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

Tracen und Visualisieren von Entwickler-Interaktionen in Jira

Tracing and Visualizing of Developer-Interactions in Jira

Bachelorarbeit

im Studiengang Informatik

von

Quang Phu Quang Le

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Prof. Dr. Joel Greenyer
Betreuer: M.Sc. Fabian Kortum

Hannover, 27.03.2019

Erklärung der Selbstständigkeit

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

Hannover, den 27.03.2019

Quang Phu Quang Le

Zusammenfassung

Die inneren Kommunikationsstrukturen und -Angewohnheiten von Entwickler-Teams haben einen großen Einfluss auf den fortlaufenden Erfolg eines Softwareprojektes. Je mehr Informationen Entwickler austauschen können, desto klarer sind die Ziele, und desto höher ist die Anzahl der Entwickler, die motiviert sind, sich an einem Projekt zu beteiligen und aktiv daran mitzuarbeiten [15]. Um die Kommunikationsstrukturen zu verstehen, wurden Programme entwickelt, die Kommunikationsdaten analysieren und grafisch darstellen, wie zum Beispiel FlowExplorer [11]. Die meisten dieser Anwendung sammeln die subjektiven Rückmeldungen der Teammitglieder bezüglich des aufgebrachten Kommunikationsverhalten. Die Datenerhebung ist daher aufwendig.

Diese Arbeit schlägt einen anderen Ansatz zur Optimierung der Datenerhebung, nämlich ein zentrales Plug-In *Interaction Revealer* für das JIRA Projekt- und Aufgabenverwaltungssystem vor. Dieses Plug-In erfasst die historischen und realen Dateninteraktionen zwischen Entwicklern in JIRA und visualisiert den Zusammenhang davon. Mit dem *Interaction Revealer* gewinnen sowohl Projektleiter als auch Teammitglieder eine komplexitätsreduzierende, objektive und übersichtliche Überblick vom Kommunikationsverhalten, ohne zu viel Aufwand für den Prozess der Datenerhebung zu betreiben.

In dieser Arbeit wird die Konzeptionierung und Implementierung von dem Plug-In näher eingegangen.

Abstract

The internal communication structures and habits of developer team have a major impact on the ongoing success of a software project. The more information developers are able to exchange, the clearer are the goals, and the higher is the number of developers who are motivated to collaborate and actively participate in a project [15]. In order to understand the communication structures of team, programs have been developed that analyze and graphically display communication data, such as FlowExplorer [11]. Most of this application collects the subjective feedback from the team members regarding the provided communication behavior. Therefore, the method for collecting data is in this case expensive and costs much effort.

This bachelor thesis proposes a different approach to optimize the data collection, namely a central plug-in *Interaction Revealer* for the JIRA project and task management system. This plug-in captures the historical and real-world data interactions between developers in JIRA and visualizes the context of it. With the *Interaction Revealer*, both project leaders and team members gain a complexity-reducing, objective, and clear overview of team behavior without spending too much effort on the data collection process.

In this thesis, the conception and implementation of the plug-in is discussed in more detail.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ein entscheidender Aspekt bei großen Softwareprojekten ist die Zusammenarbeit zwischen Entwicklern. Laut Herbsleb et al. [9] kann die mangelnde Kommunikation die fehlenden Kenntnisse über die Projektzustände verursachen und kann dazu führen, dass die im Projekt entstandenen Probleme nicht adressiert werden. Des Weiteren betont das Agile Manifesto den wesentlichen Einfluss von Entwickler-Kommunikation in agilen Projekten, „Menschen und Interaktionen gelten mehr als Prozessen und Werkzeugen“ [1]. Die Analyse der menschlichen Aspekte, darunter Kommunikation und Interaktion, ist daher notwendig geworden, um die Schwierigkeiten schnell zu identifizieren, beheben und das Projekt zum Erfolg zu bringen [5].

In diesem Zusammenhang wurden im Rahmen einer Bachelorarbeit die nachfolgende Problemstellung untersucht, ein Lösungskonzept ergründet sowie eine praktisch orientierte Softwarelösung entwickelt.

1.1 Problemstellung

Die erste Möglichkeit zur Untersuchung der Kommunikation ist, die Umfragen durchzuführen. Dabei soll jeder Entwickler die Berichte über seinem Kommunikationsverhalten anhand der Fragebögen abgeben. Der Nachteil dieses Verfahrens ist die subjektiven Rückmeldungen vom Teammitgliedern und die aufwendige manuelle Erfassung der Daten. Jedoch gibt es andere technische Vorgehensweise, die einerseits objektive Kommunikationsdaten untersucht und andererseits den Fragebogenanteil verkleinert. Das ist das Tracen der Entwickler-Aktivitäten im Issue-Tracking System JIRA. Der Nachteil dieses Verfahrens besteht darin, dass die systeminterne Dateninteraktionen nicht immer direkt einsehbar oder nicht aufbereitet sind. Daher um möglichst viele Erkenntnisse aus den Systemdaten zu ziehen, sollen diese aufbereitet und visualisiert werden.

Die Motivation dieser Bachelorarbeit stellt die Konzeptionierung und Implementierung einer Softwareerweiterung dar, welche automatisch die

Entwickler-Interaktionen in JIRA verfolgt und den Zusammenhang visualisiert. Der wesentliche Vorteil dieser Anwendung soll grundsätzlich darin bestehen, dass keine Fragebögen erforderlich ist und dass aus der anschauliche Darstellung der Interaktionen das maximale Informationsgehalt abzuleiten ist.

1.2 Lösungsansatz

Der allgemeine Lösungsansatz dieser Problemstellung ist eine Erweiterung für die JIRA Projektmanagementsoftware, das einerseits die relevanten JIRA-Interaktivitäten der Teammitglieder sammelt und andererseits diese Daten vernünftig und passend aufbereitet. Damit lassen sich wöchentliche JIRA-Projektdaten hinsichtlich der Teaminteraktionen zentral, und somit über das Einzelsystem auswerten und bereitstellen.

Um die relevanten JIRA-Interaktionen zu erheben, wird die GQM-Methode verwendet, wobei die Interaktionsarten identifiziert werden und deren Aussagekraft analysiert wird.

Zur Erstellung der JIRA-Erweiterung und zur Implementierung der Datenerfassung dient die Programmiersprache *Java*, deren bereits verfügbaren API durch den Produkthersteller gewährleistet wird.

Des Weiteren kann die Visualisierung mit Hilfe der Bibliothek *D3.JS* [1] in Javascript erledigt werden. D3 listet zudem alle gängigen und relevanten Visualisierungsformen im Bereich der sozialen Networkanalyse.

1.3 Struktur der Arbeit

Diese Arbeit ist wie folgt strukturiert. In dem Kapitel 2 werden die grundlegende Erklärungen und Definitionen präsentiert, darunter eine Einführung in die JIRA Umgebung und die Goal-Question-Metric Methode. Das darauffolgende Kapitel 3 beinhaltet die Beschreibung der Planung und Vorgehensweise für das zu implementierende Programm. Anschließend folgt das Kapitel 4 , in dem die Entwicklung der Software beschreibt wird. Des Weiteren wird in Kapitel 5 die Analyse und Interpretation der durchgeführten Evaluation vorgestellt. Im letzten Kapitel 6 wird ein Fazit dieser Arbeit gezogen und ein Ausblick auf zukünftige mögliche Erweiterungen gegeben.

[1] https://d3js.org

Kapitel 2

Grundlagen

Dieses Kapitel umfasst die wichtigsten thematischen Grundkonzepte und Technologien, welche als Basiswissen für den Inhalt dieser Arbeit vorausgesetzt werden.

2.1 Grundlagen der JIRA-Software

Das Ziel dieser vorliegenden Arbeit ist die Implementierung einer Anwendung, welche systematische Verfahren zur Erfassung und Analyse von Entwickler-Interaktionen ermöglicht. Um die Interaktionsdaten systematisch zuzugreifen, wird die JIRA Software [12] als Basisplatform verwendet, mit der die Entwickler täglich anstehenden Aufgaben verwalten und dokumentieren können. Dieser Abschnitt behandelt einige wesentliche Grundlagen von JIRA und ihre Kernfunktionalitäten sowie Add-on, welches die JIRA-Platform um zusätzliche Funktionen erweitert.

2.1.1 Funktionsweise

JIRA ist ein von der Firma Atlassian entwickelte Webanwendung zur Fehlerverfolgung, Problemverfolgung und zum agilen Projektmanagement. Für die agile Softwareentwicklung wird JIRA primär eingesetzt, wo es das Anforderungsplanung und die Verfolgung von Problemen und Fehlern in Softwareanwendung unterstützt. JIRA bietet viele nützlichen Funktionen, die Handhabung von Problemen erleichtern. Einige der wichtigsten werden im nächsten Abschnitt gezeigt.

JIRA-Issue

Ein JIRA-Issue (Ticket oder Vorgang) repräsentiert einen Fehler, ein Problem oder eine Aufgabe in einem Softwareprojekt. Jedes Issue wird einem spezifischen Projekt zugeordnet und enthält die Beschreibung und weitere Metadaten zum Beispiel Zustand des Issues (offen, gelöst), Issue-Type,

Priorität, Issue-Ersteller und -Bearbeiter. Außerdem können während der Bearbeitung eines Issues weitere Entwickler-Aktivitäten wie Kommentare und Dateien hinzugefügt werden. Aus diesem Grund wird die Verfolgung von Issues in JIRA benutzt, um herauszufinden, wie die Entwickler miteinander kommunizieren und interagieren. Abbildung 2.1 zeigt eine Beispielübersicht eines JIRA-Issues mit der Typ Bug. Grundlegend beinhaltet alle notwendigen Informationen, um eine Entwickleraufgabe umsetzen zu können.

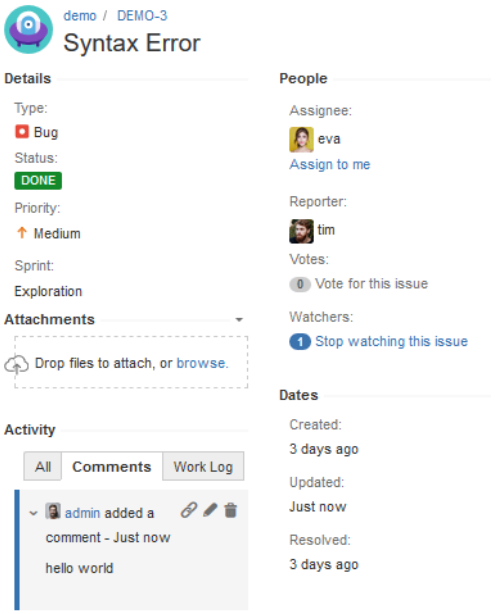


Abbildung 2.1: Übersicht eines JIRA-Issues

JIRA Sprintverwaltung

In der agilen Softwareentwicklung wird ein Projekt in mehreren regelmäßigen und wiederholbaren Arbeitsabläufe unterteilt. Diese Zyklen werden Sprint genannt und sind zeitlich beschränkt. Der Zweck eines Sprint ist es, ein funktionsfähiges Zwischenprodukt zu entwickeln. Die einzelnen Sprints bauen aufeinander auf, wobei die Zwischenprodukte der vorherigen Sprints das Grundlage für die nächsten sind [6]. Deswegen ist es sinnvoll, das Entwicklerteamverhalten in einzelnen Sprints zu betrachten und analysieren.

JIRA bietet eine benutzerfreundliche Ansicht für die Verwaltung von Sprints. Das Bild 2.2 zeigt ein Scrum-Board, welches eine aktive Sprint-Phase zeigt, auf dem der Bearbeitungsstatus eines Issues verfolgt, bzw. von Entwicklern aktualisiert werden kann.

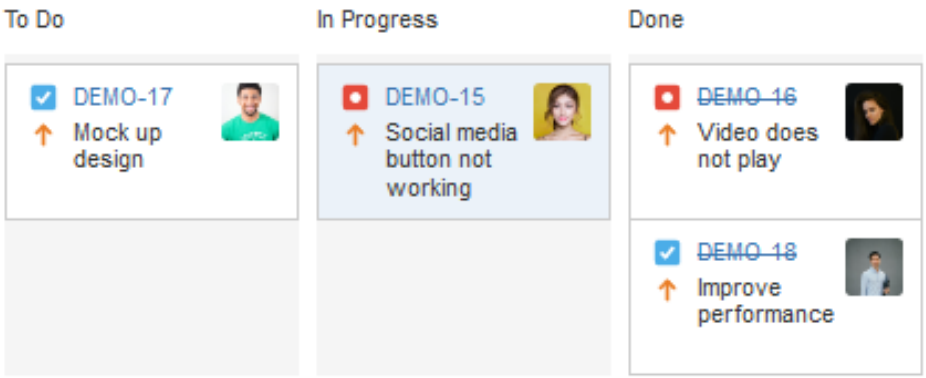


Abbildung 2.2: JIRA Scrum-Board Ansicht

Wenn man auf ein Issue klickt, kann man sich die Detail des Issues genauer anschauen. Die Inhalte sind dabei in der Abbildung 2.1 dargestellt.

