Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik

ERIK BARTON
BACHELOR THESIS

SEMANTIKSPEZIFISCHE VISUALISIERUNG VON BAUMBASIERTEN DIFFS IN WIKIS

Eingereicht am 30. März 2015

Betreuer: Dipl.-Inf. Hannes Dohrn, Prof. Dr. Dirk Riehle, M.B.A.
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg

Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 30. März 2015

License

Erlangen, 30. März 2015

Abstract

Semantics-specific Visualization of Tree-based Diffs in Wikis

In this thesis possible changes by a user in a wiki article are specified and classified. Afterwards a visualization of those changes is presented. Instead of a text based diff, which is for instance used in Wikipedia, a tree based diff is taken as groundwork. Because a text based diff displays unnecessary information and what is more does not recognize contexts between text passages a tree based diff allows a more understandable representation of differences between two versions of an article. The developed representation of those changes will be implemented and integrated into the existing Sweble Wiki. The advantages of this visualization will then be pointed out by a comparison with a row based diff, which for instance can be found in Wikipedia.

Zusammenfassung

Semantikspezifische Visualisierung von baumbasierten Diffs in Wikis

Im Rahmen dieser Arbeit werden mögliche Änderungen durch einen Benutzer an einem Artikel eines Wikis spezifiziert und klassifiziert. Anschließend wird eine Visualisierung dieser Änderungen vorgestellt. Statt des textbasierten Vergleichs, der zum Beispiel von Wikipedia verwendet wird, wird hier ein baumbasierter Vergleich als Grundlage genommen werden. Da ein textbasierter Vergleich dem Nutzer überflüssige Informationen anzeigt sowie Zusammenhänge zwischen Textstellen oft nicht erkennt oder unverständlich darstellt, ermöglicht ein baumbasierter Vergleich eine besser verständliche Darstellung der Differenzen zwischen zwei Versionen eines Artikels. Die erarbeitete Darstellung für diese Änderungen wird implementiert und in das existierende Sweble Wiki integriert. Die Vorteile dieser Visualisierung werden dann mithilfe einer Gegenüberstellung mit einem zeilenbasierten Vergleich, wie z.B. in Wikipedia zu finden ist, aufgezeigt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AST	Abstract Syntax Tree
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
ID	Identifikator
SCSS	Syntactical Awesome Stylesheets CSS
WOM	Wiki Object Model
XML	Extensible Markup Language

1 Einführung

1.1 Ziele der Arbeit

Heutzutage verwenden die meisten Wikis Texte, die mit der Aufzeichnungssprache Wikitext erstellt werden, um Artikel zu bearbeiten und zu speichern. Die *Open Source Research Group* hat einen formellen Parser entwickelt, der ein baumbasiertes Format von Versionen eines Wiki Artikels liefert. Dieses Format wird als *Wiki Object Model* bezeichnet. Die Änderungen in einem Wiki werden in Form von Versionen eines Artikels gespeichert. Diese Arbeit definiert verschiedene Möglichkeiten zur Visualisierung für die spezifischen Operationstypen, die ein Benutzer an einem Artikel vornehmen kann. Diese Visualisierungen werden anschließend in der Sweble Wiki Software implementiert.

2 Forschung

2.1 Einführung

Die Besonderheit an Wiki Plattformen ist, dass Artikel ohne große vorherige Einarbeitungszeit von jedem Benutzer bearbeitet werden können. Dadurch werden der Austausch und die Sammlung von vielen Informationen ermöglicht, da jeder Benutzer sein Wissen zu entsprechenden Artikeln beitragen kann. Aufgrund dieser großen Menge an Benutzern und deren Bearbeitungen ist es besonders wichtig, eine übersichtliche und gut verständliche Darstellungsform zur Anzeige dieser Änderungen anzubieten.

Insbesondere Artikel von Wikipedia dem bekanntesten Vertreter der Wiki Plattformen werden heutzutage in vielen verschiedenen Sprachen erstellt, aufgerufen und bearbeitet. Allein die englische Version von Wikipedia hat 4.688.763 Artikel mit insgesamt 750.840.257 Revisionen. Dazu kommen bei Wikipedia und Schwesterprojekten mehr als 10 Bearbeitungen pro Sekunde (Wikipedia, 2015). Alle diese Änderungen werden in einer Versionsgeschichte zum jeweiligen Artikel gespeichert und jede einzelne Version kann auf Wunsch des Benutzers angezeigt werden. Über die Versionsgeschichte ist es möglich zu jeder Version die gespeicherte Seite aufzurufen, Änderungen zu vorherigen Versionen anzeigen zu lassen, Zeitpunkt und Autor der Bearbeitung nachzuvollziehen sowie eventuell hinzugefügte Kommentare des Nutzers zu lesen (Viègas et al., 2004).
Die große Anzahl an Artikeln sowie die vielen Versionen zu einem Artikel führen zu einer hohen Quantität an Daten und somit auch zu vielen vorhanden Informationen zur Interaktion zwischen potenziellen Autoren eines Artikels (Nunes, Ribeiro & David, 2008).

Um deshalb die Interaktion der Autoren zu unterstützen, muss die Darstellung der Differenzen zwischen zwei Versionen möglichst intuitiv und schnell verständlich sein (Rönnau et al., 2009). Häufig werden diese Differenzen mittels einem textbasierten Vergleich bestimmt und angezeigt. Der textbasierte Algorithmus hat die Nachteile, dass meist überflüssige Informationen für den Nutzer angezeigt, Zusammenhänge zwischen Textstellen oft nicht erkannt oder unverständlich dargestellt sowie syntaktische Feinheiten der Dokumentstruktur nicht erkannt werden (Dohrn & Riehle, 2014). Dagegen ist es möglich, mithilfe eines baumbasierten Vergleichs eine verbesserte Darstellung für den Benutzer anzubieten. Für diesen Vergleich müssen die vorliegenden Versionen eines Artikels in der Form eines Abstrakten Syntaxbaumes abgespeichert sein (Jones, 2003). Aufgrund dieser Speicherung ermöglicht es der baumbasierte Vergleich die Differenzen in nachvollziehbarer Art und Weise darzustellen als ein rein textbasierter Vergleich, da nicht nur der Text, sondern auch die Struktur der einzelnen Versionen in den Vergleich mit einbezogen wird. Aus zwei als AST gespeicherten Versionen erzeugt der baumbasierte Vergleich ein sogenanntes *edit script* (Chawathe, Rajaraman, Garcia-Molina & Widom, 1996). Dieses *edit script* stellt eine Liste von Operationen dar, die nötig sind, die ältere Version eines Artikels durch Anwenden dieser Operationen in die neuere Version umzuwandeln. Der Algorithmus von Dohrn & Riehle (2014) stellt ein Beispiel für einen solchen baumbasierten Vergleich dar und wird in dieser Arbeit als Grundlage verwendet. Dieser verwendet die Datenstruktur *Wiki Object Model*, die auch auf der Struktur eines AST basiert (Dohrn & Riehle, July 2011). Um reinen Text eines Artikels in einem Wiki in diese Datenstruktur umzuformen, kann der *Sweble Wikitext Parser* von Dohrn & Riehle (2011) verwendet werden.

Auf der Basis des erstellten *edit script* wird in dieser Arbeit eine Visualisierung für die einzelnen Änderungen entwickelt. Dabei sollen die Vorteile, die ein baumbasierter Vergleich bietet, eine übersichtliche und schnell verständliche Darstellung ermöglichen. Dafür müssen zunächst mögliche Bearbeitungen von einem Benutzer an einem Artikel spezifiziert und klassifiziert werden, um anschließend eine sinnvolle Darstellung für jede Art von Änderung zu erarbeiten. Diese Darstellungen werden anschließend implementiert und in das Sweble Wiki integriert.[1]

[1] http://sweble.org/

Die nachstehende Arbeit ist folgendermaßen aufgebaut. In Abschnitt 2.2 werden Grundlagen und verwandte Arbeiten zum Thema Visualisierung von Differenzen näher betrachtet. Anschließend werden in Abschnitt 2.3 mögliche Änderungen von Benutzern an einem Artikel analysiert und klassifiziert sowie die Forschungsfrage formuliert. Im folgenden Abschnitt 2.4 wird die Forschungsfrage behandelt und mögliche Visualisierungen dafür erarbeitet. Danach wird die Implementierung für die verschiedenen Darstellungen in Abschnitt 2.5 vorgestellt. Abschnitt 2.6 stellt die entwickelte Visualisierung vor. In Abschnitt 2.7 wird die entwickelte Visualisierung einem textbasierten Vergleich gegenübergestellt und bewertet sowie ein Ausblick über weiterführende Möglichkeiten für die Visualisierung gegeben. Zum Schluss werden die wichtigsten Punkte in Abschnitt 2.8 zusammengefasst.

2.2 Verwandte Arbeiten

In diesem Kapitel werden zunächst in 2.2.1 die nötigen Grundlagen für diese Arbeit erläutert. Anschließend werden verschiedene Ansätze zur Visualisierung von Änderungen in Abschnitt 2.2.2 aufgezeigt.

2.2.1 Grundlagen

Wikitext

Wikitext ist eine Auszeichnungssprache, die eine vereinfachte Alternative zur *Hypertext Markup Language* darstellt. Diese wird zur Bearbeitung von Artikeln in Wikis verwendet und mithilfe der *MediaWiki* Software in HTML umgewandelt, um den Artikel letztendlich im Browser anzeigen zu können. Durch die Verwendung von Wikitext statt HTML soll erreicht werden, dass jeder Benutzer einer Wiki Plattform Artikel bearbeiten kann ohne vorher komplexere Sprachen wie zum Beispiel HTML erlernen zu müssen. Wikitext verwendet spezielle Zeichen zur Formatierung von Textstellen. Mithilfe dieser vorgegebenen Zeichenkombinationen wird zum Beispiel reiner Text für die Anzeige des Artikels zu kursiv oder fett gedruckter Schrift konvertiert.

