*-Umgebungen mit unterstützenden Touch-Display bei der Verwendung des Prototypens von Vorteil sein. Wie in der Einleitung beschrieben, werden sich zukünftig Unternehmen näher mit dem Aspekt der „Verteilung von Teams im Kontext der agilen Entwicklung“ beschäftigen müssen. Hieraus ergeben sich neue Herausforderungen für das mittlere Management, indem die Frage des Präsentseins von Teammitgliedern mit der Unterstützung von geeigneter technischer Ausstattung neu definiert werden könnte. Insbesondere das Sprint-Planning, welches im beobachteten Entwicklungsteam das Reisen von Mitarbeitern fordert, könnte mit weiteren Funktionen der Anwendung, wie dem Zuweisen

eines neuen Bearbeiters oder Erstellen eines Tickets, eine neue Fragestellung formulieren. Die Frage ob Tools *Face to Face*-Meetings im herkömmlichen Sinne (Offline) ersetzen können, gilt es weiterführend zu untersuchen. Es hat sich allerdings gezeigt, dass die durchschnittliche Kommunikationsdauer zwischen verteilten zu nicht-verteilten Teams im Verhältnis 45 Minuten zu 75 Minuten liegt (Niinimäki, 2011) und somit gegenüber aktuell verwendeten Strategien noch ein deutlicher Nachteil zu erkennen ist.

Die Theorie, dass Enterprise 2.0 im Aspekt der agilen und verteilten Entwicklung über passende Tool Changes unterstützend sein kann, scheint sich zu bestätigen. Allerdings sind im Bezug auf die Ausdehnung der Verteilung weitere Punkte wahrzunehmen. So sind unter anderem auch kulturelle Differenzen zu berücksichtigen, welche Blockaden in internationalen Teams setzen können. Große Zeitzonenunterschiede verhindern zusätzlich synchrone Kommunikation und fördern asynchrone Kanäle. Inwieweit sich diese allerdings eignen, ist kritisch zu betrachten.

Literaturverzeichnis

[Balzert 2011] Balzert, Helmut: Architektur- und Entwurfsmuster. In: *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. Spektrum Akademischer Verlag, 2011, S. 37–107. – URL http://dx.doi.org/10.1007/978-3-8274-2246-0_8. – Abruf: 2014-02-21. – ISBN 978-3-8274-1706-0

[Barkalov u. a. 2013] Barkalov, Igor ; Martin, Johannes ; Mume, Sibylle ; Metz, Michael ; Wagner, Philipp ; Jiang, Kevin: *Forward Visibility Studie (2. Auflage)*. Capgemini Consulting, 2013. – URL http://www.de.capgemini-consulting.com/resource-file-access/resource/pdf/forward-visibility-edition-2.pdf. – Abruf: 2014-05-10

[Benedetti und Cranley 2012] Benedetti, Ryan ; Cranley, Ronan: *jQuery von Kopf bis Fuß : [ein Buch zum Mitmachen und Verstehen]*. Beijing u. a. : O'Reilly, 2012. – URL http://d-nb.info/1019450436/04. – Abruf: 2014-02-21. – ISBN 978-3-86899-189-5; 3-86899-189-1

[Deitel und Deitel 2008] Deitel, P.J. ; Deitel, H.M.: *Ajax, Rich Internet Applications, and Web Development for Programmers*. Prentice Hall, 2008 (Deitel developer series). – ISBN 9780131587380

[Eckstein 2012] Eckstein, Jutta: *Agile Softwareentwicklung in großen Projekten Teams, Prozesse und Technologien - Strategien für den Wandel im Unternehmen*. 2. Heidelberg : dpunkt.verlag GmbH, 2012. – ISBN 978-3-89864-790-8

[Fischbach und Putzke 2012] Fischbach, Kai ; Putzke, Johannes: *Wissensarbeiter*. In: Kurbel, Karl (Hrsg.) ; Becker, Jörg (Hrsg.) ; Gronau, Norbert (Hrsg.) ; Sinz, Elmar J. (Hrsg.) ; Suhl, Leena (Hrsg.): *Enzyklopädie der Wirtschaftsinformatik*, URL http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/daten-wissen/Wissensmanagement/

Wissensorganisation--Instrumente-der-/Wissensarbeiter, 2012. – Online-Lexikon ; Abruf: 2014-02-21

[Franz 2007] Franz, Klaus: *Handbuch zum Testen von Web-Applikationen : Testverfahren, Werkzeuge, Praxistipps.* URL http://dx.doi.org/10.1007/978-3-540-68185-4, 2007. – Abruf: 2014-02-21. – ISBN 978-3-540-68185-4

[Freeman und Robson 2012] Freeman, Eric ; Robson, Elisabeth: *HTML5-Programmierung von Kopf bis Fuß : Webanwendungen mit HTML5 und JavaScript.* Beijing u. a. : O'Reilly, 2012. – URL http://d-nb.info/1018961895/04. – Abruf: 2014-02-21. – ISBN 978-3-86899-182-6; 3-86899-182-4

[Goll 2011] Goll, Joachim: Einführung in standardisierte Diagrammtypen nach UML. In: *Methoden und Architekturen der Softwaretechnik.* Vieweg+Teubner Verlag, 2011, S. 383–494. – URL http://dx.doi.org/10.1007/978-3-8348-8164-9_11. – Abruf: 2014-02-21. – ISBN 978-3-8348-1578-1

[Hanser 2010] Hanser, Eckhart: Extreme Programming (XP). In: *Agile Prozesse: Von XP über Scrum bis MAP.* Springer Berlin Heidelberg, 2010 (eXamen.press), S. 13–45. – URL http://dx.doi.org/10.1007/978-3-642-12313-9_3. – Abruf: 2014-02-21. – ISBN 978-3-642-12312-2

[Heinrich 2007] Heinrich, G.: *Allgemeine Systemanalyse.* Oldenbourg, 2007. – URL http://books.google.de/books?id=OXDevFRps7IC. – Abruf: 2014-02-21. – ISBN 9783486583656

[Häuslein 2004] Häuslein, A.: *Systemanalyse: Grundlagen, Techniken, Notierungen.* Vde Verlag GmbH, 2004. – URL http://books.google.de/books?id=x3VEAQAACAAJ. – Abruf: 2014-02-21. – ISBN 9783800727155

[Kaack u. a. 2014] Kaack, Alexander ; Schwarzer, Jan ; Barnkow, Lorenz ; Luck, Kai von: Ein interaktives Taskboard zur Unterstützung von verteilten Scrum-Teams. In: *erscheint als Proceedings der WIWITA*, 2014

[Mcafee 2006] Mcafee, A.P.: Enterprise 2.0: the dawn of emergent collaboration. In: *Engineering Management Review, IEEE* 34 (2006), Nr. 3, S. 38–47. – ISSN 0360-8581

[Meidl 2014] Meidl, Oliver: Technische Anforderungen. In: *Globales Webdesign.* Springer Fachmedien Wiesbaden, 2014 (essentials), S. 9–12. – URL http://dx.doi.org/10.1007/978-3-658-04088-8_3. – Abruf: 2014-02-21. – ISBN 978-3-658-04087-1

[Müller 2014] Müller, Peter: *Einstieg in CSS : Webseiten gestalten mit HTML und CSS ; [Grundlagen: Schriften, Farben und Box-Modelle ; Konzepte: Spezifität, Kaskade und Positionierung verständlich erklärt ; Gestaltung: Navigation, mehrspaltige Layouts und Media Queries ; aktuell zu HTML5 und CSS3].* Bonn : Galileo Press, 2014. – URL `http://d-nb.info/1037931475/04`. – Abruf: 2014-02-21. – ISBN 978-3-8362-2776-6; 3-8362-2776-2

[Niinimäki 2011] Niinimäki, Tuomas: Face-to-Face, Email and Instant Messaging in Distributed Agile Software Development Project. In: *Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on*, Aug 2011, S. 78–84

[OMG 2011] OMG: *Business Process Model and Notation.* URL `http://www.omg.org/spec/BPMN/2.0/PDF`, 2011. – Abruf: 2014-02-10

[Paasivaara u. a. 2009] Paasivaara, M. ; Durasiewicz, S. ; Lassenius, C.: Using Scrum in Distributed Agile Development: A Multiple Case Study. In: *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, July 2009, S. 195–204

