Verteiltes Dokumentenmanagement

Langzeitarchivierung - Revisionssichere Archivierung

John Bechara

23. Mai 2009

der Bundeswehr Universität München

Inhaltsverzeichnis

Universität der Bundeswehr München

Einleitung

- Jeder PC Benutzer hat heute mehr Rechenleistung auf seinem Schreibtisch stehen, als Wissenschaftler noch 1976, mit dem ersten offiziellen Supercomputer Cray-1, zur Verfügung stand
- Kosten für Computer 2007 um 2/3 günstiger im Vergleich zum Vorjahr
- 2003 setzten bereits 80% aller Unternehmen Computer ein

Elektronische Dokumente

- Elektronische Dokumente wurden und werden immer wichtiger / größer / komplexer
- Vor allem bei Source Code Dokumenten in der Softwareentwicklung

SLOC bei Windows

Year	Operating System	SLOC (Million)
1993	Windows NT 3.1	4-5
1994	Windows NT 3.5	7-8
1996	Windows NT 4.0	11-12
2000	Windows 2000	more than 29
2001	Windows XP	40
2003	Windows Server 2003	50

Quelle: `http://en.wikipedia.org/wiki/Source_lines_of_code`

SLOC bei diversen anderen Betriebssystemen

Operating System	SLOC (Million)
Debian 2.2	55-59
Debian 3.0	104
Debian 3.1	215
Debian 4.0	283
OpenSolaris	9.7
FreeBSD	8.8
Mac OS X 10.4	86
Linux kernel 2.6.0	5.2
Linux kernel 2.6.29	11.0

Quelle: `http://en.wikipedia.org/wiki/Source_lines_of_code`

Versionskontrollsysteme

- Grund für Etablierung von Versionskontrollsystemen
- Im Bereich der Softwareentwicklung größte Verbreitung
- Verwendung aber, unter best. Vorraussetzungen, bei zahlreichen Dokumententypen denkbar
- Daher: Schwerpunkt auf Source Code Dokumenten

Aufgaben von VCS

- Versionskontrolle: Aufzeichnen und Abrufen von Änderungen in einem Projekt
- Sicherung und Wiederherstellung:
  - Sicherstellen einer Speicherung
  - Möglichkeit unerwünschte Änderungen zu revidieren
- Festhalten von Änderungen
  - Welcher Nutzer hat welche Dateien zu welchem Zeitpunkt in welcher Art und Weise geändert?

Grundbegriffe

- Repository
  - Kern des VCS
  - Speicherung aller Daten in baumartiger Ordner- /Dateistruktur
- Sandbox
  - Kopie des Repository
  - Arbeiten in der Sandbox, nicht im Repository
  - Physikalischer Speicherort ist lokal oder auf Fremdsystem
  - Bei lokaler Speicherung auch als "working copy" bezeichnet

Grundbegriffe

- Diff
  Prozess, bei dem Unterschiede zwischen zwei Versionen einer Datei festgestellt wird
- Delta
  Ergebnis eines Diffs
- Patch
  Anwenden eines Deltas auf ein Dokument