Für zum Beispiel kursiven Text ist es das Hinzufügen von zwei Apostrophen vor und nach dem zu formatierenden Text oder für fettgedruckten Text drei Apostrophen. Somit wird zum Beispiel ″kursiv″ bei der Anzeige zu *kursiv* oder ‴fett‴ zu **fett**. Es kann aber auch bestimmten Textstellen eine besondere Bedeutung zugewiesen werden. So können unter anderem Überschriften oder Listen durch die entsprechenden Zeichen erstellt werden. Zum Beispiel wird eine nummerierte Liste mithilfe des Symbols # vor den entsprechenden Elementen der Aufzählung erstellt. [2]

Wiki Object Model

Der in dieser Arbeit verwendete Algorithmus zur Erstellung der Differenzen zwischen zwei Versionen basiert auf dem *Wiki Object Model*. Diese Datenstruktur speichert den vorhandenen Wikitext in Form eines Abstrakten Syntaxbaums ab und ist eine Spezifikation der Schnittstelle für den Zugriff auf Elemente des Wikitextes (Dohrn & Riehle, 2011). Erstellt wird der Baum aus Wikitext eines Artikels mithilfe des *Sweble Wikitext Parser*. Ein solcher Baum präsentiert jeweils eine Version eines Artikels. Die Datenstruktur WOM ist folgendermaßen aufgebaut. Zunächst besteht der Baum aus Knoten, die entweder Textknoten oder Elementknoten sind. Dabei beinhalten Textknoten den eigentlichen Wikitext und Elementknoten dienen zur Formatierung des entsprechenden Bereichs, der alle Kindsknoten vom Elementknoten umfasst. (Dohrn & Riehle, July 2011) Abbildung 2.1 zeigt beispielhaft einen solchen WOM Baum.

[2] http://www.mediawiki.org/wiki/Help:Formatting

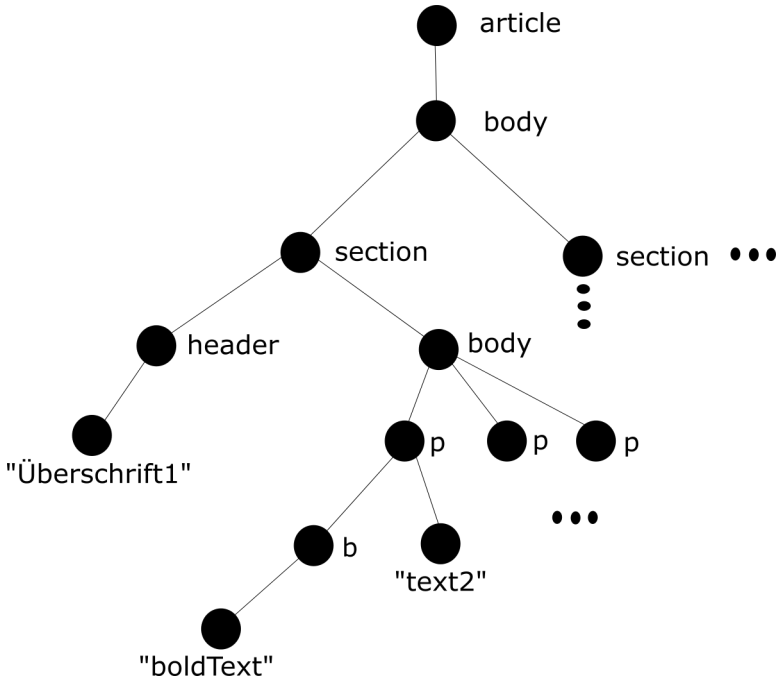

Abbildung 2.1: Exemplarische Darstellung eines WOM Baums

Dabei stellen Knoten, die ohne Anführungszeichen beschriftet sind, Elementknoten dar und die anderen die entsprechenden Textknoten des Wikitextes. Der dargestellte Baum ist ein Artikel, der zunächst einmal aus einem *body* besteht. Dieser *body* teilt sich wiederum in mehrere *sections* auf. Dabei ist hier nur die erste *section* exemplarisch dargestellt. Diese besteht hier aus einer Überschrift, die mittels dem Elementknoten *header* markiert ist, und einem eigenen *body*, der sich wiederum in Paragraphen (hier *p*) aufteilt. Der eigentliche Text dieser *section* befindet sich somit unterhalb dieser Paragraphen und ist eventuell mithilfe von weiteren Elementknoten speziell formatiert. Dies ist in der Abbildung mit dem *b* Elementknoten angedeutet, der fettgedruckte Schrift erzeugt.
Dieser Baum lässt sich sowohl vertikal als auch horizontal durchlaufen. Vertikal meint von einem Vaterknoten zu dessen Kindsknoten und horizontal ist das Durchlaufen der Geschwisterknoten. Des Weiteren kann jeder Knoten ein Set aus Attributen besitzen, das nähere Informationen über den entsprechenden Knoten liefert. (Dohrn & Riehle, July 2011)

Edit script

Der baumbasierte Vergleich von Dohrn & Riehle (2014) erstellt aus zwei Versionen eines Artikels, die als WOM Baum gespeichert sind, ein sogenanntes *edit script*. Dies ist eine Liste von Operationen, die benötigt werden, um die ältere Version des Artikels in die neuere Version umzuwandeln. Somit liefert diese Liste auch alle Änderungen zwischen den beiden Versionen. Die Operationen in der Liste können vom Typ *insert*, *delete*, *move* und *update* sein. In Abbildung 2.2 sind zwei Versionen eines Artikels, gespeichert als WOM Baum, gegenübergestellt und Änderungen zwischen ihnen visuell markiert. Dabei existiert hier jeder Operationstyp genau einmal und ist mithilfe der entsprechenden Beschriftung markiert. Dabei bedeutet *deleted*, dass der entsprechende Knoten der älteren Version gelöscht wurde und nicht mehr im Baum der neuen Version existiert. Des Weiteren zeigen Knoten mit dem Vermerk *inserted* Elemente an, die zum rechten Baum hinzugefügt wurden. Knoten mit *updated* wurden von der älteren Version zur neueren Version verändert. Dagegen sind Knoten mit *moved* inhaltlich unverändert geblieben, aber innerhalb des Baumes verschoben worden. (Dohrn & Riehle, 2014)

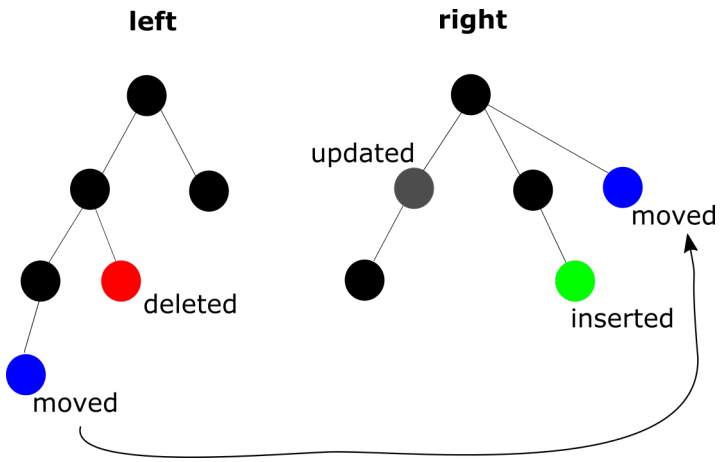

Abbildung 2.2: Gegenüberstellung und Darstellung von Änderungen zweier Versionen eines Artikels, die jeweils als WOM Baum dargestellt sind

Für das dargestellte Beispiel ist das *edit script* die folgende Liste von Operationen:
[*deleted, inserted, updated, moved*]
Diese Liste von Operationen wird für die Visualisierung in dieser Arbeit verwendet (Dohrn & Riehle, 2014).

2.2.2 Ansätze zur Visualisierung von Änderungen

Die Visualisierung der Unterschiede zwischen zwei Versionen eines Artikels basiert in den meisten Fällen auf einem textbasierten Algorithmus. Ein bekanntes Beispiel dafür ist die Anzeige von Wikipedia, bei der ein von Myers entwickelter Vergleichsalgorithmus als Grundlage verwendet wird (Myers, 1986). Dieser Algorithmus wird aufgrund seiner Einfachheit und umfangreichen Einsetzbarkeit in vielen Fällen als Basis zur Erstellung des textbasierten Vergleichs genommen. Die Alternative dafür ist ein baumbasierter Vergleichsalgorithmus. Hier ist beispielhaft der Algorithmus von Dohrn & Riehle (2014) zu nennen, der in dieser Arbeit als Grundlage dient.
Außer der Wahl des Algorithmus, der zum Vergleich verwendet wird, ist es wichtig, die Ergebnisse angemessen darzustellen. Im Artikel von Cicchetti *et al.* wird eine mögliche allgemeine Herangehensweise an die Erarbeitung einer Präsentation von Unterschieden zwischen zwei Versionen eines Dokuments beschrieben. Dabei werden Anforderungen aufgeführt die eine Visualisierung erfüllen sollte, nämlich modellbasiert, minimalistisch, umformend, kompositionell und Metamodell unabhängig. Außerdem ist laut Cicchetti *et al.* der momentane Stand der Technik für Visualisierungen die Verwendung eines *edit script* und die Hervorhebung mithilfe von Farbschemata. Auch in dieser Arbeit ist es das Ziel, die beiden Techniken in der Präsentation zu vereinen und somit eine übersichtliche und intuitive Darstellung anzubieten. (Cicchetti, Ruscio & Pierantonio, 2007)

Von Neuwirth *et al.* werden allgemeine Fragen für die Anzeige aufgegriffen und abgehandelt. Zuerst die Frage, welche Änderungen überhaupt angezeigt werden sollen, damit eine Reizüberflutung vermieden wird, aber weiterhin genug Informationen angezeigt werden. Dabei soll die Auswahl der Änderungen abhängig von der entsprechenden Benutzergruppe gemacht werden und flexibel einstellbar sein. Des Weiteren wird die Frage behandelt, wie die Änderungen, die angezeigt werden, präsentiert werden sollen. Dafür werden einstellbare Optionen für den Benutzer aufgeführt. So soll dieser unter anderem wählen können, wie detailliert Änderungen angezeigt werden und inwiefern dabei umliegender Text um die Änderung herum mit in die Darstellung einbezogen wird. Zuletzt wird die Benutzeroberfläche der Änderungsanzeige angesprochen, wobei vor allem hervorzuheben ist, dass Text vom Benutzer horizontal schneller als vertikal gelesen wird. (Neuwirth et al., 1992)

Ball und Eick geben einen Überblick über konkrete Präsentationsmöglichkeiten von Software, um das Nachvollziehen der Programmierung zu vereinfachen. Diese sind Zeilenpräsentation, Pixelpräsentation, Dokumenten zusammenfassende Präsentation und hierarische Präsentation. (Ball & Eick, 1996)