[Pichler 2009] Pichler, Roman: *Scrum - Agiles Projektmanagement erfolgreich einsetzen.* Heidelberg : dpunkt-Verl., 2009. – ISBN 3898644782 9783898644785

[Schneider und Romano 2012] Schneider, Gabriel ; Romano, Roger: *ICT-Geschäftsprozessunterstützung und Akzeptanzförderung : Grundlagen zur Prozessoptimierung und Veränderungsbegleitung mit Beispielen, Fragen und Antworten.* 1. Aufl. Zürich : Compendio Bildungsmedien, 2012. – 272 S

[Schwaber 2011] Schwaber, J.: *The Scrum Guide.* URL `http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf`, 2011. – Abruf: 2014-04-23

[Schwarzer 2012] Schwarzer, Jan: *Enterprise Mirrors - Interaktive und ubiquitäre Benutzungsschnittstellen für Unternehmen*, Hamburg, Hochschule für Angewandte Wissenschaften, Fak. Technik und Informatik, Dep. Informatik, Diplomarbeit, 2012. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/schwarzer.pdf`. – Abruf: 2013-05-06

[Sommerville 2007] Sommerville, Ian: *Software-Engineering.* München; Boston [u. a.] : Pearson Studium, 2007. – ISBN 9783827372574 3827372577

[Stapelkamp 2010] Stapelkamp, Torsten: Interfacedesign. In: *Interaction- und Interfacedesign*. Springer Berlin Heidelberg, 2010 (X.media.press), S. 150–287. – URL http://dx.doi.org/10.1007/978-3-642-02074-2_3. – Abruf: 2014-02-21. – ISBN 978-3-642-02073-5

[Stray u. a. 2012] Stray, V.G. ; Moe, N.B. ; Aurum, A.: Investigating Daily Team Meetings in Agile Software Projects. In: *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, Sept 2012, S. 274–281

[VersionOne 2013] VersionOne: *8th Annual State of Agile Survey*. URL http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf, 2013. – Abruf: 2014-04-23

[Wirdemann 2011] Wirdemann, Ralf: *Scrum mit User Stories*. München [u. a.] : Hanser, 2011. – ISBN 9783446426603 3446426604

[Zeitelberger 2012] Zeitelberger, J. L.: Management für Organisationen – Jetzt mit 2.0. In: *e & i Elektrotechnik und Informationstechnik* 129 (2012), Nr. 2, S. 76–78. – URL http://dx.doi.org/10.1007/s00502-012-0081-5. – Abruf: 2014-02-21. – ISSN 0932-383X

Abbildungsverzeichnis

Tabellenverzeichnis

A. Anlagen

A.1. Erfassung des Ist-Zustandes

Gedächtnisprotokoll zur Analyse des Daily Scrums *Datum: 28.11.2013*

1. Rahmeninformationen

Anwesende Rollen:
Entwickler_____
Hamburg *München*
Scrum-Master:
Hamburg ☐ München ☐
Product Owner:
Hamburg ☐ München ☐
Sonstige anwesende Personen:
Teilnehmer Hamburg:_____

Teilnehmer München:_____

Dauer(Von/Bis):_____

Räumlichkeiten:
Hamburg:_____

München:_____

2. Eingesetzte Medien:

_____

3. Tickets

Anzahl genannter Tickets:_____

Wechsel von Ticket kleiner 5 Sekunden:_____

Wechsel von Ticket größer gleich 5 Sekunden:_____

4. Besonderheiten:

_____

_____

_____

_____

1

Gruppeninterview zur Analyse des Daily Scrums *Datum: 28.11.2013*

1. Rahmeninformationen

Anwesende Rollen:
Entwickler_____
Hamburg *München*
Scrum-Master:
Hamburg ☐ München ☐
Product Owner:
Hamburg ☐ München ☐
Sonstige anwesende Personen:
Teilnehmer Hamburg:_____

Teilnehmer München:_____

Dauer(Von/Bis):_____

2. Seht Ihr Störfaktoren im Daily Scrum?

_____

_____

_____

_____

3. Was könnte zusätzlich zur Effizienz des Meetings beitragen?

_____

_____

_____

_____

4. Zusätzliche Bemerkungen:

_____

_____

Fragebogen zur Analyse des Daily Scrums *Datum: 28.11.2013*

1. **Deine Vorbereitungszeit zum Daily Scrum beträgt:**

   _____

2. **Wenn ein Ticket-Kürzel genannt wird, weißt Du sofort worum es geht.**
   Bitte beantworte die Frage auf einer Skala von 1 bis 6.

trifft voll zu	trifft zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	trifft gar nicht zu
☐	☐	☐	☐	☐	☐
1	2	3	4	5	6

3. **Die eingesetzten Medien: Monitor + Maus und Telefon mit Lautsprecher reichen völlig aus.**
   Bitte beantworte die Frage auf einer Skala von 1 bis 6.

trifft voll zu	trifft zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	trifft gar nicht zu
☐	☐	☐	☐	☐	☐
1	2	3	4	5	6

4. **Das Daily Scrum der verstreuten Teams wird durch die eingesetzten Medien sinnvoll unterstützt.**
   Bitte beantworte die Frage auf einer Skala von 1 bis 6.

trifft voll zu	trifft zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	trifft gar nicht zu
☐	☐	☐	☐	☐	☐
1	2	3	4	5	6

5. **Der aktuelle Ablauf und die Struktur des Daily Scrums ist optimal.**
   Bitte beantworte die Frage auf einer Skala von 1 bis 6.

trifft voll zu	trifft zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	trifft gar nicht zu
☐	☐	☐	☐	☐	☐
1	2	3	4	5	6

Gedächtnisprotokoll des 1. Beobachtungstermins

Daily Scrums - Standort Hamburg

Alexander Kaack
alexander.kaack@haw-hamburg.de

20. November 2013

1 Dauer

- 11:45 Uhr bis 12:00 Uhr →15 Minuten

2 Teilnehmer

- Das Team besteht aus 15 Mitgliedern.
- Anwesende Mitglieder
  - 2 Entwickler aus München, beide männlich
  - 2 Entwickler + Product Owner aus Hamburg, alle männlich

3 Räumlichkeiten innerhalb der Standorte

Das Daily Scrum findet in einem regulären Büro statt. Die Mitglieder versammeln sich entsprechend am jeweiligen Standort in einem geeigneten Büro.

4 Eingesetzte Medien

Monitor + Maus mit Sicht auf das JIRA Taskboard. Telefon + Hörer + Lautsprecher. Telefonhörer als Speech-Token.

5 Ablauf

1. Eröffnung des Daily Scrum aus Hamburg.
2. Derjenige, der die Runde eröffnet hat, startet.
3. Generell wird sich an die drei Fragen gehalten:

> 1. Was habe ich seit dem letzen Daily Scrum erreicht?
> 2. Was plane ich bis zum nächsten Daily Scrum?
> 3. Welche Hindernisse gibt es?
>
> (Wirdemann, 2011, S. 158)

5.1 Fazit

Der Ablauf orientiert sich stark am Standardablauf eines Daily Scrum und zeigt keine direkten Abweichungen.

6 Tickets

Es wurden insgesamt hörbar direkt sieben Ticketkürzel angesprochen. Davon wurden drei aus Hamburg und vier aus München genannt. Der kürzeste Zeitabstand zwischen zwei genannten Ticketkürzeln lag bei unter fünf Sekunden.

7 Besonderheiten

- Wenn ein Ticket erwähnt wurde, schauten alle Teilnehmer schnell zum Monitor.
- Ab und zu wurde die Maus genutzt um das Ticket zu finden.
- Je nach Tempo des Berichterstatters, war es nicht möglich, den Titel bzw. Status des Tickets über den Monitor zu erfassen.

8 Sonstiges

- Es war der erste Tag nach dem Sprint Planning und das Team war dünn besetzt.
- Die Zeit von 15 Minuten wurde nicht ausschließlich für eine kurze Berichtserstattung genutzt und es fand ein *Story Telling* sowie *Problem Solving* statt.
- Es wurden auch Aufgaben genannt, die nicht als Ticket erfasst wurden.

