Jörg Schilling

Ursprünge der Versionsverwaltung und Revival vom Source Code Control System (SCCS)

Fraunhofer FOKUS

Anfänge, SCCS als erste Versionsverwaltung

- Im Jahre 1972 schreibt Marc Rochkind SCCS in *Snobol* für eine IBM-370. Das Dateiformat ist binär.
- Später wurde es von ihm in C neu geschrieben um es auf einer PDP-11 unter UNIX laufen zu lassen
- Am 18. Februar 1977 hat AT&T SCCSv4 freigegeben, das History-Dateien vollständig als Text-Dateien verwaltet
- Danach gab es nur wenige Neuerungen in SCCS:
  - Mai 1980 Eric Allman (BSD) schreibt sccs(1)
  - 1988 mit SunOS-4.0 binäre Inhalte durch uuencode
  - 1999 SCCS wird *Y2k clean*

Wichtige Versionsverwaltungen im Überblick

- SCCS (seit 1972 einzeldateiorientiert POSIX Standard)
- RCS (seit 1982 einzeldateiorientiert)
- CVS (seit ca. 1986 projektorientiert zentral)
- TeamWare (seit ca. 1990 projektorientiert verteilt)
- BitKeeper SCCS (seit ca. 1997 projektorientiert verteilt)
- Subversion (seit ca. 2000 projektorientiert zentral)
- GIT (seit April 2005 projektorientiert verteilt)
- Mercurial (seit April 2005 projektorientiert verteilt)

Schwächen von SCCSv4

- Altes Datumsformat funktioniert nur bis 1999 bzw. 2068
- Zeitstempel mit Ortszeit (schlecht für globales Arbeiten)
- Metadaten müssen zuerst komplett gelesen werden
- Zu spät in OpenSource umgewandelt
- SCCSv4 unterstützt noch keine Projekte
- Keine Signaturen

Schwächen von RCS

- Keine Checksumme (Defekte bleiben lange unbemerkt)
- Schwer parsierbares Meta-Datenformat
- Metadaten müssen zuerst komplett gelesen werden
- Sehr langsam (mindestens 6x langsamer als SCCS)
- Inverse Deltas (Zugriff auf alte Versionen langsamer)
- Branches/Merges sind normale Deltas
- Keine Signaturen/Verifikation

Schwächen von CVS

- Alle Schwächen des RCS History-Formates
- Zentrale Aufbewahrung statt verteiltem Arbeiten
- Tags werden in jede Datei geschrieben
- Noch langsamer als RCS
- Keine Signaturen/Verifikation

Schwächen von TeamWare

- Alle Schwächen des SCCSv4 History-Formates
- Closed Source
- `Clone` benötigt NFS Zugriff auf den Server
- `Enter` benötigt direktes Arbeiten auf dem Server
- Keine Signaturen/Verifikation
- Auslaufmodell

Schwächen von BitKeeper

- Closed Source
- Nutzung nicht kostenlos
- … schade
- Flaschenhals: Changeset Datei

Schwächen von Subversion

- Die meisten Schwächen von RCS/CVS bleiben
- Nur globale Versionsnummern
- Ältere Versionen waren unzuverlässig
- Keine Signaturen/Verifikation

Schwächen von GIT

- Keine Diff-Daten (daher hoher Platzbedarf)
- Branches/Merges sind normale Deltas
- Relativ hoher Aufwand beim initialen Einchecken
- Erkennen des Status erst nach Lesen vieler Dateien
- Steigender Aufwand bei älteren/aktiven Projekten
- Nur globale Versionsnummern
- Keine Schlüsselwortexpansion

Schwächen von Mercurial

- Dateinamen vermutlich begrenzt auf US-ASCII
- Nur globale Versionsnummern
- Keine hierarchischen Projekte
- Keine Schlüsselwortexpansion
- Abhängigkeit von Python

Vorteile von SCCS

- Schnell
- Keine Datenverluste durch SCCS Fehler seit >25 Jahren
- Datenstrukturen unterstützen aktiv Annotation
- Datenstrukturen unterstützen aktiv Branches/Merges
- Leicht erweiterbares History-File-Format
- MR-Nummern können einzelne Aktionen erlauben

Das historische SCCS Konzept

- Versionsverwaltung von disjunkten Einzeldateien
- Checksumme zur Sicherung der Integrität
- Atomare Updates für jede Einzeldatei
- Platzsparende Speicherung durch *Interleaved deltas* (alle Zeilen aus allen Versionen nebeneinander)
- Kein Extra-Aufwand für das Anlegen von Branches
- Kein Extra-Aufwand für das Mischen von Branches
- Gleichzeitige Nutzung durch mehrere Nutzer
- Verwaltung von Nutzer-Zugriffslisten pro Datei
- Spezielles Auschecken zum Editieren notwendig

