Digitale Retrospektive mit Scrum
Woran hat es gelegen?

Erwin Lang

Hochschule für angewandte Wissenschaften Hamburg
Email: erwin.lang@haw-hamburg.de

I. Einführung

A. Motivation

Softwareentwicklung ist ein Feld mit zunehmend steigender Relevanz. Die entwickelten Systeme haben jedoch eine immer größer werdende Komplexität und erfordern daher ein geordnetes und dennoch flexibel gestaltetes Vorgehen. Aus diesem Grund werden heute eine Vielzahl von Softwareprojekten mit agilen Prozessen umgesetzt. Die Vorteile sind unter anderem eine höhere Qualität, Transparenz und Motivation der Entwickler. Laut einer Umfrage von VersionOne [1] sind im Jahr 2013 52% der Softwareprojekte in den befragten Firmen mit einem agilen Modell umgesetzt worden. Das am weitesten verbreitete dieser Methoden war mit großem Abstand Scrum mit 55%.

Doch die Anforderungen an ein Team durch Methoden wie Scrum sind trotz der oftmals positiven Ergebnisse nicht zu unterschätzen. So ist es häufig nicht möglich, dass alle Entwickler gemeinsam und zur gleichen Zeit arbeiten. Dies erschwert die persönliche Kommunikation welche zentral für agile Methoden ist. Diese Ausarbeitung beschäftigt sich damit, wie Scrum mit Hilfe von einer eigenen Software unterstützt werden kann um derartige Probleme zu lösen. Der klare Fokus liegt hierbei auf dem Aspekt der Retrospektive. Die Ansätze hierfür basieren auf den Forschungsgebieten *Enterprise 2.0* und *Computer Supported Collaborative Work (CSCW)*. Im folgenden werden zunächst die festgestellten Probleme erläutert. Anschließend werden bekannte Ansätze vorgestellt auf denen weiter aufgebaut wird. Ziel ist es den Grundstein für ein Projekt zu legen in dem die vorgestellten Lösungen umgesetzt und validiert werden sollen.

B. Einleitung in Scrum

Um das Verständnis für das Themengebiet dieser Ausarbeitung zu verbessern folgt nun eine kurze Einleitung zu dem Kern des Scrum-Frameworks. Scrum wurde im Jahre 1993 von Jeff Sutherland entwickelt. Es gibt ein simples Rahmenwerk aus Rollen, Meetings und Artefakten vor, welche im Entwicklungsprozess verwendet werden. Wie diese Teile des Prozesses konkret umgesetzt werden ist dabei nicht vorgegeben. Lediglich der Zweck der einzelnen Aspekte ist definiert. Im Laufe der Zeit haben sich jedoch einige *Best Practices* entwickelt welche über das grundlegende Scrum hinausgehen. Abbildung 1 zeigt den Ablauf im Scrumprozess.

Die Prinzipien auf denen der gesamte Prozess aufgebaut ist sind *Transparency, Inspection* und *Adaption*. Im Kern bedeutet dies, dass die Aufgabe und das Ziel klar ist, die Arbeit regelmäßig evaluiert wird auf sich ändernde Umstände reagiert

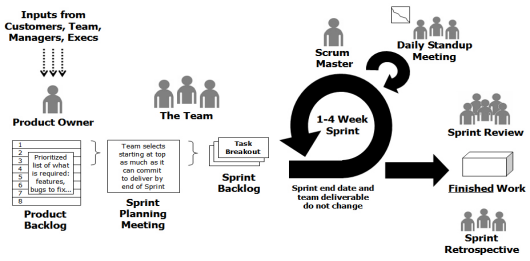

Abbildung 1. Ablauf von Scrum [2]

werden kann. Diese Prinzipien werden durch die folgenden Aspekte von Scrum gestützt [2] [3].

1) Rollen:

- **Product Owner:** Verantwortlicher für den Erfolg des Produkts.
- **Scrum Master:** Verantwortlicher für den Erfolg des Teams.
- **Developer:** Entwickler.

2) Meetings:

- **Daily:** Kurze Besprechung am Anfang des Tages um zu kommunizieren wo man gerade steht.
- **Sprint Planning:** Planung des nächsten Sprints von 2-4 Wochen. Basis der Iteration welche *Inspection* und *Adaption* fördert.
- **Sprint Review:** *Inspection* der Arbeit des vergangenen Sprints bezüglich des Produkts.
- **Sprint Retrospective:** *Inspection* der Arbeit des vergangenen Sprints bezüglich der Arbeitsweise. Fokus dieser Arbeit.

3) Artefakte:

- **Product Backlog:** Liste der gesamten bekannten Anforderungen im Projekt. Kann jederzeit erweitert werden um Flexibilität zu gewährleisten.
- **Sprint Backlog:** Liste der Umzusetzenden Anforderungen im aktuellen Spring
- **Increment:** Für sich funktionierendes Produkt am Ende eines jeden Sprints.

