



Dokumentation zum Projekt Mensch-Maschine-Systemtechnik

# WEITERENTWICKLUNG EINER REVISIONSDARSTELLUNG FÜR DAS REVISIONSVERWALTUNGSSYSTEM R43PLES

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



# INHALTSVERZEICHNIS

1	Motivation	5
2	Stand der Technik	7
2.1	Grundlegende Richtlinien . . . . .	7
2.2	Bisherige Visualisierungen . . . . .	7
2.3	Weitere Beispiele . . . . .	8
2.3.1	Git2PROV . . . . .	8
2.3.2	GitHub Desktop . . . . .	9
2.3.3	Tortoise Git . . . . .	9
3	Anforderungsdefinition	11
3.1	Einleitung . . . . .	11
3.2	Allgemeine Beschreibung . . . . .	11
3.2.1	Produktumgebung . . . . .	11
3.2.2	Benutzereigenschaften . . . . .	11
3.3	Spezifische Anforderungen . . . . .	11
3.3.1	Funktionale Anforderungen . . . . .	11
3.3.2	Qualitätsanforderung . . . . .	12
3.3.3	Einschränkungen und Randbedingungen . . . . .	12
4	Gestaltungsentwurf und Implementierung	13



# 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: Repositories), 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 Ästhetikkriterien 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<sup>1</sup> angezeigt

---

<sup>1</sup>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.

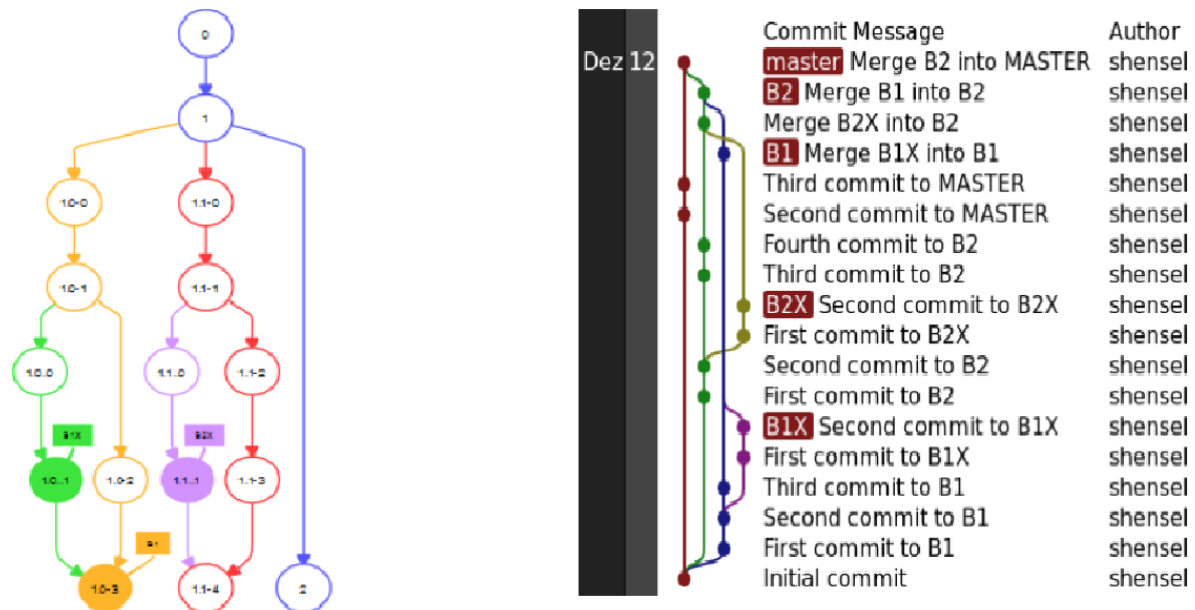


Abbildung 2.1: Visualisierungen der vorherigen Gruppen, links Gruppe 2.1, rechts Gruppe 2.3

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 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.

Beide Visualisierungen sind in Abbildung 2.1 zu sehen.

