Computergestützte Gruppenarbeit

Übungsblatt 8 - Lösung

Dr. Jürgen Vogel

European Media Laboratory (EML)
Heidelberg

FSS 2007

Timewarp

Klausur-Aufgabe vom SS 2005 – 10 Punkte

Implementieren Sie das Timewarp-Verfahren in Pseudo-Code. Entwickeln Sie dazu die Funktion **`Receive(Op o)`**, die beim Empfang einer Operation o ausgeführt wird. Verwenden Sie dabei ausschließlich die angegebenen Datentypen und Funktionen.

Kommentieren Sie Ihre Vorgehensweise!

```
Type Op                              // Operation
Type History = List of Op            // Operationshistorie
Type Iterator                        // Iterator für die Historie

Boolean IsState(Op o)                // TRUE falls o State ist
Timestamp GetTime(Op o)              // Ausführungszeitpunkt
Execute(Op o)                        // führt o aus

Boolean IsEmpty(History h)           // TRUE falls h leer
Iterator Start(History h)            // Referenz auf älteste Op
Iterator End(History h)              // Referenz auf jüngste Op
Next(Iterator i)                     // setzt i auf nächste Op
Previous(Iterator i)                 // setzt i auf vorige Op
Op GetOp(Iterator i)                 // referenzierte Operation
Insert(History h, Iterator i, Op o)  // fügt o bei i ein
Delete(History h, Iterator i)  // löscht Op an der Stelle i
```

Timewarp - Lösung

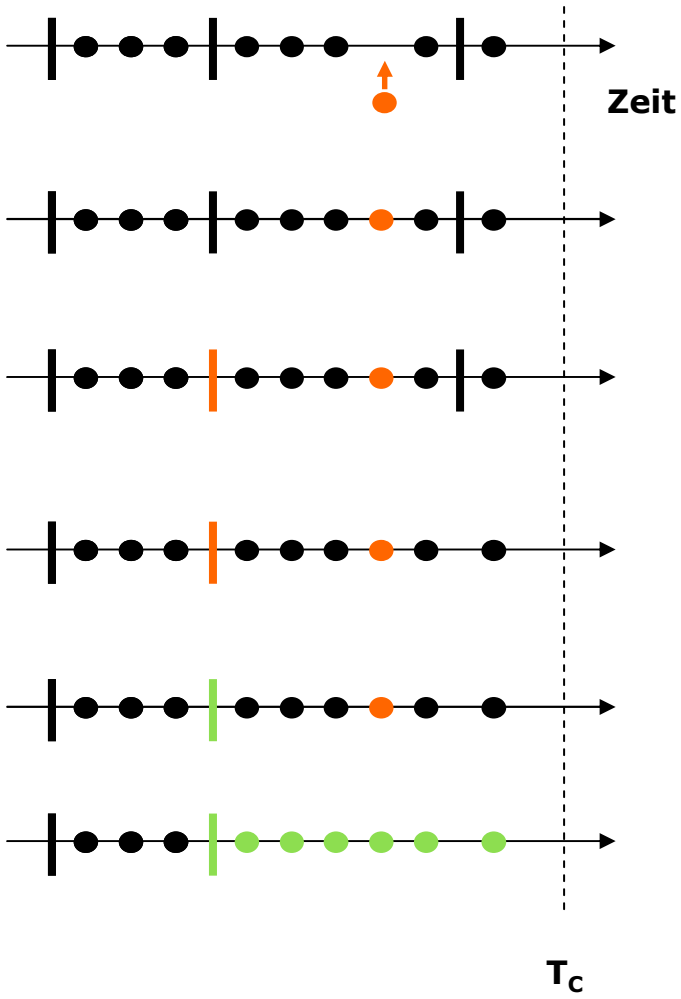
```
Receive(Op o)
  IF IsEmpty(h)
    Insert(h, Start(h), o)
    Execute(o)
    RETURN

  Iterator i = End(h)
  WHILE ((GetTime(o) < GetTime(GetOp(i)))
         && (i != Start(h)))
    IF (IsState(GetOp(i)) Delete(h, i)
    ELSE Previous(i)
  Insert(h, i, o)

  IF (i == End(h))
    Execute(o)
    RETURN

  Previous(i)
  WHILE (!IsState(GetOp(i)))
    Previous(i)

  WHILE (i != End(h))
    Execute(GetOp(i)
  Execute(GetOp(i)
```



