TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Fakultät Elektro- und Informationstechnik Institut für Automatisierungstechnik

DOKUMENTATION PRAKTIKUM MENSCH-MASCHINE-SYSTEMTECHNIK

ENTWURF EINER VISUALISIERUNG FÜR DAS SEMANTISCHE REVISIONSVERWALTUNGSSYSTEM R43PLES

Gruppe 2.3: Lukas Buntkiel, Alexander Lehmann, Miao Zhang, Sven Schönfeld, Falk-Jonatan Strube

29. Januar 2015

INHALTSVERZEICHNIS

1 EINLEITUNG

1.1 EINORDNUNG

Hauptbestandteil der Aufgabenstellung ist das Entwerfen einer interaktiven Darstellung der Revisions-Struktur des Versionsverwaltungssystems (Version Control System, VCS) R43ples.

R43ples kann zur Versionsverwaltung von Named Graphs genutzt werden, dem Schlüssel-Bestandteil des Semantic Web [1]. Es verwendet dabei zur Verwaltung der Revisionen wiederum Named Graphs, in denen auch sämtliche, zur Darstellung der Struktur notwendigen, Informationen in Form von Linked Data enthalten sind [2]. R43ples verwendet dabei ein ähnliches Konzept wie klassische Versionsverwaltungssysteme (wie z.B. git [3]) indem es Verzweigungen von Revisionen in Form von Branches sowie das Kennzeichnen spezieller Revisionen mit Tags unterstützt [2].

Der Hauptunterschied zu klassischen VCS liegt also weniger im Konzept der Versionsverwaltung selbst, als in der Anwendung dieses Konzeptes auf einen neuen Typ von Ressource (Named-Graphs). Es kann daher angenommen werden, dass durch andere VCS bereits Lösungen für die graphische Darstellung von Revisionen vorhanden sind, die im Verlauf dieser Arbeit analysiert werden können, um günstige Merkmale herauszuarbeiten.

1.2 THEORIEBILDUNG

Die durch den Lehrstuhl gegebene Aufgabenstellung lautet wie folgt:

> Ziel dieser Aufgabe ist es, eine gebrauchstaugliche Visualisierung für das semantische Revisionsverwaltungssystem R43ples) zu implementieren, die Revisionsgraphen mit Branches, Merges und Tags darstellen kann. Das Werkzeug soll sich nahtlos in die HTML-Oberfläche des Gesamtsystems einfügen und dem Nutzer Überblicksinformationen über den Revisionsverlauf und Detailinformationen zu den einzelnen Revisionen geben.

Dem entsprechend wurden bei der Theoriebildung folgende Kernanforderungen an das Projekt herausgearbeitet.

Schnittstelle zu bestehender Software Zuerst muss eine Schnittstelle zur bestehenden Software hergestellt werden. Das wäre möglich, indem man beispielsweise die turtle Datensätze ausliest die der laufende Server ausgibt. Dadurch hätte man ein separates System das unabhängig vom Revisions-Server turtle Datensätze auslesen und darstellen könnte.

Eine andere Möglichkeit stellt die direkte Integration in das R43ples-Projekt auf github dar, in der Visualisierung direkt im System integriert wäre. Diese Herangehensweise hätte den Vorteil, dass man nicht erst eine turtle Datei parsen müsste, sondern direkt auf die Datensysteme des Servers zugreifen könnte. Durch einen fork des repositorys könnte man an einer Lösung der Aufgabe arbeiten und hätte bei einer guten Lösung einen einfachen Weg die Ergebnisse schnell im Hauptprojekt zu nutzen.

Graphische Darstellung realisieren Nachdem die Daten für das System über die Schnittstelle beschafft wurden, ist eine Hauptaufgabe diese als graphische Darstellung zu realisieren.

Eine erste Realisierung liegt bereits als einfache Implementierung mit Viz.js vor. Diese ist allerdings als nicht besonders gut und lediglich als Platzhalter zu bewerten.

