TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik, Institut für Automatisierungstechnik

Dokumentation zum Projekt Mensch-Maschine-Systemtechnik

WEITERENTWICKLUNG EINER REVISIONSDARSTELLUNG FÜR DAS REVISIONSVERWALTUNGSSYSTEM R43PLES

Gruppe 6: Dominik Baumann, Johannes Börnicke, Katrin Messbauer, Angela Wobar

Betreut durch:
Dipl.-Ing. M. Graube
Eingereicht am 05. Februar 2016

INHALTSVERZEICHNIS

1 MOTIVATION

Um mit mehreren Personen koordiniert an großen Softwareprojekten arbeiten zu können, ist ein Versionsverwaltungssystem unerlässlich. Das Ziel dieser Systeme ist eine inkrementelle Entwicklung des Projektes, bei der die Historie jederzeit verfügbar ist. Dies bietet den Vorteil, dass die Änderungsgeschichte nachvollziehbar bleibt und jederzeit auf alte Versionen zurückgegriffen werden kann. Außerdem kann parallel auf verschiedenen Branches (Zweigen) entwickelt werden, so können also mehrere Personen gleichzeitig am gleichen Projekt arbeiten und das Versionsverwaltungssystem sollte dann auch eine Möglichkeit bieten, diese verschiedenen Versionen nachher in eine zu integrieren (auch: mergen). Der Begriff Revision bezeichnet dabei einen eindeutigen Versionsstand des Projektes (auch: Repository), der durch eine Revisionsnummer eindeutig bestimmt ist [4]. Um eine gute Nutzbarkeit des Systems zu gewährleisten, sind zudem eine ansprechende Visualisierung sowie die Bedienbarkeit essentiell.

Insbesondere für das Semantic Web mangelt es derzeit noch an einem solchen System, weshalb es in der Industrie noch nicht akzeptiert ist. Das Semantic Web stellt eine Erweiterung des Webs dar, mit der Informationen einfacher austausch- und verwertbar werden sollen, indem zu Begriffen zusätzliche semantische Informationen hinzugefügt werden [6]. In [5] wird mit R43ples ein Konzept für ein solches Versionsverwaltungssystem vorgestellt. Zur Visualisierung werden hier gerichtete Graphen verwendet. Es existieren eine komplette semantische Beschreibung und Mergingmöglichkeiten. Zur Datenabfrage gibt es zudem ein SPARQL Interface.

In der vorliegenden Arbeit soll die vorhandene Visualisierung erweitert werden, um die Changesets, der einzelnen Revisionen sichtbar zu machen. Dabei werden zunächst bestehende Konventionen und Ästhetikkriterien für Graphen untersucht und die bisherigen Arbeiten an R43ples, sowie andere Konzepte für Versionsverwaltungssystemen für das Semantic Web bzw. Linked Data im Allgemeinen analysiert werden, um auf dieser Basis anschließend eine eigene Visualisierung zu entwerfen und zu implementieren. Diese soll sich, um eine unkomplizierte Integration zu gewährleisten, direkt in die HTML-Oberfläche des bestehenden Gesamtsystems einfügen.

2 STAND DER TECHNIK

In [5] wurde das Versionsverwaltungssystem R43ples etabliert und, wie bereits erwähnt, mittels gerichteter Graphen visualisiert. Die Visualisierung wurde bereits in [1, 7] weiterentwickelt, diese Entwicklung soll in der vorliegenden Arbeit fortgesetzt werden. Da die Visualisierung weiterhin mittels gerichteter Graphen erfolgen soll, werden zunächst grundlegende Richtlinien und Ästhtetikkriterien zur Darstellung gerichteter Graphen diskutiert. Anschließend werden die vorhandenen Arbeiten an R43ples sowie weitere Darstellungen von Versionsverwaltungssystemen aus der Literatur und dem Internet analysiert.

2.1 GRUNDLEGENDE RICHTLINEN

Ein Graph ist definiert als eine Menge an Knoten und Kanten, wobei jede Kante zwei Knoten aus dieser Menge verbindet [2, Seite 2]. Sie dienen zur Veranschaulichung von Relationen zwischen Objekten, so können also sehr kompakt Informationen und Beziehungen zwischen den Informationen dargestellt werden. Graphen können dabei u.a. in gerichtete und ungerichtete Graphen unterteilt werden. Bei ungerichteten Graphen wird nicht zwischen Start- und Zielknoten unterschieden, bei gerichteten Graphen ist eine solche Unterscheidung vorhanden [3]. Für die Aufgabenstellung liegt also ein gerichteter Graph vor, da jeder Knoten einen Vorgänger und einen Nachfolger hat und diese klar unterscheidbar sind.