2.1.2 JIRA Plug-In (Add-on)

Ein entscheidender Faktor, der JIRA zum Erfolg bringt, sind die zahlreichen Möglichkeiten, die Funktionalität durch ein Plug-In anzupassen und zu erweitern. Zur Erstellung und Integration der JIRA Add-on gibt es die von Atlassian JIRA bereitgestellten Java-APIs und -Bibliotheken, die den Zugriff auf JIRA Funktionen ermöglicht. Außerdem versetzt die JIRA Architektur den Benutzer in der Lage, die Modifikation zur Anpassung der Benutzeroberfläche (durch HTML/CSS) und Scripten (durch JavaScript) für Datenerfassung und Visualisierung zu realisieren.

Ein JIRA Erweiterung besteht aus mehreren Module. Einige davon werden im Folgenden verkürzt vorgestellt, die für diese Arbeit wichtig sind.

- Webwork wird verwendet, um neue URLs und Webseite in JIRA zu definieren.
- JQL Function: JQL steht für JIRA Query Language. Mit diesem Modul kann der Nutzer alle Issues nach gewünschtem Kriteria suchen. Die Befehle ähneln dabei der SQL-Syntax.
- REST: Die JIRA-REST-APIs werden für die Remote-Kommunikation mit den JIRA Server-Anwendungen benutzt (siehe Abschnitt 4.1).
- Servlet-Filter: Mit diesem Modul kann man Aufrufe von JIRA-URLs abfangen und ändern, was zurückgegeben wird, sowie neue Java-Servlets verwenden. Dieses Modul wird in dieser Arbeit für die Verfolgung von Herunterladen der Attachments verwendet.

2.1.3 Entwickler-Aktivitäten in JIRA

JIRA bietet dem Entwicklerteam mehrere Möglichkeiten, bei gemeinsamen Arbeiten miteinander zu kommunizieren, interagieren und zusammenarbeiten. Dadurch kann sich die Entwickler-Interaktion charakterisieren.

Die Entwickler-Interaktion in JIRA bezeichnet den Austausch oder Übertragung von Daten zwischen Entwicklern (nicht mit sich selbst), die auf verschiedenen Arten in JIRA System auftreten können. Das heißt, in einer Interaktion existieren immer ein Absender und Empfänger. Die Interaktion der Teammitglieder besteht aus vier folgenden Typen:

- **Assignment** (Aufgabenzuweisung) ist eine Aktivität des Benutzer, wenn ein Issue vom ihm erstellt und an den anderen zugewiesen ist. Hier wird die Selbstzuteilung nicht gezählt, da es keine Interaktion mit den anderen Benutzer gilt. Aus der Anzahl der Assignment kann man die Produktivität schätzen, beispielsweise wie viele Arbeit von dem Entwicklern erledigt wurden, und ob Tendenzen bestehen, dass unbeliebte Aufgaben immer weitergeschoben werden. Abbildung 2.3 weist die Interaktion zwischen zwei Entwicklern bezüglich von Assignment auf, welche gerichtet vom Issue-Erstellern zum Issue-Bearbeitern ist.

Abbildung 2.3: Assignment-Interaktion zwischen 2 Benutzern

- **Comment** (Kommentar): Jedes Issue verfügt über eine Feld zur Diskussion. Dazu kann man die Kommentar schreiben, um seine Meinung und Informationen über das Ticket beizutragen, wie zum Beispiel Issues, Kunden-Feedback, Beobachtung während der Entwicklung und so weiter. In einer Kommentar hat man noch eine Möglichkeit, mit dem Befehl „@user“ Teammitglieder schnell einmal über Arbeiten zu benachrichtigen. Wenn kein Benutzer in Kommentar erwähnt wird, dann wird angenommen, dass diese Kommentar an allen gesendet wird. Abbildung 2.4 stellt die Struktur der Kommentar-Interaktion dar, in der Startknoten den Absender repräsentiert und der Endknoten den Empfänger.

Abbildung 2.4: Comment-Interaktion zwischen 2 Benutzern

- **Watch** (Beobachtung): Anders als Erwähnung (Mention) in Kommentar, wenn ein Nutzer ein Issue betrachtet (watch), erhält er während des gesamten Bearbeitungsdauer des Issues eine Menge von Benachrichtigungen. Im Prinzip kann der Systemadministrator einstellen, welche Benachrichtigungen Beobachter bekommt. Standardmäßig sehen der Beobachter alle wichtigen Ereignisse eines Vorgangs wie Feldänderung, Kommentar, angehängte Dateien, Neuzuweisung des Vorgangs.

  Die Beobachtungsfunktion ist sehr nützlich zur Verfolgung eines Tickets. Zum Beispiel wenn in der Software ein kritischer Fehler vorliegt, möchte man immer den Überblick über alle Aktualisierung erhalten, die das Team zu diesem Ticket gemacht hat. Die Verfolgung des Issues hilft dabei, mit dem Problem in Auge haben, von der Erstellung, Entwicklung bis zur Lösung. Der Beobachter verfolgt nicht nur das Issue, sondern auch den Issue-Bearbeiter (Assignee), der dieses Issue übernimmt. Dies führt zu einer Beobachtung-Interaktion zwischen Teammitgliedern. Anhand der Abbildung 2.5 wird dieser Typ der Entwickler-Interaktion verdeutlichen.

Abbildung 2.5: Watch-Interaktion zwischen 2 Benutzern

- **Download/View** (Austauschen von festen Informationen [21]): Die Teammitglieder sind auch in der Lage, miteinander zu interagieren, indem sie separate feste Informationen (z.B Dokumente, Audio, Video, Quellcode) an einem Ticket anhängen, um weitere Details und Erklärungen zur Verfügung zu stellen. Diese Interaktionsart hilft dabei, die Relevanz eines Attachments für die Lösung des Tickets zu identifizieren. Die Beziehung zwischen dem Hochladern und dem Downloadern bildet die Entwickler-Interaktion ab (siehe Abbildung 2.6). Hier wird der Pfeile nach recht ausgehend definiert, weil der Downloader in dem Fall aktiv die Interaktion betreibt. Anders gesagt, der Downloader schickt eine Downloadanfrage an den Uploader.

Abbildung 2.6: Download-Interaktion zwischen 2 Benutzern

Kapitel 3

Konzept und Planung

In folgenden Kapitel sollen die Konzeptionierung und die Planung des Interaction Revealer näher betrachtet werden. Dazu zählt neben verwandte Arbeiten, auch die Berücksichtigung von Bestandssoftware sowie ein zielgerichtetes Lösungskonzept für die Problemstellung dieser Arbeit.

3.1 Verwandte Arbeiten

Es gibt einige Arbeiten und bestehende Software, die Menschen-Faktoren in einem Projekt untersuchen, messen und visualisieren, darunter Kommunikation, Produktivität.

Die Recherche ergab eine Vielzahl von wissenschaftlichen Artikeln zur Analyse des Kommunikationsverhaltens zwischen Teammitgliedern. Kortum et al. [8] erfassen auch die Kommunikationsdaten über Feedbacks von Team und visualisieren sie in einem sozialen Netzwerk. Die Feedbacks werden aber subjektiv über digitalen Fragebogen gesammelt.

Des Weiteren untersuchen Schneider et al. [19] die Charakterisierung von Kommunikation und entwickeln dazu den FLOW-Netzwerk zur Modellierung von Entwicklerteam und seiner Kommunikation. Die in diesem Paper [19] genannten Metriken sind für dieser vorliegenden Arbeit themenrelevant.

Bezüglich der Effektivität von Entwicklergesellschaft messen und bewerten Ortu et al. [15], wie sich das Team hinsichtlich der Problemlösungszeit eines bestimmten Problems verhalten, in dem die Autoren 7 populäre Projekte in JIRA analysieren. Dabei wurden die Datensatz durch die Sammlung der Daten aus dem JIRA System erfasst. Basierend auf diesen Daten wurden das Entwicklernetzwerk extrahiert.

Außerdem existieren eine bestehenden JIRA-Erweiterungen bezüglich des Konzepts von Teamstatus in JIRA. Die erste themenverwandte Software heißt Screeningful Metrics for JIRA. Es handelt sich primär um visuellen Dashboards und automatisierte Teamstatusberichte. Screeningful Metrics for JIRA ermöglicht den Benutzer, den Projektfortschritt zu visualisieren und zu

teilen. Durch die Funktion der Datenvorhersage kann der Nutzer den Projekt besser planen.

3.2 Anforderung an den Interaction Revealer

In Softwareprojekten gelten die Anforderungen als Basis für die effiziente Softwareplanung und -entwicklung. Daher werden im Zuge der Konzeption die Anforderungen an die Interaction Revealer Anwendung aufgestellt. Dabei werden neben funktionale auch nicht-funktionale Anforderungen definiert, die Software abbilden sollte. In diesem Abschnitt werden diese Anforderungen anhand zwei User Stories beschrieben.

1. **User Story 1**

> Als Projektleiter (Administrator) möchte ich separat die Dateninteraktionen in einzelnen Sprint erfassen, um das Interaktionsverhalten jeder Iteration zu verfolgen und zu vergleichen.

Funktionale Anforderungen

- Das Programm sammeln, speichern und aktualisieren die systeminternen Aktivitätsdaten des Entwicklers, sobald der Administrator diese Aktion startet.
- Die gespeicherte Interaktionsdaten müssen nach Sprint sortiert. werden.

Nicht-funktionale Anforderungen

- Diese obigen Funktionen sind nur für den Administrator zugänglich.

2. **User Story 2**

> Als Projektleiter und Entwickler möchte ich die Visualisierung der Dateninteraktionen von den bisherigen geschlossenen Sprints haben, um das Kommunikationsverhalten vom Team besser zu verstehen.

Funktionale Anforderungen

- Die Anwendung stellt die Entwickler-Interaktionen und deren Zusammenhang in Grafik dar.
- Der Nutzer können zwischen angezeigten Sprints navigieren.

Nicht-funktionale Anforderungen

- Diese obigen Funktionen sind für allen authentifizierten Benutzer zugänglich.
- Die Visualisierung soll möglichst einfach sein, aber das maximale Informationsgehalt bringen.

3.3 Analyse relevanter Interaktionen durch GQM

Ziel des Interaktion Revealer ist zur Charakterisierung von Teamstruktur bezüglich der Interaktionsdaten. Jedoch sind nicht alle Interaktionen in JIRA relevant in Hinblick auf dieses Ziel. Deswegen muss bereits vor der eigentlichen Datenerfassung untersucht und definiert werden, welche JIRA-Interaktionsdaten abgerufen und interpretiert werden können. Dazu wird die Goal-Question-Metric (GQM) Methode angewandt. Dieses Verfahren hilft dabei, zielgerichtete Fragen zu entwickeln, deren Antwort nützliche Informationen zur Erfüllung des Ziels liefern [18].

3.3.1 Zielfacetten

Der erste Schritt der GQM-Methode ist die Zielfacetten zu ermitteln, welche das Hauptziel definiert und verfeinert. Da die Eigenschaft jeder Interaktionsart (siehe Abschnitt 2.1.3) strukturell gleich ist, wird in dieser Arbeit ein zusammenfassendes Ziel gegeben. Dieses Ziel bezieht sich auf die Untersuchung von Team-Interaktion und -Struktur bezüglich der in JIRA existierenden Entwickler-Aktivitäten. Die dazu passende Perspektive sind die Projektleitung und der Entwickler, weil das Programm Interaktion Revealer dem Projektleiter und dem Team dabei, die Entscheidung zu treffen, sodass die Leistung, Effizienz und Produktivität des Team verbessert wird. Das Hauptziel is in Facettentablle 3.1 dargestellt, darunter können weitere Unterziele erweitert werden, z.B Untersuche die Entwickler-Interaktion anhand der Kommentar, Assignment, Watcher und Attachment aus der Perspektive der Projektleitung.