Der Graph wird anscheinend nach keinen gestalterischen Grundsätzen generiert. Bis auf die chronologische Auflistung der Revisionen von oben nach unten und die Pfeile als optische Hilfestellung wird keine weitere Hilfestellung geboten, die die Erfassung des Graphen erleichtern würde.

Des weiteren ist die Auswahl der Informationen die von den Revisionen angezeigt werden nicht optimal. Sie bietet dem Nutzer nur zwei Informationen, wobei eine davon für den Nutzer uninteressant ist. Es fehlen aber andere wie beispielsweise der Autor oder das Datum.

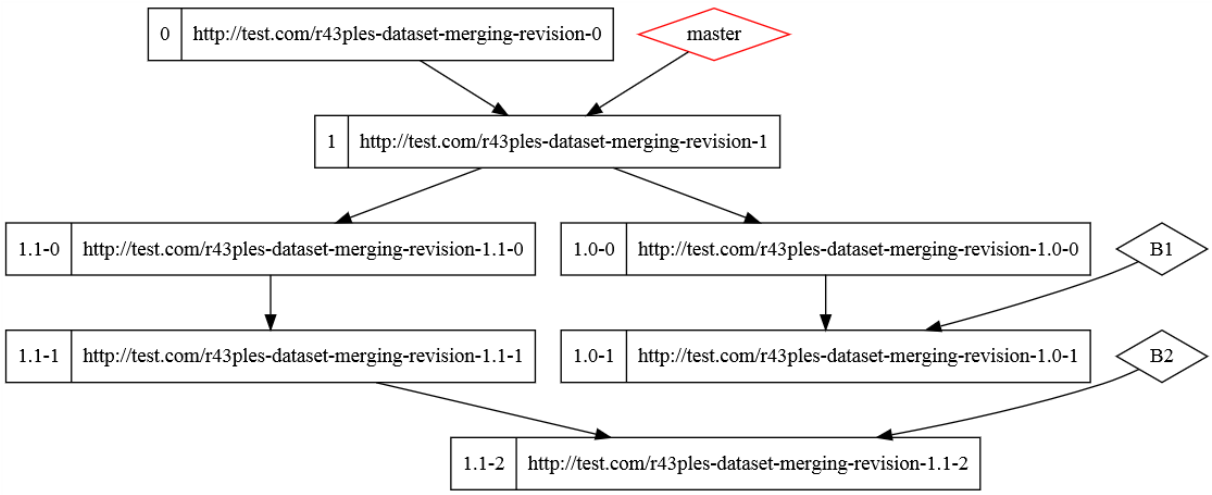

Abbildung 1.1: Vorgegebene Implementierung mit Viz.js

Überblicks- und Detailsicht Wenn man die Idee der vorhandenen Implementierung beibehalten möchte, die die Informationen der Revisionen direkt in den Knoten des Graphen darstellt muss man sich überlegen, wie man bei großen Revisionsbäumen eine übersichtliche Darstellung erreicht.

Um einen guten Kompromiss aus Informationsfülle und Übersichtlichkeit zu finden würde sich also vor allem bei vielen Daten ein Trennung des Graphen in eine Überblicks- und eine Detailansicht anbieten. So könnte man in einem Überblicksgraphen die wichtigsten Informationen anzeigen und durch das Navigieren auf eine spezielle Revision eine Detailansicht erreichen, die sämtliche relevanten Informationen darstellt.

WIE VISUALISIERT MAN EIN SEMANTISCHES VERSIONSVERWALTUNGSSYSTEM?