C. Problemstellung

Im folgenden werden die Probleme genauer erläutert welche durch eine digitale Unterstützung gelöst werden sollen. Ein entscheidender Punkt bei allen agilen Techniken ist ein Fokus auf persönliche und direkte Kommunikation [3]. In einer idealen Welt würde das gesamte Entwicklungsteam durchgehend gemeinsam im selben Büro sitzen. Dies ist in der Realität allerdings nicht immer umzusetzen. Flexible Arbeitszeiten, Verantwortung für mehrere Projekte und Homeoffice sind Alltag in vielen Softwarefirmen. Hinzu kommt noch das Outsourcing von ganzen Teilen der Entwicklung an andere Standorte, oftmals über Landesgrenzen hinweg. Typische Herausforderungen bei der verteilten Softwareentwicklung sind [4]

- **Strategisch:** *Best Practice* Methoden können nicht verwendet werden, da das Verhältnis zum Aufwand sie nicht rechtfertigt. Beispiele hierfür kann ein *Daily Scrum* sein welches über Zeitzonen hinweg durchgeführt werden muss.
- **Projekt- und Prozessmanagement:** Die Arbeit der verschiedenen Teams muss korrekt Synchronisiert werden. der Managementaufwand von 2 oder mehreren isolierten Teams ist naturgemäß größer als bei einem lokalen Team.
- **Kommunikation:** Die Möglichkeiten der Entwickler miteinander zu kommunizieren ist eingeschränkt. Es muss stärker auf asynchrone Kommunikation ausgewichen werden wie E-Mail. Synchrone Mittel wie Sofortnachrichten sind oft nicht für die Diskussion komplexer Themen geeignet und sind schwer für dritte Nachzuvollziehen. Die Meetings im Scrumprozess sind nicht leicht umzusetzen [5].
- **Kulturelle Aspekte:** Landeskultur und Unternehmenskultur haben einen Einfluss auf das Verständnis und die Arbeitsweise von Mitarbeitern. Dieses Verständnis zu vereinen wird schwieriger je mehr unterschiedliche Kulturen beteiligt sind und je distanzierter sie voneinander sind [6].
- **Technisch:** Unterschiedliche Dateiformate, Schnittstellen und Entwicklungswerkzeuge führen zu Schwierigkeiten bei der Integration eines Projektes.
- **Sicherheit:** Die verwendeten Kommunikationsmittel müssen sicherstellen, dass die Daten vertraulich behandelt werden und sicher sind.

Diese Punkte sind für alle verteilten Projekte relevant. Bezogen auf die Retrospektive ist es jedoch besonders wichtig darauf zu achten, dass vereinbarte *Best Practices* umgesetzt werden und die Kommunikation unterstützt wird. Kulturelle Unterscheide sind ebenfalls besonders interessant, da es in einigen Kulturen nicht unüblich ist Kritik als respektlos zu empfinden. Eine digitale Plattform kann helfen derartige Missverständnisse zu lösen.

Doch auch in nicht verteilten Projekten gibt es Potenzial zur Verbesserung. Nach einer Studie von [7] führen nur 62% aller Scrumprojekte eine Retrospektive nach jedem Sprint durch. 13% haben eine Retrospektive nach Durchführung des Projektes und 25% arbeiten völlig ohne. Dies zeigt ein Problem des Scrum-Frameworks auf. Da viele Aspekte von Scrum flexibel gestaltet sind kann es leicht passieren, dass auch essentielle Punkte in der Praxis entfernt werden. Da die Retrospektive auf den ersten Blick nicht unmittelbar mit dem Projekterfolg zusammenhängt passiert dies hier besonders oft. Die Umfrage zeigt auch, dass 35% der befragten keine Scrum Master Rolle definiert haben. Eine digitale Plattform kann hier helfen den Zweck der Retrospektive deutlicher zu machen und *Best Practices* zu fördern.

Hinzu kommt noch, dass auch wenn die Retrospektive durchgeführt wird es für die Entwickler nicht immer leicht ist potenzielle Probleme auch zu erkennen und wiederzugeben. Auch kann es dazu kommen, dass nicht alle Teammitglieder ihre Meinung aktiv wiedergeben und es zu einem Ungleichgewicht in der Beteiligung bei der Retrospektive gibt. Hier kann beispielsweise eine Visualisierung des Projektverlaufes bereits eine große Unterstützung darstellen und die beteiligten Personen ermutigen ihre Meinung zu äußern.

II. Schnittstellen zu anderen Projekten

Das Projekt zur Erstellung einer Software um den agilen Prozess zu unterstützen soll gemeinsam mit den Kommilitonen Leon Fausten und Alexander Kaack durchgeführt werden. Dabei soll eine gemeinsame Grundlage geschaffen und implementiert werden auf der die Arbeiten aufbauen können. Während sich diese Ausarbeitung vorwiegend mit der Verbesserung der Retrospektive beschäftigt liegt der Fokus bei Leon Fausten auf Softwarearchitekturen im Rahmen des *Sprint Plannings* und Alexander Kaack konzentriert sich besonders auf *Stakeholder Awareness*.

Die Problematik bei Softwarearchitekturen im Rahmen von agilen Methoden besteht darin, dass die Anforderungen, funktionierende Software so schnell wie möglich zu entwickeln, und gleichzeitig eine qualitativ hochwertige Architektur zu haben, nicht einfach miteinander zu vereinbaren sind. Dies kann darin resultierten, dass die Architektur nicht ausreichend durchdacht wird. Ein möglicher Lösungsansatz ist hier das *Agile Model Driven Development* [8].