 © Jürgen Vogel

Verbesserung von Timewarp - Lösung

Entwerfen Sie geeignete Methoden, um die Nachteile des Timewarp-Algorithmus zu beheben bzw. abzumildern:

Visuelle Artefakte

- "sanfte" Änderung - aber möglichst schnell zur Vermeidung von sekundären Inkonsistenzen
- Awareness-Informationen für den Anwender → Kapitel 9

Rechenaufwand für die Ausführung eines Timewarps

- nachgeführte Zustände
- rundenbasierter Timewarp

Speicherbedarf für Verwaltung der kompletten Operationshistorie

- Beschränkung der Historie

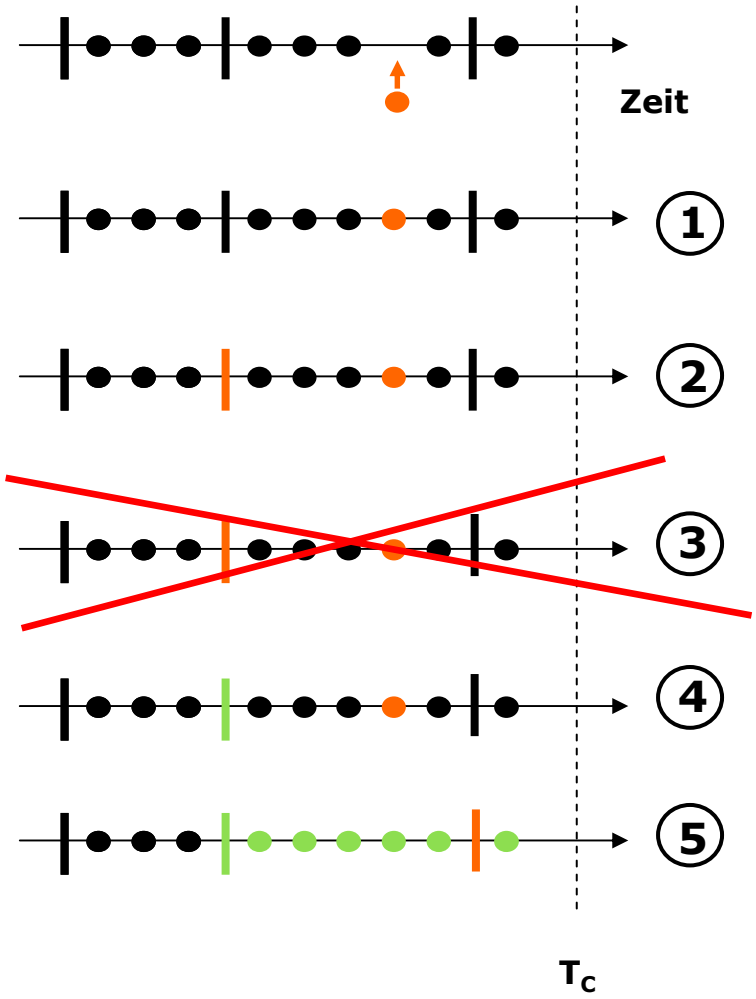
Nachgeführte Zustände ("Trailing States")

Timewarp: lösche alte Zustände $S_{i,t'}$ (Schritt 3)

- H_i enthält weniger Zustände
- zukünftige Timewarps erfordern größere Rücksprünge und $H_{i,t}$ beinhaltet mehr Operationen

Idee: ersetze $S_{i,t'}$ während des Timewarps (Schritt 5) durch einen aktualisierten Zustand

- Anzahl von Zuständen in H_i bleibt konstant
- im Durchschnitt höherer Speicherbedarf



 © Jürgen Vogel

Rundenbasierter Timewarp (1)

Timewarp: jede Operation mit $t^* < T_C$ löst einen Timewarp aus

- Timewarp verbraucht gewisse Rechenzeit

➡ im ungünstigsten Fall selbstverstärkender Effekt

Idee: rundenbasierter Timewarp

- Anwendung sammelt alle Operationen innerhalb einer Zeitspanne T (=Runde)
- die Operation mit der kleinsten Ausführungszeit bestimmt den Startzustand $S_{i,t}$ und den auszuführenden Teil der Historie $H_{i,t}$
- $S_{i,t}$ ist entweder der in der letzten Runde berechnete Zustand oder ein älterer

➡ pro Runde T wird höchstens ein Timewarp ausgeführt, unabhängig von der Anzahl der verspäteten Operationen