Wie bereits in der Einordnung erwähnt, kann als Konzept ein ähnliches verwendet werden wie ein klassischen Versionsverwaltungssystemen. Folgende Richtlinien wurden als Gestaltungsgrundsatz festgelegt:

- Übersichtlichkeit – alle wichtigen Informationen sollen mit einem Blick gesehen werden
- Verständlichkeit – alle gesehenen Informationen sollen möglichst schnell im Zusammenhang verstanden werden
- Angemessener Informationsgehalt – Je nach Ansicht sollen angemessen viele Informationen dargestellt werden nach dem Motto: „so wenig wie möglich, so viel wie nötig“

1.3 LITERATUR

Nachdem in der Einordnung die grundlegende Literatur für das Thema dargelegt wurde ([1], [2], [4]), finden sich auch noch mehr Informationen in verschiedensten wissenschaftlichen Publikationen:

Der Artikel „Special Graph Representations And Visualization Of Semantic Networks“ [5] steigt genau bei der Fragestellung nach der Visualisierung von semantischen Netzwerken ein und bietet einen grundlegenden Überklick für die Herangehensweise.

Die Masterarbeit „Ein empirischer Vergleich von semistrukturierter und unstrukturierter Konfliktbehandlung in Versionsverwaltungssystemen“ [6] gibt dabei einen tieferen Einblick zu einem Aspekt der Versionsverwaltung, ist bei der Visualisierung aber nur bedingt hilfreich.

In „Approaches to Visualising Linked Data: A Survey“ [7] wird zwar die Visualisierung von Linked Data vorgestellt, allerdings weniger im Bereich der Graphenbildung. Dadurch ist der Artikel für dieses Projekt in dem die Daten durch einen Graphen geordnet darstellen wollen weniger bedeutsam.

Ganz ähnlich bietet „A General Introduction To Graph Visualization Techniques“ [8] einen Überblick über die Visualisierungs-Optionen, wobei hier vermehrt Lösungen durch Graphen vorgestellt werden, was den Artikel etwas interessanter für das Projekt macht.

Dann bietet sich nach der Theoriebildung an sich Literatur zu den gefundenen Problem- und Fragestellungen zu suchen.

Der Artikel „Eye tracking for visualization evaluation: Reading values on linear versus radial graphs“ [9] führt auf, was die Unterschiede sind, ob man Graphen linear oder radial darstellt. Dabei wird der Schluss präsentiert, dass linear dargestellte Graphen deutlich einfacher auf den ersten Blick zu erfassen sind, weswegen in dem Projekt eine solche Darstellung verwendet werden soll.

Die im Internet zu findende Seite „A successful Git branching model“ [10] zeigt schon auf eine gute Weise, wie man ein Versionsprotokoll darstellen kann und worauf zu achten ist.

Die DIN „Software-Ergonomie für Multimedia-Benutzungsschnittstellen“ DIN EN ISO 14915 [11] [12] [13] gibt entscheidende Richtlinien für die Gestaltung des Projektes im Rahmen einer Benutzerschnittstelle, was vor allem dann von Bedeutung ist, falls die verschiedenen Ansichten in ihren Zoom-Stufen umgesetzt werden soll.