Alexander Kaack hingegen betrachtet welche Herausforderungen es im Bezug auf die Stakeholder in einem agilen Projekt gibt und wie diese angegangen werden können. Stakeholder sind die jeweiligen Interessenten des Projektes. Bei *Stakeholder Awareness* geht es vor allem darum den Informationsfluss zu den jeweiligen Parteien zu optimieren. Je besser die Informationen für die Stakeholder aufgearbeitet sind, desto schneller und besser lassen sich wichtige Entscheidungen treffen und Probleme finden und lösen.

Das Ziel ist es ein gemeinsames Werkzeug zu entwickeln, welches diese Probleme löst und als ein Ganzes verwendet werden kann. Die Basis dieser Arbeiten wird es sein eine gemeinsame Schnittstelle zu bestehenden Ticketsystemen zu nutzen.

III. Bestehende Arbeiten

In diesem Abschnitt wird auf den aktuellen Forschungsstand näher eingegangen und relevante Arbeiten werden vorgestellt. Interessant sind hier Ansätze um agile Softwareentwicklung mit digitalen Mitteln weiter zu unterstützen und Arbeiten welche die Retrospektive im speziellen betrachten.

A. Activity Based Computing

Der vorgestellte Ansatz des *Activity Based Computing* konzentriert sich vor allem auf die Probleme der verteilten

Softwareentwicklung in Scrum. Eine Abgrenzung dieses Ansatzes gegenüber dem Projekt dieser Arbeit ist, dass nicht im speziellen auf die Retrospektive eingegangen wird. ABC ist ein Paradigma oft assoziiert mit *Ubiquitous Computing* bei dem Aktivitäten im System verwaltet werden und für alle Nutzer sichtbar sind. Folgende Prinzipien sind grundlegend für ABC [9] [10].

- **Activity basierte Ressourcen:** Bei ABC geht es vorwiegend darum die Aktivitäten des Menschen abzubilden. Daher sollen weitere Ressourcen an diese geknüpft werden. Artefakte wie User Stories oder Dokumentationen sollen nicht alleine stehen sondern immer an einer Activity hängen. Zusätzlich sind Informationen wie verantwortliche Personen, die Historie und eine klare Beschreibung Teil einer jeder Activity. Diese Bausteine sollen die reale Welt so gut wie möglich nachbilden. Der Vorteil einer solchen Modellierung ist zum einen eine Übersicht die alle Teilnehmer unabhängig von ihrem realem Standort erhalten und zum anderen eine Festlegung auf die Werkzeuge die verwendet werden sollen. Dies hilft um die technischen Probleme und die des Managements zu lösen.
- **Activity Unterbrechen und Wiederaufnehmen:** Da ABC oftmals im Bezug auf reale Tätigkeiten umgesetzt wird ist es essentiell diese auch unterbrechen zu können. Für den Anwendungsfall der Softwareentwicklung ist dies meist weniger wichtig, da es selten positiv ist zwischen vielen verschiedenen Aktivitäten hin- und her zu springen. Interessant wird das Pausieren erst wenn mehrere Personen an einer Activity beteiligt sind.
- **Activity Bewegen:** Unter Bewegung von Activites versteht man den einfachen Transfer zwischen Workstations und Devices. Dies kann zum Beispiel für Meetings interessant sein bei denen eine Activity von einem Laptop auf den Beamer transferiert werden. Eine Aufgabe für alle sichtbar zu machen mit allen zugehörigen Ressourcen fördert die Kommunikation und Kollaboration im Team. Zusätzlich können Acitivies zwischen den Teams bewegt werden wodurch die Arbeitsteilung erleichtert wird.
- **Activity Teilen:** Das Teilen verhält sich ähnlich zum Bewegen. Alle zugeordneten Benutzer können die Activity starten und pausieren. Dabei wird zwischen synchronem, asynchronem und temporalen Teilen unterschieden. Synchron bedeutet, dass mehrere Personen gleichzeitig an der Activity arbeiten können. Dies kann mit Hilfe von Chat- oder Videofunktionen unterstützt werden. Bei asynchronen oder temporalen Teilen wird die Activity blockiert wenn ein Nutzer an diese arbeitet bzw. nach festgelegten Zeitslots. Dies kann genutzt werden um Arbeitsergebnisse zu teilen, z.B. über Zeitzonen hinweg.
- **Activity Awareness:** Awareness ist das Bewusstsein für eine Sache. Dieses Bewusstsein wird geschaffen indem Informationen klar sichtbar dargestellt. In lokalen agilen Projekten schafft ein *Taskboard* Awareness für das Projekt in dem der aktuelle Projektzustand immer für jeden Sichtbar ist. ABC bietet immer einen Überblick über alle Activities mit allen möglichen Informationen. Um Awareness auf Projektebene zu schaffen müssen die Daten jedoch noch weiter aggregiert werden.
- **Activity Integration:** Activity Integration beschreibt die Verknüpfung von den digitalen Aktivitäten mit realen Objekten und Vorgängen. Ziel ist es die reale Welt und das ABC Modell möglichst konsistent zu halten. Ein typisches Beispiel könnte eine Verbindung von physischen *Taskboards* mit ABC sein.