➡ Reduktion der Komplexität auf $O(n^2)$

➡ verhindert Folge-Timewarps

Rundenbasierter Timewarp (2)

- gut geeignet für kontinuierliche Anwendungen
  - periodisches Update des angezeigten Zustands
  - z.B. Spiele mit x FPS
  - wähle T entsprechend der Update-Frequenz (Rechenzeit pro Timewarp berücksichtigen)
  - ist T klein genug, merkt der Benutzer nichts
- diskrete Anwendungen
  - falls in T keine Operation anfällt, bleibt der Zustand gleich

Beschränkung der Operations-Historie (1)

Ziel: Speicherbedarf für Operations-Historie begrenzen

1) Ersetze Operations-Sequenzen

- ersetze lange Operations-Sequenzen durch semantisch gleichwertige Operation
- z.B. mit Events punktweise erzeugte Freihandlinie → Zustand
- Cues lösen keinen Timewarp aus und werden nicht gespeichert

Beschränkung der Operations-Historie (2)

2) Lösche sicher ausgeführte Operationen

- lösche Operationen, die von allen Instanzen ausgeführt wurden
- ➡ werden für mögliche Timewarps nicht benötigt
- implizite Bestätigung von Operationen mit Zustandsvektoren
  - Instanz i empfängt O_j von Instanz j mit Vektor SV_{O_j}
  - $SV_{O_j}[k]$ ist die SN der letzten Operation, die j von k empfangen hat

  → j hat alle älteren Operationen von k empfangen

  - falls O_j kausal ausführbar ist: temporäre Inkonsistenzen durch Operationen von j, die nebenläufig zu Operationen von k mit $SN_k \leq SV_{O_j}[k]$ sind, können nicht auftreten

  → aus der Sicht von j kann i alle Operationen von k mit $SN_k \leq SV_{O_j}[k]$ aus der Historie entfernen

  - hat i Bestätigungen von allen Instanzen: lösche Operationen

Beschränkung der Operations-Historie (3)

- explizite Bestätigung von Operationen: periodische Status-Nachrichten mit den aktuellen Zustandsvektoren
  → schnellere Bestätigung aber Kommunikations-Overhead

3) Verwende anderen Konsistenzerhaltungs-Mechanismus

- beschränke die von der Historie abgedeckte Zeitspanne T_H:
  lösche O_{j,t^o,t^*} mit $t^* < T_C - T_H$

➡ temporäre Inkonsistenzen durch empfangene Operationen O_{j,t^o,t^*_x} mit $t_x^* < T_C - T_H$ können nicht durch Timewarp behoben werden