In "*Graphical Merge Support for XML Document*" von Rönnau *et al.* wird ein Benutzerinterface zur Darstellung von Bearbeitungen vorgestellt. In diesem Interface hat der Benutzer die Möglichkeit mittels Filter die anzuzeigenden Operationen selbst mit zu bestimmen. Zur Darstellung werden zwei XML Bäume (vorher und nachher) verwendet, sowie eine Liste von Operationen auf der rechten Seite. Die einzelnen Operationen werden dabei farblich passend zum entsprechenden Operationstyp in den beiden XML Bäumen hervorgehoben. Abbildung 2.3 zeigt eine Abbildung vom Interface. (Rönnau et al., 2009)

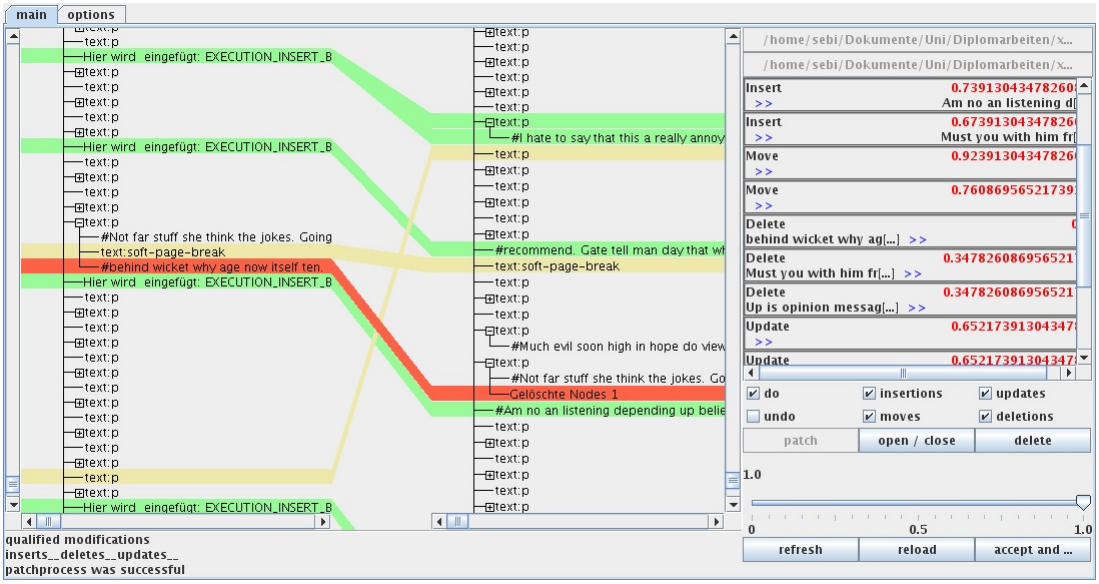

Abbildung 2.3: Benutzerinterface mit zwei XML Bäumen (vorher und nachher) sowie der Liste von Operationen auf der rechten Seite (Rönnau et al., 2009)

Viègas *et al.* beschreiben in ihrem Artikel eine Möglichkeit die komplette Versionsgeschichte zu visualisieren. Dabei präsentieren sie nicht nur zwei Versionen von einem Dokument, sondern zeigen die komplette Versionsgeschichte an. Dafür werden die Bearbeitungen von einem Autor in einer zugewiesenen Farbe angezeigt, wodurch der Nutzer einen Überblick über die Änderungen und deren Urheber an einem Artikel erhält. In Abbildung 2.4 ist eine exemplarische Darstellung für einen Artikel zu sehen. (Viègas et al., 2004)

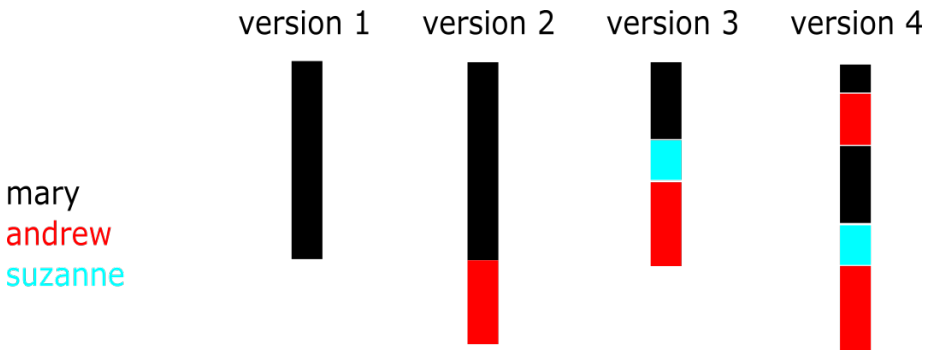

Abbildung 2.4: Visualisierung der kompletten Versionsgeschichte am Beispiel von vier Versionen und drei Autoren von einem Artikel
In Anlehnung an (Viègas et al., 2004)

2.3 Forschungsfrage

Bevor die Visualisierung für die Änderungen entwickelt werden kann, müssen zunächst mögliche Bearbeitungen durch einen Benutzer betrachtet werden. Benutzer von Plattformen wie Wikipedia haben bei der Bearbeitung von Artikeln viele verschiedene Möglichkeiten, die von der verwendeten Syntax der entsprechenden Seite vorgegeben wird. Bei Wikis, die als Software *MediaWiki* verwenden, ist dies der sogenannte Wikitext (engl. *wiki-markup*). Diese Auszeichnungssprache stellt eine vereinfachte Alternative zu HTML dar, die mögliche Nutzer innerhalb kurzer Zeit nachvollziehen können und somit auch die Möglichkeit haben einen Artikel zu bearbeiten, ohne vorher viel Arbeit in das Erlernen der Syntax zu investieren. Im Folgenden werden diese möglichen Bearbeitungen identifiziert und klassifiziert.

Zunächst werden Änderungen an einem Artikel von Wikitext in drei Kategorien aufgeteilt. Das sind die Änderungen an Seiten Metainformationen, die Bearbeitung des Inhalts und die Änderungen, die zum Verschieben einer Seite innerhalb der Plattform führen.
Die Änderungen an Seiten Metainformationen beinhaltet unter anderem die Konfiguration von Lese- und Schreibrechten für Artikel eines Wikis. Diese bestimmen die Rechte zum Aufruf und zur Bearbeitung eines Artikels für festgelegt Benutzergruppen. Da diese Änderungen meist nur von Administratoren vorgenommen werden dürfen und der normale Benutzer keine Möglichkeiten hat daran Einstellungen vorzunehmen, wird dieser Bereich in der Arbeit nicht näher behandelt. Außerdem wäre eine solche Änderung nur umständlich zu visualisieren und würde kaum Nutzen für den Betrachter von Differenzen zwischen zwei Versionen mit sich bringen.
Anders als bei der Bearbeitung der Seiten Metainformationen ist das Merkmal von Wikis, dass jeder Benutzer die Möglichkeit hat, einen Artikel inhaltlich zu bearbeiten, insofern keine bestimmten Rechte dafür nötig sind. Dieser Fall der Einschränkung wird hier nicht weiter betrachtet und es wird davon ausgegangen, dass jeder Benutzer alle Rechte zum Bearbeiten des Wikitextes hat.

Die Bearbeitung des Inhalts einer Seite lässt sich wiederum in zwei Unterpunkte aufteilen, nämlich formale Änderungen und inhaltliche Änderungen. Zur formalen Änderung zählen Rechtschreibkorrekturen, Formatierungsänderungen sowie das sogenannte Wikifizieren von Artikeln. Beispiele für Formatierungsänderungen sind *Whitespace* Änderungen, also Bearbeitungen von Leerraum, oder auch das Formatieren der Schrift an einzelnen Textstellen, wie z.B. kursiver Text. Wikifizieren ist ein Begriff, der neu eingeführt wurde mit der Entstehung von Wikis. Spricht man von Wikifizieren, so ist damit das Verlinken von Schlagwörtern sowie das Anpassen an ein einheitliches Layout gemeint. Dies ist vor allem bei reinem Text ohne Formatierung der Fall, damit das spätere Lesen des Artikels einfacher und verständlicher ist. (Wikipedia, 2014)
Auch die inhaltlichen Änderungen können verschiedene Gründe und Ziele haben. So kann ein vorhandener Text umformuliert werden oder falsche Angaben korrigiert werden. Dies kann eine unverständlich ausgedrückte Textpassage sein oder auch ein fehlerhaftes Datum, das von einem Benutzer angepasst wird. Weiterhin werden Inhalte, die hinzugefügt bzw. entfernt werden, auch zur inhaltlichen Änderung gezählt. Diese Inhalte können Textstellen sein, aber auch Bilder, mathematische Formeln, Vorlagen, Funktionen, Musik Notationen usw. Die Möglichkeiten für verwendbare Inhalte werden dabei von der verwendeten Software und deren Syntax vorgegeben. Die Größe des eingefügten bzw. gelöschten Inhalts ist dabei abhängig von der Bearbeitung, die der Nutzer vornehmen möchte. Eine weitere inhaltliche Änderung stellt die Verschiebung von Inhalten innerhalb des Artikels dar. Dies tritt zum Beispiel auf, wenn ein Benutzer bestimmte Inhalte an einer anderen Stelle im Wikitext für sinnvoller hält und den Text dementsprechend anpasst.
Die letzte Kategorie ist die Bearbeitungen eines Artikels, die zur Verschiebung von kompletten Seiten führen. Diese können unterschiedliche Ursachen haben. Die einfachsten beiden Optionen sind die Erstellung einer neuen Seite und das Umbenennen eines bereits vorhandenen Artikels. Beides führt zu einem neuen Eintrag des Artikels innerhalb der Plattform. Der Benutzer kann aber auch durch bestimmte Befehle bewirken, dass Teile einer Seite in einen neuen Artikel ausgelagert werden oder mehrere vorhandenen Seiten in einen neuen gemeinsamen Artikel integriert werden.