Die Programme von SCCSv4

- `get` **Holt eine Version aus einem Historyfile**
- `delta` **Baut neue Versionen in ein Historyfile ein**
- `admin` **Legt neue Historyfiles an oder ändert Parameter**
- `cdc` **Ändert einen Deltakommentar (Versionskommentar)**
- `comb` **Beseitigt delta-Bereiche (z.B. alles vor SID 1.9)**
- `help` **Gibt Erklärungen für Fehlermeldungen**
- `prs` **Gibt Metadaten von Historyfiles aus**
- `rmdel` **Beseitigt ein Delta am Ende eines Branches**
- `sact` **Listet Edit Aktivitäten auf**

Die Programme von SCCSv4 Fortsetzung

- `sccsdiff` **Listet Unterschiede zwischen Versionen**
- `unget` **Beseitigt die Effekte von** `get -e`
- `val` **Validiert SCCS Historyfiles auf Defekte**
- `what` **Sucht SCCS-Marker** `@(#)` **in Dateien**
- `sccs` **SCCS Frontend mit zusätzlichen Meta-Kommandos wie:**
  - `sccs create` **Überführen von Files ins SCCS**
  - `sccs diffs` **Vergleicht Edit-Version gegen ältere**
  - `sccs edit` **Checkt editierbare Version aus**
  - `sccs unedit` **Beseitigt editierte Dateien**

Das SCCSv4 Historyfile Format

- **Einmal: SCCS Magic auf erster Zeile `^Ahnnnnn`**
  - **Alle Steuerzeilen in SCCS beginnen mit `^A`**
  - **`h` als Marker für *hash***
  - ***`nnnnn`* die untersten 16 Bits der „signed“ Checksumme**
- **Eine Folge von SCCS Delta Tabelleneinträgen**
  - **Statistik-Zeile (inserted/deleted/unchanged)**
  - **Delta-Zeile (Version, Datum, Autor, Vorgängerversion)**
  - **I/X/G Zeilen (include/exclude/ignore) für Branch/Merge**
  - **MR-Nummern (Modification Request)**
  - **Delta-Kommentar (beliebig viel Text)**

Das SCCSv4 Historyfile Format Fortsetzung

- Usernamen bzw. GruppenIDs für Zugriffsrechte
- SCCS Flags
- SCCS Beschreibungsfile
- Body mit dem interleaved Delta-Block (Weave)

Das SCCSv4 Historyfile Format Beispiel

```
^Ah41388

^As 00001/00001/00013

^Ad D 1.2 07/01/27 15:05:05 joerg 2 1

^Ac man neu

^Ae

^As 00014/00000/00000

^Ad D 1.1 06/12/20 23:46:27 joerg 1 0

^Ac date and time created 06/12/20 23:46:27 by joerg

^Ae

^Au

^AU

^Af e 0

^At

^AT
```

```
^AI 1

.....
```

Fraunhofer FOKUS

Interleaved Deltas Beispiel

```
^AI 1
zeile 1
^AD 2
zeile a
^AE 2
^AI 2
zeile 2
^AE 2
^AE1
```

Vorteile des SCCS Historyfile Formats

- Einfach zu Parsieren
- Checksumme schützt vor unbemerkter Beschädigung
- Interleaved Deltas:
  - Schneller Zugriff auf beliebige Versionen
  - Annotierter Text ohne zusätzlichen Aufwand
    - Autor, Zeit, Version für jede Zeile ist in den Datenstrukturen
  - Merge (von Branches ohne Überlappung) ohne zusätzlichen Aufwand
  - Keine bekannten Probleme seit mehr als >25 Jahren

Weitere Vorteile von SCCS

- **SCCS ist Bestandteil des POSIX Standards**
- **`make` hat bereits Unterstützung für SCCS**
- **Die Web-GUI „opengrok“ unterstützt SCCS bereits**
- **SCCS ist extrem schnell (typisch ca. 5-6x schneller als andere Systeme – RCS inbegriffen)**

Grundlagen zur Nutzung von SCCS

- **File editieren mit `$EDITOR` der Wahl**
- `sccs create filename` **bringt File unter SCCS Kontrolle**
- `sccs edit filename` **checkt File zum Editieren aus**
- `sccs diffs filename` **zeigt Unterschiede**
- `sccs delget filename` **neue Version → History-Datei**
- `sccs help` **für Übersichtshilfe**
- `sccs help sccs` **für Optionen von** `sccs(1)`
- `sccs help sccs_subcmds` **für Liste von Sub-Kommandos**
- `sccs help cm3` **für Hilfe zum Fehlercode „cm3“**