## 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.

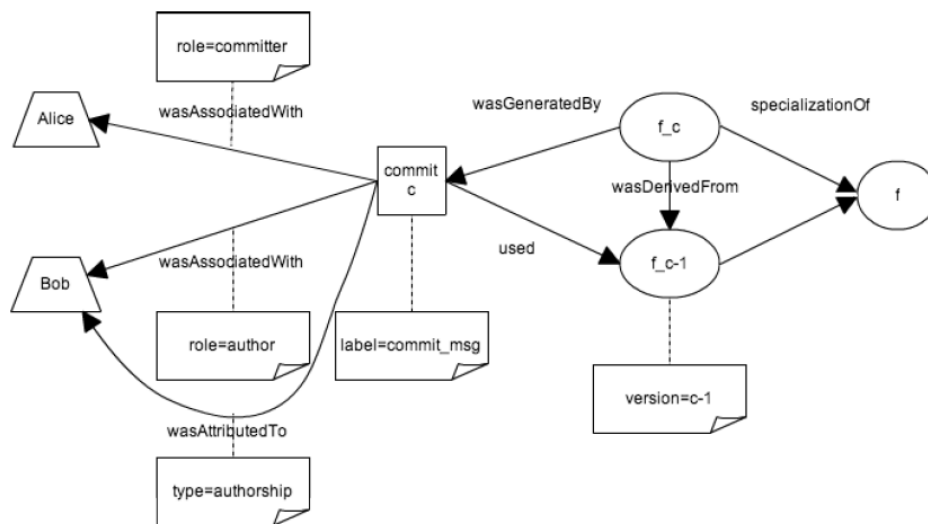


Abbildung 2.2: Darstellung eines Graphen mit Git2PROV

### 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 einzelnen 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, dargestellt in Abbildung 2.4. Dargestellt wird hier ein kleiner, von unten nach oben aufgebauter Graph, eine grafische Darstellung, ob Dateien gelöscht, hinzugefügt oder geändert 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.