Da diese Kategorie von Änderungen ihre Wichtigkeit eher für den Bereich der Speicherung von Artikeln in einer Plattform hat und nur bei der Auswahl der entsprechenden Version eines Artikels eine Rolle spielt, ist diese Form der Änderung für die eigentliche Visualisierung, genauso wie die Änderungen an Seiten Metainformationen, irrelevant. Allerdings ist es generell wichtig, dass bei der Auswahl in der Versionsgeschichte angezeigt wird, wenn Versionen eines Artikels verschoben wurden und somit an anderer Stelle abgespeichert sind.
Somit bleiben noch die inhaltlichen Änderungen für die Visualisierung von Differenzen. Von diesen Änderungen muss bei der Umsetzung der Visualisierung entschieden werden, welche Bearbeitungen bei der Anzeige der Differenz zwischen zwei Versionen berücksichtigt werden sollen und welche Bearbeitungen unwichtige Informationen für den Benutzer liefern und somit außer Acht gelassen werden können. Des Weiteren muss für jede Änderung entschieden werden, wie diese übersichtlich und verständlich für den Benutzer angezeigt werden soll. Diese Fragen werden im kommenden Abschnitt Forschungsansatz detailliert beantwortet.

2.4 Forschungsansatz

Im vorherigen Kapitel wurden mögliche Änderungen an Wikitext von einem Artikel durch einen Benutzer spezifiziert und klassifiziert. Von den drei Kategorien ist der Hauptpunkt dieser Arbeit die Änderung des Inhalts eines Artikels und die Visualisierung dieser Änderungen. Die Änderung der Seiten Metainformationen und die Bearbeitungen, die zum Verschieben von Seiten führen, werden im Folgenden nicht weiter angesprochen, da diese für die Darstellung der Differenzen zwischen zwei Versionen zunächst nicht von Relevanz sind.
Die Grundlage für die Präsentation dieser Änderungen ist die Speicherung von Versionen eines Artikels als WOM Baum nach Dohrn & Riehle (2011) und der Erstellung des *edit scripts* bei der der Algorithmus diese Baumstruktur verwendet (Dohrn & Riehle, 2014). Das *edit script* besteht aus einer Liste von Operationen. Diese Operationen können vom Typ *insert*, *delete*, *move* und *update* sein und stellen alle Änderungen zwischen den zu vergleichenden Versionen eines Artikels dar. Mithilfe dieser Liste von Operationen wird die Visualisierung der Unterschiede umgesetzt.

Für die Präsentation der Differenzen muss zunächst die Struktur der angezeigten HTML-Seite erarbeitet werden. Im Benutzerinterface von Rönnau *et al.* (2009) wird die Anzeige in zwei Teile aufgeteilt. Zum einen die Visualisierung der Änderungen und zum anderen die Auflistung aller vorkommenden Operationen sowie Möglichkeiten zur Einstellung von Anzeigefiltern. Diese Aufteilung wird auch für diese Visualisierung mit der Einschränkung übernommen, dass die Visualisierung und die Auflistung der Operationen nicht nebeneinander, sondern untereinander angezeigt werden. Das hat den Grund, das die eigentliche Visualisierung im Vordergrund stehen soll und die Auflistung der Operationen als Hilfestellung für den Benutzer bei vielen vorhandenen Operationen dient. Auch das Konzept für Anzeigefilter wird übernommen und als Checkboxen für die Auswahl der anzuzeigenden Operationstypen in die Anzeige integriert. Generell wird für die Anzeige jedem Operationstyp eine Farbe zugewiesen, um diese voneinander unterscheiden zu können. Gelöschte Elemente, also der Operationstyp *delete*, werden mittels der Farbe rot hervorgehoben. Einfügte Elemente, also Operationen vom Typ *insert*, werden mit der Farbe grün und verschobene Elemente, also *move* Operationen, mit der Farbe blau dargestellt.
Da für die eigentliche Visualisierung der Änderungen nicht der komplette Wikitext der Versionen von Bedeutung für den Benutzer ist, sondern die eigentlichen Änderungen interessant sind, wird die Anzeige auf Abschnitte mit Änderungen beschränkt. Dabei werden diese Abschnitte als einzelne Blöcke dargestellt und verschieden Abschnitte der beiden zu vergleichenden Versionen gegenübergestellt. Damit soll der Benutzer dabei unterstützt werden, sich auf die konkreten Änderungen konzentrieren zu können und nicht mit zu vielen Informationen überflutet zu werden. Die Markierung der Änderungen in den Blöcken wird mithilfe der Anpassung vom Schrifthintergrund umgesetzt. Dabei werden die entsprechenden Farben für die Operationstypen verwendet. Da eine *update* Operation die Änderung eines Elements im Wikitext bedeutet und dies somit mithilfe von eingefügten und gelöschten Zeichen dargestellt werden kann, wird dieser Operationstyp auch durch die entsprechenden Farben von *insert* und *delete* markiert.
Durch diese Visualisierung der einzelnen Textstellen soll dem Nutzer beim Aufrufen dieser Seite sofort klar sein, welche Änderungen vorgenommen wurden.

Zusätzlich soll dem Benutzer mit der Auflistung der Operationen, folglich der Beschreibung der Änderungen, die Möglichkeit gegeben werden, alle vorhandenen Änderungen nachzuvollziehen. Diese werden entsprechend ihres Operationstyps in einer Liste aufgeführt. Somit kann der Benutzer sofort sehen, wie viele Änderungen vom jeweiligen Typ existieren. Außerdem werden dem Benutzer eventuell ineinander verschachtelte Änderungen angezeigt.

Der genaue Aufbau der HTML-Seite und die Umsetzung der Visualisierung der einzelnen Operationstypen wird im Abschnitt 2.6 näher beschrieben.

2.5 Implementierung

In diesem Abschnitt wird die Architektur der Implementierung näher betrachtet und erläutert. Die Struktur der Seite für die Präsentation wird durch die HTML-Seite *CompareRevisionsPage* vorgegeben. Dadurch ist auch der Bereich festgelegt in dem die Darstellung und Beschreibung der Änderungen angezeigt werden. Zusätzlich zur Präsentation der Änderung sind Checkboxen implementiert. Diese sind mithilfe von HTML-Elementen dargestellt und deren Funktionalität durch die innere Klasse *RefForm* in der Klasse *CompareRevisionsPage* festgelegt. Für die Umsetzung der Visualisierung wird dann zunächst jeweils ein Objekt der Klassen *DiffPrinter* und *DiffDescriber* erzeugt. Dabei ist die Instanz der Klasse *DiffPrinter* für die Umsetzung der visuellen Gegenüberstellung der zu vergleichenden Versionen zuständig und die Instanz der Klasse *DiffDescriber* für die Auflistung der vorkommenden Änderungen.
Zur Anzeige der Gegenüberstellung erzeugt die Instanz der Klasse *DiffPrinter* die Tabelle und durchläuft die beiden Versionen für den Vergleich mithilfe einer Tiefensuche. Dabei soll bei jedem Knoten überprüft werden, ob eine Operation an diesem Knoten des Baumes durchgeführt wurde. Da das standardmäßige *edit script* aus einer Liste von Operationen besteht, wäre es nötig für jeden einzelnen Knoten diese Liste zu durchlaufen und zu überprüfen, ob der momentane Knoten von einem Element der Liste betroffen ist.

Um das zu vermeiden, wird eine Instanz der Klasse *AugmentedEditScript* erzeugt. Diese fügt alle Operationen als Effekte, die mithilfe der Klasse *NodeEffect* erzeugt werden, zu den betroffenen Knoten hinzu. Somit ist eine direkte Abbildung von einem Knoten des Baumes auf eine Instanz der Klasse *NodeEffect* möglich. Durch diese Abbildung kann bei der Tiefensuche für jeden Knoten sofort bestimmt werden, ob eine Änderung an ihm vorliegt oder nicht. Der Baum wird komplett durchlaufen und für jeden Knoten wird überprüft, ob ein Effekt vorliegt oder nicht. Nach der Ermittlung, ob eine Änderung an dem Knoten vorgenommen wurde, wird die spätere Darstellung des Textes mithilfe von Instanzen der Klasse *DiffFormatter* festgelegt. Dabei existiert für jede zu vergleichende Version ein Objekt dieser Klasse. Anschließend wird überprüft, ob der aktuelle Knoten als eigenes Blockelement dargestellt werden soll. Das ist mithilfe des erweiterten *edit script* umgesetzt. Denn in diesem ist für jeden Typ von Knoten des Baumes festgelegt, ob dieser als Blockelement oder *Inline* Element dargestellt werden soll. Das Resultat dieser Überprüfung wird ebenfalls in der zur Version passenden Instanz der Klasse *DiffFormatter* abgespeichert. Nachdem feststeht, wie der Text darzustellen ist und ob er ein eigenes Blockelement ist, erstellt die Instanz der Klasse *DiffFormatter* den HTML-Text mithilfe der Klasse *DocWriterBase*, die zur Erstellung der einzelnen HTML-Elemente dient. Diese HTML-Elemente werden dann durch die Instanz der Klasse *DiffPrinter* in die Tabelle zur Visualisierung der Änderungen eingefügt.

Um die Auflistung der vorkommenden Änderungen umzusetzen, wird die bereits erwähnte Instanz der Klasse *DiffDescriber* verwendet. Diese erzeugt zunächst für die Operationstypen *insert*, *delete*, *move* und *update* einen entsprechenden Bereich zur Darstellung. Jede Änderung wird somit dem passenden Operationstyp und dessen Liste zugeteilt. Das wird ähnlich wie bei der Gegenüberstellung mithilfe einer Tiefensuche durch den Baum implementiert. Wird ein Knoten mit zugehörigem Effekt gefunden, so wird dieser entsprechend seinem Operationstyp erstellt und der passenden Liste zugeteilt. Die Erstellung dieser HTML-Elemente geschieht mithilfe der Klassen *DocWriterBase* und *EditScriptSubtreePrinter*.

Eine Ausnahme bei der Zuteilung einer Operation zur entsprechenden Liste ist, wenn der aktuell betrachtete Knoten mit der Operation ein Kindsknoten von einem anderen Knoten mit Operation ist. Folglich eine Änderung die in einer anderen Änderung verschachtelt ist. Denn dann werden innere Operationen unter der äußersten Operation und nicht in der Liste für den eigenen Operationstyp aufgelistet.
Somit lassen sich die Rollen der Klassen folgendermaßen zusammenfassen. Die Instanz der Klasse *DiffPrinter* ist für das Erstellen der Tabelle und Einfügen des Textes in die Tabelle zuständig sowie dem Durchlaufen des Baumes zum Auffinden von Effekten. Dabei werden die Einstellungen zur Anzeige des Textes mithilfe der Klasse *DiffFormatter* zwischengespeichert und die HTML-Elemente durch die Klasse *DocWriterBase* erstellt. Die Instanz der Klasse *DiffDescriber* sorgt dagegen für die Umsetzung der Beschreibung der Änderungen, indem sie für die vorkommenden Änderungen die korrekte HTML-Darstellung mithilfe der Klassen *DocWriterBase* und *EditScriptSubtreePrinter* erzeugt und dann diese Elemente der passenden Liste zuteilt.