Zweck	**Qualitätsaspekt**	**Beobachtungs-gegendstand**	**Perspektive**
Untersuche	Entwickler-Interaktionen in JIRA	Interaktion jeder Art	Projektleiter

Tabelle 3.1: Tabellarische Darstellung der Facetten

3.3.2 Abstraction Sheets

Im nächsten Schritt werden die relevanten Fragen gestellt mit Hilfe eines Abstraction Sheets. Hier wird auf die Wiederholung unter den verschiedenen Interaktionsarten verzichtet, da diese stets die gleiche Eigenschaft besitzen. Die durch das Abstraction Sheet bestimmten Fragen können daher auf jede Interaktionsarten gleichermaßen verwendet werden. Das Abstraction Sheet besteht aus folgenden Angaben:

- Die Qualitätsfaktoren sind die Faktoren, die den Qualitätsaspekt charakterisierten können.
- Die Ausgangshypothesen sind die Vermutungen, wie es zur Zeit mit den Qualitätsfaktoren steht.
- Die Einflusshypothese beschreibt die Vermutungen dafür, was auf welche Weise welchen Qualitätsfaktor beeinflussen kann?
- Einflussfaktoren beziehen sich direkt auf Qualitätsfaktoren. Sie sind die Stellschrauben, mit denen man die Qualitätsfaktoren beeinflussen kann. [18]

Für das in der Tabelle 3.1 beschriebene Ziel wurde das folgende Abstraction Sheet 3.2 erstellt.

Zweck	**Qualitätsaspekt**	**Beobachtungsgegenstand**	**Perspektive**
Untersuche	Entwickler-Interaktionen in JIRA	Interaktion jeder Art	Projektleitung
Qualitätsfaktoren a. Von wem an wen b. Interaktionsarten c. Anzahl der Interaktionen jeder Art von Entwicklern d. Anzahl der gesamten Interaktionen von Entwicklern e. Sprint, in dem die Interaktion passiert f. Datum, an dem die Interaktion passiert		**Einflussfaktoren** a. Teamstruktur b. Kommunikations- und Teamfähigkeit c. JIRA Schulung d. Relevanz von Anhänge	
Ausgangshypothesen a. Nur Scrum-Master ist alle Aufgabenzuweisungen an anderen Entwicklern verantwortlich. b. Jeder Anhang an einem Issue wird mindestens einmal von dem Assignee des Issues heruntergeladen/gelesen. c. Die Anzahl an Interaktionen jeder Art von einem Entwicklern ist gleichgroß. d. Es gibt ein Mitglied, das überhaupt nicht mit anderen interagiert, das heißt das Mitglied ist vom Team ausgeschlossen. e. Es gibt ein Attachment, das über 50 mal abgerufen wurde.		**Einflusshypothesen** a. Teamstruktur beeinflusst (a). b. Kommunikations- und Teamfähigkeit wirkt sich (b) (c) positiv aus. c. Durch JIRA Schulung können Entwickler lernen, JIRA zu nutzen. d. Die Relevanz des Dokument für die Aufgabe beeinflusst (e)	

Tabelle 3.2: Abstraction Sheet für das Ziel in Tabelle 3.1

3.3.3 Fragen und Metriken

Aus dem Abstraction Sheet können nun Fragen und Metriken abgeleitet werden.

F1. Wie ist die Kommunikationsstruktur des Team in Bezug auf Dateninteraktionen?

- **M1.1** Anzahl der gesamten Interaktion des Teams
- **M1.2** Wie viel interagiert Entwickler mit wem?
- **M1.3** Anzahl der Interaktionen jeder Art
- **M1.4** Anzahl der Teammitglieder

F2. Wie aktiv interagiert sich der Entwickler?

- **M2.1** Anzahl der gesamten Interaktion des Entwicklers
- **M2.2** Mit wem interagiert er?

F3. Welche Bedeutung spielen die Attachments im Projekt?

- **M3.1** Anzahl der Abrufe von dem Dokument pro Tag, pro Sprint
- **M3.2** Anzahl der Abrufe jedes Entwicklers
- **M3.3** Das Datum, an dem Das Dokument angehängt wurde
- **M3.4** Der Nutzer, der das Dokument geschaut hat.
- **M3.5** Der Typ des Dokuments

Aus diesen gefundenen Metriken wird die Visualisierung konzipiert, welche im Folgenden näher eingegangen wird.

3.4 Visualisierungskonzept

Die Hauptaufgabe der Software ist die Visualisierung von Informationen, die aus der Entwickler-Interaktionen abgeleitet werden. Der Hauptzweck der Datenvisualisierung besteht darin, verschiedene Informationen übersichtlicher und effizienter als in einem Text oder in einer Tabelle über statistische Grafiken und Diagramme zu vermitteln. Anhand der Diagrammen kann der Benutzer beispielsweise ein bestimmtes Muster von Daten erkennen, Datengrößen miteinander vergleichen, ein Trend ablesen, Daten interpretieren, Analyseberichte erstellen und die Entscheidung treffen. Jedoch ist die Auswahl einer adequaten Visualisierungsart nicht trivial. Studien [10], [13] haben gezeigt, dass das Aussehen von grafischen Diagrammen eine wichtige Rolle bei der Interpretation durch Benutzer spielt. Daher ist es nötig, dass die Visualisierungen leicht zu interpretieren sind und das Kommunikationsverhalten mittels der relevanten Metriken so genau wie möglich charakterisieren. Zur Darstellung der Interaktionen zwischen Entwicklern wurden drei Diagrammtypen gewählt, nämlich soziales Netzwerk von Entwickler, Balken- und Flächendiagramm, welche im Folgenden beschrieben werden.

3.4.1 Entwickler-Netzwerk

Entwickler-Netzwerk wurde in mehreren Forschungen zur Untersuchung der Zusammenarbeit [20] und Kommunikationsstruktur [8] von Softwareteam, und zum Vorhersagen der Fehler [14] angewandt. Der Vorteil eines Entwickler-Netzwerks ist dabei, dass man eine Überblick und Grundstruktur über die Komponenten und wie sie sich mit einander kommunizieren, gewinnt.

Ein Netzwerk besteht aus zwei Mengen: eine Knotenmenge und eine Kantenmenge, wobei eine Kante zwei Knoten verbindet. Die Knotenmenge repräsentiert das Entwicklerteam und die Interaktionen im Team werden durch die Kantenmenge beschrieben. Der Netzwerk soll gerichtet sein, da die Entwickler-Interaktionen bidirektional sind. Um die einzelnen Interaktionsarten in dem Netzwerk mit darzustellen, werden die erweiterten Knoten eingeführt, welche aus einem Hauptnoten und andere vier kleinen dazugehörigen Knoten gebildet werden. Abbildung 3.1 zeigt einen Beispielgraph, in dem zwei erweiterten Knoten mit einander verbunden sind. Um den Hauptknoten herum befinden sich die bunten Knoten, die immer an dem Hauptknoten angebunden sind. Diese Nebenknoten dienen dazu, die vier Interaktionsarten eines Entwicklers zu kennzeichnen.

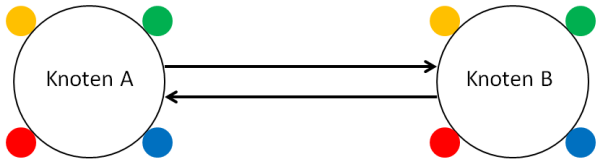


Abbildung 3.1: Konzept eines erweiterten Knoten im Netzwerk

Durch den Entwickler-Netzwerk können die Rolle des einzelnen Teammitglieds und die Verteilung der gesamten Interaktivität und Beteiligung innerhalb eines Teams charakterisiert werden. Die folgenden Tabelle 3.3 legt die im Netzwerk dargestellten Metriken dar.

Visualisierung	**Bedeutung**	**Metrik**
Anzahl von Knoten	Teamgröße	M1.4
Anzahl von Knoten, Kantendichte	Beschreibt, wie intensiv alle Entwickler mit einander interagieren	M1.1, M1.2
Kantendichte, -länge	Wie oft interagieren sich die 2 Entwickler im Projekt?	M2.2
Größe von Knoten	Wie oft interagiert Entwickler mit anderen Personen?	M2.1
Größe von kleinen Knoten	Wie viel interagiert Entwickler durch bestimmten Interaktionstypen?	M1.3

Tabelle 3.3: Die Metriken und die dazugehörigen Visualisierung

3.4.2 Flächendiagramm und Balkendiagramm

Flächendiagramm

Flächendiagramm wird verwendet, um die kumulierte Summe der Attachment-Abrufen von Team im Laufe der Iterationen darzustellen. Die Datenreihe jeder Iteration werden in verschiedenen Farbe gefärbt. Somit können die Anzahl von Abrufen jedes Sprints verglichen werden und kann man feststellen, wann die Dokumente (Anhänge) aktiv im Projekt benutzt wurden, bzw. die unhilfreichen Dokumente zu identifizieren. Die folgenden Metriken werden in Flächendiagramm aufgetragen, M3.1 wird in x-Achse dargestellt und M3.3 in y-Achse.

Balkendiagramm

Während Flächendiagramm die Datei-Abrufe vom ganzen Team beobachtet, stellt Balkendiagramm die Häufigkeit der Abrufen einer bestimmten Dokumenten dar. Auf der y-Achse kann man die Information über Dokumente (z.B Name, Erweiterung) abtragen, auf der anderen Achse die Anzahl der Zugriff. Durch die Länge der Balken kann der Benutzer die Häufigkeiten schätzen und vergleichen. Dieses Diagramm ist gewählt, um die Metriken M3.4, M3.5, M3.1 und M3.2 zu visualisieren.

3.5 Benutzeroberfläche

Da es sich bei dem Interaction Revealer um eine JIRA-Weberweiterung bezieht, welches häufig für die Analyse und Evaluation des Team hinsichtlich der Aktivitäten in JIRA genutzt werden soll, wird auf eine ansprechende und einfache Benutzeroberfläche einen besonderen Wert gelegt. Zwei separate Seiten werden für zwei unterschiedlichen Benutzertypen, Administrator und authentifizierter Benutzer erstellt.

3.5.1 Administrator-Seite

Für das User Story 1 soll eine Admin-Seite entwickelt werden, die ausschließlich vom JIRA-Administrator zugänglich ist. Hierzu werden zwei Schaltflächen und eine Dropdown-Liste von Sprint in der Seite angelegt. Eine Schaltfläche heißt "Alle Interaktionsdaten löschen", der dient dazu, dass alle gespeicherten analysierten Dateninhalten gelöscht werden. Der andere Button ist "Interaktionsdaten analysieren". Beim Betätigen dieses Buttons startet die Analyse der Dateninteraktion bezüglich des in Dropdown-Liste gewählten Sprints, wobei die neuen Daten des ausgewählten Sprints analysiert und gespeichert werden. Des Weiteren bietet die Admin-Seite ein informatives Feedback, nachdem die Daten erfolgreich bzw. fehlgeschlagen

analysiert wurden. Somit der Nutzer die Kontrolle haben, was gerade passiert ist.

3.5.2 Hauptseite

Die Hauptseite können von jedem Teammitglied zugegriffen werden, wo die Interaktionsdaten abgerufen und in den Grafiken dargestellt werden. Im Abschnitt 3.4 wurden drei Visualisierungskonzepte gewählt, welche in zwei Bereichen aufgetragen werden (siehe Abbildung 3.2). Weiterhin soll die Visualisierung so anpassungsfähig, dass der Anwender sie nach der Größe des Webbrowserfenster vergrößern und verkleinern kann.