9 Ergebnis

Durch die zeitliche Begrenzung des Daily Scrum ist jeder im Team bedacht, schnell die drei Fragen zu beantworten. Die Kürzel der Tickets helfen zusätzlich zur Identifikation der Aufgaben. Eine Beschreibung der Aufgabe sowie weitere Informationen finden sich am Ticket selber. Solange ein Blick auf das angesprochene Ticket zur Verfügung steht, kann jeder den Status und die Beschreibung der Aufgabe nachverfolgen. Der *Speech Token Owner* muss hierzu natürlich auf das Ticket zeigen, um alle im Raum auf das Ticket zu aufmerksam zu machen. Dies könnte mit einer Maus oder anderen Gesten geschehen. Problematisch ist hierbei die räumliche Distanz der Teammitglieder. Wird ein Ticket angesprochen, müssen die Teammitglieder der entfernten Standorte das Ticket suchen. Der zeitliche Rahmen lässt allerdings ein Warten nicht wirklich zu, bzw. würde das Meeting zusätzlich in die Länge führen. Die einzige Möglichkeit der Berichtsverfolgung ist, dass der Status und die Beschreibung des genannten Ticketkürzels bereits bekannt ist. Meine Vermutung ist zusätzlich, dass der Fokus kurz durch die Orientierungssuche unterbrochen wird.

Literatur

[Wirdemann 2011] Wirdemann, Ralf: *Scrum mit User Stories*. München [u. a.] : Hanser, 2011. – ID: 846279372. – ISBN 9783446426603 3446426604

Gedächtnisprotokoll des 2. Beobachtungstermins

Daily Scrums - Standort Hamburg

Alexander Kaack
alexander.kaack@haw-hamburg.de

27. November 2013

1 Dauer

- 11:45 Uhr bis 11:55 Uhr →10 Minuten

2 Teilnehmer

- Das Team besteht aus 15 Mitgliedern.
- Anwesende Mitglieder
  - 1 Entwickler aus München, männlich
  - 1 Entwickler + Scrum-Master aus Hamburg, beide männlich

3 Räumlichkeiten innerhalb der Standorte

Das Daily Scrum findet in einem regulären Büro statt. Die Mitglieder versammeln sich entsprechend am jeweiligen Standort in einem geeigneten Büro.

4 Eingesetzte Medien

Monitor + Maus mit Sicht auf das JIRA Taskboard. Telefon + Hörer + Lautsprecher. Telefonhörer als Speech-Token.

5 Ablauf

1. Eröffnung des Daily Scrum aus Hamburg vom Scrum-Master.
2. Derjenige, der die Runde eröffnet hat, startet.

3. Generell wird sich an die drei Fragen gehalten:

> 1. Was habe ich seit dem letzen Daily Scrum erreicht?
> 2. Was plane ich bis zum nächsten Daily Scrum?
> 3. Welche Hindernisse gibt es?
>
> (Wirdemann, 2011, S. 158)

5.1 Fazit

Der Ablauf orientiert sich stark am Standardablauf eines Daily Scrum und zeigt keine direkten Abweichungen.

6 Tickets

Es wurden insgesamt hörbar direkt vier Ticketkürzel angesprochen. Davon wurden drei aus Hamburg und eins aus München genannt. Der kürzeste Zeitabstand zwischen zwei genannten Ticketkürzeln bzw. einem anderen Thema lag bei unter fünf Sekunden. Dies kam insgesamt einmal vor. Der Wechsel bei den restlichen Ticketkürzeln lag bei größer gleich fünf Sekunden.

7 Besonderheiten

- Ein Ticket wurde aus Hamburg als letzte und aktuelle Aufgabe genannt, daraufhin wurde mittels Maus versucht, das Ticket auf dem offenen JIRA-Taskboard zu finden. Als Erstes versuchte derjenige, welcher am nächsten zur Maus war, das Ticket zu suchen und wurde vom *Speech Token Owner* zum Scrollen aufgefordert.
- Generell schauten beide Teilnehmer in Hamburg während des Meetings kontinuierlich auf das JIRA-Board.

8 Sonstiges

- Das eigentlich geplante Gruppeninterview und die anschließende Beantwortung des Fragebogens wurde mangels anwesenden Teammitgliedern auf den 28.11.2013 verschoben.
- Die Zeit von 10 Minuten wurde nicht ausschließlich für eine kurze Berichtserstattung genutzt und es fand ein *Story Telling* sowie *Problem Solving* statt.
- Es wurden auch Aufgaben genannt, die nicht als Ticket erfasst wurden.

9 Ergebnis

Auch in dieser Beobachtung werden wieder deutlich, dass eine direkte Fokussierung eines Tickets nicht immer möglich war. Ebenfalls stellt die Interaktion mit der Maus zur Erfassung bzw. Markierung eines Tickets ein Problem dar. Je nach Teamgröße muss die Maus für jeden schnell greifbar sein und somit wird oft Jemand, der in der Nähe der Maus steht zum agieren verleitet. Auch wurde diesmal besonders durch das Stoppen der Berichtserstattung für die Erfassung des Tickets auf dem JIRA-Board deutlich, dass ein Focusverlust stattfand.

9.1 Lösungsvorschläge

Eine mögliche Lösung dieser Interaktion könnte ein Touchscreen sein. So könnte durch eine schnelle und direkte Interaktion des Berichtenden der Fokus auf das erwähnte Ticket gelenkt werden. Im Gegensatz zu dieser und auch der eigenen Interaktion mit der Maus steht der Speech Token, welcher aktuell durch das kabelgebundene Telefon vorliegt. Hier könnte eine andere Wahl des Tokens und des Mikrofonmediums Abhilfe schaffen.

Literatur

[Wirdemann 2011] Wirdemann, Ralf: *Scrum mit User Stories*. München [u. a.] : Hanser, 2011. – ID: 846279372. – ISBN 9783446426603 3446426604

Gedächtnisprotokoll des 3. Beobachtungstermins

Daily Scrums - Standort Hamburg

Alexander Kaack
alexander.kaack@haw-hamburg.de

28. November 2013

1 Dauer

- 11:45 Uhr bis 11:58 Uhr →13 Minuten

2 Teilnehmer

- Das Team besteht aus 15 Mitgliedern.
- Anwesende Mitglieder
  - 2 Entwickler aus München, männlich
  - 3 Entwickler + Product Owner aus Hamburg, alle männlich

3 Räumlichkeiten innerhalb der Standorte

Das Daily Scrum findet in einem regulären Büro statt. Die Mitglieder versammeln sich entsprechend am jeweiligen Standort in einem geeigneten Büro.

4 Eingesetzte Medien

Monitor + Maus mit Sicht auf das JIRA Taskboard. Telefon + Hörer + Lautsprecher. Telefonhörer als Speech-Token.

5 Ablauf

1. Eröffnung des Daily Scrum aus Hamburg vom Scrum-Master.
2. Derjenige, der die Runde eröffnet hat, startet.

3. Generell wird sich an die drei Fragen gehalten:

> 1. Was habe ich seit dem letzen Daily Scrum erreicht?
> 2. Was plane ich bis zum nächsten Daily Scrum?
> 3. Welche Hindernisse gibt es?
>
> (Wirdemann, 2011, S. 158)

5.1 Sprechzeiten

Sprecher	Dauer
1	62 Sekunden
2	98 Sekunden
3	41 Sekunden
4	55 Sekunden
5	147 Sekunden
6	185 Sekunden

5.2 Fazit

Der Ablauf orientiert sich stark am Standardablauf eines Daily Scrum und zeigt keine direkten Abweichungen. Allerdings wird die Frage 3 "Welche Hindernisse gibt es?"gerne ausgelassen und nicht erwähnt.

6 Tickets

Es wurden insgesamt hörbar direkt 16 Ticketkürzel angesprochen. Davon wurden vier aus Hamburg und 12 aus München genannt. Der kürzeste Zeitabstand zwischen zwei genannten Ticketkürzeln bzw. einem anderen Thema lag bei unter fünf Sekunden. Dies kam insgesamt siebenmal vor. Der Wechsel bei den restlichen Ticketkürzeln lag bei größer gleich fünf Sekunden.

Anzahl	Dauer
6	< 5 Sekunden
9	> 5 Sekunden

7 Besonderheiten

- Es gab eine positive Rückmeldung, dass man sich aktuell innerhalb der geplanten *Velocity* befindet.
- Besonders nach der schnellen Hintereinaderstellung der Tickets wurde Richtung JIRA-Taskboard geschaut.
- Die größte Anzahl der genannten Tickets und auch schnell hintereinanderstellen von Tickets kam aus München.

- Nach dem Beenden des Meetings wurde kurz ohne die Teilnehmer aus München ein weiterer Punkt angesprochen.