Für das Zeichnen von Graphen existieren einige Konventionen, die die Lesbarkeit erhöhen sollen und z.B. in [8] aufgeführt sind. Zu diesen Konventionen zählt, dass alle Kanten als gerade Linie gezeichnet werden sollten. Besondere Übersichtlichkeit wird erzielt, wenn der Graph kreuzungsfrei gezeichnet werden kann, sich die gezeichneten Linien also nicht schneiden. Diese Vorgabe kann in einem Versionsverwaltungssystem nicht immer eingehalten werden, allerdings kann hier der Begriff des planaren Graphens eingeführt werden. Ein Graph ist planar, wenn er durch Punkte und Linien der Zeichenebene dargestellt werden kann und sich die Linien höchstens an den Knoten schneiden. Diese Konvention soll bestmöglich umgesetzt werden. Neben den Konventionen gibt es einige Ästhetikkriterien, auf die ebenfalls geachtet werden soll. Dazu zählt die Minimierung der Schnittpunkte von Kanten, der Fläche und der Gesamtkantenlänge.

2.2 BISHERIGE VISUALISIERUNGEN

In [7] ist eine Möglichkeit zur Visualisierung bereits dargestellt. In diesem Entwurf steht der Graph im Mittelpunkt. Die einzelnen Knoten sind mit der Revisionsnummer beschriftet und durch die Pfeile ist ersichtlich, wie der Verlauf der Branches ist. Gleiche Branches sind zudem durch gleiche Farben gekennzeichnet. Durch einen Haken können zusätzlich die Tags[1] angezeigt

[1] Unter einem Tag versteht man eine Art Etikett, der zu einem Commit hinzugefügt wird, um zum Beispiel den Stand des Projektes zu diesem Zeitpunkt festzuhalten.

werden. In einem mouseover Feld werden bei Bedarf weitere Informationen angezeigt. Die Commits sind von oben nach unten angeordnet. Hierbei ist zwar ersichtlich, welche Commits innerhalb eines Branches nacheinander kamen, für zwei Commits aus unterschiedlichen Branches ist aber nicht erkenntlich, welcher zeitlich früher stattfand. Ein Beispielgraph ist in Abbildung 2.1 links zu sehen.

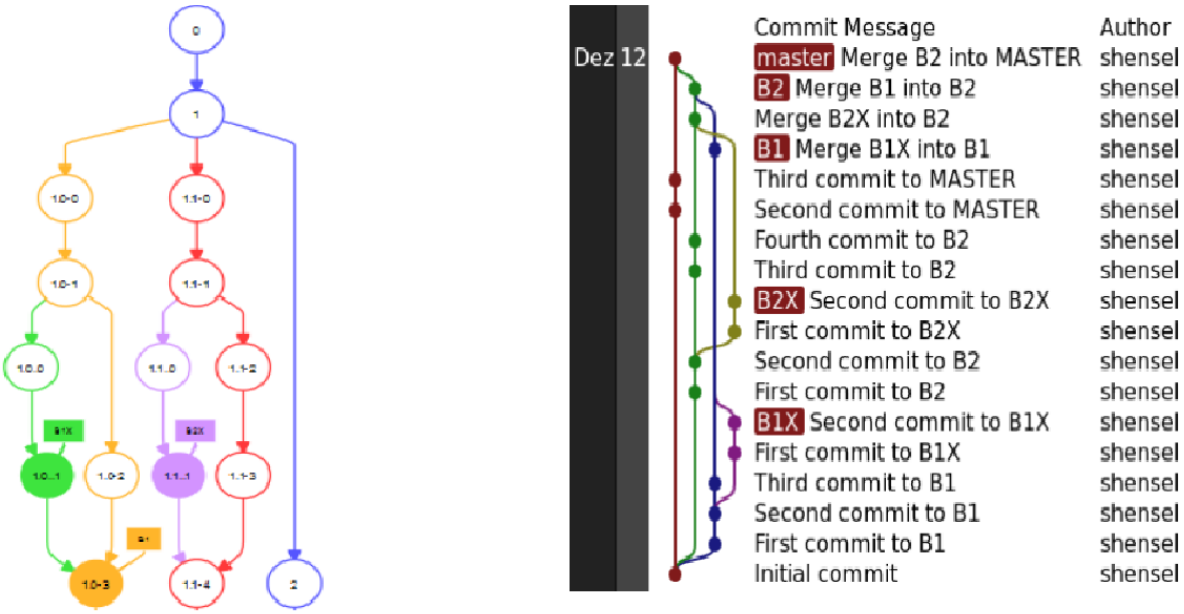


Abbildung 2.1: Visualisierungen der vorherigen Gruppen, links Gruppe 2.1, rechts Gruppe 2.3

Die zweite Visualisierung in [1] stellt den Graphen mehr an die Seite und stellt mit dem Kommentar zu jedem Commit sowie dem Autor zusätzliche Informationen dar. Die zeitliche Abfolge ist hier ersichtlich, zudem ist eine Einteilung in Monate gegeben. Der Graph wird durch diese Darstellung allerdings relativ klein und der Text dominiert die Visualisierung. Im in Abbildung 2.1 rechts dargestellten Beispiel sind die Kommentare immer sehr kurz gehalten, dies muss aber nicht zwangsläufig so sein muss, sodass es in dieser Art der Darstellung bei längeren Kommentaren zu Zeilenumbrüchen kommt, was den Graph in die Länge zieht.