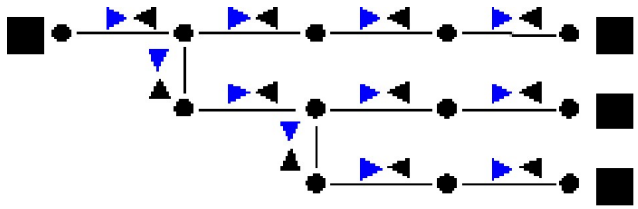
Beispiel

Einkaufsliste_V1	
1	Mehl
2	Eier
3	Kartoffeln

Einkaufsliste_V2	
1	Mehl
2	Eier
3	Hefe
4	Kartoffeln

Delta: füge Hefe an Position 3 zu Einkaufsliste_V1

Deltas



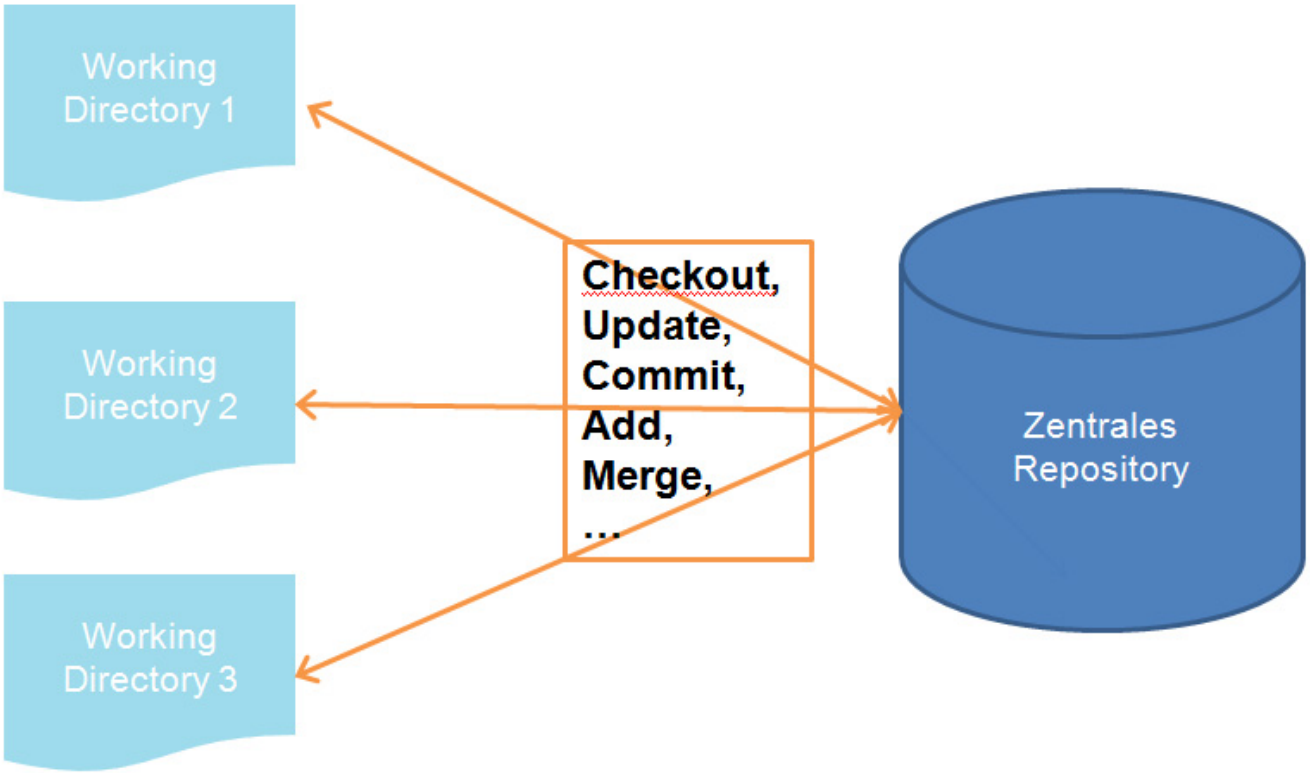
Quelle: `http://www.uni-koblenz.de/~espi/Softwaretechnik2.pdf`

- Rückwärtsdelta
  - Aktuellste Versionen wird gespeichert
  - Deltas dienen dazu Vorgängerversionen wiederherzustellen
  - Vorteil: direkter Zugriff auf aktuelle Version
  - Nachteil: mehrere Vollversionen
- Vorwärtssdelta
  - Erste Version wird gespeichert
  - Deltas dienen dazu die aktuellsten Versionen wiederherzustellen
  - Vorteil: Nur eine Vollversion
  - Nachteil: dauernde Neuberechnung

CVCS

- Client-Server-Model
- Informationen werden in Dokumenten mit einer Liste von Deltas auf Server gehalten und von Clients abgerufen
- "Quasi-gleichzeitiges" Arbeiten auf einer Datei möglich
- Nur lesender aber kein schreibender Zugriff, da in Sandbox gearbeitet wird, nicht auf Repo

Client Server Struktur



Quelle: `http://www.fh-wedel.de/~si/seminare/ws08/Ausarbeitung/06.vvw/vvw1.htm`

”how will the system allow users to share information but prevent them from accidently stepping on each other´s feet?”

Das Lock Modify Unlock Model

- Pessimistische Versionskontrolle
- Manuelle Sperrung bestimmter Abschnitte vor Beginn von Änderungsarbeiten durch Nutzer
- Üblicherweise sind solche Abschnitte mindestens Dateien
- Manuelle Freigabe, wenn Änderungsarbeiten abgeschlossen durch Nutzer