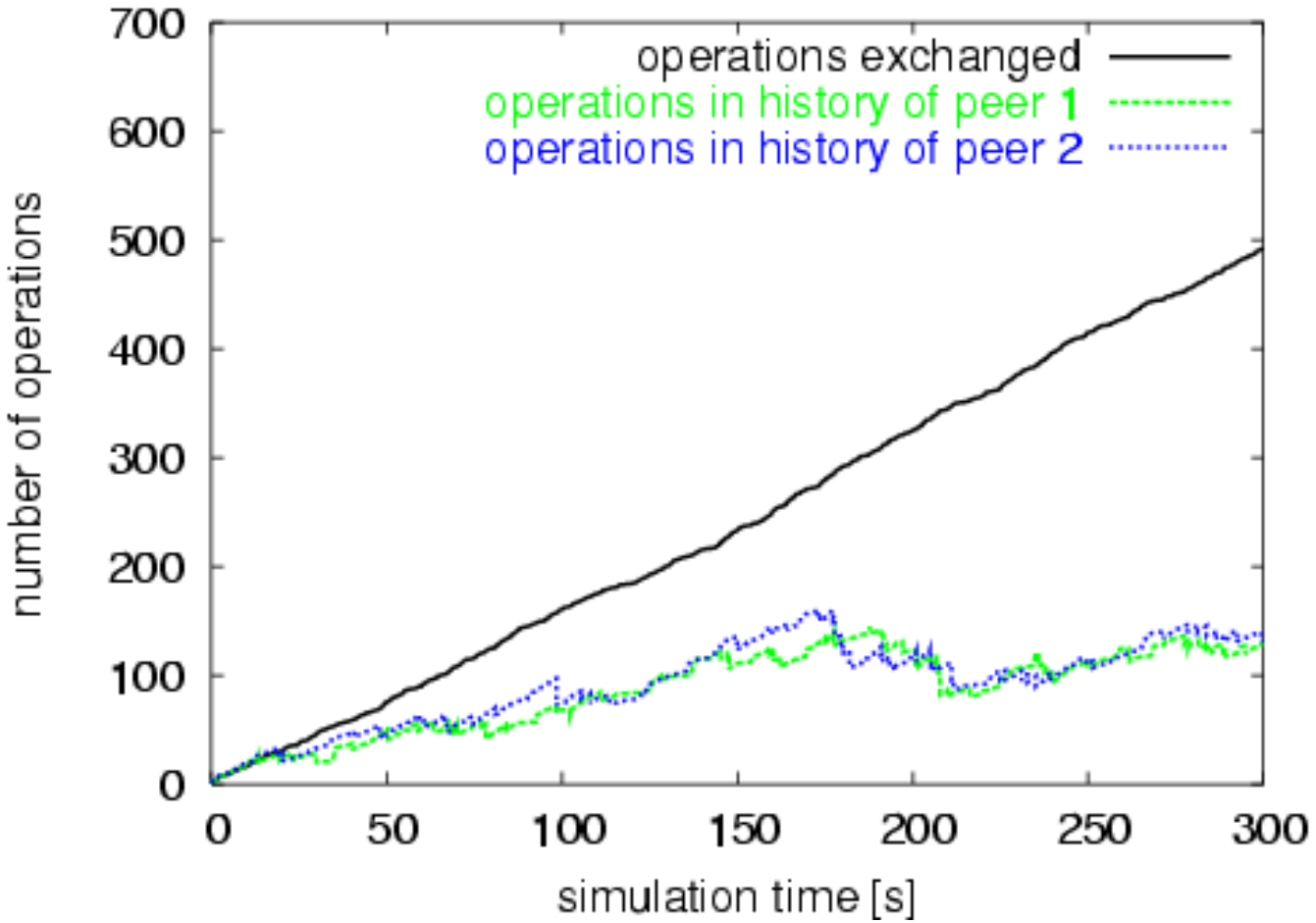
→ behebe ältere Inkonsistenzen per Zustandsanfrage

→ Trade-Off T_H: durch Timewarp abgedeckte Zeitspanne vs. Speicherbedarf (z.B. 180s beim mlb)

➡ die Historie beginnt nicht am Anfang einer Sitzung
→ Anwendung muss sicherstellen, dass immer ein Zustand $S_{i,t}$ mit $t = T_C - T_H$ in der Historie enthalten ist

© Jürgen Vogel

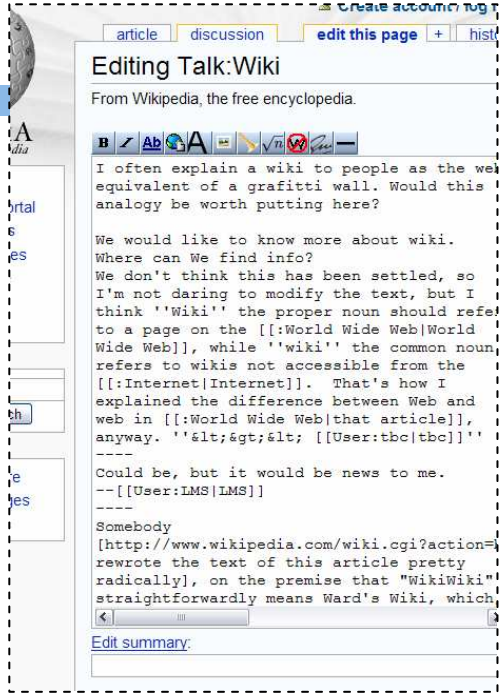
Beispiel-Ergebnis für Lösch-Algorithmus



Wiki (1)

Beschreibung und Anwendungsbereiche

- Ziel: gemeinschaftliche Erstellung einer Wissensbasis durch Zusammentragen von spezifischem Expertenwissen → Leser wird zum Editor
- Inhaltserstellung per WWW-Schnittstelle und vereinfachter Markup-Sprache (oder WYSIWYG-Editor); sofortiges Update
- starke interne und externe Verlinkung ("VerweisAufSeite") mit automatischer Erzeugung neuer Ziel-Seiten
- Änderung von Inhalten
  - meist dürfen alle Inhalte von jedermann geändert werden
  - Historie mit allen Versionen und Autorennachweis
  - Übersichtsseite mit den letzten Änderungen
  - textbasierter Vergleich zwischen Versionen
  - einfaches Wiederherstellen einer alten Version
  - zusätzliche Diskussionsseite