2.6 Forschungsergebnisse

Im folgenden Abschnitt wird der konkrete Aufbau der Seite für die Präsentation und die einzelnen Visualisierungen sowie die Beschreibung der Operationstypen im *edit script* erläutert.
Zunächst zur allgemeinen Struktur der Seite für die Präsentation der Änderungen. Die Struktur der Seite für den Vergleich lässt sich von oben nach unten aufteilen in den Titel, dem Fenster für Rückmeldungen, der Infobox, der Übersicht über die aktuell zu vergleichenden Versionen, der Checkboxen zur Konfiguration der Anzeige, der Visualisierung der Änderungen und zuletzt der Beschreibung der vorliegenden Bearbeitungen. In Abbildung 3.1 ist der Aufbau der Seite ohne Visualisierung und Beschreibung der Änderungen zu sehen. Diese befinden sich normalerweise unterhalb der Checkboxen, aber sind aus Gründen der Übersicht in dieser Abbildung nicht dargestellt.

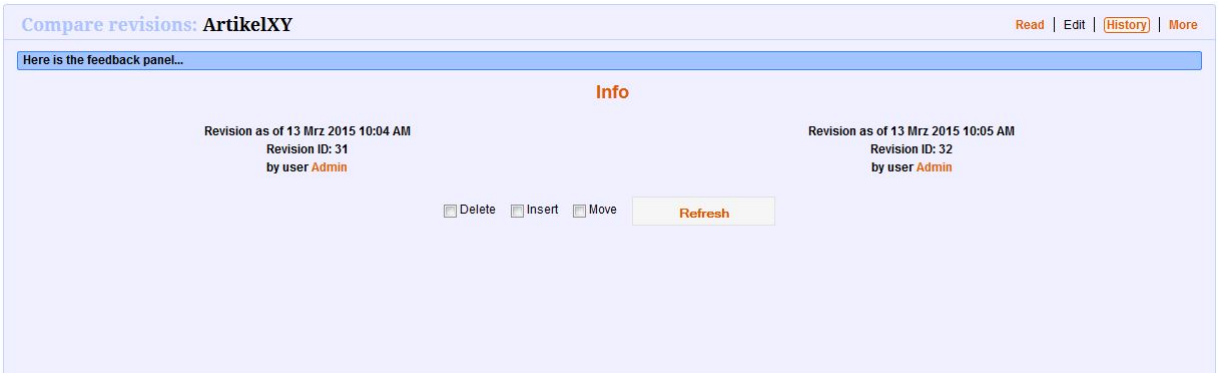

Abbildung 2.5: Aufbau der Seite zur Präsentation der Änderungen im Sweble Wiki

Die Infobox gibt dem Benutzer einen Überblick über die möglichen Einstellungen zur Anzeige und Hinweise für das Verständnis der einzelnen Umsetzungen. Diese wird durch Bewegen des Mauszeigers über das Textfeld Info angezeigt. Die Checkboxen ermöglichen es dem Benutzer zu wählen, welcher Operationstyp (*insert*, *delete* und *move*) aktuell nicht angezeigt werden soll. Dadurch kann der Nutzer die Anzeige auf einzelnen Operationen beschränken, um einen besseren Überblick zu bekommen. Der Operationstyp *update* ist dabei nicht wählbar, da dieser aus den beiden Operationen *insert* und *delete* besteht und somit durch Auswahl dieser aktiviert bzw. deaktiviert wird. Der Titel, das Fenster für die Rückmeldungen und die Übersicht über die aktuell zu vergleichenden Versionen werden automatisch durch den Aufruf generiert und der Seite hinzugefügt. Dabei zeigen der Titel und die Übersicht dem Benutzer die Informationen über die momentan zu vergleichenden Versionen des gewählten Artikels. Dagegen liefert das Fenster für die Rückmeldungen die Möglichkeit, eventuell auftretende Fehler anzeigen zu können.
Die Visualisierung der Änderungen und die Beschreibung der vorliegenden Bearbeitungen sind abhängig von den Unterschieden zweier Versionen. Allerdings gibt es auch hier ein festes Schema zur Darstellung der möglichen Änderungen. Zunächst werden alle Änderungen mithilfe einer Gegenüberstellung der Versionen angezeigt und dabei die alte Version auf der linken Seite und die neue Version auf der rechten Seite angezeigt.

Die Anzeige des Textes wird in Blöcke entsprechend vorhandener Abschnitte aufgeteilt und nur Abschnitte mit Änderungen werden angezeigt. Dabei wird jede Änderung, die auf einen Operationstyp abgebildet ist, auf eine festgelegte Art und Weise optisch hervorgehoben, um dem Benutzer eine Hilfestellung beim Erkennen von veränderten Textstellen zu geben.

Die im Abschnitt Forschungsfrage angesprochenen möglichen Bearbeitungen von Inhalten werden dabei auf den passenden Operationstyp abgebildet. Dies gilt sowohl für die formale als auch für die inhaltliche Änderung im Wikitext. Dabei ist es vor allem bei den formalen Änderungen von großer Bedeutung, inwiefern der Wikitext wirklich bearbeitet wurde. So wird zum Beispiel eine Rechtschreibkorrektur durch den Operationstyp *update* dargestellt, aber eine Formatierung der Schrift oder das Verlinken auf eine andere Seite durch die Operation *insert*. Die Gründe hierfür liegen in der Syntax von Wikitext. Denn bei einer Rechtschreibkorrektur wird Text innerhalb eines Textknotens eingefügt und gelöscht, wohingegen für die Formatierung bzw. Verlinkung bestimmte Zeichen eingefügt werden. Ein Beispiel für die Formatierung zu kursivem Text sind zwei Apostrophen, die vor und hinter dem zu formatierenden Text eingefügt werden. Außerdem wird bei diesem Beispiel ein neuer Elementknoten oberhalb der veränderten Textstelle in den Baum der Version eingefügt, der einen Abschnitt markiert in dem Text kursiv dargestellt werden soll. Dadurch ist es eine reine *insert* Operation und keine *update* Operation. Bei inhaltlichen Änderungen ist das Deuten des Operationstyps dagegen einfacher. Werden Elemente eingefügt oder gelöscht so werden diese durch die Operationen *insert* und *delete* dargestellt. Inhalte, die innerhalb der zu vergleichenden Versionen verschoben wurden, werden auf die Operation *move* abgebildet.

Nachdem jeder Bearbeitung ein Operationstyp zugewiesen wurde, werden diese ihrem Typ entsprechend dargestellt. Eine *delete* Operation wird durch Text auf der linken Seite dargestellt, der durchgestrichen ist. Des Weiteren wird der Schrifthintergrund rot eingefärbt und nicht mehr durchgestrichen, sobald man den Mauszeiger über den entsprechenden Text bewegt. Der Grund dafür ist, dass eine Reizüberflutung für den Benutzer durch zu viele leuchtende Farben vermieden werden soll.

Auf der rechten Seite wird der Text der *delete* Operation nicht angezeigt, da er in der neuen Version nicht mehr enthalten ist. Der durchgestrichene Text und das Fehlen der entsprechenden Textpassage auf der rechten Seite sind ausreichend zur Visualisierung der Änderung. Das Verändern des Schrifthintergrunds beim Bewegen des Mauszeigers über den gelöschten Text ist für nähere Betrachtungen der entsprechenden Stelle gedacht. In Abbildung 2.6 ist eine gelöschte Textpassage ohne Mauszeiger exemplarisch dargestellt und zum Vergleich in Abbildung 2.7 mit Mauszeiger über der entsprechenden Textstelle.

Abbildung 2.6: Visualisierung einer *delete* Operation ohne Mauszeiger über der Textstelle im Sweble Wiki

Abbildung 2.7: Visualisierung einer *delete* Operation mit Mauszeiger über der Textstelle im Sweble Wiki

Der Operationstyp *insert* wird durch einen grünen Schrifthintergrund hervorgehoben. Genauso wie bei der *delete* Operation wird der Text in der anderen Version nicht dargestellt (siehe Abbildung 2.8 linke Seite), da der Text in die neue Version eingefügt wurde und somit in der alten Version nicht vorkommt. In Abbildung 2.8 ist eine eingefügte Textpassage exemplarisch dargestellt.

Abbildung 2.8: Visualisierung einer *insert* Operation im Sweble Wiki

Elemente, die im Artikel verschoben wurden, sind vom Operationstyp *move* und werden zunächst auf beiden Seiten durch blauen Schrifthintergrund markiert. Zusätzlich wird das entsprechende Element in der Version auf der anderen Seite farblich stärker hervorgehoben, wenn man den Mauszeiger über das Element bewegt. In Abbildung 2.9 ist die Darstellung ohne Mauszeiger über ein Element der Operation zu sehen und in Abbildung 2.10 mit Mauszeiger.

Abbildung 2.9: Visualisierung einer *move* Operation ohne Mauszeiger über der Textstelle im Sweble Wiki

Abbildung 2.10: Visualisierung einer *move* Operation mit Mauszeiger über der Textstelle im Sweble Wiki

Zuletzt wird der Operationstyp *update* mittels den beiden bereits beschriebenen Operationen *insert* und *delete* angezeigt.
Eine Ausnahme der standardmäßigen Visualisierung bilden *Whitespace* Änderungen, da sie von den gewöhnlichen *insert* und *delete* Operationen unterschieden werden sollen. Diese werden, wenn sie als Änderung vorkommen, mittels gelben Schrifthintergrund markiert und es wird textuell unter der Visualisierung angemerkt, dass eine solche Änderung vorgenommen wurde. In Abbildung 2.11 ist exemplarisch eine Änderung des Leerraums dargestellt.