Das Modell von *Activity Based Computation* bietet Reihe von Vorschlägen um typische Probleme in der verteilten Softwareentwicklung zu lösen. Allerdings existiert zum Zeitpunkt dieser Arbeit nur ein Konzept welches noch in keiner Implementierung getestet werden konnte. Auch bietet dieses Konzept zunächst keine konkrete Methodik um die Durchführung einer Retrospektive zu fördern.

B. Fallstudie zur Retrospektive

In der nächsten Studie wird nun insbesondere die Retrospektive betrachtet. Im folgenden Fallbeispiel wurden keine agilen Methoden verwendet. Stattdessen wurde eine einzige Retrospektive am Ende des gesamten Projektes durchgeführt. In dieser wurde ein einfaches Trackingtool verwendet um die Projektarbeit zu rekonstruieren. Das Ergebnis dieser Studie sind einige Fragestellung entwickelt die helfen sollen um Tools zu entwickeln die bei der Retrospektive verwendet werden können [11].

1) Versuchsaufbau: Die Studie fand im Rahmen eines Projektes im 6. Semester eines Bachelorstudiengangs mit einem Arbeitsaufwand von etwa der Hälfte des gesamten Semesters statt. Das Projektteam bestand aus 5 Studenten welche die Spezifikation für ein komplexes Softwareprojekt erhalten und implementiert haben. Am Ende des Semesters wurde die Retrospektive durchgeführt. Hierbei wurde zunächst ohne Hilfe von Werkzeugen gearbeitet. Die Teammitglieder haben, zuerst unabhängig voneinander, Zeitleisten konstruiert und Meilensteine gesetzt. Dann folgte eine gemeinsame Diskussion des Projektes. Im Anschluss wurde das Trackingtool Trac in die Retrospektive miteinbezogen. Trac enthielt die Tickets, ein Wiki und den Changelog im Versionierungssystem. Die Studenten navigierten durch die Information in Trac und neue Erkenntnisse wurden in die Retrospektive aufgenommen.

2) Ergebnisse: Die Individuellen Zeitleisten und Meilensteine reflektierten zunächst die Rolle im Projekt. Dies ist nicht überraschend, da solche Bereiche sehr präsent sind im Gedächtnis. Der Zeitpunkt zu dem die Implementierung anfing wurde zunächst von allen Teilnehmern recht weit nach hinten gesetzt. Erst durch die Retrospektive auf Basis des Trackingtools ist herausgekommen, dass es bereits sehr viel früher erste Implementierungsversuche gab. Dies wurde anschließend diskutiert. Ohne einen Rückblick auf das Projekt mit Hilfe eines Tools wäre dieser Aspekt verlorengegangen. Aus den folgenden Einzelinterviews ergab sich, dass sich einige der Teammitglieder durchaus über das vorherige Coding im klaren waren, da aber nicht alle daran teilgenommen hatten wurde dies zunächst nicht in die Diskussion in der Retrospektive

aufgenommen. Das Tool hat hier Abläufe des Projektes hervorgebracht, die nicht für alle sichtbar waren.

Eine weitere interessante Erkenntnis ist der Nutzen der miteinander verknüpften Daten in dem Tool. Da die Tickets mit den entsprechenden Änderungen in der Versionierung verbunden waren war es recht einfach Ereignisse zu rekonstruieren, in dem die Daten chronologisch durchgegangen wurden. Dies wäre nicht möglich gewesen wenn man sich bei der Retrospektive vollständig auf aggregierte Daten verlassen hätte, da die Zusammenhänge nicht mehr zu erkennen wären. Tatsächlich können aggregierte Daten falsche Eindrücke vermitteln, beispielsweise wenn die Anzahl der Dateien die in der Versionierung verändert wurden als Coding gewertet werden. Diese Daten können auch kleine Bugfixes und minimale Anpassungen sein und sagen nicht zwingend etwas über den tatsächlichen Umfang der Programmierung aus. Gezeigt wurde, dass das Einbinden von Werkzeugen mit historischen Daten nützlich für den Prozess der Retrospektive sein kann und die Tools daher mehrfachen nutzen bringen (Siehe Abbildung 2). Die folgenden Fragestellungen wurden formuliert um den Nutzen einer potenziellen Software für die Retrospektive zu evaluieren.

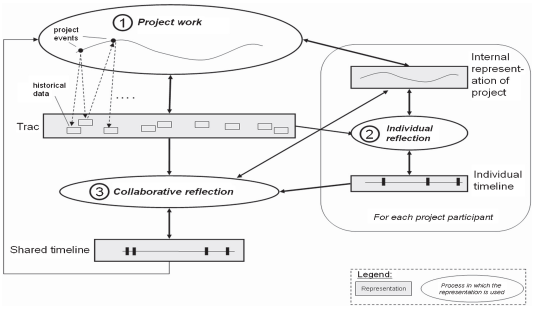

Abbildung 2. Trac als Tool während der Entwicklung und Retrospektive [11]

- Welche historischen Daten des Prozesses werden in dem Tool abgebildet?
- Können diese Daten chronologisch dargestellt werden?
- Ist es möglich unterschiedliche Versionen von Artefakten anzuzeigen?
- Werden Daten oberflächlich und im Detail angezeigt?
- Sind die Daten möglicherweise Privat?
- Wie können werden die Daten in eine Retrospektive eingebunden?