In der zum Beginn dieser Arbeit im Git Repository vorhandenen Version war der Graph zoombar. Befand sich der Mauszeiger im Feld des Graphen und es wurde gescroolt, wurde an den Graphen heran- bzw. davon weg gezoomt. Diese Funktionalität bietet nur einen geringen Vorteil, der Graph wird immer in einer gut lesbaren Größe dargestellt und im Zweifel mit einem Scrollbalken versehen, die Zoomfunktion bietet hier also keinen entscheidenden Vorteil. Allerdings muss zum Teil gescrollt werden, um alle Informationen zu erkennen. Dabei musste immer darauf geachtet werden, dass sich der Mauszeiger nicht im Feld des Graphen befand, um nicht aus Versehen zu zoomen anstatt zu scrollen. Deshalb wurde entschieden, auf diese Funktion ganz zu verzichten oder sie erst zu aktivieren, wenn zusätzlich zum Scrollen gleichzeitig beispielsweise die „Strg“ Taste gedrückt wird.

2.3 WEITERE BEISPIELE

2.3.1 Git2PROV

In [9] wird mit W3C PROV ein neuer Standard eingeführt, um Versionskontrollsysteme wie Git einfacher benutzen und veröffentlichen zu können, zu sehen in Abbildung 2.2. Dort sind viele Informationen des semantischen Netzes dargestellt, die verschiedenen Arten von Informationen sind durch unterschiedliche Formen dargestellt. So sind die Commits in Rechtecken aufgeführt, die Namen der Autoren in Trapezen, außerdem sind die Kanten mit Spezifizierungen der Beziehungen, in der die Knoten stehen, versehen. Damit bietet dieses Format sehr viele Informationen, ist dabei aber weniger intuitiv und es braucht eine gewisse Einarbeitungszeit, bis damit umgegangen werden kann. Dieses Modell ist eher als Werkzeug für Entwickler gedacht und damit als Vorlage für R43ples nicht geeignet, da hier mehr auf den Anwender eingegangen werden soll, der die wesentlichen Informationen dargestellt haben möchte anstatt einer ausführlichen Beschreibung des semantischen Netzes.

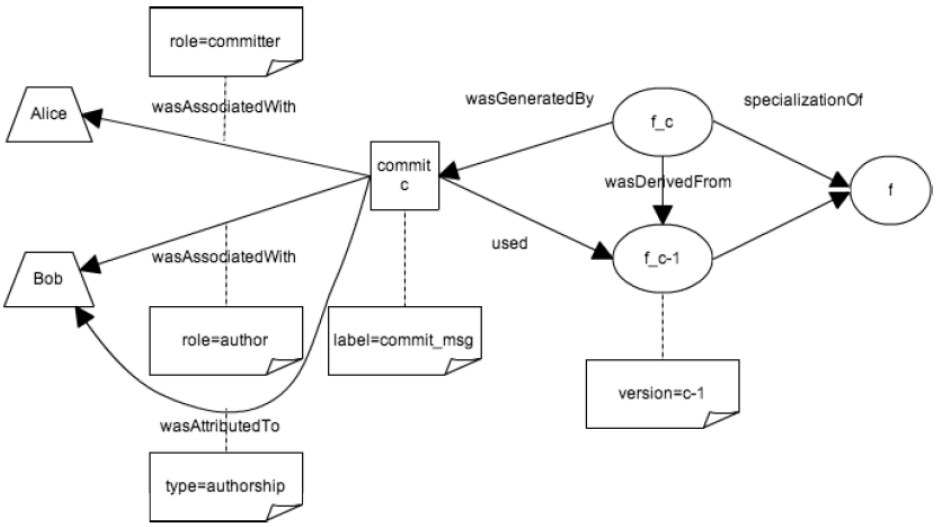


Abbildung 2.2: Darstellung eines Graphen mit Git2PROV

2.3.2 GitHub Desktop

GitHub stellt ein Desktop Tool zur Verfügung, in dem ebenfalls der Verlauf der verschiedenen Branches eines Projektes zu sehen ist, dargestellt in Abbildung 2.3. Hier werden für den markierten Commit relativ umfangreiche Informationen angezeigt, es ist genau ersichtlich, was sich im Vergleich zur vorherigen Version geändert hat, außerdem sind die Kommentare der einzelnen Commits samt Benutzerbild des Autors zu sehen. Es wird zwischen Commits und Merges unterschieden und der Zeitverlauf ist durch unterschiedliche Abstände ersichtlich, außerdem wird unter dem Kommentar angezeigt, wie viel Zeit seitdem vergangen ist. Es werden allerdings nur die Commits des markierten Branches angezeigt und dort auch nur die aktuellsten. Dies erhöht die Übersichtlichkeit, insbesondere bei umfangreichen Projekten, die Suche nach einzlnen Commits gestaltet sich aber aufwendiger, auch weil die Knoten keine Beschriftung haben, wodurch eine direkte Zuordnung nicht möglich ist, die Revisionsnummer taucht gar nicht auf.