Abbildung 2.11: Visualisierung einer *whitespace* Operation im Sweble Wiki

Nach der Gegenüberstellung der Versionen folgt die Beschreibung der vorkommenden Änderungen zwischen den beiden zu vergleichenden Versionen. Dies dient zur kompakten Übersicht über alle vorhandenen Bearbeitungen. Dabei wird jede Operation zu einer Liste hinzugefügt, die ihrem Operationstyp entspricht. Es werden *insert* Operationen untereinander gelistet genauso wie die restlichen Operationstypen. Sollten Operationen ineinander verschachtelt sein, so werden alle Operationen unter der äußersten Operation gelistet, um einen guten Überblick über verschachtelte Änderungen anbieten zu können. Verständlicher wird das an einem kleinen Beispiel. Wurde eine Textstelle an eine neue Textstelle verschoben und außerdem in kursive Schrift umgewandelt, so hat man sowohl eine *move* Operation (Verschieben von Text) als auch eine *insert* Operation (Formatieren der Schrift zu kursiv). Die äußere Operation ist dabei die *insert* Operation. Dadurch wird die *move* Operation unterhalb dieser mit aufgelistet und nicht unter ihrer entsprechenden Liste vom Operationstyp. In Abbildung 2.12 ist dieses Beispiel dargestellt.

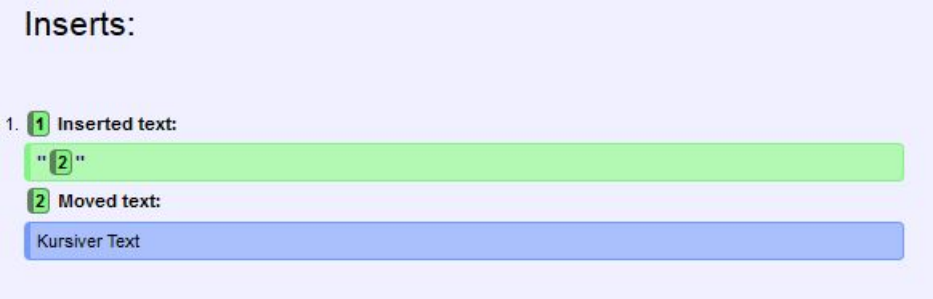

Abbildung 2.12: Beispiel für die Beschreibung der Änderungen im Sweble Wiki

2.7 Diskussion und Ausblick

Der folgende Abschnitt vergleicht die Ergebnisse dieser Arbeit mit anderen Visualisierungen von Änderungen und zeigt mögliche Erweiterungen für die entwickelte Präsentation. Zunächst wird in Abschnitt 2.7.1 die entwickelte Visualisierung mit der Darstellung von Änderungen in Wikipedia verglichen und bewertet. Anschließend befasst sich der Abschnitt 2.7.2 mit Aspekten, die die entwickelte Visualisierung verbessern und erweitern können.

2.7.1 Gegenüberstellung mit der Visualisierung von Änderungen in Wikipedia

Im Folgenden wird die in dieser Arbeit entwickelte Visualisierung mit der Darstellung der Versionsunterschiede in Wikipedia verglichen und Gemeinsamkeiten sowie Vorteile und Nachteile der Präsentationen erläutert. Dafür wurden Artikel in Wikipedia zufällig ausgewählt und zwei Versionen dieses Artikels, zwischen denen einige Bearbeitungen durchgeführt wurden, bestimmt. Diese Versionen wurden in das Sweble Wiki integriert, um eine Präsentation an identischen Texten zu ermöglichen.
Beim ersten Blick auf die beiden unterschiedlichen Darstellungen fällt als Erstes die Gemeinsamkeit auf, dass beide Darstellung mithilfe von Blockelementen umgesetzt wurden. Des Weiteren fällt auch sofort auf, dass im Gegensatz zu Wikipedia eine Anzeige der entsprechenden Zeilennummer fehlt. Dies ist vermutlich darauf zurückzuführen, dass der Vergleich nicht wie bei Wikipedia zeilenbasiert, sondern baumbasiert ist. Dadurch fehlt die Angabe der aktuellen Zeile bei der Bearbeitung einer Änderung im baumbasierten Vergleich und kann somit auch nicht dargestellt werden.
Dagegen ist die Farbwahl bei der entwickelten Visualisierung auffälliger und somit die einzelnen Änderungen auch schneller erkennbar. Denn bei der Präsentation in Wikipedia sind Farben gewählt, die eher matt sind, und somit dem Benutzer nicht auf den ersten Blick auffallen.

Des Weiteren ist es ein Vorteil der Visualisierung im Sweble Wiki, dass mehrere verschiedene Operationstypen farblich hervorgehoben sind. So existiert bei Wikipedia nur die Darstellung für Elemente, die eingefügt oder gelöscht wurden. In der entwickelten Visualisierung dagegen gibt es zusätzlich den Operationstyp *move*, der verschobenen Text beschreibt, und die Hervorhebung für *Whitespace* Änderungen. Dadurch bekommt der Benutzer bereits bei der Betrachtung der Seite einen genaueren Überblick über die Bedeutung der einzelnen Bearbeitungen. Auch die Markierung der eingefügten oder gelöschten Elemente ist bei der Visualisierung im Sweble Wiki detaillierter. Wurde zum Beispiel nur ein Buchstabe in ein Wort eingefügt, so wird in Wikipedia das komplette Wort markiert und nicht wie in der entwickelten Visualisierung nur der eingefügte Buchstabe. Trotz der Vorteile bei der Art und Weise der Markierung der Textstellen hat diese auch ihre Nachteile. Denn bei zum Beispiel vielen verschobenen Textstellen hintereinander geht der Überblick für den Benutzer über die eigentliche Änderung verloren.
Der größte Pluspunkt für die Visualisierung im Sweble Wiki ist aber die Auflistung und Beschreibung der vorkommenden Änderungen. Damit erhält der Benutzer einen Überblick über alle Änderungen und kann somit einzelne Änderungen detaillierter betrachten.

2.7.2 Mögliche Aspekte zur Erweiterung der Visualisierung

In diesem Abschnitt wird ein Ausblick über mögliche Erweiterungen der entwickelten Visualisierung von Änderungen gegeben. Ein möglicher Ansatzpunkt ist zunächst einmal der Aufbau der allgemeinen Darstellung. In dieser Arbeit wurde eine Gegenüberstellung der beiden zu vergleichenden Versionen eines Artikels verwendet. Eine Alternative dazu ist es nur die neue Version anzuzeigen. Anstatt eine Visualisierung mit vorher und nachher zu erstellen, wird der Wikitext der aktuellen Version angezeigt und darin die Änderungen visualisiert. Zusätzlich dazu wäre es zum Beispiel sinnvoll die gelöschten Elemente zusätzlich am Rand anzuzeigen, um dem Benutzer diese als Zusatzinformation zu geben. Somit hätte man eine Darstellung bei der nicht der Vergleich zwischen Versionen im Mittelpunkt steht, sondern der Wikitext der aktuellen Version.

Ein weiterer Punkt für Erweiterungen ist zum Beispiel in der Beschreibung der Änderungen. Dort könnte man anstelle der Auflistung von ineinander verschachtelten Operationen unterhalb der äußersten Operation eine automatische Erkennung von höherwertigen Operationen einfügen, wenn eine solche vorliegt. Eine höherwertige Operation ist zum Beispiel das Verlinken von Schlagwörtern. In der entwickelten Visualisierung wird das noch mithilfe einer *insert* Operation und einer *move* Operation beschrieben. Dies ist exemplarisch in Abbildung 2.13 dargestellt.

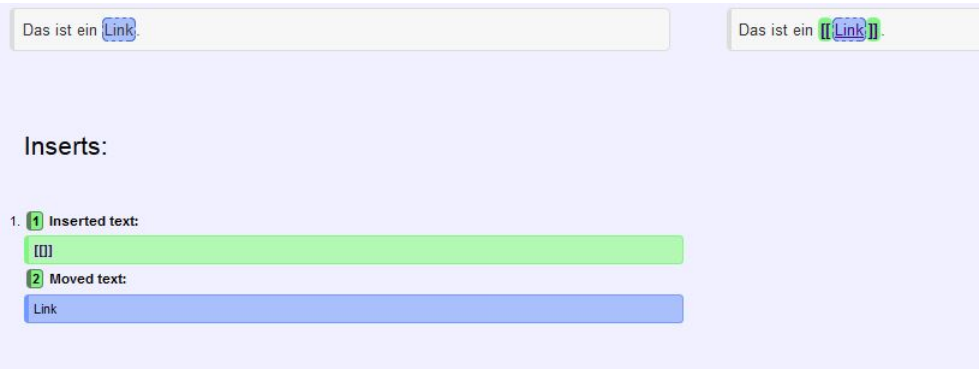

Abbildung 2.13: Beschreibung einer Verlinkung eines Schlagwortes im Sweble Wiki

Die alternative Darstellung dazu wäre zum Beispiel textuell zu beschreiben, dass eine Verlinkung an dieser Stelle vorliegt.

2.8 Zusammenfassung

Im Laufe dieser Arbeit wurde die Entwicklung der Visualisierung von Differenzen zwischen zwei Versionen eines Artikels beschrieben. Zuerst wurde in der Einführung die wachsende Bedeutung von Wiki Plattformen sowie die Notwendigkeit einer einfachen und intuitiven Visualisierung von Bearbeitungen eines Artikels dargestellt. Die Visualisierung kann auf der Basis eines textbasierten oder eines baumbasierten Vergleichs erzeugt werden. Da ein textbasierter Vergleich dem Nutzer überflüssige Informationen anzeigt sowie Zusammenhänge zwischen Textstellen oft nicht erkennt oder unverständlich darstellt, ermöglicht ein baumbasierter Vergleich eine verständlichere Darstellung der Änderungen zwischen zwei Versionen eines Artikels. In dieser Arbeit wurde dafür die WOM Datenstruktur verwendet sowie der auf dieser Baumstruktur basierende Algorithmus von Dohrn & Riehle (2014), der als Ergebnis ein *edit script* liefert. Im Abschnitt Verwandte Arbeiten wurden zunächst die nötigen Grundlagen vorgestellt und anschließend bereits existierende Präsentationen von Versionsunterschieden aufgezeigt, um einen Überblick über den momentanen Stand der Technik zu erhalten. Außerdem wurden nötige Anforderungen für die Visualisierung von Änderungen aufgezählt. In der Forschungsfrage wurden daraufhin zunächst allgemein mögliche Änderungen von Benutzern an einem Artikel, geschrieben in Wikitext, spezifiziert und klassifiziert. Des Weiteren wurden Fragen, die sich mit der Auswahl und Anzeige von Änderungen auseinandersetzen, für die eigentliche Umsetzung der Visualisierung formuliert. Im nächsten Abschnitt Forschungsansatz wurden die vorher kategorisierten Änderungen aufgegriffen und mögliche Darstellungsformen dafür entwickelt. Dabei wurde sowohl ein Konzept für die Visualisierung der Änderungen als auch für die Beschreibung dieser Änderungen erstellt. Dieses wurde mithilfe von verschiedenen Farbgebungen, Auswahlfiltern und weiteren optischen Unterstützungen umgesetzt. Die Implementierung dieses Konzepts ist in Kapitel Implementierung beschrieben. Die grobe Struktur der HTML-Seite sowie die letztendliche Darstellung der Operationstypen wurden in Kapitel Forschungsergebnisse erläutert. Im letzten Kapitel Diskussion und Ausblick wurde die entwickelte Darstellung mit einem textbasierten Vergleich gegenübergestellt und bewertet sowie ein Ausblick über mögliche Erweiterungen der Visualisierung gegeben.