8 Sonstiges

- Das eigentlich geplante Gruppeninterview und die anschließende Beantwortung des Fragebogens wurde mangels anwesenden Teammitgliedern auf den 28.11.2013 verschoben.
- Die Zeit von 13 Minuten wurde nicht ausschließlich für eine kurze Berichtserstattung genutzt und es fand ein *Story Telling* statt.
- Es wurden auch Aufgaben genannt, die nicht als Ticket erfasst wurden.
- Das Gruppeninterview findet nach der Beobachtung in Hamburg statt. Ein Fragebogen wird in Hamburg und München ausgehändigt.

9 Ergebnis

Die Ergebnisse aus dem Gedächtnisprotokoll vom 27.11.2013 wurden wieder bestätigt. Ein Großteil des *Story Telling* resultierte aus Aufgaben, welche nicht als Ticket erfasst wurden. Besonders durch die große Anzahl der hintereinander gestellten Tickets aus München, teilweise mit einem Zeitabstand unter fünf Sekunden, war am Standort Hamburg eine Orientierungslosigkeit anzumerken.

Literatur

[Wirdemann 2011] Wirdemann, Ralf: *Scrum mit User Stories*. München [u. a.] : Hanser, 2011. – ID: 846279372. – ISBN 9783446426603 3446426604

Protokoll des Gruppeninterviews

Daily Scrums - Standort Hamburg

Alexander Kaack
alexander.kaack@haw-hamburg.de

28. November 2013

1 Dauer

- 11:58 Uhr bis 12:02 Uhr →4 Minuten

2 Teilnehmer

- Das Team besteht aus 15 Mitgliedern.
- Anwesende Mitglieder
  - 3 Entwickler + Product Owner aus Hamburg, alle männlich

3 Räumlichkeiten der Befragung

Das gleiche Büro, indem das Daily Scrum stattgefunden hat.

4 Besonderheiten zu Befragung

Das Gespräch wird aufgezeichnet.

5 Frage 1: Seht Ihr Störungsfaktoren im Daily Scrum?

1. Antwort Nö.
2. Antwort Sprachqualität, die Kollegen auf der anderen Seite sieht man nicht.
3. Antwort Equipment ...
4. Antwort Kein Platz, Positionierung im Raum um einen kleinen Monitor und dann ein erwähntes Ticket zu erfassen. Scrollen notwendig.
5. Antwort Nicht die richtige Ausrüstung.

6 Frage 2: Was könnte zusätzlich zur Effizienz des Meetings beitragen?

1. Antwort Besseres Equipment.

2. Antwort Größeren Monitor, sodass sich alle im Halbkreis ran stellen können. Integriertes Mikrophone zum freisprechen.

3. Antwort Monitor mit Sprach, Video und Board. Aktuell sehen nur diejenigen, die direkt am Monitor stehen, alle Tickets.

4. Antwort Am besten einen Monitor, an dem man nicht scrollen muss.

5. Antwort Größe eines Standard-Whiteboards.

7 Zusätzliche Bemerkungen:

1. Antwort Nicht gestört werden, Tür zu. Keine Leute raus-ziehen.

2. Antwort Schwierigkeiten sich am Standardablauf zu orientieren. "Was habe ich bisher gemacht?", "Was werde ich tun?", "Was hält mich davon ab?"+ gleichzeitig auf die Tickets zu verweisen. Man kommt dadurch ins Plaudern.

Ergebnisse aus dem Fragebogen

Daily Scrums - Standort Hamburg

Alexander Kaack
alexander.kaack@haw-hamburg.de

28. November 2013

1 Dauer

- maximal 5 Minuten

2 Teilnehmer

- Das Team besteht aus 15 Mitgliedern.
- Teilnehmer des Fragebogens
  - 5 Entwickler + Product Owner aus Hamburg + Scrum Master, alle männlich

3 Räumlichkeiten der Befragung

Der Fragebogen wurde in Hamburg lokal im gleichen Raum ausgefüllt, wo auch das Daily Scrum durchgeführt wurde. Die Entwickler aus München wie auch der Scrum Master in Hamburg haben die Fragen digital beantwortet.

4 Besonderheiten zu Befragung

Der Fragebogen wird von insgesamt sieben Teilnehmern ausgefüllt. Drei Entwickler und der Product Owner beantworten die Fragen nach dem Gruppeninterview. Der Scrum Master, welcher nicht am Daily Scrum teilgenommen hat, sowie die Entwickler in München füllen die Fragen ohne das vorher geführte Gruppeninterview aus.

5 Rückmeldungen

Folgende Fragen wurden gesellt:

Fragenummer	Frage	Typ
1	Deine Vorbereitungszeit zum Daily Scrum beträgt?	Offene Frage
2	Wenn ein Ticket-Kürzel genannt wird, weißt Du sofort worum es geht.	Skalierte Antworten
3	Die eingesetzten Medien: Monitor + Maus und Telefon mit Lautsprecher reichen völlig aus.	Skalierte Antworten
4	Das Daily Scrum der verstreuten Teams wird durch die eingesetzten Medien sinnvoll unterstützt.	Skalierte Antworten
5	Der aktuelle Anlauf und die Struktur des Daily Scrums ist optimal.	Skalierte Antworten

Antworten auf Frage 1:

1. Antwort höchstens 5 min, meist gar keine.
2. Antwort 5 min
3. Antwort weniger als 5 min (meistens)
4. Antwort 0 min
5. Antwort 5 min
6. Antwort 2-3 min
7. Antwort 2 min

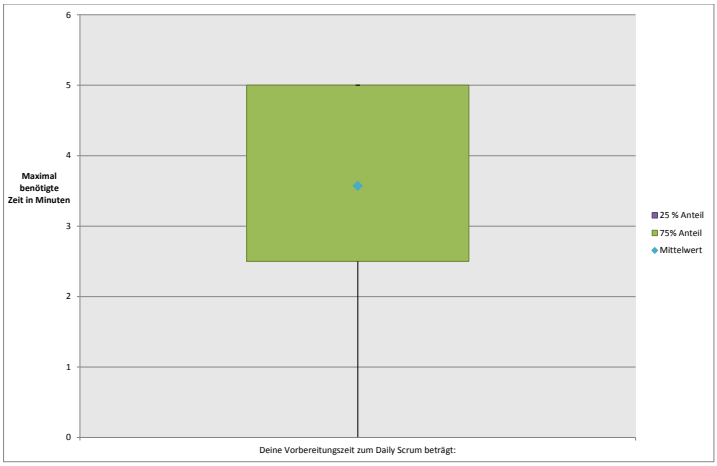

Abbildung 1: Maximale Vorbereitungszeit

Wie in der Abbildung 1 zu erkennen ist, ist die maximale Vorbereitungszeit auf 5 Minuten begrenzt.