2.3.3 Tortoise Git

Eine weitere Möglichkeit, Git zu nutzen, bietet die freie grafische Benutzeroberfläche Tortoise Git, zu sehen in Abbildung 2.4. Dargestellt wird hier ein kleiner, von unten nach oben aufgebauter Graph, eine grafische Darstellung, ob Dateien gelöscht, hinzugefügt oder geändert

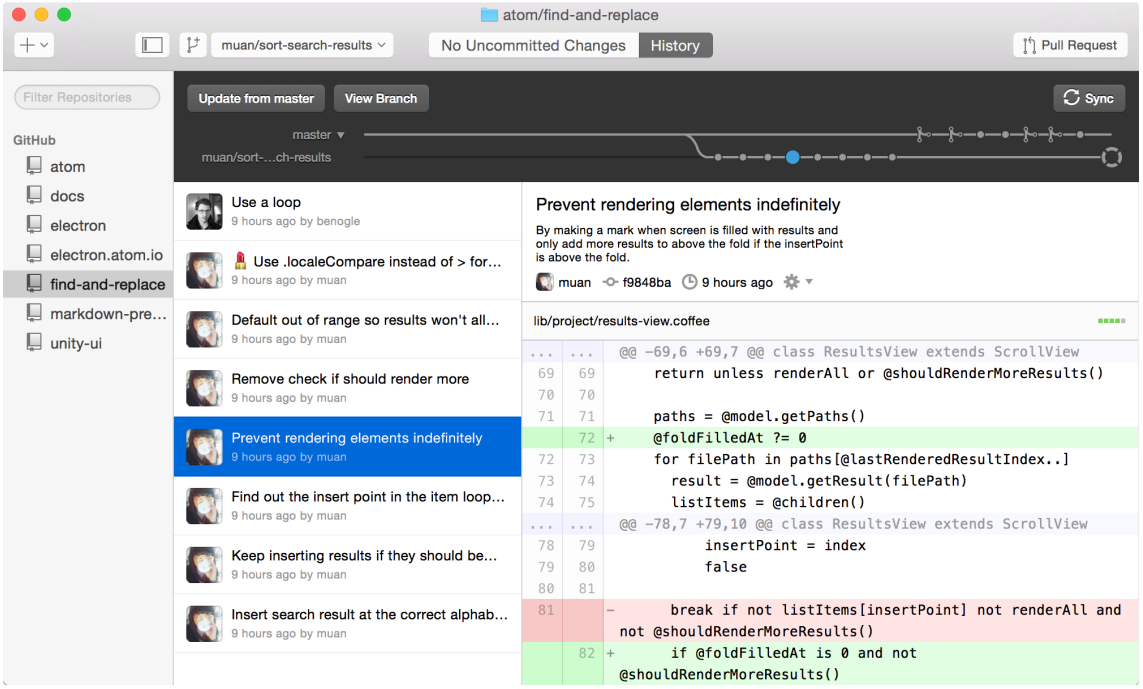


Abbildung 2.3: Darstellung eines Graphen mit GitHub Desktop

wurden, die Commit Nachricht, der Autor und das Datum. Durch Klicken auf den jeweiligen Commit können genauere Infos abgerufen werden, z.B. welche Dateien genau geändert/hinzugefügt/gelöscht wurden, außerdem wird die Commit Nachricht komplett angezeigt, falls sie vorher aus Platzgründen nur zum Teil dargestellt werden konnte.

Hier zeigt sich bereits ein Nachteil dieser Art der Darstellung. Lange Commit Nachrichten können nicht dargestellt werden, aus denen ersten Worten, die gezeigt werden können, ist nicht zwingend der Kontext ersichtlich. Zudem ist der Platz für den Graphen relativ gering, sodass große, weit verzweigte Graphen nicht übersichtlich dargestellt werden können.