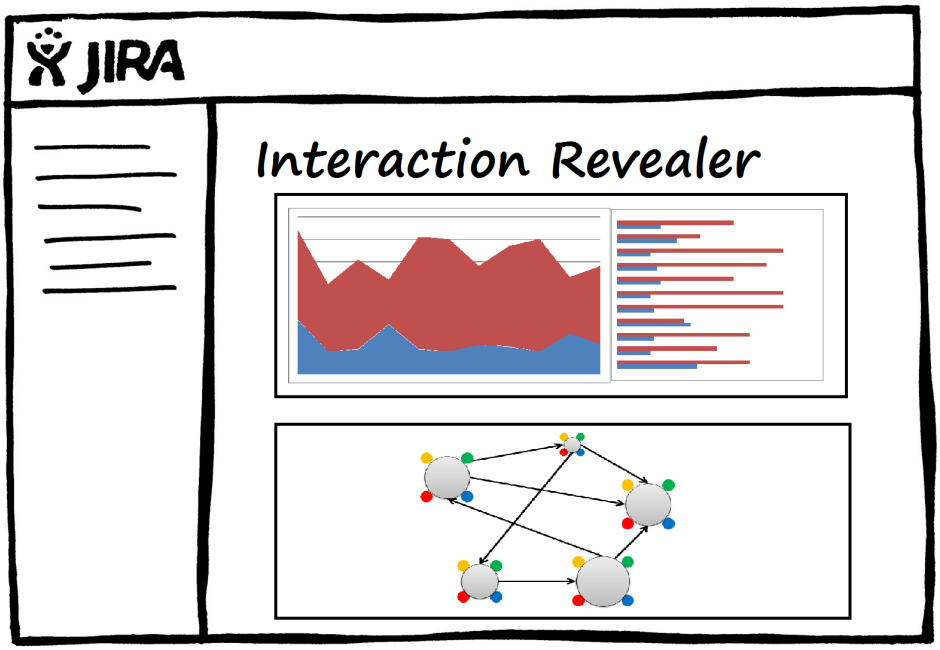


Abbildung 3.2: Mockup des Interaction Revealers

Das Flächendiagramm und Balkendiagramm werden in dem oberen Bereich zusammengefasst, der unteren Bereich ist für den Entwickler-Netzwerk reserviert.

3.6 Datenbankkonzept

Nach der Identifizierung und Erfassung der relevanten Metriken der Interaktionen, sollen diese Daten gespeichert, also an der JIRA-Datenbank ausgelagert. Dies ermöglicht, die Analyse und der Abruf der Daten getrennt zu halten. Die Grafik 4.6 verdeutlicht den Einsatz der Datenbank im Programm. Wenn der Administrator Analysephase startet, liest die Datenbank

die Interaktionsdaten aus dem JIRA System und speichert diese ab. Später wenn der Nutzer diese Daten aus der Datenbank anfordert, werden sie zur Verfügung gestellt, welche zur Visualisierung dienen.

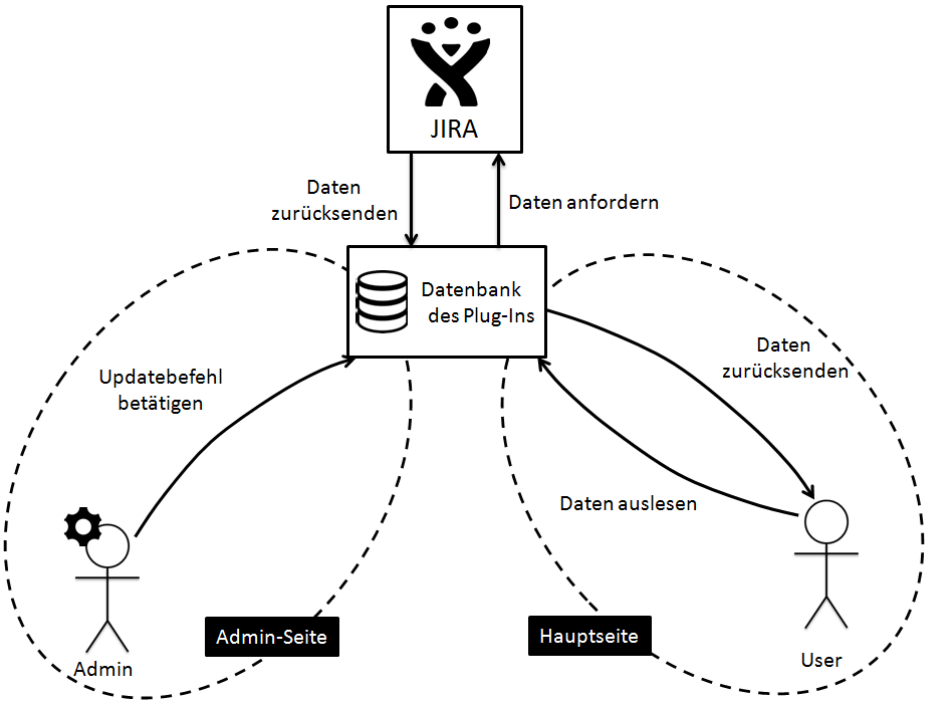


Abbildung 3.3: Einsatz von Datenbank in der Anwendung

Kapitel 4

Implementierung

In diesem Kapitel wird die Implementierung des Programm Interaction Revealers erläutert. Das Kapitel ist in sechs Teile gegliedert. Erstens werden die in dieser Arbeit eingesetzten Technologien beschrieben. Anschließend wird die Grundstruktur der Anwendung erläutert. Der dritte Teil besteht darin, die gewünschten Interaktionsdaten im JIRA abzurufen. Um dies zu erreichen, wird die REST APIs benutzt. Als Nächstes werden diese Daten in JSON-Format umgewandelt, welche von der Webanwendung lesbar sind. An dieser Stelle werden diese Daten zusätzlich auch in der Datenbank gesendet und abgespeichert. Sobald die Daten vorbereitet sind, können wir mit der Visualisierung mit Hilfe von den C3JS und D3JS Bibliotheken beginnen. Der letzte Teil des Kapitels sind die Probleme und Änderungen, die während der Implementierung aufgetreten haben.

4.1 Eingesetzte Technologien

In diesem Abschnitt werden die wichtigsten Technologien erörtert, welche für die Umsetzung dieser Arbeit erforderlich sind.

4.1.1 REST

REST ist eine Programmierschnittstelle, die den Austausch von Informationen zwischen Client und Server ermöglicht. Die wesentlichen Merkmale von REST lauten: [17]

- Client Server: Die Implementierung von REST folgt das Client-Server-Struktur Prinzip. Dabei stellt der Server den Dienst zur Verfügung, der bei Bedarf vom Client angefragt werden kann.
- Zustandslosigkeit: Die Kommunikation zwischen Client und Server verläuft zustandslos. Das bedeutet, dass alle Anfragen vom Client zum Server alle relevanten Informationen enthalten müssen. Server hat

kein Vorwissen über den Kontext von Client bzw über vorangegangene Anfragen.

- Caching: Client kann Antworten des Servers im Cache speichern und bei weiteren Anfragen auf diese zurückgreifen.
- Vereinheitliche Schnittstelle: REST stellt einen einheitlichen Satz von Standardmethoden, um auf eine Ressource zuzugreifen, wie zum Beispiel Standard-HTTP-Methoden, GET, POST, PUT, usw.
- Adressierbarkeit: Jede Ressource wird durch ein Unique Ressource Identifier (URI) eindeutig identifiziert.

Eine der Haupteigenschaft von REST ist die Nutzung von HTTP-Methode, durch die Client mit Server kommuniziert. Die typischen HTTP-Operationen und ihre Funktionen werden in Tabelle 4.1 aufgelistet [16].

HTTP-Methode	**Beschreibung**
GET	Liefert die angeforderte Ressource zurück.
PUT	Legt die angegebene Ressource an. Falls die Ressource existiert, ersetzt diese mit der anderen.
POST	Fügt die neue Ressource hinzu und liefert diese zurück.
DELETE	Löscht die Ressource.

Tabelle 4.1: Typische HTTP-Methode

4.1.2 JSON

JavaScript Object Notation (JSON) ist ein Textformat für die Serialisierung strukturierter Daten [7]. Vor allem im Umfeld von Web-Services, insbesondere bei REST ist JSON eingesetzt, um Daten auszutauschen. JSON bietet den Vorteil, dass es von Menschen lesbar und von Computer einfach übersetzt ist. JSON ist direkt in Javascript unterstützt und eignet sich am besten für JavaScript-Anwendungen. Außerdem kann diese Notation beliebige Strukturen darstellen, darunter primitive Type, Objekte, Sammlungen, Liste von Werten. Abbildung 4.1 beschreibt ein Beispiel, in dem JSON verwendet wird, um einen Vornamen und Nachnamen zu codieren.

```
{
        firstname: "Thomas",
        lastname: "Müller"
}
```

Abbildung 4.1: Ein einfache JSON-Struktur beschreibt die Kodierung des Namen

4.1.3 XML

Neben JSON wird XML auch für den Datenaustausch zwischen Systemen eingesetzt. XML ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten im Format einer Textdatei, die sowohl von Menschen als auch von Maschinen lesbar ist [2]. So wie JSON, ist XML ein textbasiertes Dateiformat, aber die Struktur ist anders. Ein Beispiel für ein in XML codiertes Objekt ist in Abbildung 4.2 zu finden.

```
<name>
        <first>Thomas<first>
        <last>Müller<last>
<name>
```

Abbildung 4.2: Eine hierarchische XML-Struktur für einen Namen

4.1.4 D3.JS

D3.JS (Data-Driven Documents) ist eine JavaScript-Bibliothek zur Erleichterung von Erstellung dynamischer und interaktiver Datenvisualisierung in Webbrowsern. D3.JS ermöglicht die HTML-, vektorgrafische (SVG) oder Canvas-Elemente im Kontext eines Datensatzes zu manipulieren. Diese Elemente können flexibel benutzt sein, wie zum Beispiel entsprechend dem Inhalt des Datensatzes eingefügt, entfernt oder bearbeitet sein. D3.JS hilft primär bei der Datenexploration, insbesondere es unterstützt die Kontrolle über die Veranschaulichung der Daten und das Hinzufügen von Interaktivität. D3.JS Hauptmerkmale sind Folgendes:

- Flexibilität
- Schnelligkeit und Bedienbarkeit
- Unterstützung von großen Datensatz
- Gute Dokumentation
- Deklarative Programmierung
- Zahlreiche Diagrammvorlagen

4.1.5 C3.JS

C3.JS ist eine wiederverwendbare auf D3 basierte Diagrammbibliothek, die eine tiefere Integration von Diagrammen in Webanwendungen ermöglicht. Der Hauptunterschied zwischen D3.JS und C3.JS liegt bei der Flexibilität und Interaktivität. C3.JS gibt Entwicklern weniger Flexibilität beim Modifizieren der Grafik als D3.JS. Jedoch biete C3.JS die Diagramme mit

guten voreingestellt Interaktivitäten. Die Hauptvorteile von C3.JS sind unten gelistet:

- Kleiner Aufwand beim Programmieren: C3.JS macht es einfach, D3-basierte Diagramme zu generieren.
- Voreingestellte Interaktivität

4.2 Interner Aufbau

Grundsätzlich ist die JIRA-Erweiterung eine Webanwendung, die mit in JIRA integriert wird. Für die Webentwicklung wird die Arbeit in zwei konkreten Teilbereichen klassifiziert, also Frontend- und Backend-Entwicklung. Backend umfasst den Teil eines Systems, der sich mit der Datenverarbeitung im Hintergrund beschäftigt. In dem Kontext des Interaction Revealers spiegelt sich das Backend die Datenerfassung und -speicherung in einer externen Datenbank. Als Frontend werden die Benutzeroberflächen bezeichnet, welche die Daten und Prozesse im Backend zugreifen und darstellen. Dies entspricht genau, was das zu realisierende Programm tun soll, nämlich die Visualisierung von Interaktionsdaten. Deswegen ist es sinnvoll, das Interaction Revealer in zwei separierte Schichten zu implementieren. Dadurch wird die Modularität des Programms gesichert. Abbildung 4.3 stellt die Zusammenhänge der Softwarekomponenten in einer vereinfachten Ansicht dar.

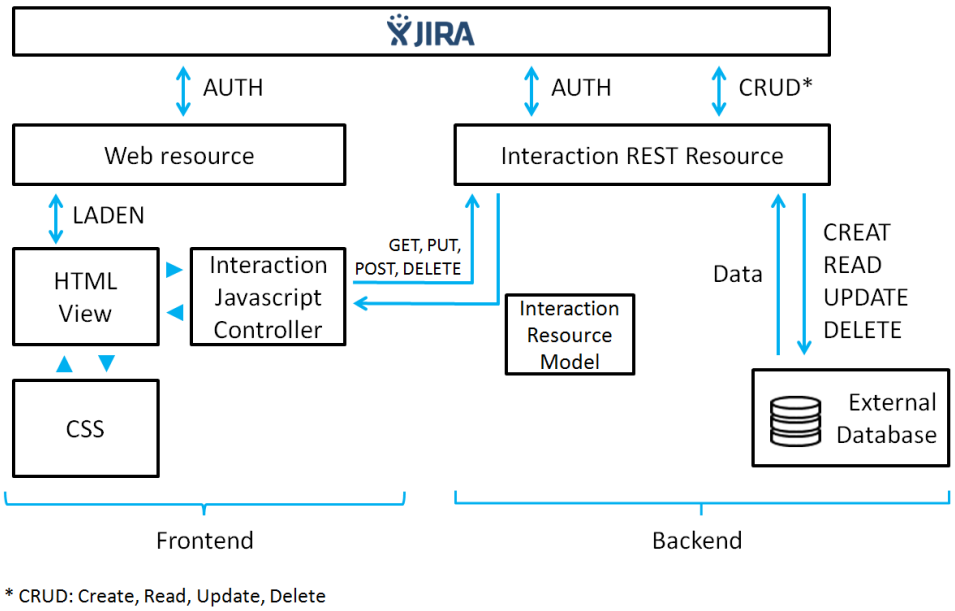


Abbildung 4.3: Vereinfachte Struktur des Interaction Revealers

4.2.1 Backend Struktur

Die Struktur vom Backend besteht aus zwei Hauptelementen: Interaction REST Resource zur Analyse der Daten und die externe Datenbank, wo solche Daten gespeichert werden. Die beiden Komponenten werden in Java programmiert, da JIRA dafür die Java-APIs anbietet.