3

5.1 Grafische Darstellung der Ergebnisse

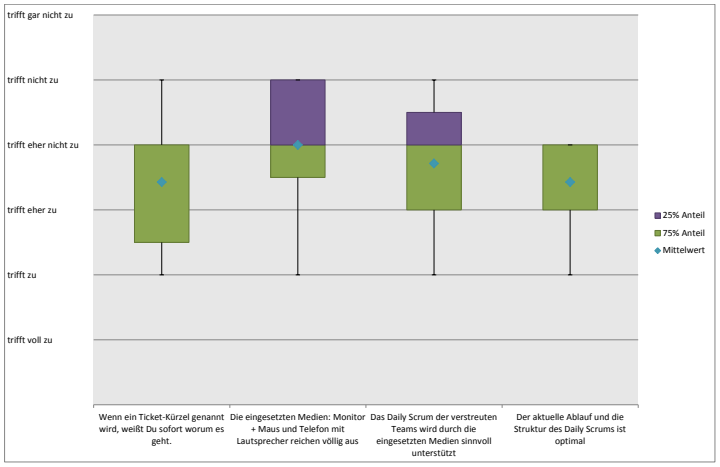

Abbildung 2: Rückmeldung gesamt

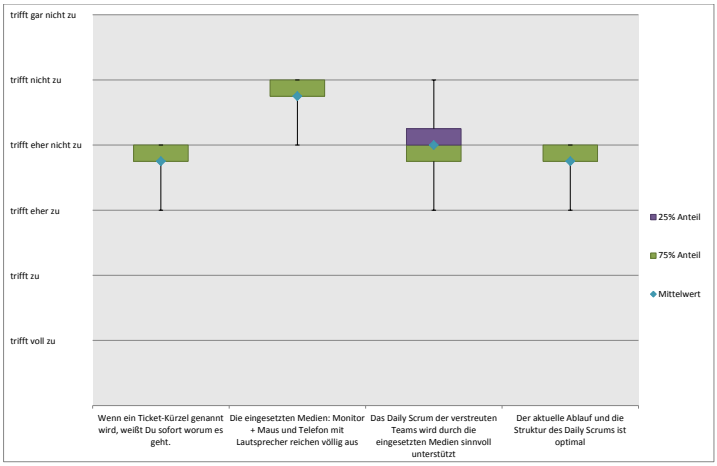

Abbildung 3: Rückmeldung ohne Gruppeninterview

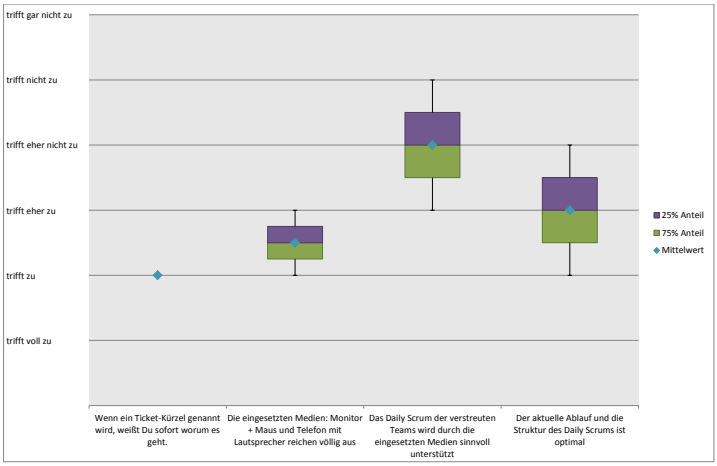

Abbildung 4: Rückmeldung mit Gruppeninterview

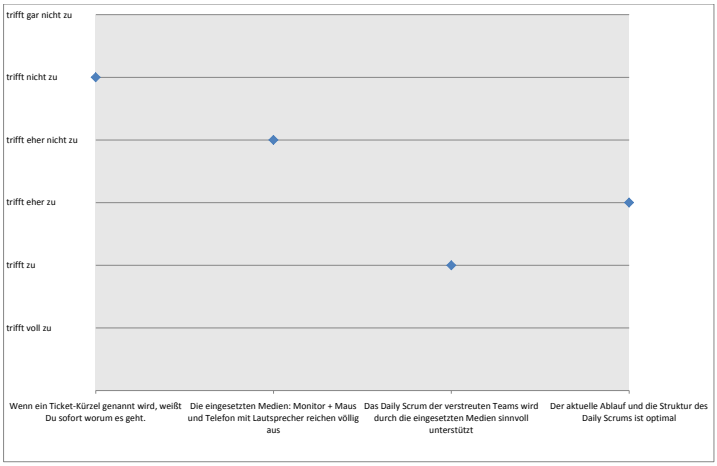

Abbildung 5: Rückmeldung ScrumMaster

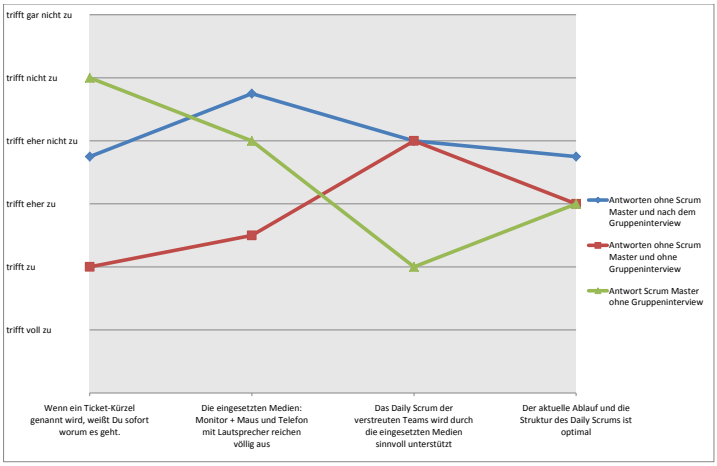

Abbildung 6: Vergleich der Mittelwerte

5.2 Tabellarische Darstellung der Ergebnisse

Alle Rückmeldungen				
Statistic	**Frage 2**	**Frage 3**	**Frage 4**	**Frage 5**
Median	4	4	4	4
25% Anteil	2,5	3,5	3	3
75% Anteil	4	5	4,5	4
Mittelwert	3,42857143	4	3,71428571	3,42857143
Min	2	2	2	2
Max	5	5	5	4

Antworten nach dem Gruppeninterview				
Statistic	**Frage 2**	**Frage 3**	**Frage 4**	**Frage 5**
Median	4	5	4	4
25% Anteil	3,75	4,75	3,75	3,75
75% Anteil	4	5	4,25	4
Mittelwert	3,75	4,75	4	3,75
Min	3	4	3	3
Max	4	5	5	4

Antworten ohne Gruppeninterview				
Statistic	**Frage 2**	**Frage 3**	**Frage 4**	**Frage 5**
Median	2	3	3	3
25% Anteil	2	2,5	2,5	2,5
75% Anteil	3,5	3,5	4	3,5
Mittelwert	3	3	3,33333333	3
Min	2	2	2	2
Max	5	4	5	4

Abbildung 7: Rückmeldungen Tabelle

A.2. Systemtests

Testnummer	1
Name	Boardauswahl.
Beschreibung	Es wird ein Board aus JIRA Agile über die Administrationsoberfläche gewählt und eine korrekte Darstellung auf allen Clients geprüft.
Fokus	Anzeigen des Boards.
Vorbedingung	Alle Clients müssen bereits die Seite des Interaktionsbereiches offen haben.

Tabelle A.1.: Systemtests - 1. Testfall

Testnummer	2
Name	Setzen von farblichen Markierungen.
Beschreibung	Ein Client markiert via Mausklick ein Ticket. Das Ticket sollte eine rote Umrandung bei allen Clients besitzen. Die Ausnahme ist eine schwarze Umrandung beim Client, welcher die Markierung gesetzt hat.
Fokus	sychrone Kennzeichnung des Tickets.
Vorbedingung	Alle Clients müssen bereits die Seite des Interaktionsbereiches offen haben und der Testfall 1. sollte erfolgreich abgeschlossen sein.

Tabelle A.2.: Systemtests - 2. Testfall

Testnummer	3
Name	Fokus des Tickets.
Beschreibung	Ein Client markiert via Mausklick ein Ticket. Der Fokus sollte nun bei allen Clients auf diesem Ticket liegen
Fokus	sychrone Fokussierung.
Vorbedingung	Alle Clients müssen bereits die Seite des Interaktionsbereiches offen haben und der Testfall 1. sollte erfolgreich abgeschlossen sein.

Tabelle A.3.: Systemtests - 3. Testfall

Testnummer	4
Name	Sortieren nach Bearbeiter.
Beschreibung	Ein User gibt Buchstabe für Buchstabe den Namen eines Bearbeiters an. Nach jedem Zeichen sollte die Auswahl automatisch angepasst werden.
Fokus	sychrone Darstellung.
Vorbedingung	Offene Session und ausgewähltes Board.

Tabelle A.4.: Systemtests - 4. Testfall

Testnummer	5
Name	Sortieren nach Ticket-Kürzel.
Beschreibung	Ein User gibt Buchstabe für Buchstabe das Kürzel eines Tickets an. Nach jedem Zeichen sollte die Auswahl automatisch angepasst werden.
Fokus	sychrone Darstellung.
Vorbedingung	Offene Session und ausgewähltes Board.

Tabelle A.5.: Systemtests - 5. Testfall

Testnummer	6
Name	Prüfen der Korrektheit der Boardinformationen.
Beschreibung	Jedes Ticket sollte im unmarkierten und markierten Zustand geprüft werden, ob die dargestellten Informationen den Informationen im JIRA Agile-System entsprechen.
Fokus	Korrektheit der Informationen.
Vorbedingung	Offene Session und ausgewähltes Board sowie Zugang zum JIRA bzw. JIRA Agile- System.