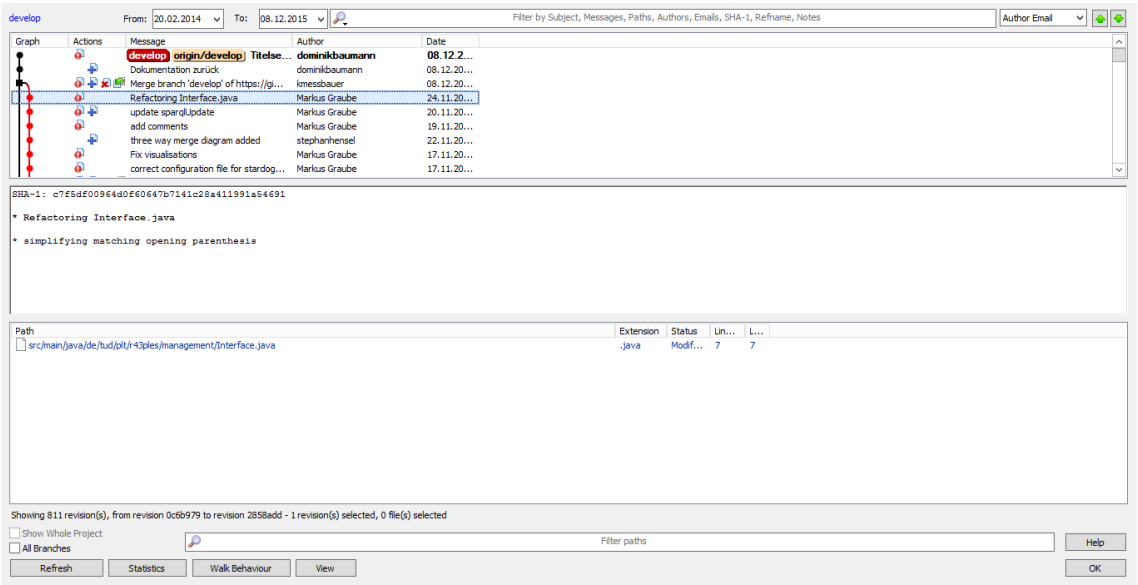


Abbildung 2.4: Darstellung eines Graphen mit Tortoise Git

3 ANFORDERUNGSDEFINITION

3.1 EINLEITUNG

Die im vorangegangenen Abschnitt analysierte, vorhandene Visualisierung für das semantische Revisionsverwaltungssystem R43ples soll erweitert werden, sodass die Inhalte der Changesets der Revisionen sichtbar werden. Der neue Entwurf soll sich dabei gut in die bisherige HTML-Oberfläche integrieren und gut skalieren in Bezug auf die Größe der Changesets.

3.2 ALLGEMEINE BESCHREIBUNG

3.2.1 Produktumgebung

Da Revisionsverwaltungssysteme im Allgemeinen vorwiegend auf Desktop-PCs oder Laptops, evtl. auch auf Tablets, verwendet werden, soll das Produkt für eine Nutzung an einem Bildschirm im Querformat optimiert werden. Das System muss sich in die bereits existierende Weboberfläche einfügen.

3.2.2 Benutzereigenschaften

Anders als die in Abschnitt 2.3.1 beschriebene Visualisierung, die eher für Entwickler gedacht ist, ist R43ples für Normalanwender gedacht, die keine ausführliche Darstellung des semantischen Netzes, sondern eine übersichtliche Visualisierung der wichtigsten Informationen benötigen.

3.3 SPEZIFISCHE ANFORDERUNGEN

3.3.1 Funktionale Anforderungen

F1 Der Graph muss durch geeignete Darstellung eine schnelle Auffindbarkeit der gesuchten Revision gewährleisten.

- F1.1 Die Revisionen werden automatisch nach dem Zeitpunkt des Commits von links nach rechts angeordnet, da ein Bildschirm im Querformat so mehr Platz für den Graphen bietet, ohne dass der Benutzer scrollen muss.

F1.2 Die Revisionen sind den einzelnen Branches durch einheitliche Farbgebung aller Revisionen eines Branches und lineare Anordnung eines Branches zuordenbar zu machen.

F1.3 Die Branches sind entsprechend ihrer Benennung zu beschriften.

F1.4 Der Master-Branch muss durch eine immer gleiche Farbgebung und seine Position besonders leicht erkennbar sein.

F1.5 Direkt auf den Knoten der Revisionen wird die Revisionsnummer dargestellt.

F1.6 Die Knoten müssen immer die gleiche Größe haben, um sie gut erkennen zu können. Ist der Graph zu groß, um in einem Fenster dargestellt zu werden, wird ein Scrollbalken hinzugefügt. Dabei wird immer auf die aktuellste Version fokussiert.

F1.7 Am oberen Bildschirmrand soll eine Zeitleiste für die Commit-Zeitpunkte hinzugefügt werden.

F1.8 Revisionen mit Tags sollen gesondert markiert werden.

F2 Durch zusätzliche Darstellung von Tooltips beim Mouseover sollen dem Benutzer die wichtigsten Information schnell zugänglich gemacht werden.

F2.1 Beim Mouseover über eine Revision soll ein Tooltip-Fenster geöffnet werden, das sich automatisch wieder schließt, sobald der Benutzer den Mauszeiger vom Knoten wegbewegt.