Wiki (2)

- große Gruppen mit vielen passiven Benutzern
- aktive Benutzer sind nur im Hintergrund präsent
- z.B. WikiWiki, MediaWiki (Wikipedia), TikiWiki, …

Klassifikation und technische 3K-Protokolle

- indirekte und asynchrone Kommunikation steht im Vordergrund
- zusätzlich themenspezifische Diskussion
- Koordination und Kooperation über soziale Protokolle, die Vergabe von Aufgaben ("Tickets") und Floor/Session Control

Daten, Datenmodell und Architektur

- Client-Server-Architektur (Web-Client, Web- und Datenbank-server)
- vernetztes Objektmodell
- Unterstützung von States

Wiki (3)

WYSIWIS und Awareness

- relaxiertes WYSIWIS mit (Web-)Client-spezifischer Darstellung
- (Email-)Benachrichtigung über Änderungen an einer Seite an spezielle Benutzer ("Editor")

Floor und Session Control

- Vergabe von Schreib- und Lese-Rechten für Objekte
- meist offene Gruppenstruktur
- Rollenmodell mit assoziierten Rechten (Gast, Autor, Editor)

Konsistenzerhaltung

- serverseitige Serialisierung

Wieso funktionieren Wikis? (1)

Herausforderungen

- freiwillige Inhaltserstellung
- keine explizite Qualitätskontrolle
- Konsensfindung erforderlich/schwierig für kontroverse Themen
  → Wiki-Seite reflektiert von allen Benutzern anerkanntes Wissen
- offen für bewusste oder unbewusste Falschinformationen
- Vandalismus

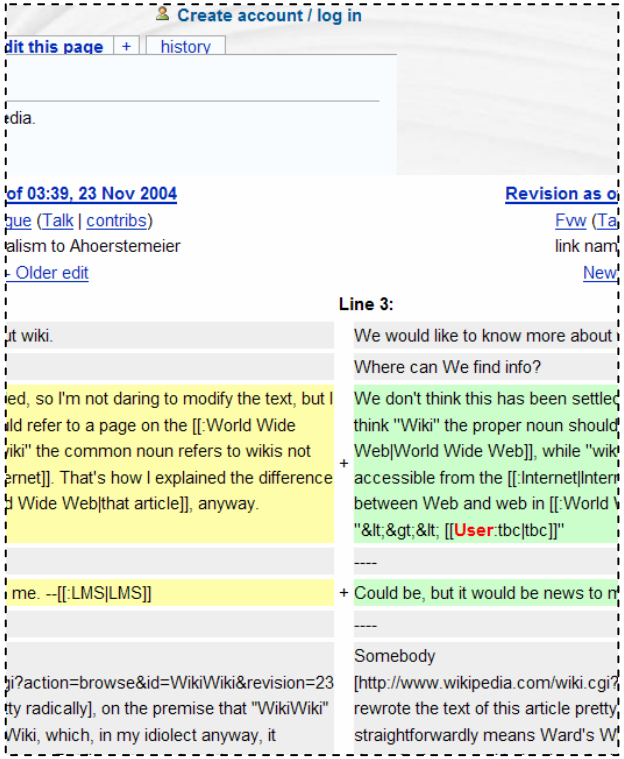
Wieso funktionieren Wikis? (2)

Gründe für den Erfolg von Wikis

- sehr einfache Inhaltserstellung und Verlinkung
- viele kleine Beiträge einer großen aktiven Gemeinde summieren sich schnell (kritische Masse); ein schnelles Erreichen der kritischen Masse ist aber essentiell
- Belohnungsmechanismen: Bewertung von Beiträgen bzgl. Qualität und Quantität (z.B. Amazon)
- "Patenschaft" ist weit verbreitet: Wiki-Autoren fühlen sich für Themen, die sie privat/beruflich interessieren, verantwortlich
  → Email-Benachrichtigung bei Änderung und Anzeige der letzten Änderungen ("Recent Changes") erleichtern das Überwachen
  → schnelle Reparatur von Vandalismus-Schäden
- Versionshistorie ermöglicht einfache Reparatur von Schäden
- Autoren überwachen sich gegenseitig
- allgemein anerkanntes Ziel ist der "Neutral Point of View", d.h. möglichst objektive Themendarstellung
- Gutmensch-Prinzip (solange "gutartige" Benutzer überwiegen) und einfache Revision