Entwicklung anderer Systeme nach 1972

- 1982: Walter Tichy schreibt RCS, ein SCCS Nachbau
  - Keine Checksummen, daher damals 20% schneller
  - Umgekehrte Deltas, langsam für alte Versionen
  - Dateiformat aufwendiger zu parsieren
  - Insgesamt keine Vorteile zu SCCS
- 1986 Sun veröffentlicht NSE (Network Software Engineering)
- 1986 CVS als Frontend zu RCS (ab 1992 eigenständig)
- 1990 Sun TeamWare als SCCS Frontend
- 1997 BitKeeper SCCS von Larry McVoy

- 1986 NSE von Sun als Frontend zu SCCS
- 1986 CVS als Frontend zu RCS
- 1990 TeamWare von Sun als Frontend zu SCCS
- 1997 BitKeeper SCCS (Neuentwicklung v. Larry McVoy)
- 2000 Subversion, gedacht als Nachfolger zu CVS
- 2005 GIT
- 2005 Mercurial

Verwandtschaften

- Alle modernen verteilten OSS Versionsverwaltungen basieren letztlich auf den Ideen von BitKeeper SCCS
- BitKeeper SCCS basiert auf den Ideen von TeamWare
- TeamWare basiert auf den Ideen von NSE
- NSE ist ein Frontend zu SCCS
- Damit bauen alle modernen Systeme letztlich auf SCCS auf
- Verteilt arbeiten TeamWare, BitKeeper, GIT, Mecurial

Warum ist SCCS heute nicht dominierend?

- 1985 waren sowohl SCCS als auch RCS Closed Source
- SCCS war binär kostenlos auf allen Systemen
- RCS kostete sogar Geld
- Das unterlegene RCS wurde jedoch zum richtigen Zeitpunkt in OSS umgewandelt. Der richtige Zeitpunkt war vor dem Aufkommen von FreeBSD und CVS
- SCCS wurde zu spät OpenSource (erst 2006)

Entwicklung bei SCCS nach 1977 (SCCSv4)

- 1986 NSE Erweiterungen von Sun
- 1988 Unterstützung von Binärfiles ab SunOS-4.0
- Nach 1990 Unterstützung für beliebig lange Text-Zeilen
- 2000-2001 Diskusionen mit SCO über OSS
- 2004-2006 Diskussionen mit Sun über OSS
- 20. Dezember 2006 Sun gibt SCCS unter der CDDL frei
- 21. Dezember 2006 Schily SCCS kompiliert auf Solaris
- 7. Januar 2007 Schily SCCS-0.3 läuft auf Solaris, Linux, FreeBSD, Mac OS X, Cygwin

Schily-SCCS seit Dezember 2006

- 26. Dezember 2006 SCCS-0.1 erste publizierte Version
- 11.Februar 2007, SCCS-1.0 läuft auf nahezu allen Platt-formen
- 11.Juli 2011, SCCS-5.01 bringt viele Neuigkeiten, wie z.B. Rekursive Bearbeitung in Directory Bäumen
- Aktuelle Entwicklung geht zu SCCS-6.0 als verteiltes System mit Unterstützung für *Changesets* in Projekten
  - Projekte statt Einzeldateien (in Entwicklung)
  - Netzwerkunterstützung (geplant)

Neue Features in Schily-SCCS

- 4,7x weniger CPU-Zeit-Bedarf seit Dezember 2006
- Unterstützung bis zum Jahr 9999 (64 Bit Versionen)
- 4-stellige Jahreszahlen auch in Keywords
- Rekursives Arbeiten mit `sccs -R command`
- Schnelle Massenübernahme in SCCS, OpenSolaris mit 46000 Files und 500 MB benötigt nur 8 Sekunden (GIT 100s)
- Erheblich verbesserte Fehlererkennung in val(1)
- Bessere man Pages und besseres `sccs help`
- Automatische Regressionstests zur Qualitätssicherung

Neue Features in Schily-SCCS

- `^A` **am Anfang von Textzeilen und Dateien ohne** `\n` **am Ende als „Textfiles“ (zur Kompatibilität nur aktiv mit v6)**
- **Neues erweiterbares SCCSv6 Historyfile Format**
- **Neue versionsspezifische Checksummen (ähnlich BK)**
- `sccslog` **Kommando neu (seit 1997)**
- `rcs2sccs` **Kommando neu**
- `sccs istext` **neu, prüft ob Text Deltas möglich sind**
- `sccs editor` **neu, startet** `$EDITOR` **– evt. vorher Checkout**

Ausblicke/Planung zum verteilten SCCS

- **Einführung einer Directory `.sccs` im Project-Set Home**
- **Einführung einer ChangeSet Datei zur Beschreibung von Projekten**
- **`sccs init` initialisiert ein neues leeres Projekt (schaltet um in den Projekt Modus)**
- **`sccs clone` kopiert ein Projekt von einem Master**
- **`sccs export` erzeugt ein Changeset-Diff-File**
- **`sccs import` importiert ein Changeset-Diff-File (wird hg export Files unterstützen)**