2 ANALYSE UND ENTWURF

2.1 ANALYSE

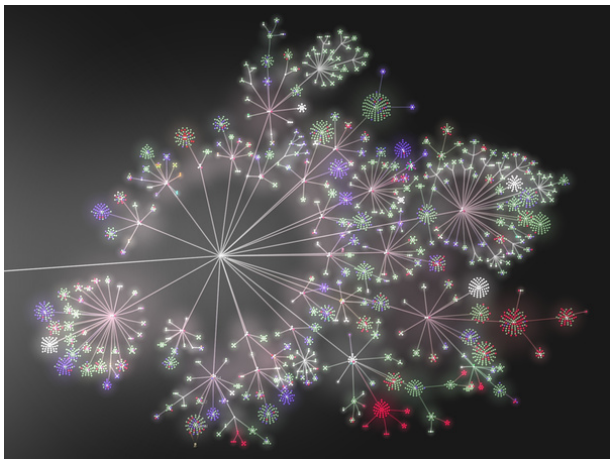

Abbildung 2.1: gource Visualisierung

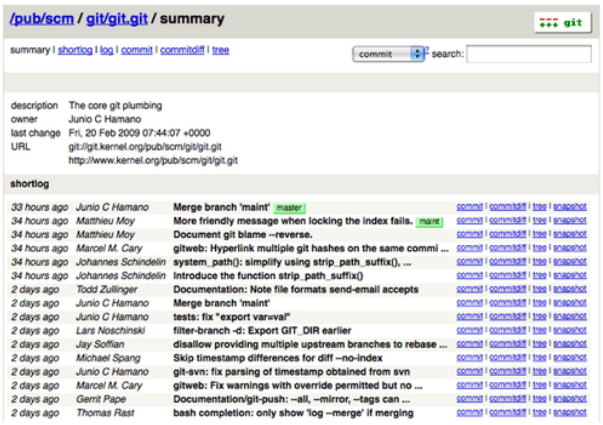

Abbildung 2.2: gitweb Visualisierung

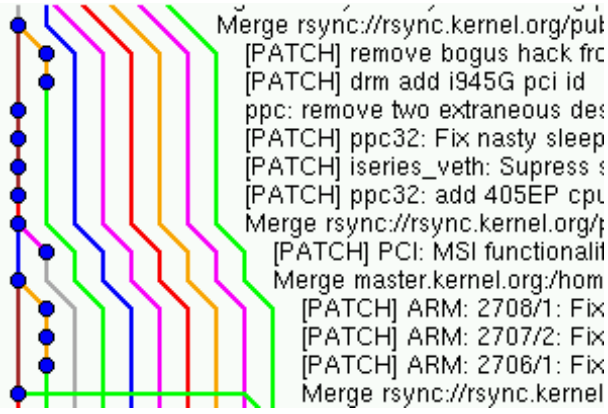

Abbildung 2.3: gitready Visualisierung

Darstellungen anderer Subversion Systeme

- SVN [14]

- GIT [4]
- GIT [15]