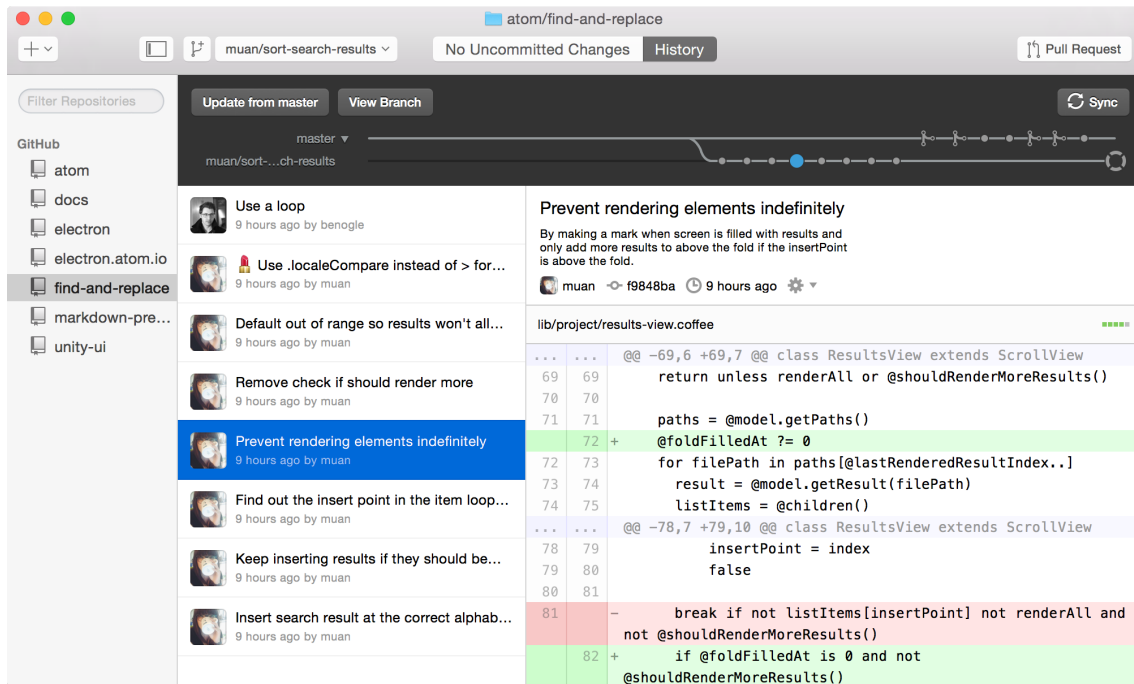


Abbildung 2.3: Darstellung eines Graphen mit GitHub Desktop

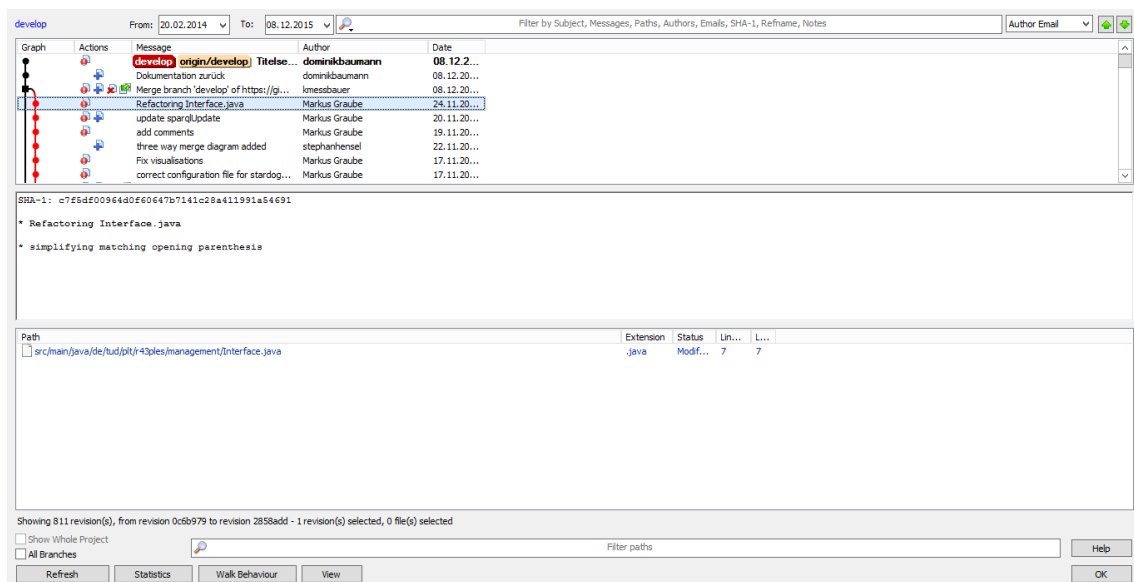


Abbildung 2.4: Darstellung eines Graphen mit Tortoise Git

## 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.

#### 3.2.2 Benutzereigenschaften

Anders als die in Abschnitt 2.3.1 beschriebene Visualisierung, die eher für Entwickler gedacht ist, ist R43ples für Anwender 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 soll durch geeignete Darstellung eine schnelle Auffindbarkeit der gesuchten Revision gewährleisten.

F1.1 Die Revisionen sollen nach dem Zeitpunkt des Commits automatisch von links nach rechts angeordnet werden, 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 soll durch eine immer gleiche Farbgebung und dessen zentrale Position besonders leicht erkennbar sein.
- F1.5 Direkt auf den Knoten der Revisionen soll die Revisionsnummer dargestellt werden.
- F1.6 Am oberen Bildschirmrand ist eine Zeitleiste für die Commit-Zeitpunkte hinzuzufügen.
- 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 rechts unten neben dem Knoten 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 sollen Detailinformation in einem Fenster unterhalb des Graphen dargestellt werden.
  - F3.1 Die Detailinformationen sollen nochmals die Informationen aus dem Tooltip-Fenster enthalten, sowie zusätzlich die Changesets detailliert auflisten.
  - F3.2 Die Detailansicht einer Revision bleibt solange erhalten, bis auf eine andere Revision geklickt wird und somit deren Detailansicht geöffnet wird.

### 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 soll als Fork auf GitHub durchgeführt werden.
- R3 Als Implementierungssprache ist Javascript vorgesehen.
- R4 Das Produkt ist in die bestehende Anwendung zu integrieren.

## 4 GESTALTUNGSENTWURF UND IMPLEMENTIERUNG

Ideen:

- Mouseover für mehr Informationen
- Evtl. durch Klick dauerhaftes Hinzufügen der Informationen
- Show und Hide Haken
- evtl Änderung der Form auf Rechteck, um mehr Platz für zusätzliche Informationen zu erhalten
- Commits und Tags sollen weiterhin klar unterscheidbar sein, evtl. durch Farbe
- Vermeiden von Kreuzungen
- Ausgleich zwischen Information und Übersichtlichkeit finden
- Neuestes nach oben
- Evtl. Master in der Mitte, alles andere drum herum anordnen
- Evtl. Einfügen einer Suche
- Entscheidendes Ziel der zusätzlichen Informationen: Schnelles Finden der Version, die wiederhergestellt werden soll
- add und remove durch Symbol darstellen
- Mit Klick/Mouseover Informationen, was geaddet bzw. removed wurde
- Sortierung nach Datum? Evtl. auch grobe Zeiteinteilung
- Bei Klick werden Add und Delete Sets mit angezeigt (was wurde hinzugefügt, was wurde gelöscht)
- Anzeige in Fenster am unteren Bildschirmrand
- Zeitleiste



# 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 - Revisionsverwaltung: 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

2.1	Visualisierungen der vorherigen Gruppen, links Gruppe 2.1 (aus [7]), rechts Gruppe 2.3 (aus [1]) . . . . .	8
2.2	Darstellung eines Graphen mit Git2PROV, entnommen aus [9] . . . . .	9
2.3	Darstellung eines Graphen mit GitHub Dekstop, entnommen von <a href="https://assets-cdn.github.com/images/modules/home/desktop-mac.png?v=1">https://assets-cdn.github.com/images/modules/home/desktop-mac.png?v=1</a> , abgerufen am 24.11.2015	10
2.4	Darstellung eines Graphen mit Tortoise Git . . . . .	10