Diese Fragen erscheinen auch relevant wenn es um die Umsetzung eines Tools für agile Retrospektiven geht. Die zusätzliche Perspektive einer detaillierten Sicht für chronologische Daten ist interessant, da das Wort Unterstützung zunächst eine Vereinfachung der Daten suggeriert. Diese Studie zeigt jedoch, dass auch eine große Menge von kaum oder nicht gefilterten Informationen hilfreich sein kann.

C. Fallstudie zu Participation Balance

Die letzte hier vorgestellte Arbeit beschäftigt sich mit der Kollaboration an einem digitalem Tisch im Kontext des gemeinsamen Lernens. [12]. Im speziellen geht es darum Gruppen mit einem *Awareness-Mirror* zu zeigen welche Mitglieder der Gruppe besonders viel oder wenig reden und somit dazu ermutigt sich stärker einzubringen bzw. etwas zurückzuhalten. Vorherige Studien haben bereits gezeigt, dass ein Ungleichgewicht in der Beteiligung die Motivation negativ beeinflusst und den Lernerfolg beeinträchtigt [13]. Der Kontext der Softwareentwicklung ist hier nicht gegeben. Da es jedoch insbesondere bei der Retrospektive auch um gemeinsames Lernen geht kann ein tieferer Einblick in ein System um Awareness zu schaffen von großem Nutzen sein.

1) Funktionsweise: Der Tisch verwendet LEDs um die Beteiligung der Nutzer an dem Gespräch zu visualisieren. Diese können auf verschiedene Art angeordnet werden. Im einfachsten Fall bilden die verschiedenfarbigen LEDs einfache Balken pro Person für bis zu 4 Personen. (siehe Abbildung 4). Um festzustellen wer gerade Spricht sind in der Mitte 3 Mikrophone im Tisch angebracht. Eine Glasplatte erlaubt es das System als gewöhnlichen Tisch zu nutzen. Das Design des Tisches soll nicht zu großen Einfluss auf die reguläre Arbeit haben, jedoch trotzdem die Informationen sichtbar darstellen. Ein wichtiges Prinzip hinter Awareness ist es nicht invasiv zu arbeiten. Der Tisch stellt die Informationen lediglich zur Verfügung. Es ist der Gruppe überlassen wie sie damit umgeht.

Abbildung 3. Awareness-Mirror um Gesprächsbeteiligung zu visualisieren. [12]

2) Versuchsaufbau: Um zu untersuchen ob diese Art der Visualisierung das Bewusstsein für den Umfang der Kommunikation steigert und ob Gruppen welche mit dieser Visualisierung arbeiten diesbezüglich ausgeglichener sind wurde ein Versuch mit Studenten durchgeführt. 18 Gruppen aus je 4 Studenten wurden gebeten eine Aufgabe zu lösen bei der jede Person andere und unvollständige Informationen hatte, so dass sie gezwungen waren miteinander zu kommunizieren um die Aufgabe zu lösen. Die Hälfte dieser Gruppen arbeitete an dem Tisch mit der Visualisierung der Gesprächsbeteiligung. Die Andere Hälfte hatte eine Visualisierung bezüglich Länge mit der sie ein Thema diskutierten als Vergleichsgruppe. Diese Arbeiten wurden aufgenommen und gemeinsam mit abschließenden Fragebögen ausgewertet.

3) Ergebnisse: Ein Großteil der Studenten gab an, dass sie die Angaben auf dem Tisch wahrgenommen haben. Nur 5% fanden die Visualisierung der Gesprächsbeteiligung störend. 25% gaben jedoch an davon abgelenkt gewesen zu sein. Die Studenten, welche besonders viel oder besonders wenig während des Versuchs gesprochen haben, passten sich in den Gruppen welche ein Feedback darüber erhielten häufig über den Verlauf des Gesprächs an (siehe Abbildung 4).

Insgesamt zeigt die Studie einen klaren Effekt bei der Anwendung des *Awareness-Mirrors*. Es kann jedoch keine Aussage darüber gemacht werden wie ein solches System in einem Arbeitsumfeld aufgenommen werden würde. Auch die häufige Nutzung hat unter Umständen Auswirkungen darauf ob eine Änderung in den Verhaltensweisen stattfindet oder nicht. Interessant ist auch, dass eine Änderung der Verhaltensweise nur bei denjenigen stattfand, welche angegeben haben es wäre wichtig, dass alle Mitglieder einer Gruppe ähnlich viel Sprechen. Wenn die angezeigten Informationen nicht als negativ Empfunden werden findet daher auch keine Veränderung statt. Dies ist wichtig wenn dieses Awarenesskonzept auf andere Bereiche übertragen werden soll.