Tabelle A.6.: Systemtests - 6. Testfall

Testnummer	7
Name	Abmelden und Anmelden.
Beschreibung	Ein User, welcher selbst keine Markierung gesetzt hat, sollte seine Session schließen (Browser schließen) und sich anschließend über den Browser neu anmelden. Das Ticket sollte nun weiterhin bzw. sollte nun markiert sein.
Fokus	sychrone Darstellung.
Vorbedingung	Offene Session und ausgewähltes Board und markiertes Ticket.

Tabelle A.7.: Systemtests - 7. Testfall

Testnummer	8
Name	Gleichzeitiges Markieren.
Beschreibung	Mehrere Clients sollte versuchen verschiedene Tickets quasi gleichzeitig zu markieren. Das Ergebnis sollte nur ein gleicher Zustand sein und nach dem markieren des ersten Tickets alle anderen Clients in ihrer Eingabe blockieren.
Fokus	sychrone Darstellung.
Vorbedingung	Offene Session und ausgewähltes Board.

Tabelle A.8.: Systemtests - 8. Testfall

Testnummer	9
Name	Abmelden des Markierenden.
Beschreibung	Ein Client sollte ein Ticket markieren und sich abschließend abmelden. Der View sollte nun bei allen Clients blockiert sein, allerdings nach dem definierten Zeitlimit wieder freigegeben werden.
Fokus	Prüfen des Zeitlimits zur Freigabe des Views.
Vorbedingung	Offene Session und ausgewähltes Board.

Tabelle A.9.: Systemtests - 9. Testfall

Testnummer	10
Name	Ausloggen einer Authentifizierung.
Beschreibung	Im Administrationsbereich soll ein eine Authentifizierung abgemeldet werden. Diese sollte danach nicht mehr verfügbar sein, solange die Authentifizierungsinformationen erneut eingeben werden.
Fokus	Authentifizierung.
Vorbedingung	Offene Session und angemeldete Authentifizierung.

Tabelle A.10.: Systemtests - 10. Testfall

Testnummer	11
Name	Zurücksetzen der Session.
Beschreibung	Im Administrationsbereich soll die Session zurückgesetzt werden und somit ein leeres Board auf allen Clients produzieren. Keinerlei Informationen zu Tickets und Administration der Anwendungen sollte noch vorhanden sein.
Fokus	synchrone Darstellung.
Vorbedingung	Offene Session und ausgewähltes Board.

Tabelle A.11.: Systemtests - 11. Testfall

Testnummer	12
Name	Prüfen der definierten maximalen Anzahl an Sessions.
Beschreibung	Es sollen sich zusätzlich Clients anmelden um das definierte Maximum an Sessions zu prüfen.
Fokus	Definierte Maximum an Sessions.
Vorbedingung	Offene Session.

Tabelle A.12.: Systemtests - 12. Testfall

Testnummer	13
Name	Dauerhafte Lauffähigkeit über 1.5 Wochen.
Beschreibung	Über die Laufzeit von 1.5 Wochen sollen regelmäßig Interaktionen und an und abmelden von Sessions vorgenommen werden.
Fokus	Lauffähigkeit.
Vorbedingung	Offene Session und ausgewähltes Board.

Tabelle A.13.: Systemtests - 13. Testfall

Testnummer	14
Name	Änderung - Ticketstatus.
Beschreibung	Das Ticket soll in einen neune Ticketstatus gesetzt werden.
Fokus	Statuswechsel.
Vorbedingung	Offene Session und ausgewähltes Board.

Tabelle A.14.: Systemtests - 14. Testfall

Testnummer	15
Name	Änderung - Ticketstatus - mehrere gleichzeitige Clients
Beschreibung	Das Ticket soll in einen neune Ticketstatus gesetzt werden, allerdings versuchen gleichzeitig weitere User diese Aktion mittels eines eigenen Statuswechsel zu behindern.
Fokus	Statuswechsel.
Vorbedingung	Offene Session und ausgewähltes Board.

Tabelle A.15.: Systemtests - 15. Testfall

Testnummer	16
Name	Änderung - ungültiger Ticketstatus.
Beschreibung	Es soll versucht werden, dass Ticket in einen ungültigen Ticketstatus zu setzen. Dies soll nicht akzeptiert werden.
Fokus	Statuswechsel.
Vorbedingung	Offene Session und ausgewähltes Board.

Tabelle A.16.: Systemtests - 16. Testfall

A.3. Erfassung der Evaluierungsergebnisse

Gedächtnisprotokoll zur Evaluierung des Daily Scrums *Datum: 18.03.2014*

1. Rahmeninformationen

Anwesende Rollen:
Entwickler_____
Hamburg *München*
Scrum-Master:
Hamburg ☐ München ☐
Product Owner:
Hamburg ☐ München ☐
Sonstige anwesende Personen:
Teilnehmer Hamburg:_____

Teilnehmer München:_____

Dauer(Von/Bis):_____

Räumlichkeiten:
Hamburg:_____

München:_____

2. Eingesetzte Medien:

_____

3. Tickets

Anzahl genannter Tickets:_____

Wechsel von Ticket kleiner 5 Sekunden:_____

Wechsel von Ticket größer gleich 5 Sekunden:_____

4. Besonderheiten:

_____

_____

_____

_____

1

Gruppeninterview zur Evaluierung des Daily Scrums *Datum: 18.03.2014*

1. Rahmeninformationen

Anwesende Rollen:
Entwickler_____
Hamburg *München*
Scrum-Master:
Hamburg ☐ München ☐
Product Owner:
Hamburg ☐ München ☐
Sonstige anwesende Personen:
Teilnehmer Hamburg:_____

Teilnehmer München:_____

Dauer(Von/Bis):_____

2. Was hat Euch an der Anwendung gefallen? (Mehrwert im Meeting)

_____

_____

3. Was hat Euch bei der Anwendung gefehlt bzw. gestört? (Features, Bugs)

_____

_____

4. Was hat Euch davon abgehalten die Software zu nutzen? (Team, Bedienungszeit, Ausstattung)

_____

_____

5. Zusätzliche Bemerkungen:

_____

Fragebogen zur Evaluierung der Software *Datum: 18.03.2014*

1. **Die Anwendung trägt zum Verständnis des genannten Ticket-Kürzels bei.**
   Bitte beantworte die Frage auf einer Skala von 1 bis 6.

trifft voll zu	trifft zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	trifft gar nicht zu
☐	☐	☐	☐	☐	☐
1	2	3	4	5	6

2. **Die Software gibt mir innerhalb des Meetings einen besseren Gesamtüberblick über den aktuellen Sprint.**
   Bitte beantworte die Frage auf einer Skala von 1 bis 6.

trifft voll zu	trifft zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	trifft gar nicht zu
☐	☐	☐	☐	☐	☐
1	2	3	4	5	6

3. **Die zur Verfügung stehende Ausstattung (Räumlichkeiten, Monitor, Tastatur, Maus und Telefon mit Lautsprecher) stellt kein Hindernis bei der Verwendung der Software dar.**
   Bitte beantworte die Frage auf einer Skala von 1 bis 6.

trifft voll zu	trifft zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	trifft gar nicht zu
☐	☐	☐	☐	☐	☐
1	2	3	4	5	6

4. **Das Daily Scrum der verstreuten Teams wird durch die eingesetzte Software sinnvoll unterstützt.**
   Bitte beantworte die Frage auf einer Skala von 1 bis 6.

trifft voll zu	trifft zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	trifft gar nicht zu
☐	☐	☐	☐	☐	☐
1	2	3	4	5	6

Fragebogen zur Evaluierung der Software *Datum: 18.03.2014*

5. Der aktuelle Ablauf und die Struktur des Daily Scrums ist optimal.
Bitte beantworte die Frage auf einer Skala von 1 bis 6.

trifft voll zu	trifft zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	trifft gar nicht zu
☐	☐	☐	☐	☐	☐
1	2	3	4	5	6

6. Die Funktionsweise der Software ist intuitiv.
Bitte beantworte die Frage auf einer Skala von 1 bis 6.

trifft voll zu	trifft zu	trifft eher zu	trifft eher nicht zu	trifft nicht zu	trifft gar nicht zu
☐	☐	☐	☐	☐	☐
1	2	3	4	5	6

Gedächtnisprotokoll des Beobachtungstermins zur Evaluierung der Software

Daily Scrums - Standort Hamburg

Alexander Kaack
alexander.kaack@haw-hamburg.de

26. Februar 2014

1 Dauer

- 11:45 Uhr bis 12:00 Uhr →15 Minuten

2 Teilnehmer

- Das Team besteht aus 15 Mitgliedern.
- Anwesende Mitglieder
  - 1 Entwickler aus München, männlich
  - 2 Entwickler + Product Owner aus Hamburg, alle männlich

3 Räumlichkeiten innerhalb der Standorte

Das Daily Scrum findet in einem regulären Büro statt. Die Mitglieder versammeln sich entsprechend am jeweiligen Standort in einem geeigneten Büro. Hierbei sind nur Teilnehmer des Meetings anwesend.