3 Ausarbeitung der Forschung

In diesem Kapitel wird die Implementierung der Visualisierung detailliert vorgestellt. Zu Beginn wird in Abschnitt 3.1 die Struktur der Implementierung näher erläutert. Im folgenden Abschnitt 3.2 wird die Umsetzung der Visualisierung der einzelnen Änderungen beschrieben. Abschnitt 3.3 behandelt anschließend Implementierungsdetails der Beschreibung der vorkommenden Operationen.

3.1 Struktur

In diesem Teilabschnitt wird die Struktur der Implementierung zur Anzeige von Versionsunterschieden beschrieben. Die Umsetzung wurde in das bestehende Sweble Wiki integriert.[1] Für die Implementierung wurden die Sprachen Java, HTML, JavaScript und Cascading Style Sheets verwendet. Dabei wurde für CSS der Präprozessor SCSS benutzt.

3.1.1 Aufbau der HTML-Seite

Der grobe Aufbau der Seite wird durch HTML-Seite *CompareRevisionPage* festgelegt und besteht aus den festen Elementen Titel, Feedback, Infobox, Informationen zu den beiden Versionen, Checkboxen und zuletzt der Abschnitt zur Visualisierung und Beschreibung der vorhandenen Änderungen. In Abbildung 3.1 ist der Aufbau der Website dargestellt.

[1] http://sweble.org/

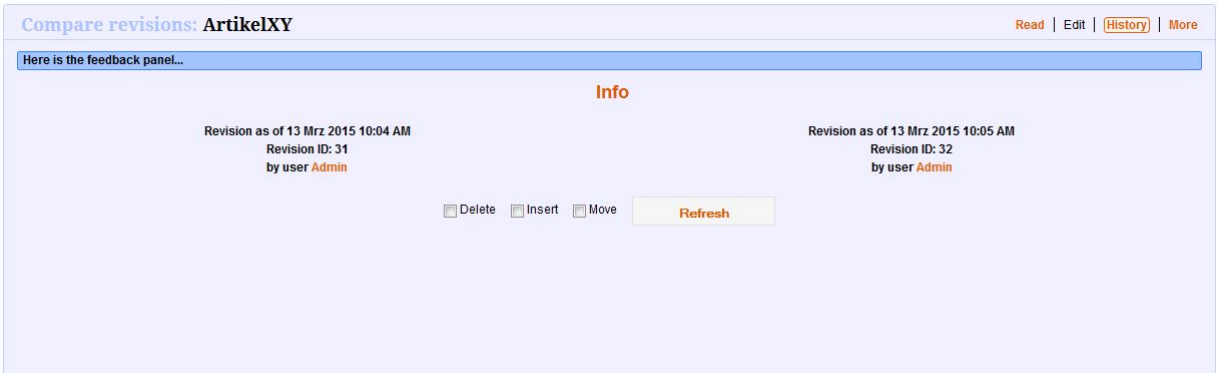

Abbildung 3.1: Aufbau der Seite zur Visualisierung von Änderungen im Sweble Wiki

Der Titel und die Informationen zu den beiden Versionen zeigen den aktuell betrachteten Artikel und die zu vergleichenden Versionen an. Zur Rückmeldung bei auftretenden Fehlern dient das Feedback Feld. Die Infobox gibt dem Benutzer die Möglichkeit einen Überblick über die möglichen Einstellungen sowie ein Verständnis für die vorliegenden Visualisierungen zu bekommen. Diese Infobox wird mittels einer Mausbewegung über das HTML-Element Info angezeigt. Diese Funktionalität wurde mithilfe der *hover* Option in der SCSS Datei *_config* implementiert.
Des Weiteren wird dem Benutzer die Funktion zur Verfügung gestellt mithilfe der Checkboxen auszuwählen, welcher Operationstyp von *insert*, *delete* und *move* nicht angezeigt werden soll. Durch Auswählen der entsprechenden Felder und bestätigen werden der Klasse *CompareRevisionPage* die ausgewählten Optionen als *PageParameter* hinzugefügt, um diese zu speichern und später weiter verwenden zu können. Der Rest der Seite wird zur Anzeige und Beschreibung der Änderungen benutzt und wird in Abschnitt 3.2 und 3.3 detaillierter beschrieben.

3.1.2 Zusammenhang der einzelnen Klassen

Wird im Sweble Wiki eine Anfrage zum Vergleich von verschiedenen Versionen gesendet, wird zuerst mithilfe der Java Klasse *CompareRevisionPage* eine Grundversion der HTML-Seite erstellt. Diese hat am Anfang als *PageParameter* die Nummern der zu vergleichenden Versionen. Sollte der Benutzer die Checkboxen zur Einstellung der Anzeige verwenden, so werden dann auch die entsprechenden Variablen zu den *PageParameter* hinzugefügt, nämlich *hideDel*, *hideIns* und *hideMov*. Die Klasse *CompareRevisionPage* erstellt dann eine neue Instanz der Klasse *TreeDiffPanel*, um die Präsentation der Änderungen zu erstellen. Dieser Instanz wird beim Erzeugen die Parameterliste der *PageParameter* übergeben. Diese Parameter werden zur Erstellung der Visualisierung und Beschreibung der Änderungen benötigt. Dafür wiederum werden die Parameter und der entsprechende Wikitext der Versionen an die Funktion *generateDiff* der Klasse *TreeDiffTools* weiter geleitet. Dort wird der Wikitext der beiden Versionen zu jeweils einem Baum im *Wiki Object Model* Format konvertiert und es wird anschließend das *edit script* mittels der Klasse *HDDiff* erzeugt (Dohrn & Riehle, July 2011, 2014). Dieses *edit script* besteht aus einer Liste von Objekten der Klasse *EditOp*. Diese Objekte haben einen der Operationstypen *insert*, *delete*, *move* und *update*. Um diese Liste von Operationen im Weiteren sinnvoll verwenden zu können, wird das *edit script* anschließend mittels der Klasse *AugmentedEditScript* erweitert. Diese Erweiterung erstellt eine Abbildung von einem Knoten (*DiffNode*) auf den entsprechenden *NodeEffect*. Ein solcher *NodeEffect* bestimmt für einen Knoten im Baum, ob der Knoten selbst oder ein Kindsknoten von einer Operation betroffen ist. Somit kann man nun von einem Knoten im Baum direkt folgern, ob eine Operation vorliegt, und muss nicht die Liste von Operationen auf der Suche nach dem entsprechenden Knoten durchlaufen. Dadurch ist es direkt möglich herauszufinden, ob ein entsprechender Knoten im Baum eingefügt, gelöscht, verschoben oder aktualisiert wurde oder ob Kindsknoten eingefügt wurde. Diese Abbildung wird allerdings nur zum linken Baum hinzugefügt, weshalb im Abschnitt Visualisierung der Änderungen nur der linke Baum bei der Erstellung der Visualisierung durchlaufen wird.

Außerdem erhält bei der Erweiterung des *edit script* jede Operation vom Typ *move* eine eindeutige ID, um später direkt mithilfe von JavaScript auf die einzelnen *move* Operationen zugreifen zu können. Zusätzlich bietet die Klasse *AugmentedEditScript* die Möglichkeit für verschiedene Elemente festzulegen und zu überprüfen, ob sie als ein Blockelement dargestellt werden sollen. Das ist nötig, um Textstellen im vorliegenden Wikitext in entsprechende Blöcke einteilen zu können, die auch im Wikitext einen Abschnitt bilden.
Nach dem Erstellen dieses erweiterten *edit scripts* wird zunächst eine Instanz der Klasse *DiffPrinter* erstellt, um die eigentliche Visualisierung, also die Gegenüberstellung der beiden Versionen, zu erzeugen. Danach werden zusammengehörende Operationen vom Typ *insert* und *delete* im *edit script* zusammengefasst, um bei der anschließenden Beschreibung der Visualisierung einen kompakten Überblick über die einzelnen Operationen geben zu können. Dafür wird eine Instanz der Klasse *DiffDescriber* erstellt. Der genaue Ablauf der Visualisierung und Beschreibung werden in den Abschnitten 3.2 und 3.3 erläutert.

3.2 Visualisierung der Änderungen

Für die Visualisierung der Differenzen zwischen zwei Versionen eines Artikels wird eine Instanz der Klasse *DiffPrinter* erzeugt. Dafür wird dieser eine Instanz der Klasse *AugmentedEditScript*, einen *StringWriter* zum Schreiben der Ausgabe und die Belegung der oben genannten Checkboxen als *boolean Array* übergeben. Zu Beginn werden die Variablen für die Anzeige der Operationstypen gesetzt, um die Konfiguration des Benutzers korrekt wiederzugeben. Anschließend wird der benötigte Bereich zur Darstellung in der HTML-Seite erzeugt. Die Gegenüberstellung der beiden Versionen wird mithilfe einer Tabelle umgesetzt. Danach wird der linke Baum, also der der früheren Version, zweimal durchlaufen, nämlich einmal für die linke Seite und einmal für die rechte Seite. Dies ist durch die Tatsache begründet, dass die Effekte vom *edit script* nur zu den Knoten des linken Baumes hinzugefügt wurden.