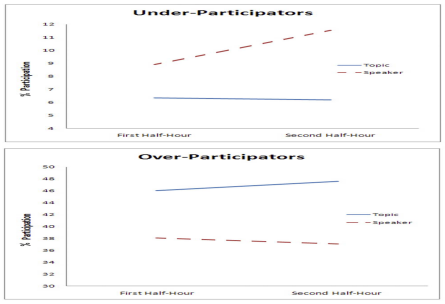

Abbildung 4. Verlauf der Gesprächsbeteiligung bei Ausreißern [12]

IV. Projektansatz

Das Ziel der weiteren Arbeit ist es ein System zu entwickeln welche die Retrospektive in der Softwareentwicklung fördert und bekannte Probleme vermeidet.In diesem Abschnitt werden die Ansätze welche im Verlauf des Grundprojektes weiter verfolgt werden sollen genauer beschrieben. Ein zentraler Unterschied zu den vergleichbaren Arbeiten ist die Konzentration auf die Retrospektive in der agilen Softwareentwicklung. Die Umsetzung ist mit der Entwicklung von Plugins für Das Trackingssystem *Jira* von Atlassian geplant. Ein Grund hierfür ist, dass mit dem Fokus auf ein zentrales Softwaresystem die Probleme besser anzugehen sind.

A. Awareness für die Retrospektive

Etwa 25% aller agiler Projekte führen keine Retrospektiven in ihrem Scrumprozess durch [7]. Dies begründet sich oft dadurch, dass die Retrospektive keinen unmittelbaren Nutzen für das aktuelle Projekt zu bringen scheint und sich nur auf längere Sicht auszahlt. Dies ist schwer zu kommunizieren. Eine Möglichkeit den Nutzen der Retrospektive deutlicher zu machen könnte sein einen *Awarenes-Monitor* im Büro unterzubringen. Dieser Sollte Informationen bezüglich der Arbeit des Teams beinhalten anstatt sich auf den Zustand des Projektes zu beziehen. So könnte man darstellten, welche Ziel sich das Team in der letzten Retrospektive gesetzt hat und der Grad zu dem diese zu dem Zeitpunkt Team als erfüllt angesehen werden.

Alternativ zu einem Monitor auf Hardwarebasis ließe sich etwas derartiges auch in einem konkreten Screen innerhalb des Trackingsystems umsetzen. Hier würde allerdings der unvermeidbare Blick auf den Monitor nicht gewährleistet sein. Ein Vorteil wäre allerdings die direkte Beteiligung des Teams, zum Beispiel über direktes Feedback in dem System. So ließen sich bereits vor der Retrospektive Vorschläge sammeln welche Dinge besprochen werden sollten.

B. Kommunikation in Meetings

Die Probleme in der Kommunikation, insbesondere in der verteilten Softwareentwicklung betreffen alle Prozesse. Meetings werden länger wenn diese nicht persönlich im gleichen Raum geführt werden können und sind gleichzeitig weniger effektiv. Auch die Kritikfähigkeit ist wesentlich stärker in einem lokalen Umfeld, anstatt in einer beispielsweise einer Telefonkonferenz. Um solche Probleme zu lösen können Audio- und Videoinformationen in das vorhandene Trackingsystem integriert werden. Ein wichtiger Faktor hier ist das dieser Aspekt in die bereits genutzte Software eingebaut wird um den Fluss des Meetings nicht zu stören [14].

C. Visualisierung in Meetings

Neben der Kommunikation soll auch die Visualisierung währen der Retrospektive verbessert werden. Diese Hilfestellung soll den Entwicklern erlauben ihre Problematik einfacher erläutern zu können. Ein Beispiel für eine solche Hilfe wäre die Übertragung des Bildschirms der sprechenden Person auf einen Beamer für alle sichtbar, ähnlich wie es im *Activity Based Computing* [9] vorgesehen ist. Dies ist auch für lokale Teams nützlich, besonders essentiell jedoch für die Nachvollziehbarkeit in verteilten Teams [15].

D. Monitoring des Projektes

Um den Entwicklern weitere Informationen über die geredet werden kann zu bieten muss das Projekt analysiert, und die wichtigsten Punkte aggregiert zur Verfügung gestellt werden. Anders als der bereits vorgestellte Ansatz liegt hier jedoch der besondere Fokus auf der Filterung auf wichtige Inhalte. Dennoch sollten bei Bedarf Details sichtbar gemacht werden können. Diese Daten können entweder als **Awareness-Monitor** eingesetzt werden oder speziell für das Meeting in der Retrospektive aufbereitet werden. Mögliche interessante Informationen sind beispielsweise durchschnittliche Bearbeitungsdauer von User Stories im Vergleich zum Schätzwert bis hin zu Anzahl der Tickets welche nach Abschluss nochmals geöffnet wurden. Diese Informationen weisen nicht zwingend auf Fehler hin. Jedoch bieten sie in jedem Fall Gesprächsstoff für Verbesserungspotenzial des Teams. [16].

V. Arbeitsplan und Risiken

In dem folgenden Teil der Arbeit soll noch genauer auf die Vorgehensweise während des Grundprojektes eingegangen und mögliche Risiken für das Projekt sollen deutlich gemacht werden.

A. Arbeitsplan

Das Grundprojekt wird vorwiegend mit dem gesamtem Team gemeinsam durchgeführt werden. Das Ziel wird es vor allen sein eine gemeinsame Basis zu entwickeln auf der die späteren Arbeiten aufgebaut werden können. Als Grundlage für die Software wurde *Jira* von Atlassian ausgewählt. Einer der ersten Schritte der Projektarbeit wird es sein *Jira* näher zu beurteilen. Hier müssen die vorhandenen Features und Plugins analysiert und für den Zweck der Retrospektive bewertet werden. In diesem Schritt wird auch die Entwicklung von neuen Plugins für *Jira* näher beleuchtet werden. Die Schnittstelle und

die Art und Qualität der abfragbaren Daten wird analysiert und ein erstes minimales Plugin soll entwickelt werden.