F2.2 Im Tooltip-Fenster sollen der Name des Autors, der Kommentar, sowie Datum und Uhrzeit angezeigt werden.

F3 Beim Klick auf eine Revision werden Detailinformation in einem Fenster unterhalb des Graphen dargestellt.

F3.1 Die Detailinformationen enthalten nochmals die Informationen aus dem Tooltip-Fenster, sowie zusätzlich eine Auflistung der Changesets.

F3.2 Wenn mehr als eine maximale Anzahl an Adds/Deletes auftritt, wird das Fenster scrollbar.

F3.3 Die Adds und Deletes sollen jeweils per Klick ein- und ausgeblendet werden können.

F3.4 Die Adds und Deletes sollen nebeneinander aufgeführt werden.

F3.5 Die Detailansicht einer Revision bleibt solange erhalten, bis auf eine andere Revision geklickt wird und somit deren Detailansicht geöffnet wird.

F3.6 Die aktuell angeklickte Revision soll hervorgehoben werden.

F3.7 Damit die Detailinformationen sicher erreicht werden können, muss die Zoomfunktion entweder ausgeschaltet oder mit einer zusätzlichen Bedingung, zum Beispiel das Drücken der „Strg“ Taste, versehen werden.

F3.8 URLs müssen in spitzen Klammern, Literale in Hochkommata dargestellt werden.

3.3.2 Qualitätsanforderung

Q1 Die in Abschnitt 2.1 erläuterten Konventionen sollen eingehalten werden.

Q1.1 Alle Kanten sollen gerade Linien sein.

Q1.2 Kreuzungen von Graphen sollen nur an Knoten auftreten.

Q1.3 Die Fläche und Gesamtkantenlänge soll minimiert werden.

3.3.3 Einschränkungen und Randbedingungen

R1 Das Produkt muss kompatibel mit aktuellen, verbreiteten Browsern (Firefox, Chrome) sein.

R2 Die Entwicklung muss als Fork auf GitHub durchgeführt werden.

R3 Die Implementierungssprache ist Javascript.

R4 Das Produkt ist in die bestehende Anwendung zu integrieren.

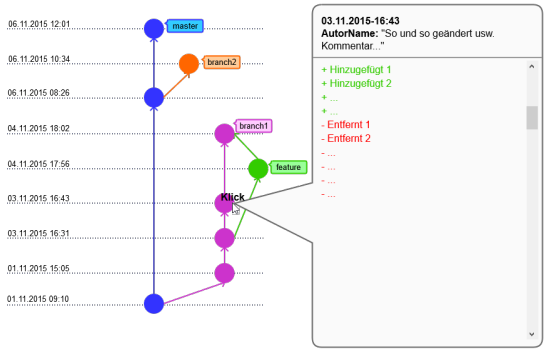
4 GESTALTUNGSENTWURF UND IMPLEMENTIERUNG

Für den Gestaltungsentwurf wurden zunächst Mockups entworfen und auf Realisierbarkeit geprüft, anschließend wurde das Konzept umgesetzt und implementiert.

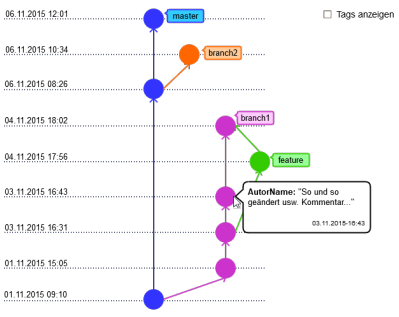
4.1 MOCKUPS

Ein erster Mockup ist in Abbildung 4.1 zu sehen. Hier ist bereits eine Zeitleiste dargestellt, die einzelnen Branches sind in der gleichen Farbe gezeichnet und mit Namen beschriftet. Der Master Branch verläuft geradlinig, nebenläufige Branches sind rechts davon angeordnet. Wird auf einen Knoten geklickt, werden wie links gezeigt, zusätzlich der Name des Autors, der Kommentar sowie die Adds und Deletes angezeigt. Wird die Maus auf den Knoten gefahren, ohne zu Klicken, sieht man lediglich den Autorennamen und den Kommentar.

Dieser Entwurf ist bereits übersichtlich, der zeitliche Verlauf kann nachvollzogen werden und es sind alle wichtigen Informationen einfach abrufbar, lediglich die Revisionsnummer fehlt hier noch, die im späteren Entwurf in jedem Knoten angezeigt wird. Weitere Änderungen betreffen die Anordnung, die, wie in den Anforderungsdefinitionen festgelegt von links nach rechts sein soll, außerdem sollten die nebenläufigen Branches zu beiden Seiten des Masters aufgeführt werden.