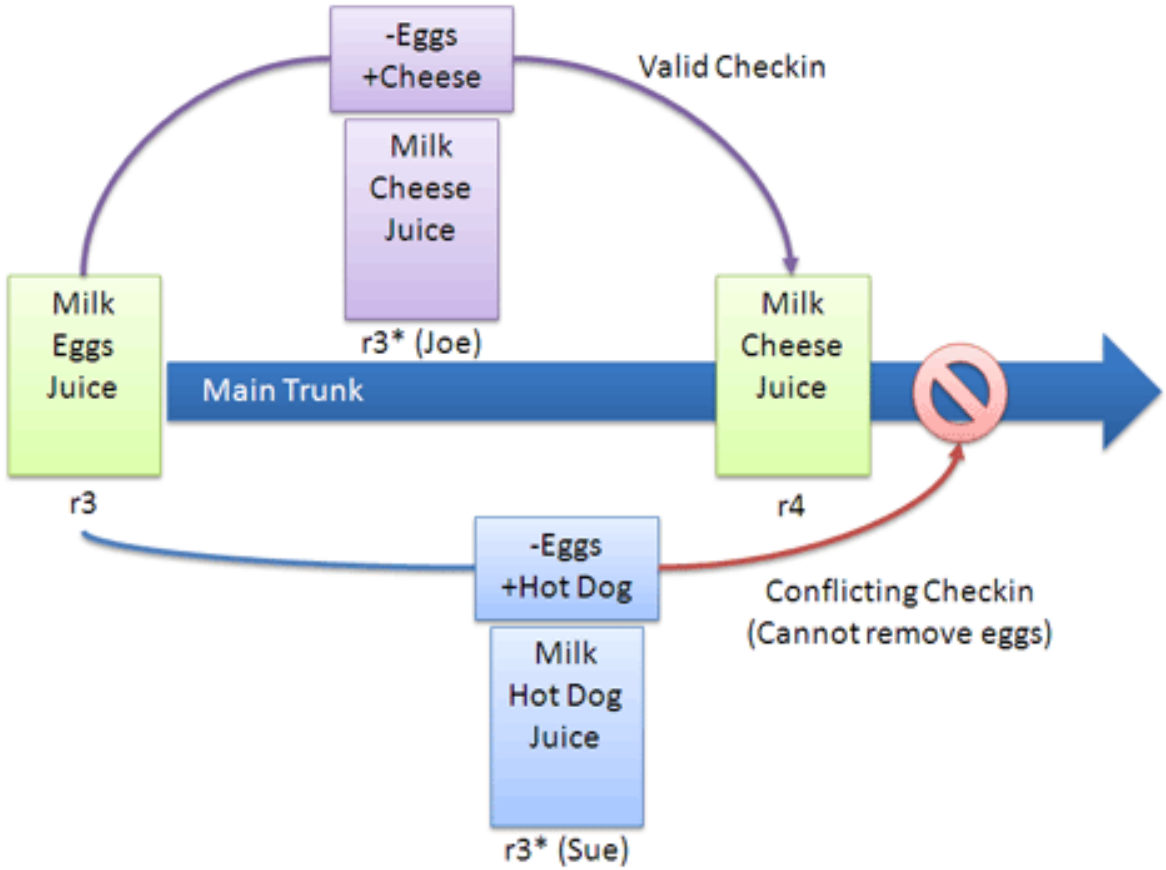
Nachteile des Lock Modify Unlock Model

- Hoher Aufwand bei Nichtaufhebung der Sperre durch Nutzer
- Sperrung einzelner Dateien garantiert keine Konfliktfreiheit
  Hier täuscht das CVCS sogar konfliktfreies Arbeiten vor, obwohl es das nicht garantieren kann
- Erzwingung eines seriellen Entwicklungsprozesses, obwohl nicht immer notwendig $\Rightarrow$ Unproduktivität

Das Copy Modify Merge Model

- Optimistische Versionskontrolle
- Jeder Nutzer arbeitet ohne Sperren in seiner Sandbox und lädt Änderungen dann auf den Server

Conflicts



Quelle: `http://betterexplained.com/articles/a-visual-guide-to-version-control/`

Merge

- Three-Way-Merging vs. Two-Way-Merging
- Three-Way-Merging
  - Informationen des Nearest Common Ancestor werden mitverwendet
  - Nearest Common Ancestor zweier Versionen v4 und v5 ist die Version v3, die von allen gemeinsamen Vorgänger-Versionen am Nächsten ist