Dann werden einzelne Ansätze implementiert und bereits grundlegend evaluiert. Dies geschieht auf Basis einer Webanwendung, so dass die Implementierung sowohl für normale Monitore, als auch für digitale *Taskboards* und potenziell auch Multitouch Tische anwendbar ist. In dieser Phase wird Usability einen Schwerpunkt der Implementierung ausmachen. Die Umsetzung selbst erfolgt nach einem agilen Vorgehensmodell, aller Wahrscheinlichkeit nach Scrum.

Nachdem die Methoden in einer relativ simplen Form implementiert worden sind soll dieser Prototypen in der Praxis getestet werden. Hierzu soll das System für einige Wochen in einer oder mehrerer Unternehmen eingesetzt werden. Umfragen und einzelne Beobachtungen der Meetings helfen bei der Auswertung. Ziel dieser Phase ist den praktischen Nutzen des Systems zu evaluieren. Dazu gehören unter anderem die Relevanz der dargestellten Daten, der Aufwand der durch das System wegfällt bzw. hinzukommt und die generelle Bedienbarkeit des Systems beispielsweise als *Taskboard*. Im Idealfall findet dieser Praxistest sowohl in einem verteilten, als auch lokalen Softwareprojekt statt. Mit den Ergebnissen dieses Tests wird das System nun weiter optimiert.

Das Gesamtziel des Projektes ist es gemeinsam mit Leon Fausten und Alexander Kaack einen Prototypen zu entwickeln mit dem die gefundenen Ansätze weiter untersucht werden können. Die Basis dieses System wird eine Schnittstelle zu *Jira* bilden welche alle gemeinsam Nutzen. Die Evaluierung wird sowohl für das Gesamtsystem als auch für die einzelnen Aspekte stattfinden.

B. Risiken

Die folgenden Risiken sind für das Projekt identifiziert worden.

- Die Schnittstelle zu *Jira* ist bisher unbekannt. Die Art und Qualität der Daten welche abgefragt werden können ist essentiell für ein funktionierendes System.
- Die praktische Evaluierung kann nicht zwingend auf alle Softwareprojekte generalisiert werden. Unternehmen und Softwareentwickler haben ihre Eigenarten welche sich auf die Nutzung des Systems auswirken können. Das erhaltene Feedback sollte daher nicht blind umgesetzt werden.
- Im Prozess der Softwareentwicklung werden bereits eine Vielzahl von Werkzeugen zur Unterstützung von verschiedenen Prozessen angewendet. Es besteht die Gefahr ein weiteres System zu bauen, welches manuell gestartet und verwaltet werden muss. Eine automatische Integration in vorhandene Systeme ist hier der Schlüssel um dies zu vermeiden.
- Es ist möglich, dass durch die Software das Arbeitsklima verschlechtert wird. Hier ist es wichtig zu entscheiden welche Daten den Teammitgliedern zur Verfügung gestellt werden. So hätte eine Auflistung der Commits nach Person beispielsweise Implikationen bezüglich der Produktivität jedes einzelnen. ähnlich ist wenn Schuldzuweisungen durch die Informationen möglich sind. Hier wird es wichtig sein genau abzuwägen welche Daten dargestellt werden sollten um die Produktivität zu verbessern.
- Die Entwickler könnten sich womöglich beobachtet und kontrolliert fühlen durch das System. Da es in der Retrospektive um die Arbeitsweise jedes einzelnen geht, was um einiges persönlicher ist als der Stand des Projektes, ist es von großer Bedeutung möglichst keine Wertung vorzunehmen sondern nur Hilfestellungen zu bieten mit denen sich die Benutzer selbst verbessern können. Auch hier ist die Form der verwendeten Daten wichtig.

VI. Zusammenfassung

Im Rahmen dieser Ausarbeitung wurde eine grundlegende Untersuchung bezüglich der potenziellen digitalen Unterstützung für Scrum im Bezug auf die Retrospektive durchgeführt. Im ersten Abschnitt dieser Arbeit wurde die Motivation dargestellt. Agile Verfahren werde immer beliebter, wobei Scrum das mit Abstand am weitesten verbreitete Verfahren ist. Danach wurde noch eine kurze Einführung in Scrum gegeben. Klar ist, dass auch im Scrummodell nicht alles perfekt ist. Einige Herausforderungen für Scrum Teams wurden identifiziert und erklärt. Diese gelten besonders für verteilte Softwareprojekte, jedoch nicht ausschließlich.

Diese Arbeit findet nicht in einem Vakuum statt. Leon Fausten und Alexander Kaack befassen sich ebenfalls mit einer digitalen Unterstützung für agile Methoden. Die Perspektive ist hier jedoch eine andere. Leon Fausten konzentriert sich auf Architekturplanung in Verbindung mit dem *Sprint Planning* während Alexander Kaack seinen Fokus auf *Stakeholder Awareness* legt.