Dabei wurden die Effekte *insert*, *delete*, *move* und *update* folgendermaßen mithilfe einer Instanz der Klasse *AugmentedEditScript* an die Knoten angehängt:

- *insert*
  Dem eingefügten Knoten wird der Effekt *inserted* angehängt und dem neuen Vaterknoten wird der eingefügte Knoten zur Liste *inserts* hinzugefügt.
- *delete*
  Ein Knoten, der gelöscht wurde, wird mit dem Effekt *deleted* gekennzeichnet.
- *move*
  Dem Knoten, der verschoben wurde, wird der Effekt *moved* angehängt und dem neuen Vaterknoten wird der verschobene Knoten in die Liste *inserts* eingefügt.
- *update*
  Einem Knoten, an dem eine *update* Änderung durchgeführt wurde, wird der Effekt *updated* angehängt.

Das Durchlaufen des Baumes wird für beide Seiten mithilfe einer Tiefensuche durchgeführt. Dabei wird bei jedem besuchten Knoten überprüft, ob eine Operation an diesem Knoten ausgeführt wurde, die zur durchlaufenden Seite passt. Also *delete* und *move* Operationen für die linke Seite, die der älteren Version entspricht, und *insert* und *move* für die rechte Seite, die der neueren Version entspricht. Der Effekt *update* wird bei beiden Seiten berücksichtigt. Zum Erzeugen der Textdarstellung der einzelnen Knoten wird für jede Seite eine Instanz der Klasse *DiffFormatter* zur Verfügung gestellt. Besitzt also ein Knoten einen entsprechenden Effekt, so wird dieser im *DiffFormatter* gesetzt, um ihn später korrekt darstellen zu können.
Genauso wird für jeden Knoten überprüft, ob dieser als neues Blockelement dargestellt werden soll. Jedes Blockelement wird bei der Visualisierung dann als eigener Block dargestellt, wobei nur die Blöcke angezeigt werden, bei denen eine Änderung vorgenommen wurde. Nachdem nun der *DiffFormatter* entsprechend konfiguriert ist, kann die Einstellung der Texthervorhebung mithilfe der Funktion *highlightText* vorgenommen werden und der nötige Bereich durch die Methode *applyScope* deklariert werden.

Die Texthervorhebung, die bereits konfiguriert wurde, wird im Folgenden erläutert. Grundsätzlich wird jedem Knoten mit entsprechendem Operationstyp die passende CSS Klasse zugeteilt, um die Visualisierung durchführen zu können. Diese Klassen sind in der SCSS Datei *_diff-tree* aufgeführt. Durch diese Zuteilung werden Elemente entsprechend ihrer Effekte dargestellt. Dabei wird bei jedem Operationstyp die Farbe des Schrifthintergrunds angepasst. Dies ist bei *insert* grün, *delete* rot und *move* blau. Operationen vom Typ *update* werden mittels *insert* und *delete* visualisiert. Die Operationen *delete* und *move* haben noch weitere Elemente zur Darstellung. So wird gelöschter Text zunächst mithilfe von Durchstreichen angezeigt und erst durch das Bewegen der Maus über ein gelöschtes Element wird der Schrifthintergrund rot gefärbt, was mittels der *hover* Funktion in *CSS* umgesetzt ist. Bei Elementen, die verschoben wurden, wird das dazugehörige verschobene Element farblich stärker hervorgehoben, wenn der Mauszeiger sich über das andere Element bewegt. Dies wird durch die JavaScript Funktionen *showMove* und *hideMove* implementiert.
Eine Ausnahme von den Operationstypen bildet die Visualisierung der *Whitespace* Änderung. Diese bezeichnet alle Änderungen bei denen nur Leerraum gelöscht oder eingefügt wurde. Wenn eine solche Änderung vorkommt, wird sie mittels gelbem Schrifthintergrund markiert und unterhalb der Visualisierung wird textuell auf eine solche Änderung hingewiesen. Dabei wird der Schrifthintergrund mit der *CSS* Klasse *_diff-tree* umgesetzt und das Textfenster zur Unterstützung unterhalb der Visualisierung mithilfe eines HTML-Elements erstellt.

Nachdem der nötige Bereich für die Blöcke deklariert wurde, wird der Text des Knotens, falls er ein Textknoten ist, mit der vorgenommenen Texthervorhebung in den deklarierten Block eingefügt. Wenn dann der linke Baum für jede Version einmal durchlaufen worden ist und jeder Textknoten mit der entsprechenden Texthervorhebung einem Block zugewiesen wurde, werden die einzelnen Blöcke in der richtigen Reihenfolge in die bereits angesprochene Tabelle eingefügt und die Visualisierung erzeugt.

3.3 Beschreibung der Änderungen

Um die Beschreibung der Änderungen zu erstellen, wird eine Instanz der Klasse *DiffDescriber* erstellt. Dafür bekommt das Objekt zum einen eine Instanz der Klasse *AugementedEditScript* und zum anderen einen *StringWriter*. Nach dem Erstellen des Objekts wird in der Methode *describe* der benötigte Bereich für die Anzeige der Beschreibung deklariert. Innerhalb dieses Bereichs werden vier Listen erstellt, die jeweils für einen Operationstyp verwendet werden und an den die vorkommenden Operationen angehängt werden. Dann wird der linke Baum mithilfe einer Tiefensuche in der Funktion *depthFirstSearch* durchlaufen und jeder Knoten wird auf Effekte überprüft. Diese Effekte sind im vorherigen Abschnitt näher erläutert. Besitzt also ein Knoten zum Beispiel eine nichtleere Liste *inserts*, so lässt sich daraus folgern, dass diesem Knoten neue Kinder hinzugefügt wurden. Dann muss noch überprüft werden, ob die Kindsknoten mithilfe der *move* oder der *insert* Operation im Baum eingefügt wurden. Anschließend wird die dem Operationstyp entsprechende *visit* Funktion aufgerufen. Genauso wird die *visit* Funktionen für die Effekte *delete* und *update* aufgerufen, wenn ein solcher Effekt vorliegt. In diesen *visit* Funktionen wird die Darstellung der entsprechenden Operation erstellt. Das Erzeugen der Anzeigeboxen wird mithilfe einer Instanz der Klasse *EditScriptSubtreePrinter* und der Methode *createSourceBox* in der Klasse *DocWriterBase* umgesetzt. In Abbildung 3.2 ist eine exemplarische Darstellung der Beschreibung von Operationen dargestellt.

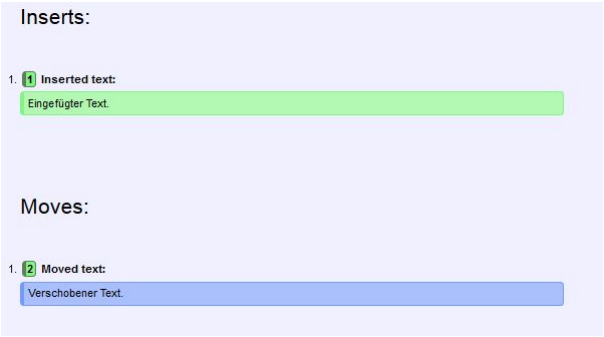

Abbildung 3.2: Exemplarische Darstellung der Beschreibung von Änderungen im Sweble Wiki

Um zu gewährleisten, dass verschachtelte Effekte in derselben Liste unterhalb der äußersten Operation aufgelistet werden, wird der Funktion *depthFirstSearch* immer das neu erzeugte Element einer Liste übergeben. Erreicht man also einen Knoten im Baum, der einen Effekt hat, wird überprüft, ob der übergebene Bereich gültig ist. Ist das der Fall, wird die Darstellung der Operation in diesem Listenelement erzeugt und somit die Änderung unter der äußeren Operation dargestellt. Ist dagegen der Bereich nicht gültig, wird ein neues Element für die entsprechende Liste erzeugt und dieses Element wiederum der Tiefensuche für spätere Knoten mitgegeben.

Literaturverzeichnis

Ball, T. A. & Eick, S. G. (1996). Software Visualization in the Large. *IEEE Computer*, *29* (4), 33–43.

Chawathe, S. S., Rajaraman, A., Garcia-Molina, H. & Widom, J. (1996). Change Detection in Hierarchically Structured Information. *In Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, 493–504.

Cicchetti, A., Ruscio, D. D. & Pierantonio, A. (2007). A Metamodel Independent Approach to Difference Representation. *Journal of Object Technology*, *6* (9), 165–185.

Dohrn, H. & Riehle, D. (2011). Design and Implementation of the Sweble Wikitext Parser: Unlocking the Structured Data of Wikipedia. *In Proceedings of the 7th International Symposium on Wikis and Open Collaboration (WikiSym'11)*, 72–81.

Dohrn, H. & Riehle, D. (2014). Fine-grained Change Detection in Structured Text Documents. *In Proceedings of the 2014 ACM symposium on Document engineering (DocEng'14)*, 87–96.

Dohrn, H. & Riehle, D. (July 2011). WOM: An object model for Wikitext. *Technical Reports*, *CS-2011-05*.

Jones, J. (2003). Abstract Syntax Tree Implementation Idioms. *In Proceedings of the 10th conference on pattern languages of programs (plop2003)*.

Myers, E. W. (1986). An O(ND) Difference Algorithm and Its Variations. *Algorithmica*, *1* (1–4), 251–266.

Neuwirth, C. M., Chandhok, R., Kaufer, D. S., Erion, P., Morris, J. & Miller, D. (1992). Flexible Diff-ing In A Collaborative Writing System. *In Proceedings of the 1992 ACM conference on Computer-supported cooperative work (CSCW 92)*.

Nunes, S., Ribeiro, C. & David, G. (2008). WikiChanges - Exposing Wikipedia Revision Activity. *In Proceedings of the 4th International Symposium on Wikis (WikiSym'08)* (25).

Rönnau, S., Teupel, M. & Borghoff, U. M. (2009). Graphical Merge Support for XML Documents. *In Klempous et al. (eds.): Proc. of the 4th Int. Conf. on Broadband Communication, Information Technology and Biomedical Applications*.

Viègas, F. B., Wattenberg, M. & Dave, K. (2004). Studying Cooperation and Conflict between Authors with history flow Visualizations. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'04)*, *6* (1), 575–582.

Wikipedia, O. V. (2014). *Wikipedia:Wikifizieren.* Zugriff am 17.03.2015 auf `http://de.wikipedia.org/wiki/Wikipedia:Wikifizieren`

Wikipedia, O. V. (2015). *Wikipedia:Statistics.* Zugriff am 08.01.2015 auf `http://en.wikipedia.org/wiki/Wikipedia:Statistics`