Semantische Konflikte

- manuelle Auflösung von semantischen Konflikten
- Unterstützung durch
  - Versions-Log mit Autor, Änderungs-datum, …
  - textbasierten Versionsvergleich
  - Benachrichtigung über Änderungen
  - separate Seite zur Diskussion abweichender Meinungen
- Design
  - hauptsächlich textuelle Darstellung
  - Farbkodierung
  - explizite Abfrage (pull)



Visualisierung der Seiten-Historie (1)

F.B. Viegas, M. Wattenberg, K. Dave, Studying Cooperation and Conflict between Authors with history flow Visualizations, In: Proc. ACM CHI, Vienna, Austria, April 2004

Was ist der fundamentale Unterschied von Wikipedia im Vergleich zu traditionellen Lexika?

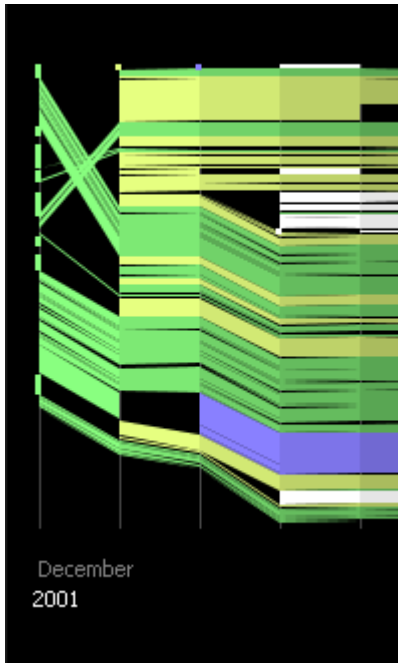
- traditionell
  - Artikel wird durch einige wenige, explizit ausgewählte Autoren erstellt und
  - vor der Veröffentlichung begutachtet
  - Leser sind rein passiv
- Wikipedia
  - Artikel werden von "Laien" erstellt
  - alle Leser sind gleichzeitig potentielle Autoren
  - Diskussionen und Kommentare sind möglich
  - schnelle Überarbeitung von Inhalten

Visualisierung der Seiten-Historie (1)

- häufige Änderung von populären/kontroversen Inhalten
- textbasierte Analyse umfangreicher Historien ist aufwendig
- ➡ Visualisierung der Historie einzelner Wiki-Seiten

Visualisierungs-Technik

- Abbildung von Text auf ein Balkendiagramm
- Höhe des Diagramms = Textlänge
- Änderungen durch einen Autor werden mit dessen Farbe auf dem Textbalken markiert
- Verlauf zwischen zwei Versionen durch Verbindung korrespondierender Textabschnitte
- optionale Abbildung der zeitlichen Abstände auf die Distanz zwischen zwei Versionslinien
- Verknüpfung zwischen Diagramm und Text



Visualisierung der Seiten-Historie (2)

Wie werden konfliktäre Änderungen sichtbar?

Konflikte sind erkennbar an

- vielen Versionen (in kurzer Zeit)
- wiederholtes gegenseitiges Überschreiben (überkreuzende Linien): "Edit Wars" (bis zu 20 Versionen lang)
- stark schwankender Textlänge

Wie werden Konflikte in Wikipedia meist gelöst?

- Konsensfindung durch ausführliche Kommentierung der Änderungen
- oder durch Diskussion von Änderungen außerhalb des Artikels

➡ ein Artikel stellt den kleinsten gemeinsamen Kompromiss dar

- ggf. Restrukturierung der Seiten bei nebensächlichen Inhalten
- implizite Gewichtung des Beitrags nach der Reputation des Autors (Bevorzugung bekannter Autoren)

Visualisierung der Seiten-Historie (3)