4 Eingesetzte Medien

Monitor + Maus mit Sicht auf die Anwendung. Telefon + Hörer + Lautsprecher. Telefonhörer als Speech-Token.

5 Ablauf

1. Eröffnung des Daily Scrum aus Hamburg vom Scrum-Master.
2. Derjenige, der die Runde eröffnet hat, startet.
3. Generell wird sich an die drei Fragen gehalten:

> 1. Was habe ich seit dem letzen Daily Scrum erreicht?
> 2. Was plane ich bis zum nächsten Daily Scrum?
> 3. Welche Hindernisse gibt es?
>
> (Wirdemann, 2011, S. 158)

5.1 Fazit

Der Ablauf orientiert sich stark am Standardablauf eines Daily Scrum und zeigt keine direkten Abweichungen. Allerdings wird die Frage 3 "Welche Hindernisse gibt es?"gerne ausgelassen und nicht erwähnt.

6 Tickets

Es wurden insgesamt hörbar direkt zwei Ticketkürzel angesprochen. Davon wurden zwei aus Hamburg. Der Wechsel der genannten Ticketkürzels lag bei größer gleich fünf Sekunden.

Anzahl	Dauer
00	< 5 Sekunden
2	> 5 Sekunden

7 Besonderheiten

- Wenig Tickets wurden genannt
- Die Beteiligung am Meeting (Anzahl der Teilnehmer) war diesmal sehr gering.
- Es wurde sichtbar, dass eine Beschreibung einer Aufgabe durch einen Klick in der Anwendung mit einen Ticket assoziiert werden konnte.
- Die Größe des Displays war ein Problem, trotz der geringen Anzahl der Teilnehmer.
- Die Verwendung der Software ergab keine Fehler oder Probleme bei Verständnis zur Bedienung.

8 Sonstiges

- Es wurden auch Aufgaben genannt, die nicht als Ticket erfasst wurden. Diese Gespräche überwiegten stark.

9 Ergebnis

Die Ergebnisse aus dem Gedächtnisprotokoll vom 28.11.2013 wurden wieder bestätigt. Ein Großteil des *Story Telling* resultierte aus Aufgaben, welche nicht als Ticket erfasst wurden. Die Beteiligung am Meeting war sehr rar und trotzdem wurde sichtbar, dass die Größe des Monitors (19 Zoll) ein Problem war, was auch in Verbindung mit der Platzierung des Geräts selber und der Positionierung der Teilnehmer liegt.

Literatur

[Wirdemann 2011] Wirdemann, Ralf: *Scrum mit User Stories*. München [u. a.] : Hanser, 2011. – ID: 846279372. – ISBN 9783446426603 3446426604

Gedächtnisprotokoll des Beobachtungstermins zur Evaluierung der Software

Daily Scrums - Standort Hamburg

Alexander Kaack
alexander.kaack@haw-hamburg.de

18. März 2014

1 Dauer

- 11:47 Uhr bis 11:56 Uhr →9 Minuten

2 Teilnehmer

- Das Team besteht aus 15 Mitgliedern.
- Anwesende Mitglieder
  - 2 Entwickler aus München, davon einer im Home-Office männlich
  - 3 Entwickler + Product Owner + Scrum Master aus Hamburg, alle männlich

3 Räumlichkeiten innerhalb der Standorte

Das Daily Scrum findet in einem regulären Büro statt. Die Mitglieder versammeln sich entsprechend am jeweiligen Standort in einem geeigneten Büro.

4 Eingesetzte Medien

Monitor + Maus mit Sicht auf die Anwendung. Telefon + Hörer + Lautsprecher. Telefonhörer als Speech-Token.

5 Ablauf

1. Eröffnung des Daily Scrum aus Hamburg vom Scrum-Master.
2. Derjenige, der die Runde eröffnet hat, startet.
3. Generell wird sich an die drei Fragen gehalten:

> 1. Was habe ich seit dem letzen Daily Scrum erreicht?
> 2. Was plane ich bis zum nächsten Daily Scrum?
> 3. Welche Hindernisse gibt es?
>
> (Wirdemann, 2011, S. 158)

5.1 Fazit

Der Ablauf orientiert sich stark am Standardablauf eines Daily Scrum und zeigt keine direkten Abweichungen. Allerdings wird die Frage 3 "Welche Hindernisse gibt es?"gerne ausgelassen und nicht erwähnt.

6 Tickets

Es wurden insgesamt hörbar direkt vier Ticketkürzel angesprochen. Davon wurden zwei aus Hamburg und zwei aus München genannt. Der kürzeste Zeitabstand zwischen zwei genannten Ticketkürzeln bzw. einem anderen Thema lag bei unter fünf Sekunden. Dies kam insgesamt einmal vor. Der Wechsel bei den restlichen Ticketkürzeln lag bei größer gleich fünf Sekunden.

Anzahl	Dauer
1	< 5 Sekunden
3	> 5 Sekunden

7 Besonderheiten

- Wenig Tickets wurden genannt
- Eine Entwickler aus München war im Home-Office dazugeschaltet
- Es wurde erstmals sichtbar, dass eine Beschreibung einer Aufgabe durch einen Klick in der Anwendung mit einen Ticket assoziiert werden konnte.
- Ein Ticket wurde aus dem Home-Office gemeldet.

8 Sonstiges

- Es wurden auch Aufgaben genannt, die nicht als Ticket erfasst wurden.
- Das Gruppeninterview findet nach der Beobachtung in Hamburg statt sowie ein Fragebogen wird in Hamburg und München ausgehändigt.

9 Ergebnis

Die Ergebnisse aus dem Gedächtnisprotokoll vom 28.11.2013 wurden wieder bestätigt. Ein Großteil des *Story Telling* resultierte aus Aufgaben, welche nicht als Ticket erfasst wurden. Die Einbindung eines Mitarbeits im Home-Office stellt kein Problem dar. Es war gut zu sehen, dass eine Aufgabe als Ticket erfasst war, allerdings erst durch den Klick in der Anwendung mit einem Ticket assoziiert werden konnte. Auch diesmal wurden wie in der vorherigen Beobachtung wenig Tickets genannt.

Literatur

[Wirdemann 2011] Wirdemann, Ralf: *Scrum mit User Stories*. München [u. a.] : Hanser, 2011. – ID: 846279372. – ISBN 9783446426603 3446426604

Protokoll des Gruppeninterviews

Evaluierung der Software - Standort Hamburg

Alexander Kaack
alexander.kaack@haw-hamburg.de

18. März 2014

1 Dauer

- 11:56 Uhr bis 12:02 Uhr →6 Minuten

2 Teilnehmer

- Das Team besteht aus 15 Mitgliedern.
- Anwesende Mitglieder
  - 3 Entwickler + Product Owner + ScrumMaster aus Hamburg, alle männlich

3 Räumlichkeiten der Befragung

Das gleiche Büro, indem das Daily Scrum stattgefunden hat.

4 Besonderheiten zu Befragung

Das Gespräch wird aufgezeichnet.

5 Frage 1: Was hat Euch an der Anwendung gefallen?

1. Antwort Team findet eine Verwendung.
2. Antwort Strukturiert sichtbar woran man gerade arbeitet.
3. Antwort In Gedanken rufen, was man gerade macht.
4. Antwort Welche Tickets man noch offen hat.
5. Antwort Wenn man nicht an Tickets arbeitet.