Interaction REST Resource

REST ist eine Webprogrammierschnitstelle, die einen Ansatz für die Kommunikation zwischen Client und Server beschreibt. Auf der REST-basierte Schnittstellen setzt JIRA. REST stellt eine Vielzahl an Standard-Funktionen zur Verfügung, um Daten aus dem JIRA-System zu lesen, zu schreiben oder JIRA zu konfigurieren. Außerdem hat man mit REST die Möglichkeit, eigene REST-Module für speziellen Zweck zu entwickeln.

Interaction REST Resource (IRR) ist ein einfacher Controller im Model-View-Controller Pattern, der auf bestimmte HTTP-Anforderungen (GET, PUT, POST, DELETE) mit einer bestimmten Antwort reagiert. Die IRR verfügt über die 4 Operationen zur Erfassung der JIRA-Daten: Erstellen, Lesen, Aktualisieren und Löschen. Jedoch muss der Nutzer authentifiziert werden, um auf die internen Operationen Hand zu legen. Dafür gibt es einen Lösungsansatz, Nutzung von Cookie. Cookie ist Daten, die beim Anmelden in JIRA die Benutzerinformation speichern. Cookie wird von Interaction Javascript Controller an IRR gesendet und zur Authentifizierung verwendet.

Externe Datenbank

Die von IRR erfassten Daten sollen dann in der JIRA Datenbank gespeichert werden. Je nachdem sich die Interaktionsdaten mit der Zeit ändern, kann sich die Interaction REST Resource die Datenbank durch die CRUD Operationen anpassen. Der Vorteil der Verwendung einer Datenbank besteht in der strukturellen Datenverwaltung, die Datenerfassung und -verarbeitung systematisiert und vereinfacht.

Interaction Resource Model

Um die Kommunikation zwischen Interaction REST Resource und Interaction JavaScript Controller zu gelingen, soll ein Interaction Resource Model verwendet werden. Interaction Resource Model ist ein Java-Model, das von Interaction REST Resource von Java-Objekt zu einem für Javascript lesbaren XML/JSON Datentyp konvertiert wird. Die Interaktion zwischen Interaction REST Resource und Interaction Javascript Controller läuft so ab, Interaction Javascript Controller sendet Interaction REST Resource eine HTTP-Request, Interaction REST Resource antwortet mit einem Interaction Resource Model als JSON-Representation.

4.2.2 Frontend

Frontend wird mit den Programmiersprachen HTML, CSS und JavaScript implementiert. Das Frontend des Programm basiert auf folgenden Komponenten:

- Die **Web resource** ist ein Webserver, über den die statischen Ressourcen wie zum Beispiel Javascript, CSS-Stylesheet und HTML geladen werden können.
- Das **HTML-View** ist die Vorlage, die grundlegende HTML-Komponenten für den JavaScript-Kontroller bereitstellt.
- Die **CSS-Datei** beschreibt, wie die HTML-Komponenten im HTML-View aussehen sollen.
- **Interaction Javascript Controller** ermöglicht die Interaktion zwischen dem HTML-View und die Interaction REST Resource. Dabei bekommt Interaction JavaScript Controller die Daten von Interaction REST Resource und stellt diese in HTML-View dar.

4.3 Abrufen der Interaktionsdaten von JIRA

Der erste grundlegende Schritt der Implementierung ist das Abrufen der Entwickler-Interaktionsdaten. Es gibt vier relevante Interaktionsarten in JIRA: Comment, Watcher, Assignment und Download. Obwohl die vier Arten gleiche Struktur haben, sind die Methode zur Datenerfassung jeweils unterschiedlich.

4.3.1 Abrufen von Kommentaren und Assignments

Um die interne Daten von JIRA abzurufen, bietet JIRA die Standard-REST-APIs an, die den Zugriff auf JIRA-Ressourcen über URI-Pfade ermöglichen. Die URIs für REST API haben die folgende Struktur:

```
http://host:port/context/rest/api-name/api-version/resource-name
```

Indem man JQL Search in REST-URI integriert, kann man alle Issues eines spezifischen Projekt und deren dazugehörigen Informationen suchen.

```
http://host:port/jira/rest/api/latest/search?jql=project=<PROJECT_ID>
```

Interaction REST Resource sendet eine HTTP-GET auf den obigen URI an den JIRA-Server und bekommt Daten als JSON/XML-Repräsentation. Diese Daten werden anschließend analysiert und die wichtigen Informationen werden aus diese Daten extrahiert. Mit dem Verfahren können die Daten von Kommentar und Assignment erfasst werden. Deren Ablauf wird in Abbildung 4.4 schematisch dargestellt.

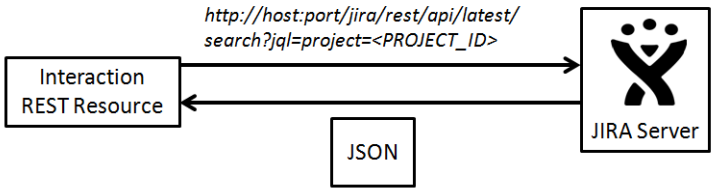


Abbildung 4.4: Alle Issue-Informationen abfragen

4.3.2 Abfangen von Herunterladen der Daten

Da die Information über die Anzahl von Downloads nicht über REST-APIs verfügbar sind, muss eine andere Methode verwenden, nämlich eine ServletFilter Modul. Bei der Beobachtung von Attachment in jedem Issue kann festgestellt werden, dass die Attachment-URIs immer gleichen Form haben, und zwar:

```
http://host:port/jira/secure/attachment/<ISSUE_ID>/<FILE_NAME>
```

Die Idee zum Abfangen von Download ist ein Listener zu implementieren, das obiges URL-Muster überwacht und beobachtet. Immer wenn ein Nutzer einen Zugriff auf dem URI macht, welcher dieses URL-Muster entspricht, wird der Abruf von Attachment erkannt. Dafür gibt es ein JIRA-Modul, das diese Überwachung von URI ermöglicht, ServletFilter. Die Funktionsweise von ServletFilter wird in Abbildung 4.5 dargestellt.

4.3.3 Abrufen von Watcher Daten

Durch REST APIs können auch Watcher-Daten abgerufen werden. Trotzdem ist durch REST nicht angegeben, in welchem Sprint der Nutzer ein Ticket beobachtet. Daher ist es nicht möglich, durch REST das Sprint einer Beobachtung-Aktivität zu bestimmen. Eine mögliche Lösung ist die Implementierung eines Watcher-Listeners für Issue. Dieser Listener beobachtet die Aktionen des Benutzers und informiert die Applikation über die anfallenden Beobachtung-Ereignisse.

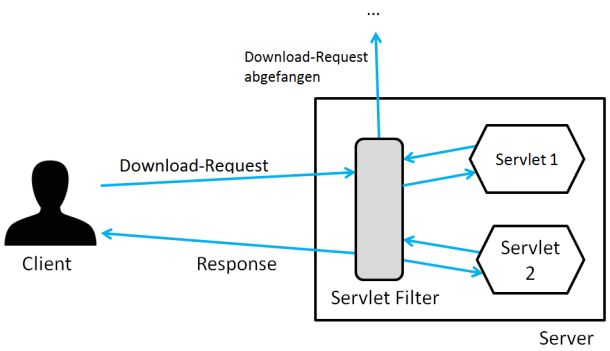


Abbildung 4.5: Zugrif auf Attachment mittels Servlet Filters abfangen

4.4 Implementierung der Datenbank

Nachdem der Prozess der Datenerfassung beendet ist, sollen die Daten in der Datenbank abgespeichert werden, damit sie im nächsten Schritt zur Visualisierung wiederverwendet werden. In diesem Abschnitt wird der Entwurf der Datenbank Schritt für Schritt beschrieben.

4.4.1 Informationsbedarfanalyse an die Datenbank

Um die Daten effizient gespeichert zu werden, soll die Interaktion-Datenbank so gut wie möglich die Realwelt spiegeln. Dafür wird ein Ausdrucksmittel gebraucht, mit denen der Realausschnitt so vereinfacht dargestellt werden kann. Im Folgenden werden die wichtigen Informationen informell natürlich-sprachlich beschrieben, die durch das Datenbanksystem verwaltet werden sollen.

- Ein Softwareprojekt wird durch einen ID identifiziert.
- In jedem Projekt befinden sich mehrere Sprints, die einen eindeutigen ID haben.
- Ein Benutzer wird durch seinen ID eindeutig bestimmt.
- Nutzer können von anderen Benutzern durch Issues zugewiesen werden.
- Jedes Issue hat einen eindeutigen ID, kann zu einem Sprint zugeordnet werden und darf nur zu einem einzigen Projekt gehören. Hier wird der Kontext betrachtet, dass kein Entwickler projektübergreifend arbeitet.
- In jedem Issue können Dateien von einem Nutzer angehängt werden. Jeder Benutzer kann diese Dateien herunterladen.
- Nutzer können Issues beobachten. Nach der Definition von Watch (Absch. 2.1.3) beobachtet dieser Nutzer auch den Assignee (Issue-Bearbeiter) des Issues.

- Zu jedem Issue können Nutzer Kommentar schreiben. Wenn in Kommentar kein anderer User erwähnt wird, wird so angenommen, dass die Kommentar an allen Nutzer gerichtet wird.

- Da die Interaktionen in jedem Sprint analysiert werden, sind die durch den Sprint ID identifiziert.

4.4.2 Entity-Relationship-Schema der Datenbank

Aus den gesammelten Informationen im Abschnitt 4.4.1 lässt sich die Interaktionsdaten durch den Entity-Relationship-Schema (siehe Abbildung 4.6) modellieren.

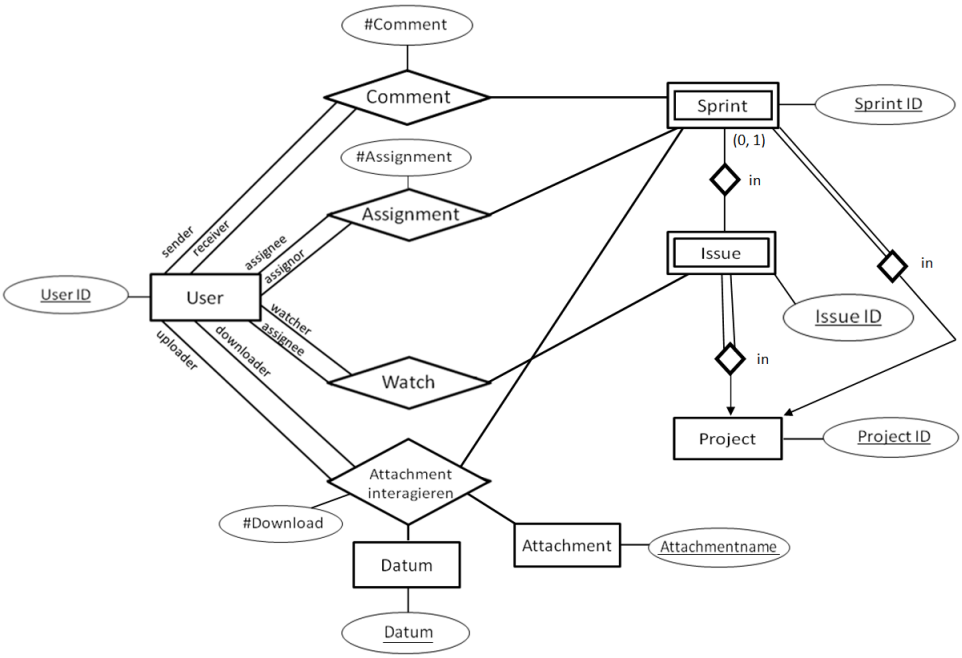


Abbildung 4.6: ER-Diagramm der Datenbank

4.4.3 Relationales Datenbankschema

In dem nächsten Schritt wird der Entity-Relationship-Schemata in äquivalente Relationale Datenbank-Schema (siehe Tabelle 4.2) transformiert, welche den in JIRA Datenbank gespeicherten Tabellen entspricht.