Ausblicke/Planung zum verteilten SCCS

- `sccs pull` **und** `sccs push` **zum automatischen Update gegen andere Repositories**
- `sccs outgoing` **und** `sccs incoming` **zur Anzeige was dabei transportiert würde**
- `sccs add`, `sccs remove`, `sccs rename`, `sccs revert` **zum Modifizieren von Projekten**
- `sccs commit` **erzeugt eine neue Projekt-Release**
- `sccs tag` **zum Hinzufügen eines Versionstags**

Ausblicke/Planung zum verteilten SCCS

- `sccs smoosh` **um zwei unterschiedliche Kopien in eine Version mit Branch zu wandeln (Idee von BK)**
- **Netzwerkunterstützung**
- **Merge Tools für nicht-triviale Konflikte**

Skalierbarkeit - Projekt-Statisktik

- OpenSolaris hatte in 5 Jahren 13150 Changesets (bei ca. 46000 Files)
  - ca. 168000 Modifikationen an Dateien (insgesamt)
  - ca. 12,8 modifizierte Dateien pro Changeset
  - ca. 8,3 Mercurial Changesets pro Tag (106 Dateiänderungen pro Tag)
- Linux hat ähnliche Änderungsfrequenzen wie OpenSolaris (aber nur 11000 Files)
- Solaris hatte in 25 Jahren ca. 190000 Changesets
  - ca. 465000 Modifikationen an Dateien (insgesamt)
  - ca. 2,4 modifizierte Dateien pro Changeset
  - ca. 21 äquivalente Changesets pro Tag (50 Dateiänderungen pro Tag)
- BSD UNIX hatte in 15 Jahren ca. 48000 Changesets (bei ca. 13000 Files)
  - ca. 108000 Modifikationen an Dateien (insgesamt)
  - ca. 2,2 modifizierte Dateien pro Changeset
  - ca. 19.8 äquivalente Changesets pro Tag (44 Dateiänderungen pro Tag)

Anforderungen für ein Projekt über 50 Jahre

- ca. 350000 Changesets bei Arbeiten wie an BSD/UNIX
  - d.h. 350000 Änderungen an der Changeset Datei
- ca. 132000 Changesets bei Arbeiten wie an OS/Linux
- Mindestens 1000 Änderungen an normalen Dateien

Skalierbarkeit von SCCS

- SCCS ist unabhängig von der Anzahl der Dateien
- Zeit für das initiale Einchecken von OpenSolaris:
  - GIT: 1:40 → 300 MB Repository
  - SCCS (erwartet) 0:20 → 120 MB Repository (komprim.)
- Zeit für das Bearbeiten eines Commits für OpenSolaris mit SCCS nach 50 Jahren Entwicklung:
  - Geschätzt unter einer Sekunde mit HW von vor 5 Jahren
- Zur Zeit läuft ein Dauertest mit SCCS, >2 Millionen Deltas
  - Platzbedarf mit SCCS ca. 300 MB (komprimiert 160 MB)
  - Platzbedarf mit GIT geschätzt: ca. 15 TB

Hilfe wird benötigt

- Diskussionspartner zum Besprechen der Konzepte
- Definitiv Autoren für eine GUI zu SCCS
- Aber auch für alle anderen Bereiche:
  - Regressionstests schreiben
  - Testkompilationen und Tests auf diversen Systemen
  - Importskripte für andere Versionsverwaltungen
  - Doku schreiben
  - Bug-Reports
  - ...

SCCS Download

- ftp://ftp.berlios.de/pub/sccs/ Nur SCCS Quellen
- ftp://ftp.berlios.de/pub/schily/ Alle „Schily“ Quellen

SCCS Dokumentation

- http://sccs.berlios.de
- http://sccs.berlios.de/man/index.html
- http://sccs.berlios.de/man/sccs.1.html
- http://sccs.berlios.de/man/sccsfile.4.html

Weiterführende Literatur

- http://pubs.opengroup.org/onlinepubs/9699919799/utilities/sccs.html
- http://basepath.com/aup/talks/SCCS-Slideshow.pdf
- http://minnie.tuhs.org/cgi-bin/utree.pl?file=PWB1/usr/news/pibs
- http://download.oracle.com/docs/cd/E19504-01/802-5880/6i9k05dhp/index.html
- http://www.catb.org/~esr/writings/version-control/version-control.html
- http://hub.opensolaris.org/bin/view/Project+opengrok/WebHome

Dieser Vortrag liegt online unter

- http://sccs.berlios.de/SCCS-Chemnitz-2012.pdf

Die Mailing-Liste ist unter:

- https://lists.berlios.de/mailman/listinfo/sccs-devel

Danke