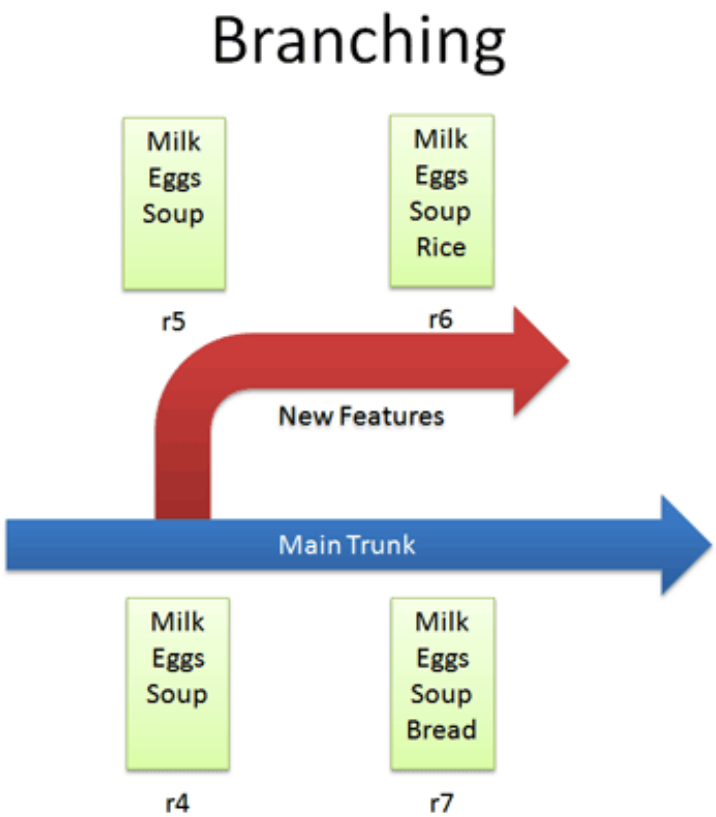
- Das macht Three-Way-Merge mächtiger als Two-Way-Merge

Nachteile des Copy Modify Merge Model

- Vom System nicht lösbare Konflikte müssen dennoch manuell gelöst werden
- Produktivität wird durch Schreibrechtevergabe eingeschränkt

Parallele Entwicklung eines Projekts

Heutige Systeme ermöglichen Verzweigungen



Quelle: `http://betterexplained.com/articles/a-visual-guide-to-version-control/`

Trunks and Branches

- Verzweigung ⇒ Branch
- Hauptordner ⇒ Trunk
- Trunk dient Release Management
- Paralleles Arbeiten wird ermöglicht
- Förderung experimentellen Vorgehens