Der nächste Teil der Ausarbeitung hat einige Ansätze vorgestellt welche für das Projekt relevant sind. *Activity Based Computing* macht die Arbeit der Teammitglieder auch über weite Entfernung für alle Sichtbar. Der Informationsaustausch ist simpel wodurch die verteilte Arbeit erleichtert wird [9] [10]. Diese Prinzipien lassen sich auch auf die Retrospektive übertragen. Die zweite vorgestellte Arbeit untersuchte eine Retrospektive am Ende eines Projektes mit Hilfe eines einfachen Trackingtools. Die Erkenntnisse hieraus zeigen, dass der Zugang zu detaillierten historischen Informationen neue Aspekte der vergangenen Arbeit hervorbringen kann welche in der Retrospektive diskutiert werden können [11]. Bei der letzten vorgestellten Studie ging es um *Participation Balancing* mit einer digitalen Anzeige auf einem Tisch. Hier wurde gezeigt wie eine Hardwarelösung Einfluss auf das Verhalten der Teilnehmer in einem Gespräch haben kann und ein ausgeglicheneren Austausch fördert [13].

Auf Basis dieser und weiterer Arbeiten wurden nun eine Ansätze Punkte für die Unterstützung der Retrospektive formuliert. So könnte ein *Awareness-Monitor* dabei helfen das Verständnis für die Retrospektive zu steigern. Die Kommunikation könnte durch Integration von Audio und Video in bereits genutzten Trackingsystemen verbessert werden. Das Monitoring und die Visualisierung von relevanten Daten auf beispielsweise einem *Taskboard* kann von großem Wert sein um die Qualität des Meetings zu verbessern und Teammitglieder zu ermutigen ihre Meinung zu äußern.

Im letzten Abschnitt dieser Arbeit ging es nun darum einen passenden Arbeitsplan für das weitere Vorgehen in dem Projekt auszuarbeiten und die Risiken des Projektes zu formulieren. Im Grundprojekt soll zunächst *Jira* als potenzielles Grundsystem bewertet werden und ein Prototyp entwickelt werden.

Dieser soll dann in einem oder mehreren Unternehmen durch mehrwöchige Tests evaluiert werden.

LITERATUR

[1] VersionOne, "8th annual state of agile survey," 2014. [Online]. Available: http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf

[2] J. S. P. D, K. Schwaber, C. creators Of Scrum, C. J. Sutherl, and P. D, "The scrum papers: Nuts, bolts, and origins of an agile process."

[3] K. Schwaber and J. Sutherland, "The scrum guide," 2013.

[4] K. E. Nidiffer and D. Dolan, "Evolving distributed project management," IEEE Software, vol. 22, no. 5, 2005, pp. 63–72.

[5] J. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," IEEE Trans. Softw. Eng., vol. 29, no. 6, Jun. 2003, pp. 481–494. [Online]. Available: http://dx.doi.org/10.1109/TSE.2003.1205177

[6] J. S. Olson and G. M. Olson, "Culture surprises in remote software development teams," Queue, vol. 1, no. 9, Dec. 2003, pp. 52–59. [Online]. Available: http://doi.acm.org/10.1145/966789.966804

[7] ScrumAlliance, "The state of scrum: Benchmarks and guidelines," 2014. [Online]. Available: https://www.scrumalliance.org

[8] Y. Zhang and S. Patel, "Agile model-driven development in practice," Software, IEEE, vol. 28, no. 2, march-april 2011, pp. 84 –91.

[9] J. E. Bardram, "Activity-based computing support for agile and global software development," 2008. [Online]. Available: http://www.citeulike.org/user/bardram/article/6535062

[10] M. Esbensen and J. Bardram, "Tool support for globally distributed scrum," 2014. [Online]. Available: http://nexgsd.org/wp-content/uploads/2014/02/Esbensen-Bardram-2014-Tool-Support-for-Globally-Distributed-Scrum_CSCW_2014_GSD_Workshop.pdf

[11] B. Krogstie and M. Divitini, "Supporting reflection in software development with everyday working tools," 2010.

[12] K. Bachour, F. Kaplan, and P. Dillenbourg, "An interactive table for supporting participation balance in face-to-face collaborative learning," TLT, vol. 3, no. 3, 2010, pp. 203–213. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TLT.2010.18

[13] J. M. DiMicco, A. Pandolfo, and W. Bender, "Influencing group participation with a shared display," in Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, ser. CSCW '04. New York, NY, USA: ACM, 2004, pp. 614–623. [Online]. Available: http://doi.acm.org/10.1145/1031607.1031713

[14] S. Dorairaj, J. Noble, and P. Malik, "Understanding lack of trust in distributed agile teams: A grounded theory study," in Evaluation Assessment in Software Engineering (EASE 2012), 16th International Conference on, May 2012, pp. 81–90.

[15] J. Paredes, C. Anslow, and F. Maurer, "Information visualization for agile software development," in Software Visualization (VISSOFT), 2014 Second IEEE Working Conference on, Sept 2014, pp. 157–166.

[16] I. Omoronyia, J. Ferguson, M. Roper, and M. Wood, "A review of awareness in distributed collaborative software engineering," Software: Practice and Experience, vol. 40, no. 12, 2010, pp. 1107–1133. [Online]. Available: http://dx.doi.org/10.1002/spe.1005