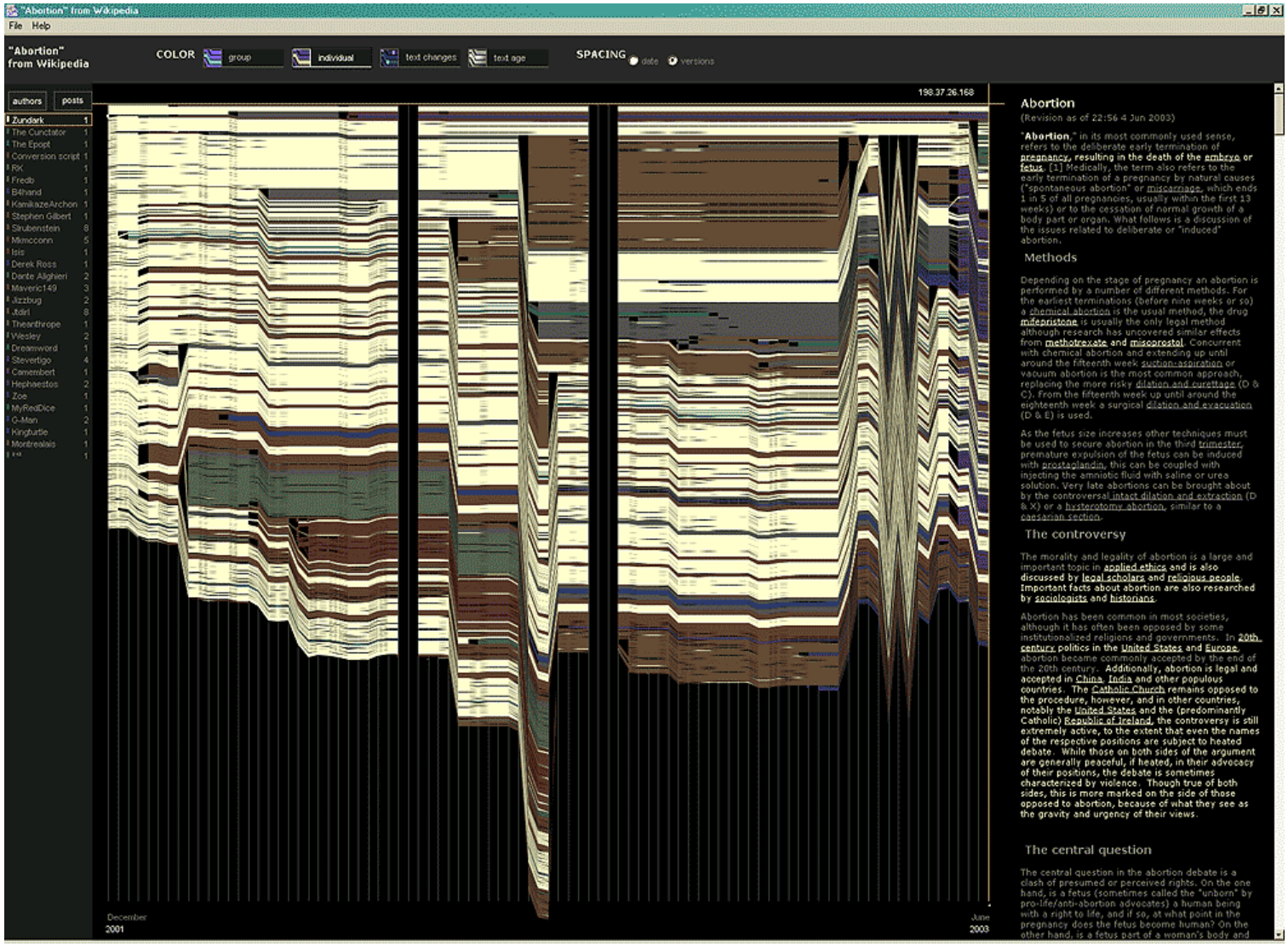
Vandalismus

- häufige Arten:
  - (1) Löschen der ganzen Seite: Lücke
  - (2) Überschreiben mit irrelevantem Text: durchgehender Balken
  - (3) vulgärer oder beleidigender Text
  - (4) falsche Verlinkung
  - (5) subjektive Meinungen, zu langer Text, etc.
- Seiten mit mindestens 50 Versionen wurden mindestens einmal attackiert
- schnelle Erkennung und Reparatur: 50% innerhalb von 2-3 Minuten

Weitere Ergebnisse

- häufig gibt es einige Autoren, die sich für eine Seite verantwortlich fühlen (z.B. Überwachung von Änderungen)
- kontinuierliche Veränderung bei fast allen Seiten
- initialer Text einer Seite hat hohe "Überlebenswahrscheinlichkeit"

Visualisierung der Seiten-Historie (3)



Visualisierung der Seiten-Historie (4)