USER	(User_ID)
PROJECT	(Project_ID)
SPRINT	(Sprint_ID, Project_ID → PROJECT)
ISSUE	(Issue_ID, Project_ID → PROJECT, (Sprint_ID^{null}, Project_ID^{null}) → PROJECT
ATTACHMENT	(Filename, Downloader → USER, Uploader → USER, Download_Date, #Download, (Sprint_ID, Project_ID) → PROJECT)
COMMENT	(Sender → USER, Receiver → USER, #Comment, (Sprint_ID, Project_ID) → PROJECT)
WATCHER	(Watcher → USER, Assignee → USER, (Issue_ID, Project_ID) → ISSUE, (Sprint_ID, Project_ID) → PROJECT)
ASSIGNMENT	(Assignor → USER, Assignee → USER, #Assignment, (Sprint_ID, Project_ID) → PROJECT)
OVERALL_INTERACTIONS	(User → USER, #Assignment, #Comment, #Download, #Watcher, (Sprint_ID, Project_ID) → PROJECT)

Tabelle 4.2: Relationales DB-Schemata

4.5 Implementierung der Visualisierung

In diesem Abschnitt wird die Visualisierung der Entwickler-Interaktionen sowie die Funktionsweise präsentiert.

4.5.1 Attachment Tracker

Der *Attachment Tracker* besteht aus drei Diagramme, ein Flächendiagramm und zwei Balkendiagramme. Abbildung 4.7zeigt ein Beispielvisualisierung von Attachment Tracker. In dem Flächendiagramm kann man mit der Legende interagieren, in dem man auf sie klickt. Beim Drücken einer Schaltfläche der Legende wird die Daten der entsprechenden Sprint in dem Flächendiagramm ausgeblendet. So werden auch die Daten des gewählten Sprints in den Balkendiagramm rausgefiltert. Somit hat der Nutzer die Möglichkeit, das einzelne Sprint zu untersuchen.

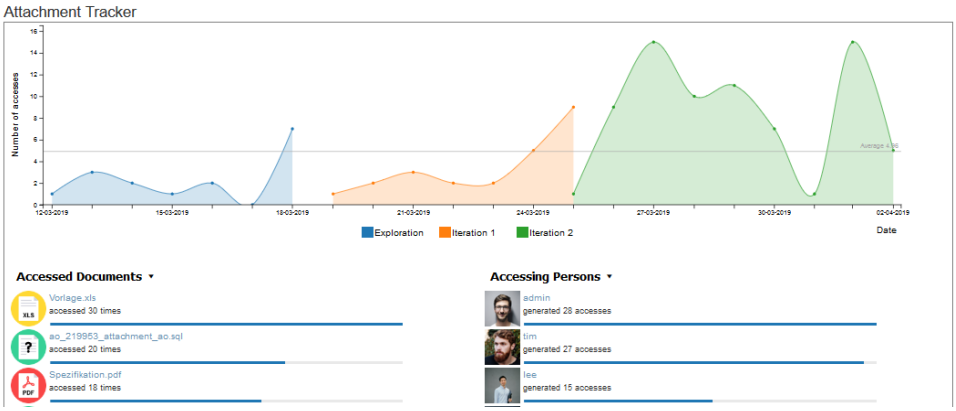


Abbildung 4.7: Visualisierung der Interaktionen bezüglich von Attachments

Wenn der Benutzer den Mauszeiger über einen Balken bewegt, erhält

er zusätzliche Informationen, die in einem Tooltip dargestellt sind (siehe Abbildung 4.10)

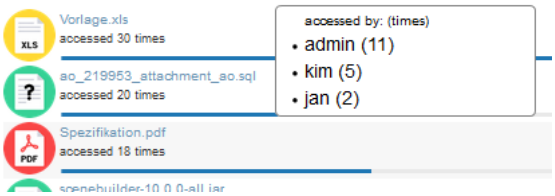


Abbildung 4.8: Detaillierte Informationen in einem Tooltip

Außerdem kann man bei Interesse auf der Seite des Issues, wo die Dateien hochgeladen wurde, gehen, in dem man auf dem Namen dieser Datei klickt.

4.5.2 Entwickler-Interaktion-Netzwerk

Abbildung 4.9 stellt ein Beispiel für den Entwickler-Netzwerk dar. Unten befindet sich der Sprint-Slider, womit man die Dateninteraktionen Sprint für Sprint zeigen lässt. Tooltips werden verwendet,um eine Menge der zusätzlichen Informationen abzudecken. Dadurch erreicht man die maximalen Informationsgehalte, ohne die Übersichtlichkeit des Netzwerks zu verletzen. Immer wenn der Nutzer den Mauszeiger über eine Schaltfläche bewegt, wird diese Schaltfläche hervorgehoben und dazu wird ein Tooltip mit relevanten Informationen gezeigt.

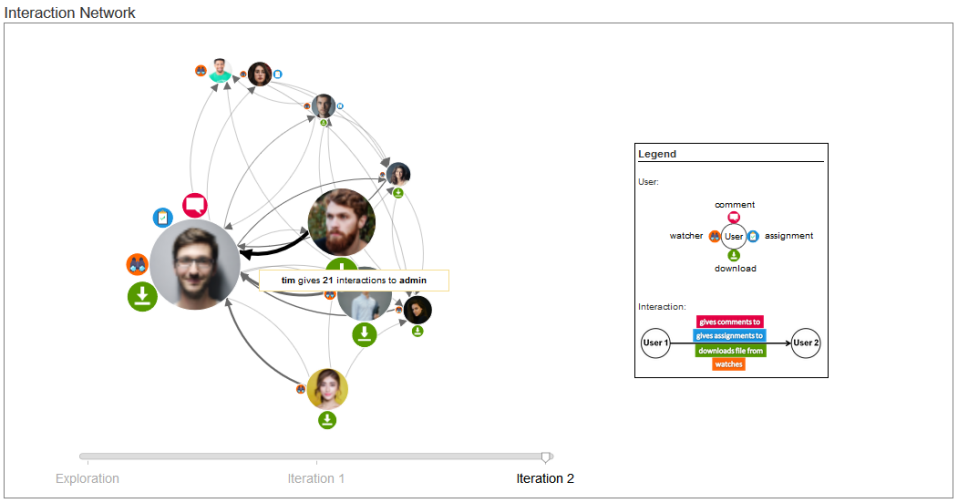


Abbildung 4.9: Beispiel eines Entwickler-Netzwerks

Außerdem kann der Nutzer freiwillig die Hauptknoten durch Drag & Drop bewegen. Die Nebenknoten darf dagegen nicht flexibel positioniert werden, weil sie immer fest an den Hauptknoten verbunden sind. Trotzdem kann die

Nebenknoten um den Hauptknoten herum bewegen. Diese Funktion ist eine Lösung für den Fall, wenn die Nebenknoten die Pfeile abdecken. Dadurch hat der Benutzer viel Flexibilität, den Entwickler-Netzwerk nach Wunsch einzustellen.

Falls der Nutzer keine Vorkenntnisse über die Interaktionsdaten in JIRA, kann ihm die Legende helfen (siehe Abbildung 4.10).

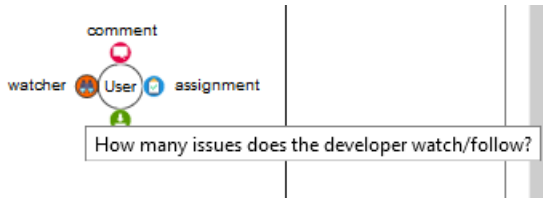


Abbildung 4.10: Kurze Erklärung der Definition von Watcher im Tooltip

Zusammenfassend wird es bei der Visualisierung versucht, die notwendigen Daten darzustellen, die zusätzlichen Informationen abzudecken und sie bei Bedarf zu zeigen. Somit ist die Übersichtlichkeit der Grafik gehalten und der entscheidende Mehrwert an Informationen ist erreicht.

4.6 Probleme und Änderungen

Während der Implementierung des Interaction Revealers trat ein Problem auf, die gelöst werden musste. Das erste Problem ist die Implementierung der Erfassung von Downloads. Die Anzahl des Abruf von Dateien stellt die JIRA über REST API nicht zur Verfügung. Außerdem bietet JIRA keinen Attachment-Listener. Auf dem Internet wurde ein Lösungsansatz vorgestellt, wobei JIRA User Logging benutzt wird, um die Aktivität von Nutzer zu verfolgen. Wenn User Logging aktiviert ist, dann wird jede Aktivität von Nutzer in einem txt-Datei abgespeichert, darunter die Download-Aktivität. Dieser Nachteil des Ansatzes besteht darin, dass die heruntergeladene Datei in Browse-Cache temporär gespeichert wird. Beim mehrmals Herunterladen wird die Datei aus der Browse-Cache abgerufen, daher ist keine Verbindung mit JIRA Server vorhanden. Das führt dazu, dass das Download von Dateien nur einmal gezählt wird, solange diese Dateien noch in der Browse-Cache gelagert sind.

Kapitel 5

Evaluation

5.1 Erklärung und Durchführung

Die Evaluation des Interaction Revealers wurde nach der Implementierung durchgeführt. Die Intention dieser Studie bestand darin, einen Vergleich zwischen einer konventionellen Analyse der Interaktionsdaten mit der JIRA-Standard-Funktion und der Vorgehensweise mittels Interaction Revealers zu ziehen. Dabei ging es um eine vollständig geführte Usability-Studie, bei der beide Verfahren in zwei ähnlichen Szenarios durchgeführt werden sollten.

Der Evaluationsbogen setzt sich aus drei Teilbereichen zusammen: die Abfrage der persönlichen Informationen, die konventionelle Methode zur Datenanalyse und die entsprechende Vorgehensweise mit dem Interaction Revealer (siehe Anhang A). In jedem Verfahren wird zuerst die Beschreibung des Szenario vorgestellt und danach schließt sich fünf Fragen an, in denen der Befragte angefordert wird, die in dem Szenario auftretenden Interaktionsdaten zu zählen. Für die Erhebung werden zwei Evaluations-Datensätze erstellt, die gleiche Struktur haben aber nicht identisch sind. Somit kann ein präziser und objektiver quantitativer Vergleich der Verfahren sichergestellt.

Am Ende jedes Vorgehens wird der Proband seine subjektive Rückmeldung über die Gebrauchstauglichkeit dieses Vorgehens befragt. Dazu wird der Ansatz *System Usability Scale (SUS)* [4] verwendet. SUS ist ein einfacher Fragebogen mit zehn Fragen nach Likert-Skala. Sie bietet eine Formel zur Berechnung von SUS-Punkte, wodurch sich verschiedene Systeme vergleichen lassen.

Die Evaluation findet in dem Computerraum und InfoLounge der Leibniz Universität statt und wird mit dem Webbrowsern FireFox durchgeführt. Die Probanden werden nach einer kurzen Einführung mit dem System vertraut gemacht. Nach der Durchführung werden die Kritik, Anregung sowie Vorschlag zur Verbesserung gesammelt.

5.2 Auswertung

An der Evaluation nehmen 11 Informatik-Studierenden teil, die über Grundkenntnisse von JIRA als Voraussetzung für die Teilnahme verfügen. Das Alter der Testpersonen liegen zwischen 19 und 31 Jahren. Abbildung 5.1 stellt die ausgewertete Metriken der Evaluation dar.Das linke Säulendiagramm trägt den Unterschied in Effektivität zwischen der Nutzung konventioneller Methoden und der von Interaction Revealer auf. Das rechte Diagramm bildet die Vorkenntnisse der Probanden ab.