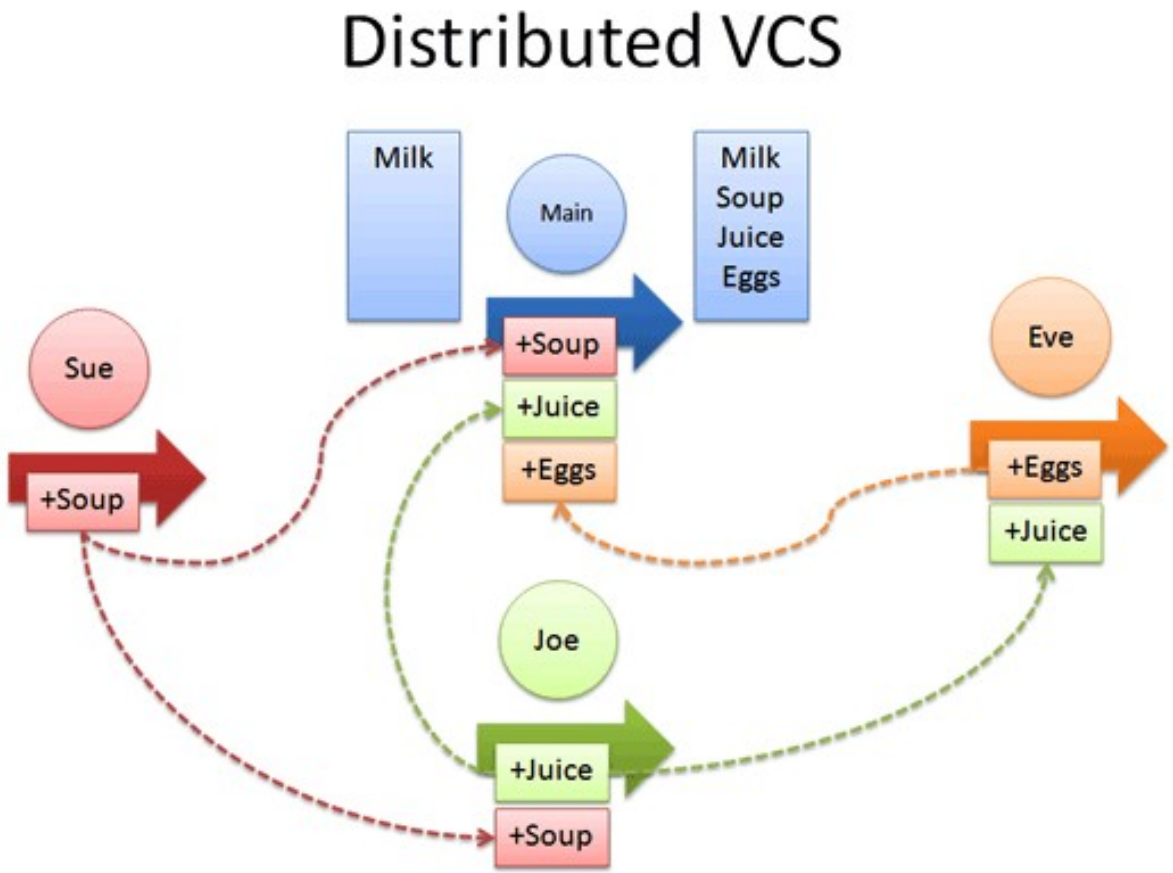
Nachteile des Branching

- Sowohl der Trunk, als auch die Branches befinden sich auf dem Server
- Es müssen eindeutige Namensräume geschaffen werden
- Oft aufwendiges Mergen notwendig

DVCS

- Selbe Fähigkeiten wie zentrale Pendants
- Jeder Nutzer hat sein eigenes Repo
- Entwicklungslinie bei Client-Server-Architektur wird zum gerichteten azyklischen Graphen (DAG)

DAG



Quelle: `http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/`

DVCS vs. CVCS

- Größter Nachteil: Wegfall der Entwicklungslinie und Aufbau eines DAG
- Kontrollverlust über den Source Code
- Abhilfe: Schaffen eines Main-Servers
  - Main-Server nur zum Release Management
  - Qualitätsmanagementgruppe zur Verwaltung des Main-Servers und Sichtung der Releases denkbar (Linus Torvalds)
  - Auch ein Ablegen verschiedener Zweige auf Main-Server denkbar

DVCS vs. CVCS

- Arbeiten ist offline möglich
  - Mergen und Konfliktbehebung ebenfalls offline möglich
  - Deltas können auch per Email verschickt werden
  - Performancesteigerung durch Wegfall der Notwendigkeit der Netzwerkoperationen
- Wegfall der Abhängigkeit von einer zentralen Instanz
  - Keine Probleme durch Serverausfall
  - Keine Engpasssituation bei Zugriffen

DVCS vs. CVCS

- Jeder Entwickler hat eigene Sandbox
  - Fehlgeschlagene Programmierversuche sind nicht öffentlich
  - Experimentieren und kreatives Arbeiten wird gefördert
  - Alle weiteren Probleme von Branches entfallen
- Direkte Kommunikation zwischen Entwicklern
  - Keine Notwendigkeit der Schaffung von Branches
  - Schreibrechtevergabe weitgehend obsolet
  - Auch ein Ablegen verschiedener Zweige auf Main-Server denkbar

DVCS vs. CVCS

- Push statt Pull
  - Jeder Nutzer bekommt Änderungen zugeschickt
  - Einbringen der Änderungen optional (Network of Trust)
- Produktivere Arbeitseinteilung
  - Keine Überlastung von Nutzern mit Schreibrechten
  - Entwickler ist Spezialist für seine eigene Arbeit
  - Jeder Entwickler übernimmt selbst Mergen und Konfliktbehebung

DVCS außerhalb der Softwareentwicklung

- Theoretisch bei zahlreichen Dokumententypen möglich
- Vorraussetzung ist Unterstützung von Delta, Diff, Patch
  - SCMS arbeiten zeilenbasiert $\Rightarrow$ Diff vergleicht Zeile für Zeile
  - Office Dokumente werden in Binär- oder XML-Datein gespeichert
  - Diff möglich, aber führt nicht zu einem Delta (zeigt nur Unterschiede zwischen zwei Versionen)
- Alle Vorteile, die bei DVCS in der Softwareentwicklung zu erkennen sind, würden auch für andere Dokumententypen gelten

Mögliche Ansätze zur LZA

- Main-Server könnte aufgrund seiner strikten Release-Management-Funktion zur LZA genutzt werden
- Push Ansatz fördert dabei die regelmäßige freiwillige Archivierung
- Qualitätsmanagementgruppe könnte für LZA Funktion verantwortlich sein
- Durch Ablegen verschiedener Zweige können auch Ideen archiviert werden

Fazit

- Wachsende Komplexität macht Alternativen zu CVCS notwendig
- DVCS zahlreiche Vorteile gegenüber ihren zentralen Pendants
- Community wächst, zahlreiche Projekte schon gewechselt
- Aber DVCS neue Technik, kaum Erfahrungen
- Anforderungen an Dokumente außerhalb der Softwareentwicklung kaum umgesetzt