(a) Darstellung des Graphen und des gesamten Changesets bei Klick



(b) Anzeige spezifischer Informationen bei Mouseover

Abbildung 4.1: Mockups

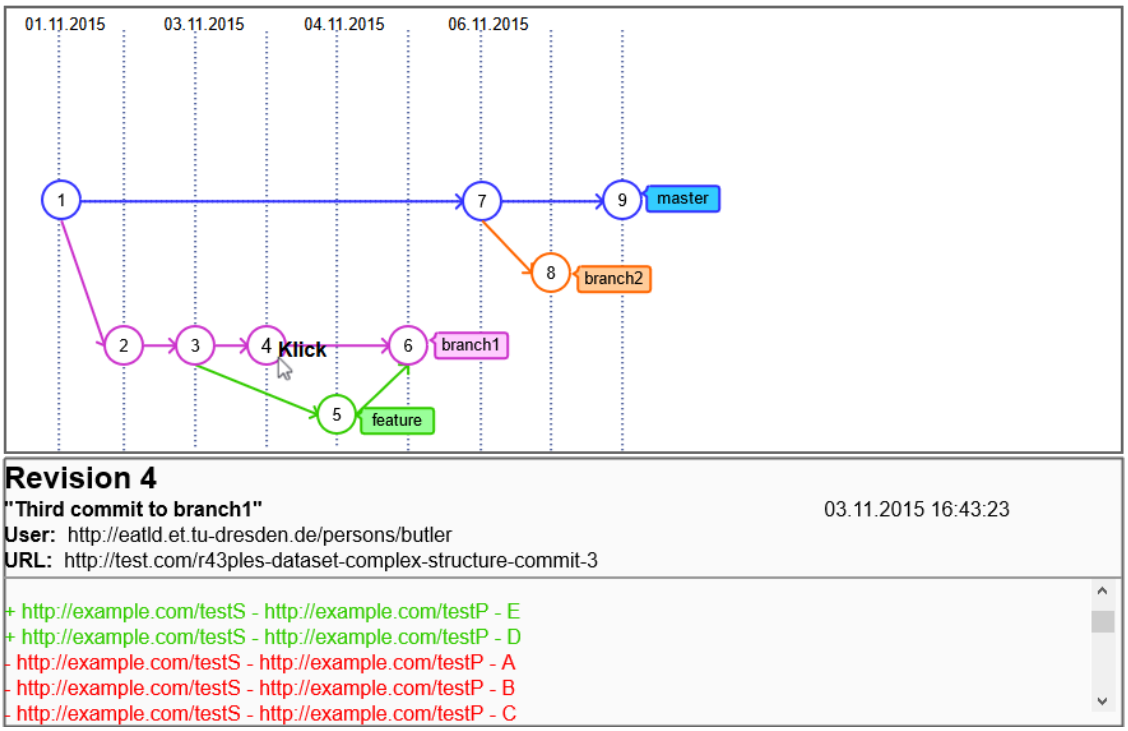


Abbildung 4.2: Mockup im Querformat mit Anzeige der Add und Delete Sets

In Abbildung 4.2 ist eine verbesserte Variante zu sehen. Der Graph wird jetzt von links nach rechts angezeigt, der Master Branch ist in der Mitte, die nebenläufigen Branches werden darunter angezeigt. Aus Gründern der Übersichtlichkeit und Darstellbarkeit wird das Datum nur noch aufgeführt, wenn sich der Tag geändert hat, außerdem stehen innerhalb der Knoten die Revisionsnummern. Bei Klicken werden am unteren Bildschirmrand die Add und Delete Sets aufgeführt, außerdem der Name des Autors und der Kommentar, sowie Datum und Uhrzeit des Commits. Da es in diesem Beispiel viele Deletes gibt wurde außerdem ein Scrollbalken hinzugefügt, um die Übersichtlichkeit zu erhalten.

Für eine weiter verbesserte Übersichtlichkeit wurde entschieden, die Add und Delete Sets nebeneinander anzuzeigen, jeweils versehen mit Scrollbalken. Die URLs sollten in spitzen Klammern angezeigt werden, Literale in Hochkommata. Im Mouseover Feld wurde zudem eine Anzeige hinzugefügt, wie viele Adds und Deletes es in dieser Revision gab. Gesonderte Markierungen wurden zum einen für die derzeit angeklickte Revision des Graphen, durch Fettschrift der Zahl und fettere Umrandung des Knotens, sowie für alle Revisionen mit Tags, indem diese Knoten ausgemalt wurden, eingeführt. Diese Erweiterungen sind in Abbildung 4.3 zu sehen.