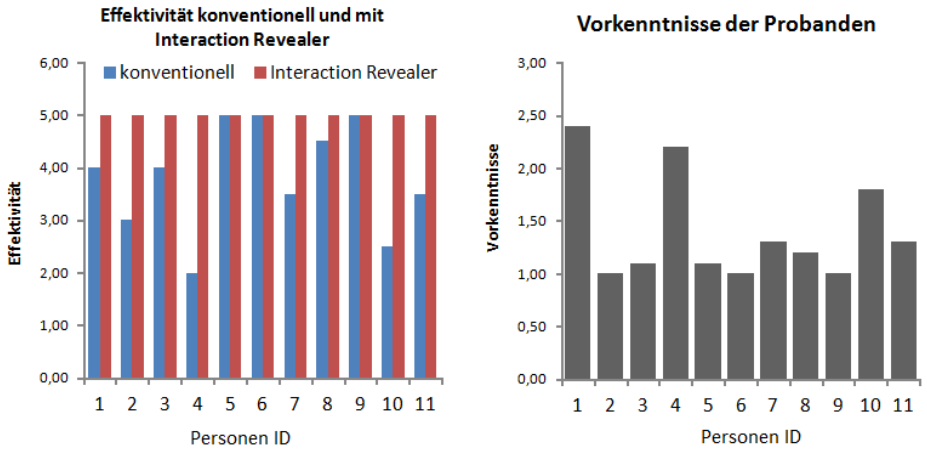


Abbildung 5.1: Graphische Darstellung der Messparameter (Effektivität, Vorkenntnisse)

Der Vorkenntnisskala interpretiert den Wert 3 als hervorragende Vorkenntnisse und 0 als keine Kenntnisse. Dieser Wert wird als Mittelwert aller Fähigkeiten der Testpersonen. Dabei wurden die Vorkenntnisse bezüglich Programmierkenntnisse, agile Softwareentwicklung, JIRA Software abgefragt. Außerdem zählt dazu auch die Kommunikationsaktivitäten in JIRA, die von Probanden in ihren Projekte aufgebracht haben, weil dieser Aspekt stark die Durchführung der konventionellen Methode beeinflussen kann. Effektivität wird anhand der Anzahl der richtig beantworteten Fragen gemessen. Der Vergleich der konventionellen Vorgehensweise mit der Nutzung des Interaction Revealers zeigt, dass der Interaction Revealer für alle Probanden immer maximal Informationsgehalt bringt und keine gute Vorkenntnisse voraussetzt. Währenddessen kann die konventionelle Methodik sehr wahrscheinlich zum Analysefehler führen, nur drei von elf Testpersonen können die maximale Informationen finden. Daraus lässt sich schließen, dass das Verfahren der Datenanalyse mittels Interaction Revealers als zuverlässig und fehlerfrei eingeschätzt ist.

Des Weiteren soll die Effizienz von beiden Verfahren verglichen werden.

In der Evaluation wurde gemessen, wie lange die Probanden brauchen, jede Methode fertigzustellen. Diese Metrik wird benutzt, um die Effizienz zu charakterisieren. Effizienz wird definiert als die Effektivität (Anzahl richtiger Lösung der Fragen) geteilt durch die Zeit. Somit lässt sich die Effizienz der beiden Verfahren in Abbildung 5.2 darstellen.

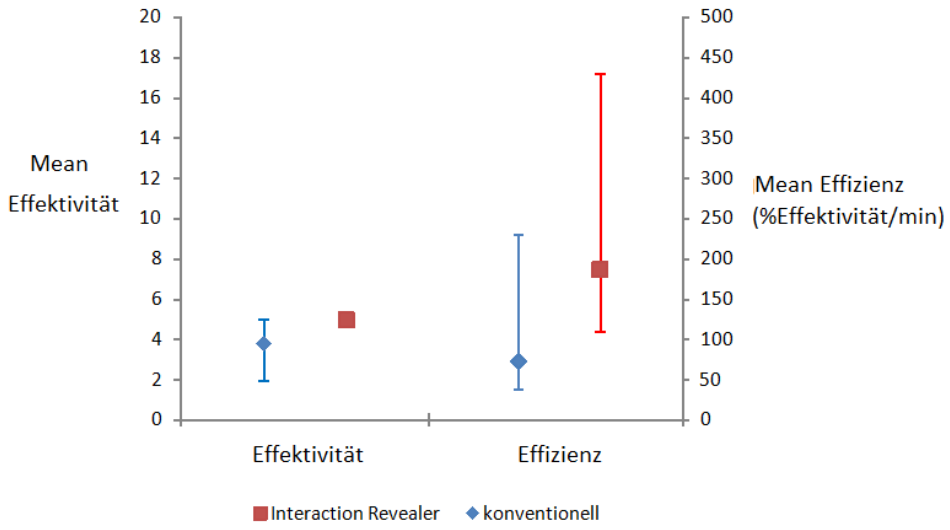


Abbildung 5.2: Darstellung von Mean Effektivität und Mean Effizienz

Aus dem Diagramm 5.2 kann entnommen werden, obwohl die Effektivität der beiden Verfahren nicht aussagekräftig ist, kann man mit dem Interaction Revealer dagegen deutlich mehr Arbeit leisten. Laut Diagramm konnten die Probanden bei der Nutzung des Programms durchschnittlich in einem Minuten 1.87 Fragen richtig beantworten, eine Frage mehr als bei dem konventionellen Prozess. Somit wird das Programm Interaction Revealer bei diesem Vergleich als effizienter eingeschätzt.

Schließlich wurde die Usability des Interaction Revealers anhand der SUS-Score bewertet. Dabei wurden die Probanden separat zwei SUS Fragebogen für die zwei Verfahren ausfüllen. Aus der SUS-Formel [3] lassen sich die durchschnittliche SUS-Punkte von beiden Verfahren berechnen. Abbildung 5.3 zeigt, dass Gebrauchstauglichkeit des Interaction Revealers als hervorragend bewertet wurde, während die meisten Probanden die Usability der konventionelle Methodik mit einer schlechten Bewertung einschätzen.

Die wichtigste Anforderung an das implementierte Programm besteht in der Benutzerfreundlichkeit und darin, dass die Visualisierung von Dateninteraktionen zwischen Entwicklern Informationsmerhwert bringt. Die Ergebnisse der Evaluation bestätigen, dass diese Anforderung vollständig

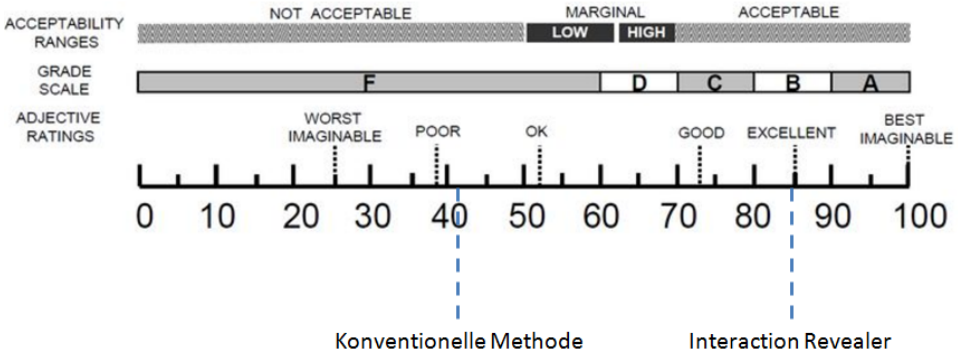


Abbildung 5.3: SUS-Punkte der beiden Verfahren (Brooke John [3])

erfüllt ist. Es lohnt sich, Interaction Revealer umzusetzen und zu nutzen.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

In der vorliegenden Arbeit sollte eine Anwendung konzeptioniert und implementiert werden, die sich zur Untersuchung von Entwickler-Interaktionen eignet. Dabei sollten die JIRA-Interaktivitäten von Teams erfasst und visualisiert werden. Um diese Anforderungen umzusetzen, wurden zu Beginn die relevanten Interaktionsdaten mittels Goal-Question-Metric Verfahrens identifiziert. Dann wurden diese Dateninteraktionen mit Hilfe von REST API zugegriffen und in der JIRA Datenbank gespeichert. Sobald die Daten vorbereitet sind, können sie in einer angemessenen Form visualisiert werden. Schließlich wurde die Software auf ihre Effektivität, Effizienz Gebrauchstauglichkeit hin anhand der Evaluation untersucht. Laut der Evaluation ist die Software im Vergleichen zu konventionellen Verfahren durchaus viel besser ist. Insbesondere wurde die Usability von Interaction Revealer von den Probanden sehr gut bewertet. Mit diesem Ergebnis konnte die Implementierung der adäquaten Visualisierung erfolgreich bestätigt werden.

6.2 Ausblick

Durch die modulare Struktur des Interaction Revealers ist eine Erweiterung jeglicher Funktionen möglich. Der erste Möglichkeit ist, weitere Metrik in dem Entwickler-Netzwerk zu integrieren, wie zum Beispiel Maverik Score [19]. Diese Metriken würden die Kommunikationsdifferenz zwischen Entwicklern besser hervorheben. Des Weiteren könnte die Legende des Flächendiagramm alternativ mit Check Box implementiert werden, da die meisten Probanden nicht wussten, dass die Legende interaktiv gestaltet wurde. Mit Check Box wäre die Filter-Funktion der Legende leichter zu erkennen.

Außerdem könnte eine Funktion entworfen werden, dass die Nebenknoten im Entwickler-Netzwerk gefiltert würden. Dies würde Möglichkeit bieten, jede einzelne Interaktionsart im Netzwerk zu untersuchen.

Anhang A

Evaluationsbogen

Evaluationsbogen: Usability Studie

Vergleich zweier Vorgehensweisen der Analyse von Entwickler-Interaktionen

Erklärung der Studie:

In dieser Studie werden zwei Verfahren zur Analyse der Dateninteraktion zwischen Entwicklern in JIRA verglichen. Dabei soll die Durchführung und Evaluation dieser Verfahren mit Ihrer Hilfe stattfinden.
Um die Kommunikationsverhalten von Entwickler-Team zu verstehen, erhebt und analysiert man die Interaktionsdaten, die es in JIRA gibt. Da hierfür verschiedene Verfahren existieren, wollen wir zwei vergleichen, die von Ihnen durgeführt werden sollen. Ihnen wird jeder Arbeitsschritt textuell vorgegeben, daher wird darum gebeten, die Anweisung genau zu lesen und anschließen durchführen. Mit der Bearbeitung dieser Evaluation willigen Sie ein, dass ihre Angaben in der Auswertung benutzt werden dürfen.

I. Persönliche Angaben

Bitte geben Sie hier einige Angaben zu Ihrer Person an.

Alter: ☐☐ Jahre alt

Studiengang bzw. Abschluss (Bitte mit Fachrichtung angeben)

.....

Wie würden Sie ihre Kenntnisse bezüglich der folgenden Angaben einschätzen?

	sehr geringe Kenntnisse	geringe Kenntnisse	gute Kenntnisse	sehr gute Kenntnisse
Programmierungskenntnisse	☐	☐	☐	☐
Agile Softwareentwicklung	☐	☐	☐	☐
JIRA Projektverwaltungsoftware	☐	☐	☐	☐

Wie würden Sie einschätzen, wie oft Sie folgende JIRA-Aktivitäten in Ihrer *Software-Projekt* Veranstaltung getätigt haben?

	selten	manchmal	oft	immer
Aufgaben zuweisen (Issue erstellen und koordinieren)	☐	☐	☐	☐
Kommentar schreiben	☐	☐	☐	☐
Herunterladen/Hochladen von Attachment	☐	☐	☐	☐
Issue verfolgen (Nutzer wird benachrichtigt, wenn Issue aktualisiert wurde)	☐	☐	☐	☐

II. Durchfühung der Analyse der Dateninteraktionen (konventionell)

Nun sollen Sie die Dateninteraktionen des Entwicklerteams in einem Sprint durch den konventionellen Weg untersuchen. Die Beschreibungen des Szenarios zeichnen sich durch die grauen Kästen aus. Bitte beanworten Sie die Fragen nach jedem Szenario.

Szenario 1: Iteration 2

Ihnen werden eine Liste von Issues gegeben. Ihre Aufage ist, alle zum Sprint *Iteration 2* gehörenden Issues durchzusuchen, im Detail anzuschauen und die folgenden Fragen zu beanworten.

1. Wie viele Issues (Assignments) hat der Nutzer *admin* erstellt?

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

2. Wie viele Anhänge (Attachments) wurden hochgeladen?

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

3. Wie viele Issues wurden vom Nutzer *eva* verfolgt (watched)?

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

4. Wie viele Kommentare wurden von *admin* an *bob* gerichtet?

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

5. Wer hat die meinsten Issues erstellt?

☐ admin ☐ bob ☐ sue ☐ eva ☐ jan ☐ lee

-

III. Durchfühung der Datenanalyse mit dem Interaction Revealer

Dieser Abschnitt führt Sie durch die Analyse der Entwickler-Interaktion mittels Interaction Revealers.