6. Antwort Es fällt auf, wenn man innerhalb des Meetings wenig erfasste Aufgaben hat.

7. Antwort Fokussierter.

8. Antwort Eingrenzen auf Namen.

9. Antwort Flüssig.

10. Antwort Gesamtüberblick.

11. Antwort Überblick über eigene Tasks.

12. Antwort Offene Tickets.

13. Antwort Es wird bemerkt, wenn keine Tickets als Aufgabe bearbeitet werden.

6 Frage 2: Was hat Euch bei der Anwendung gefehlt bzw. gestört?

1. Antwort Ticketwechsel via Tastatur pro Statusfeld.

2. Antwort Das Design könnte noch verbessert werden. An den JIRA Standard annähern.

3. Antwort Aktualisierung verbessern.

7 Frage 3: Was hat Euch davon abgehalten die Software zu nutzen?

1. Antwort Größe Monitor mit Touch wäre hilfreich.

2. Antwort Maus und Tastatur.

3. Antwort Räumlichkeiten, zu klein und jeder muss zur Tastatur/Maus hingehen.

4. Antwort Nicht dran gedacht das Board zu öffnen.

8 Zusätzliche Bemerkungen:

1. Antwort Es trifft 'nur' zu und nicht voll zu, dass die Anwendung zum Verständnis des genannten Ticket-Kürzels beiträgt, weil man parallel zuhören und lesen muss, was sich nicht vermeiden lässt.

2. Antwort Besonders durch die Filtermöglichkeit wird ein besserer Gesamtüberblick über den aktuellen Sprint durch die Software bereitgestellt.

3. Antwort Größerer Monitor und schnurlose Eingabe wäre schön um die Anwendung im vollen Umfang zu verwenden.

Ergebnisse aus dem Fragebogen

Daily Scrums - Standort Hamburg

Alexander Kaack
alexander.kaack@haw-hamburg.de

18. März 2014

1 Dauer

- maximal 5 Minuten

2 Teilnehmer

- Das Team besteht aus 15 Mitgliedern.
- Teilnehmer des Fragebogens
  - 3 Entwickler + Scrum Master aus Hamburg, alle männlich
  - 1 Entwickler aus München, männlich

3 Räumlichkeiten der Befragung

Der Fragebogen wurde in Hamburg lokal im gleichen Raum ausgefüllt, wo auch das Daily Scrum durchgeführt wurde. Die Entwickler aus München wie auch der Scrum Master in Hamburg haben die Fragen digital beantwortet.

4 Besonderheiten zu Befragung

Der Fragebogen wird von insgesamt 5 Teilnehmern ausgefüllt. 3 Entwickler und der Scrum Master beantworteten die Fragen nach dem Gruppeninterview. Ein Entwickler in München hat ohne das vorher durchgeführte Gruppeninterview am Fragebogen teilgenommen.

5 Rückmeldungen

5.1 Grafische Darstellung der Ergebnisse

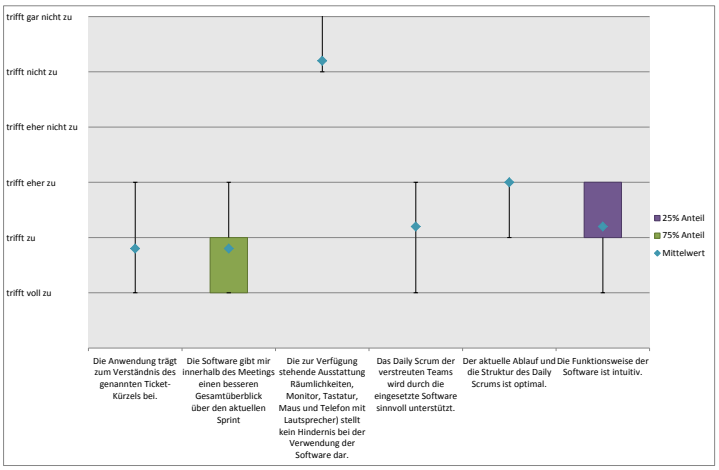

Abbildung 1: Rückmeldung gesamt

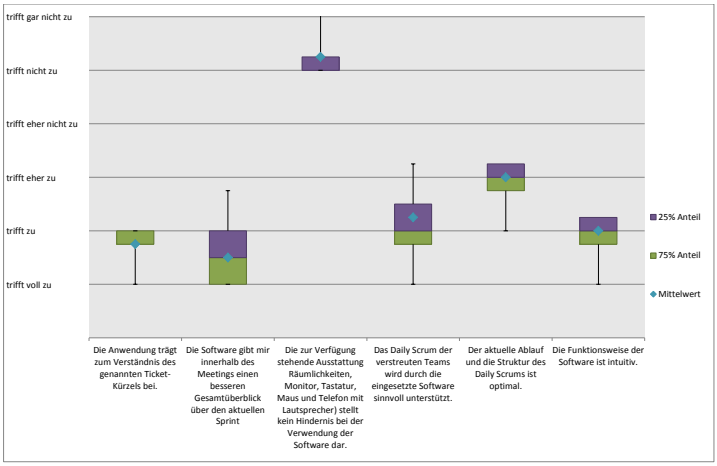

Abbildung 2: Rückmeldung ohne Gruppeninterview

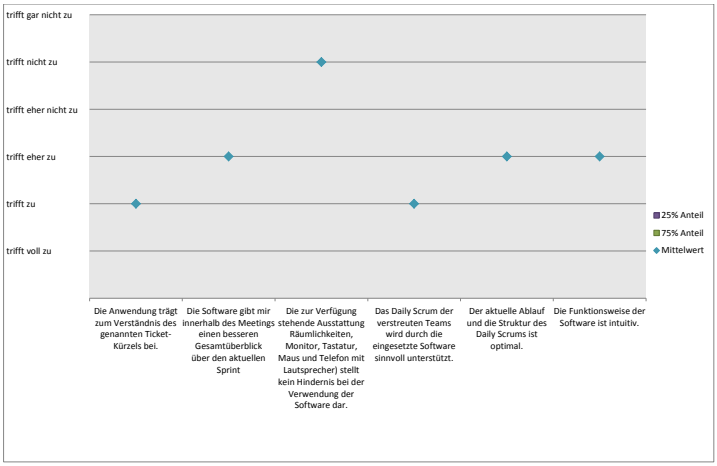

Abbildung 3: Rückmeldung mit Gruppeninterview

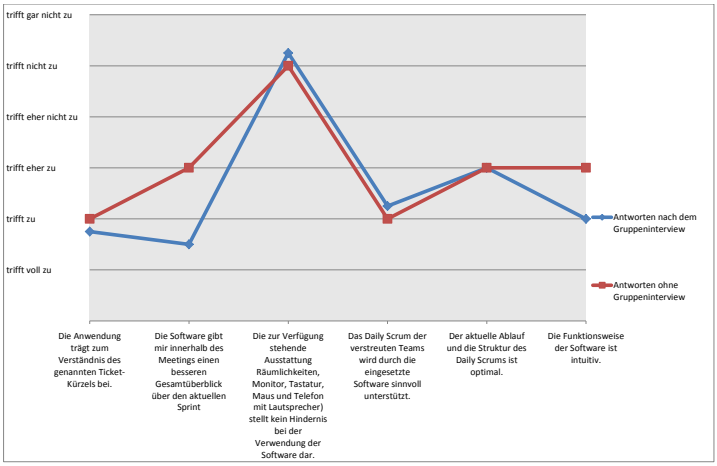

Abbildung 4: Vergleich der Mittelwerte

5.2 Tabellarische Darstellung der Ergebnisse

Alle Rückmeldungen						
Statistic	**Frage 1**	**Frage 2**	**Frage 3**	**Frage 4**	**Frage 5**	**Frage 6**
Median	2	2	5	2	3	2
25% Anteil	2	1	5	2	3	2
75% Anteil	2	2	5	2	3	3
Mittelwert	1,8	1,8	5,2	2,2	3	2,2
Min	1	1	5	1	2	1
Max	2	3	6	4	4	3

Antworten nach dem Gruppeninterview						
Statistic	**Frage 1**	**Frage 2**	**Frage 3**	**Frage 4**	**Frage 5**	**Frage 6**
Median	2	1,5	5	2	3	2
25% Anteil	1,75	1	5	1,75	2,75	1,75
75% Anteil	2	2	5,25	2,5	3,25	2,25
Mittelwert	1,75	1,5	5,25	2,25	3	2
Min	1	1	5	1	2	1
Max	2	2	6	4	4	3

Antworten ohne Gruppeninterview						
Statistic	**Frage 1**	**Frage 2**	**Frage 3**	**Frage 4**	**Frage 5**	**Frage 6**
Median	2	3	5	2	3	3
25% Anteil	2	3	5	2	3	3
75% Anteil	2	3	5	2	3	3
Mittelwert	2	3	5	2	3	3
Min	2	3	5	2	3	3
Max	2	3	5	2	3	3

Abbildung 5: Rueckmeldungen Tabelle

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 16. Mai 2014 Alexander Kaack