2.2 ENTWURF

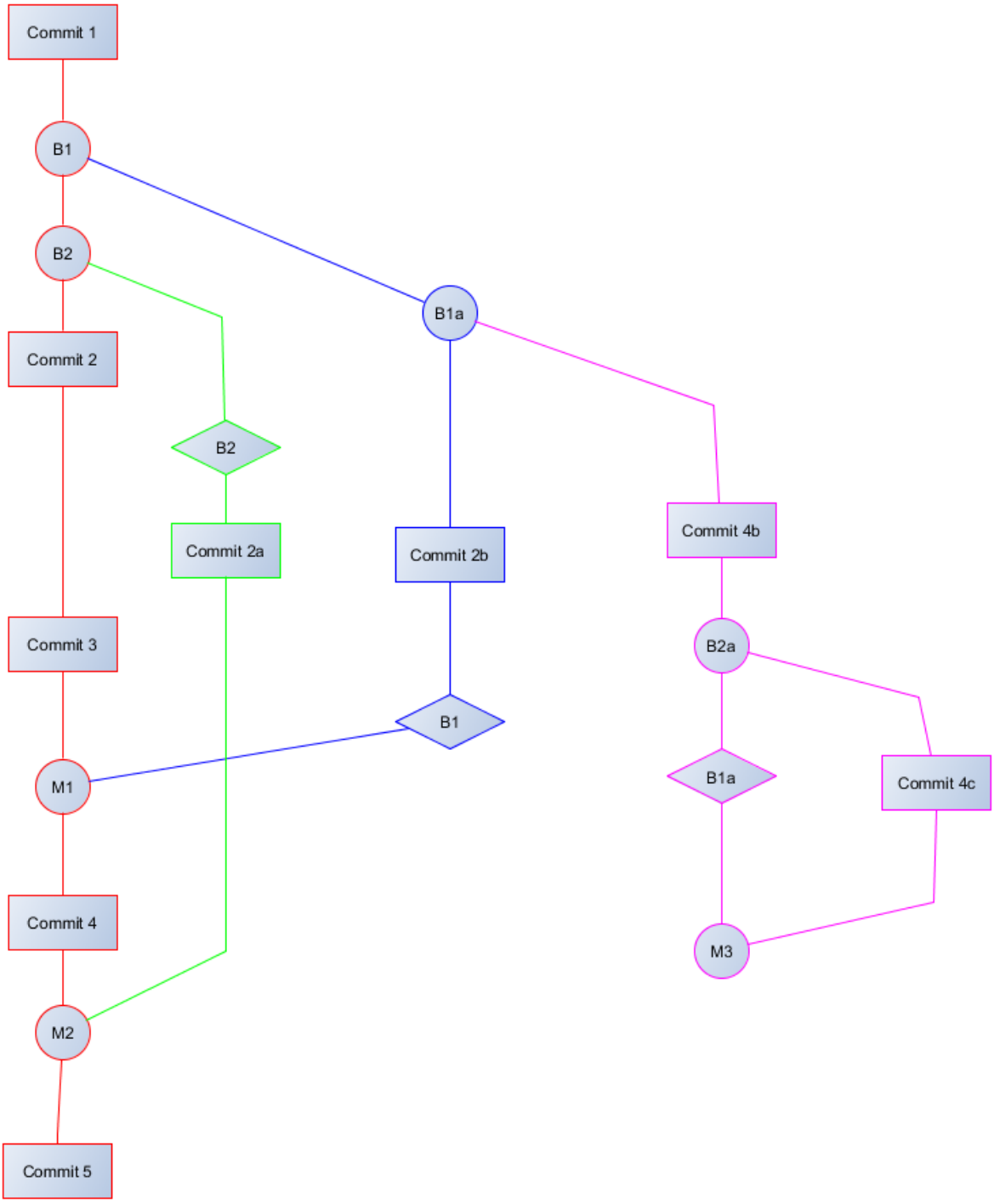

Abbildung 2.4: Erste Skizze

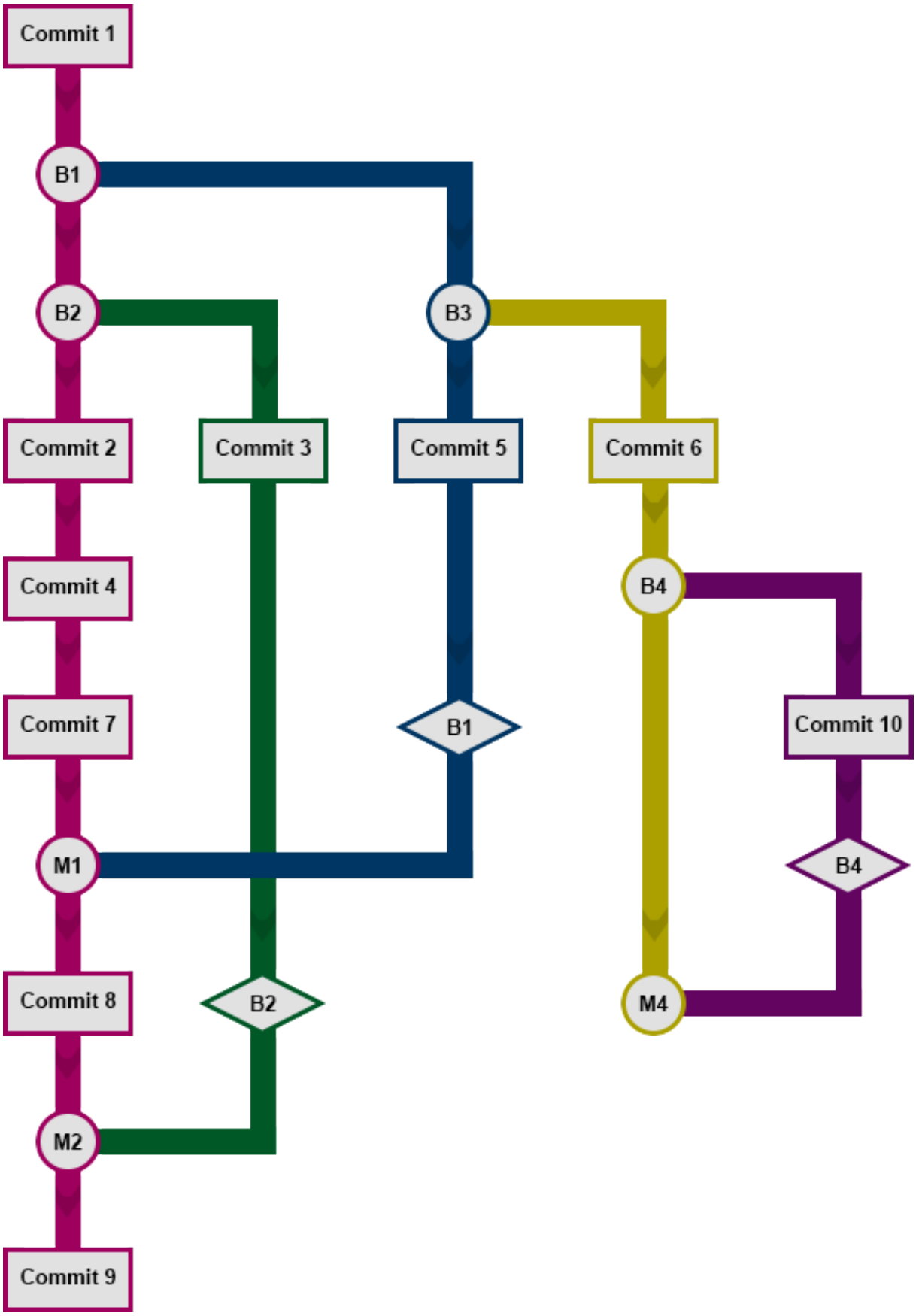

Abbildung 2.5: Überarbeitete Skizze

3 PFLICHTENHEFT

Die Erstellung des Pflichtenhefts erfolgt entsprechend der Empfehlungen des IEEE Guide for Software Requirements Specifications (SRS) [16]. Die dort empfohlene Struktur wird in deutscher Übersetzung nach Balzert [17] übernommen.

3.1 PRODUKTFUNKTIONEN

3.1.1 MUSS

- Daten (bspw. aus turtle-Datensatz) auslesen (zum weiterverarbeiten)
- Daten werden graphisch abgebildet

3.1.2 KANN

- Daten werden in in unterschiedlichen Darstellungsebenen dargestellt (Gesamt- und Detailübersicht)
- Einzelansicht für die Details von Revisions
- Daten werden tabellarisch-strukturiert abgebildet

3.1.3 SOLL

- Zweckmäßige Übergangsanimationen zwischen Darstellungsansichten

3.2 QUALITÄTSANFORDERUNGEN

- Daten werden unverfälscht abgebildet
- Durch Informationsreduzierung (auf das nötigste) wird ein höhere Übersichtlichkeit erreicht (Minimierung der Darstellung von merges und revisions durch Unterteilung in Detailansichten)