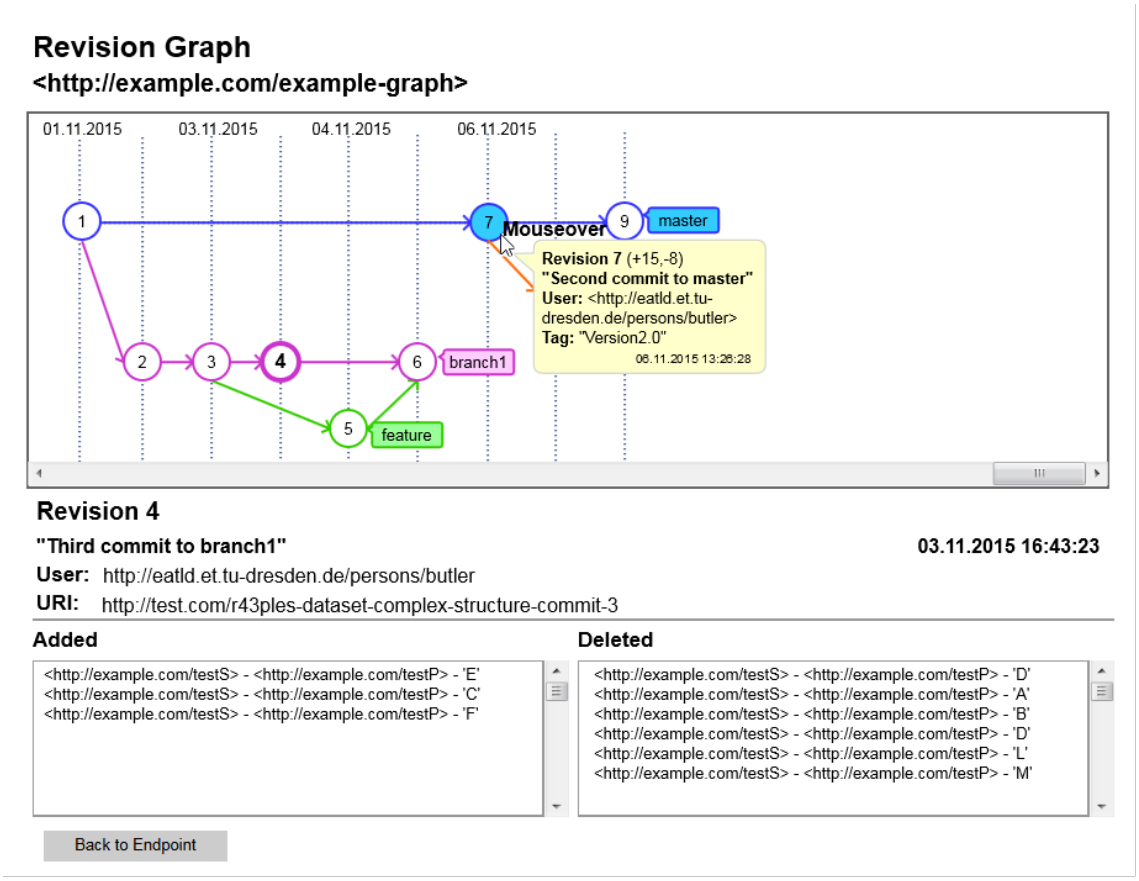


Abbildung 4.3: Add und Delete Sets nebeneinander, erweitertes Mouseover Feld, Markierung der geklickten Revision sowie gesonderte Markierung von Revisionen mit Tags

LITERATURVERZEICHNIS

[1] Buntkiel, L., A. Lehmann, M. Zhang, S. Schönfeld und F.-J. Straube: Konzeption und Gestaltung einer Revisionsdarstellung für das semantische Revisionsverwaltungssystem R43ples. Dokumentation zum Projekt Mensch-Maschine-Systemtechnik, 2015.

[2] Büsing, C.: Graphen- und Netzwerkoptimierung. Spektrum Akademischer Verlag, 2010.

[3] Diekert, V., M. Kufleitner und G. Rosenberger: Elemente der diskreten Mathematik. De Gruyter, 2013.

[4] Graube, M.: Vorlesungsunterlagen CAE - Revisisonsverwaltung: R43ples als RCS für Linked Data, 2015.

[5] Graube, M., S. Hensel und L. Urbas: R43ples: Revisions for Triples - An Approach for Version Control in the Semantic Web. Proceedings of the 1st Workshop on Linked Data Quality (LDQ), 1215, 2014.

[6] Grütter, R.: Semantic Web zur Unterstützung von Wissensgemeinschaften. Oldenbourg Verlag München, 2008.

[7] Hoffmann, F., M. Müller, W. Nebyliza und J. Riffert: Konzeption und Gestaltung einer Revisionsdarstellung für das semantische Revisionsverwaltungssystem R43ples. Dokumentation zum Projekt Mensch-Maschine-Systemtechnik, 2015.

[8] Jänicke, H.: Darstellung von Graphen. Vorlesungsunterlagen Visualisierung I. Uni Heidelberg.

[9] Nies, T. D., S. Magliacane, R. Verborgh, S. Coppens, P. Groth, E. Mannens und R. V. de Walle: Git2PROV: Exposing Version Control System Content as W3C PROV. Poster and Demo Proceedings of the 12th International Semantic Web Conference, 2013.

ABBILDUNGSVERZEICHNIS