Szenario 2: Exploration Iteration

Ihnen werden jetzt die Visualisierungen von Dateninteraktionen vorgestellt. Schauen und interpretieren Sie die Grafiken an. Die Fragen werden in Bezug auf den Issues von dem Sprint *Exploration* gestellt.

1. Wer war ausgeschlossen vom Team? Das heißt, wer hat gar nicht mit anderen Entwicklern interagiert?

☐ admin ☐ bob ☐ kim ☐ lee ☐ jan ☐ ana

2. Wer interagiert am aktivsten?

☐ admin ☐ bob ☐ eva ☐ ana ☐ tim ☐ lee

3. Wie viele Interaktionen hat *tim* an *admin* gerichtet?

☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

4. Wie viele Kommentare schrieb *tim*?

☐ 6 ☐ 8 ☐ 12 ☐ 14 ☐ 16 ☐ 18

5. Welcher Anhang im *Exploration* Sprint wurden am meisten abgerufen?

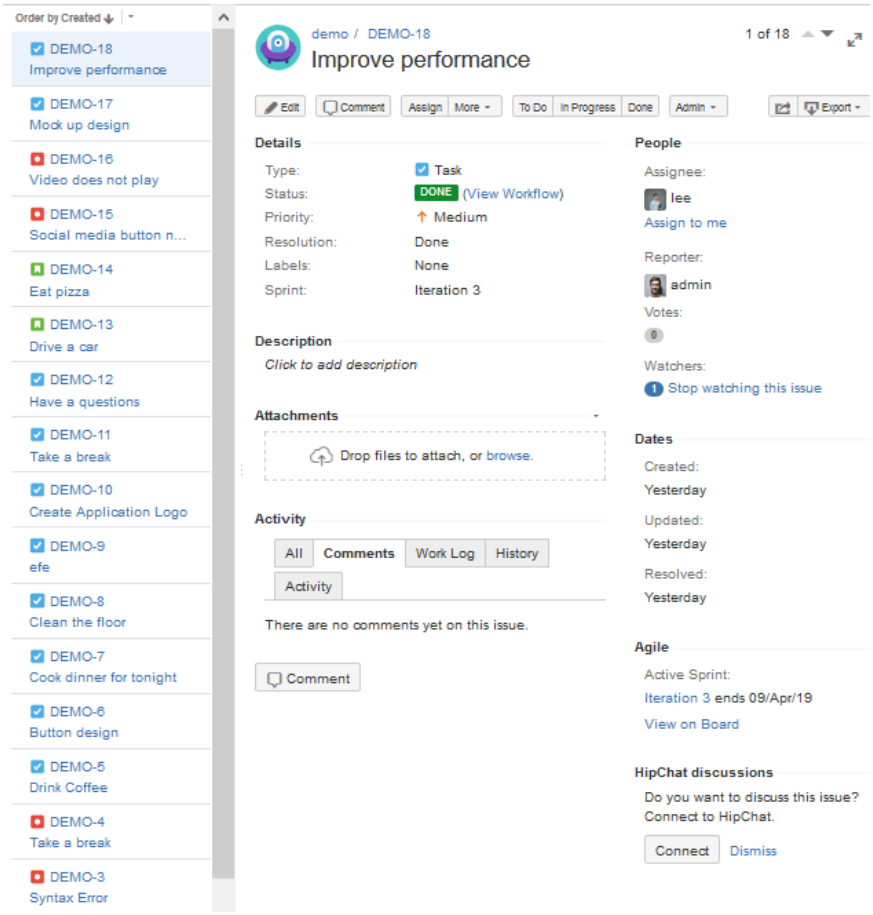
☐ test.png ☐ Vorlage.xls ☐ Spezifikation .pdf ☐ graph.zip ☐ Presentation .ppt ☐ Data.csv

IV. Bewertung der Gebrauchtauglichkeit

Nun sollen Sie Ihre Meinung über die Benutzerfreundlichkeit, die das Programm Interaction Revealer bzw. die konventionelle Methode auf Sie wirkte.

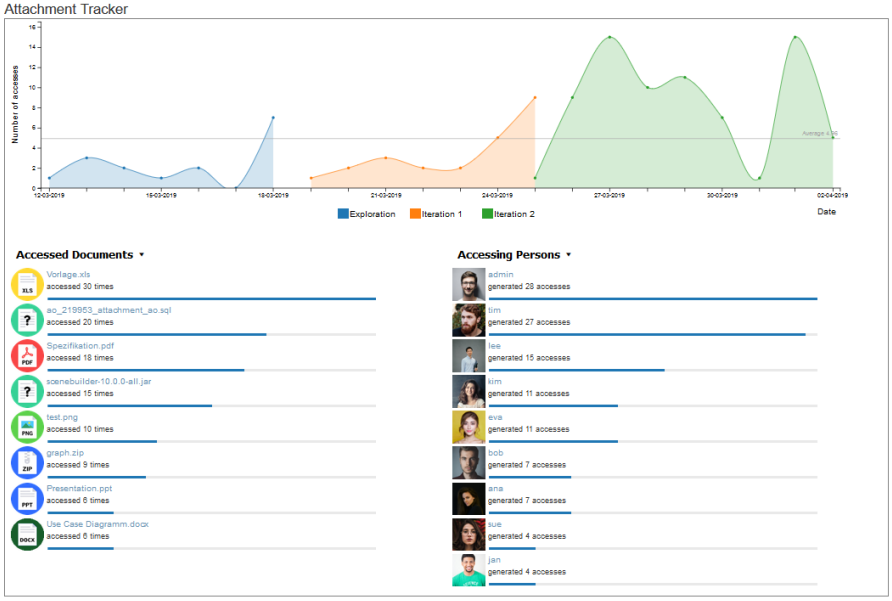
	Stimme überhaupt nicht zu	Stimme eher nicht zu	Weder Zustimmung noch Ablehnung	Stimme eher zu	Stimme völlig zu
1. Ich kann mir sehr gut vorstellen, das System regelmäßig zu nutzen.	1	2	3	4	5
2. Ich empfinde das System als unnötig komplex.	1	2	3	4	5
3. Ich empfinde das System als einfach zu nutzen.	1	2	3	4	5
4. Ich denke, dass ich technischen Support brauchen würde, um das System zu nutzen.	1	2	3	4	5
5. Ich finde, dass die verschiedenen Funktionen des Systems gut integriert sind.	1	2	3	4	5
6. Ich finde, dass es im System zu viele Inkonsistenzen gibt	1	2	3	4	5
7. Ich kann mir vorstellen, dass die meisten Leute das System schnell zu beherrschen lernen	1	2	3	4	5
8. Ich empfinde die Bedienung als sehr umständlich	1	2	3	4	5
9. Ich habe mich bei der Nutzung des Systems sehr sicher gefühlt.	1	2	3	4	5
10. Ich musste eine Menge Dinge lernen, bevor ich mit dem System arbeiten konnte.	1	2	3	4	5

Szenario 1:

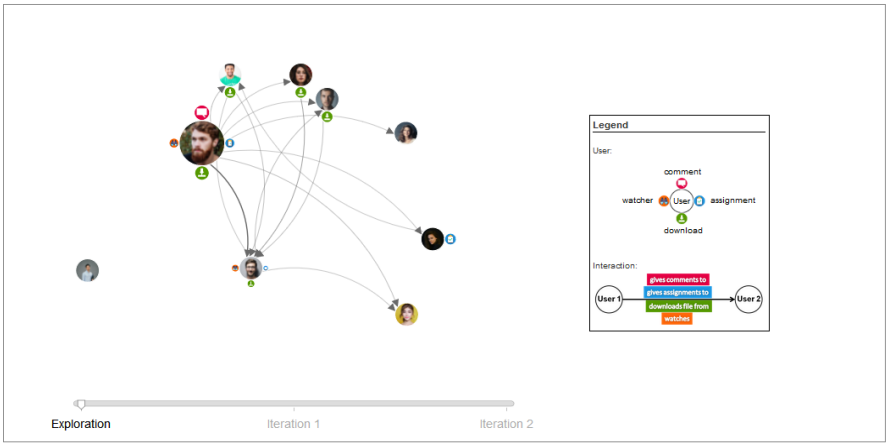

Order by Created
DEMO-18
Improve performance
DEMO-17
Mock up design
DEMO-16
Video does not play
DEMO-15
Social media button n...
DEMO-14
Eat pizza
DEMO-13
Drive a car
DEMO-12
Have a questions
DEMO-11
Take a break
DEMO-10
Create Application Logo
DEMO-9
efe
DEMO-8
Clean the floor
DEMO-7
Cook dinner for tonight
DEMO-6
Button design
DEMO-5
Drink Coffee
DEMO-4
Take a break
DEMO-3
Syntax Error
demo / DEMO-18
1 of 18
Improve performance
Edit
Comment
Assign
More
To Do
In Progress
Done
Admin
Export
Details
Type: Task
Status: DONE (View Workflow)
Priority: Medium
Resolution: Done
Labels: None
Sprint: Iteration 3
Description
Click to add description
Attachments
Drop files to attach, or browse.
Activity
All
Comments
Work Log
History
Activity
There are no comments yet on this issue.
Comment
People
Assignee:
lee
Assign to me
Reporter:
admin
Votes:
0
Watchers:
1 Stop watching this issue
Dates
Created:
Yesterday
Updated:
Yesterday
Resolved:
Yesterday
Agile
Active Sprint:
Iteration 3 ends 09/Apr/19
View on Board
HipChat discussions
Do you want to discuss this issue?
Connect to HipChat.
Connect
Dismiss

Szenario 2:

Attachment Tracker



Interaction Network



Literaturverzeichnis

[1] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. Manifesto for agile software development. 2001.

[2] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (xml) 1.0, 2000.

[3] J. Brooke. Sus: a retrospective. *Journal of usability studies*, 8(2):29–40, 2013.

[4] J. Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[5] R. N. Charette. Why software fails [software failure]. *IEEE spectrum*, 42(9):42–49, 2005.

[6] A. Cockburn. *Agile software development*, volume 177. Addison-Wesley Boston, 2002.

[7] D. Crockford. The application/json media type for javascript object notation (json). Technical report, 2006.

[8] K. S. Fabian Kortum, Jil Klünder. Behavior-driven dynamics in agile development: The effect of fast feedback on teams. 2019. In International Conference on Software and System Process (ICSSP).

[9] J. D. Herbsleb and D. Moitra. Global software development. *IEEE software*, 18(2):16–20, 2001.

[10] W. Huang, S.-H. Hong, and P. Eades. How people read sociograms: A questionnaire study. In *IN PROC. ASIA PACIFIC SYMPOSIUM ON INFORMATION VISUALISATION (APVIS2006*, pages 199–206, 2006.

[11] M. Höppner. Interaktiver editor für informationsflussmodelle. Master's thesis, Leibniz Universität Hannover, Fachgebiet Software Engineering, 2016.

[12] P. Li. *Jira Essentials.* Packt Publishing Ltd, 2015.

[13] F. Mccarey, M. Cinnéide, and N. Kushmerick. Rascal: A recommender agent for agile reuse. *Artificial Intelligence Review*, 2005.

[14] N. Murphy. Year: 2008 can developer-module networks predict failures?

[15] M. Ortu, G. Destefanis, M. Kassab, and M. Marchesi. Measuring and understanding the effectiveness of jira developers communities. In *Proceedings of the Sixth International Workshop on Emerging Trends in Software Metrics*, pages 3–10. IEEE Press, 2015.

[16] L. Richardson and S. Ruby. *RESTful web services.* O'Reilly Media, Inc., 2008.

[17] A. Rodriguez. Restful web services: The basics. *IBM developerWorks*, 33:18, 2008.

[18] K. Schneider. *Abenteuer Softwarequalität: Grundlagen und Verfahren für Qualitätssicherung und Qualitätsmanagement.* dpunkt. verlag, 2012.

[19] K. Schneider, O. Liskin, H. Paulsen, and S. Kauffeld. Media, mood, and meetings: Related to project success? *ACM Transactions on Computing Education (TOCE)*, 15(4):21, 2015.

[20] T. W. A. Schröter. Mining task-based social networks to explore collaboration in software teams. *IEEE Software*, pages 58–66, 2009.

[21] K. Stapel and K. Schneider. FLOW-Methode - Methodenbeschreibung zur Anwendung von FLOW. Technical report, Fachgebiet Software Engineering, Leibniz Universität Hannover, 2012.