3.3 EINSCHRÄNKUNGEN UND RANDBEDINGUNGEN

- Läuft auf neuerem Firefox und Chrome

3.4 ANNAHMEN UND ABHÄNGIGKEITEN

- Nutzer ist mit r43ples und dem semantic web vertraut

4 GESTALTUNGSENTWURF

- Framework zu svg-Generierung: Apache Batik
  - implementiert die Java Graphics2D API Code zum Zeichnen kann wiederverwendet werden, z.B. auf einem AWT Canvas
- Neue Datenstruktur „StructuredTree“ bildet den Commitgraph ab und vereinfacht das Arbeiten mit den Commits
  - beinhaltet doppelt verknüpften Graphen der Commits, Liste der Branches und Liste der Tags, beides jeweils mit Referenzen auf die Commits
  - kapselt die Logik zum Generieren der Datenstruktur aus dem SPARQL Result
- Trennung von Commit-Graph und Zusatzinformationen durch Regel: Ein Commit pro Zeile
  - bessere Übersichtlichkeit für den Anwender, da die Beschriftungen nicht die Außmaße des Graphen vergrößern Graph bleibt kompakt, geringe Augenbewegungen nötig um den Graphen vollständig zu erfassen.
  - Reihenfolge der Commits sofort erkennbar, da pro Zeile nur ein Commit
  - Um von Commit-„Punkt“ im Graphen zur Beschriftung zu gelangen nur eine geradlinige horizontale Augenbewegung nötig trotz Trennung hohe Lesegeschwindigkeit (siehe Literatur: „Information Visualisation“)
- Commit-Graph verläuft in Spalten, pro Spalte ein Branch gute Übersichtlichkeit
  - verschiedene Spalten außerdem verschieden eingefärbt ebenfalls Erhöhung der Übersichtlichkeit
  - Farbwahl sollte der Ermüdung des Auges verbeugen
- Zeitpunkt des Commits wird nur tagesgenau wiedergegeben + Ausgabe des Datums nur wenn Änderung zu vorhergehendem Commit Herabsetzen des Informationsgehaltes und Vermeiden von Redundanzen damit der Anwender die Daten schneller überblicken kann

LITERATURVERZEICHNIS

[1] Pascal Hitzler. *Semantic Web / Grundlagen*. Springer, 2008.

[2] Markus Graube, Stephan Hensel, and Leon Urbas. R43ples: Revisions for triples. *1st Workshop on Linked Data Quality*, 09 2014.

[3] git-SCM. http://git-scm.com/.

[4] Git auf dem Server - GitWeb. http://git-scm.com/book/de/v1/Git-auf-dem-Server-GitWeb.

[5] V. V. Borisenko, A. P. Lakhno, and Chepovskiy A. M. Special graph representations and visualization of semantic networks. *Journal of Mathematical Sciences*, 185, 08 2012.

[6] Benjamin Brandl. Ein empirischer vergleich von semistrukturierter und unstrukturierter konfliktbehandlung in versionsverwaltungssystemen. Master's thesis, Universität Passau, 04 2011.

[7] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising linked data: A survey. *Semantic Web 1*, 2011.

[8] Raga'ad M. Tarawneh, Patric Keller, and Achim Ebert. A general introduction to graph visualization techniques. *IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop*, 2011.

[9] Joseph Goldberg and Jonathan Helfman. Eye tracking for visualization evaluation: Reading values on linear versus radial graphs. *Information Visualization*, 7 2011.

[10] Vincent Driessen. A successful git branching model. http://nvie.com/archives/323, 01 2010.

[11] DIN Deutsches Institut für Normung e.V. Software-ergonomie für multimedia-benutzungsschnittstellen din en iso 14915-1, 4 2003.

[12] DIN Deutsches Institut für Normung e.V. Software-ergonomie für multimedia-benutzungsschnittstellen din en iso 14915-2, 11 2003.

[13] DIN Deutsches Institut für Normung e.V. Software-ergonomie für multimedia-benutzungsschnittstellen din en iso 14915-3, 4 2003.

[14] Visualisieren von Repositorien mit Gource. http://www.pro-linux.de/kurz-tipps/2/1674/visualisieren-von-repositorien-mit-gource.html.

[15] Das Repository visualisieren. http://gitready.com/intermediate/2009/01/13/visualizing-your-repo.html.

[16] IEEE Computer Society. Software Engineering Standards Committee and IEEE-SA Standards Board. Ieee recommended practice for software requirements specifications. Institute of Electrical and Electronics Engineers, 1998.

[17] Helmut Balzert. Lehrbuch der softwaretechnik: Basiskonzepte und requirements engineering. *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering:, ISBN 978-3-8274-1705-3. Spektrum Akademischer Verlag, 2009*, 1, 2009.