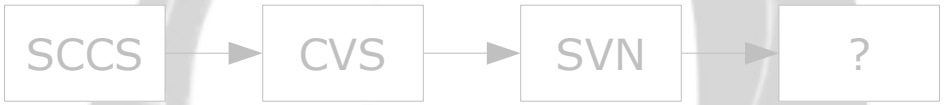
Präsentation der Bachelor-Arbeit zum Thema

„Open Source Version Control“



Autor: Thomas Keller
Betreuer: Prof. Dr. M. Frank

Überblick

- Einführung
- Open Source
  - Open Source – Was ist das?
  - Beweggründe für OS-Entwicklung
  - Mögliche Geschäftsmodelle
- Versionskontrollsysteme
  - Versionskontrolle – Was ist das?
  - Gründe für Versionskontrolle
  - Grundlagen und Konzepte
  - Quelloffene Versionskontrollsysteme

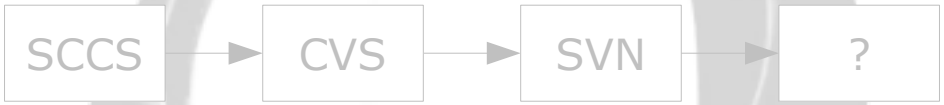
Einführung

- Vorliegende Arbeit verbindet die beiden Themen „Open Source“ und „Versionskontrolle“
- Wurde auf Englisch verfasst und ist frei verfügbar (URL am Ende)
- Motivation für den Open-Source-Teil:
  - Image von Open Source verbessern
  - Open Source und Free Software als Bewegung, als neuen Denkansatz verstehen
  - Möglichkeiten aufzeigen, als Firma Open Source zu nutzen und vielleicht sogar selbst zu partizipieren

Einführung (II)

- Motivation für den Teil über Versionskontrolle:
  - Versionskontrolle als Beispiel für Innovation aus der Open Source Gemeinschaft
  - Vorteile von Versionskontrolle aufzeigen
  - Konzepte und Architekturen vorstellen
  - Frei verfügbare (Open Source) Versionskontrollsysteme vergleichen und bewerten
  - Arbeitsweisen mit Versionskontrollsystemen aufzeigen, „Best Practises“

Open Source

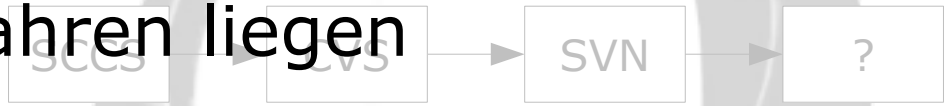


Open Source – Was ist das?

- Open Source = quelloffen, d.h. der Quelltext eines Programms ist frei verfügbar und darf modifiziert / weiterverteilt werden
- Unterschiedliche Lizenzen, die den „Freiheitsgrad“ bestimmen - Copyleft!
- 10 Attribute, die lt. der OSI eine Open Source Lizenz als solche qualifiziert, die wichtigsten: „free redistribution“, „availability of the source code“, „allow modifications and derived works“

Open Source – Was ist das? (II)

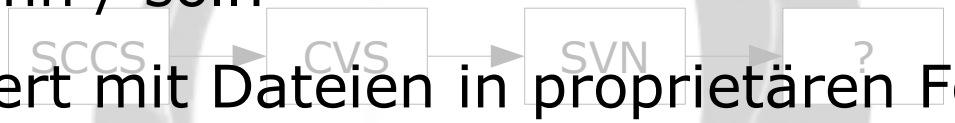
- Weitere Attribute:
  „No restrictions in use / audience (no NDAs)“,
  „Technology neutrality“, ...
- Open Source != kostenlos!
- Grundlage für Open Source ist die freie Softwarebewegung, deren Ursprünge in den 60er und 70er Jahren liegen



- Wichtigste Persönlichkeit bis heute: Richard Stallmann

Beweggründe für OS-Entwicklung

- Nutzer proprietärer (copyrighted) Software sind in Bezug auf Support vom „Wohlwollen“ des Herstellers der Software abhängig
  - Was ist, wenn die Software nicht auf älterer Hardware läuft?
  - Was ist, wenn eine Lizenz ausläuft und nicht erneuert werden kann / soll?
  - Was passiert mit Dateien in proprietären Formaten; können sie später nach dem EOL von anderen Programmen gelesen werden?
  - Bugs? Lizenzkosten? ...



Beweggründe für OS-Entwicklung (II)

- Ausweg ist Open Source
  - Geringe Lizenzkosten (geringer TCO)
  - Quellen verfügbar, eigene Anpassungen möglich
  - Support und ständige Weiterentwicklung durch eine aktive Gemeinschaft
  - Schon heute sind einige Open Source Projekte ihren kommerziellen Pendanten überlegen (bspw. Eclipse)

Mögliche Geschäftsmodelle

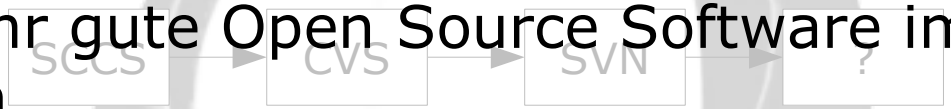
- Nutzung von Open Source Software ist eine Sache, an Open Source Projekten partizipieren eine andere
- Sponsoring ist gern gesehen, da Kosten (z.B. Bandbreite) gedeckt werden müssen (erfolgt ansonsten meist über Spenden)
- Eric Raymond beschreibt in „The Magic Cauldron“ mögl. Geschäftsmodelle, welche es Firmen erlaubt, mit einem eigentlich „kostenlosen“ Gut trotzdem Geld zu verdienen

Mögliche Geschäftsmodelle (II)

- Modell „Give away the recipe, open a restaurant“
  - Professionelle Unterstützung für Firmen
  - Individualanpassungen
  - Öffnung von Software ermöglicht hier breitere Marktakzeptanz
- Modell „Widget Frosting“
  - Insb. an Hardwarehersteller gerichtet
  - Konzentration auf Innovation, offene Treiber und UI-Entwicklung

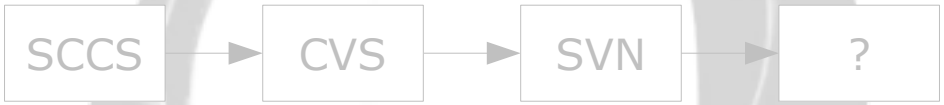
Mögliche Geschäftsmodelle (III)

- Weitere Modelle umfassen z.B. „Accessorizing“, d.h. Produkte rund um Open Source Software (Bücher, Merchandize, usw.)
- Nicht jede kommerzielle Software ist für die „Öffnung“ geeignet, Open Source Software markiert Closed-Source Entwicklung nicht obsolet
  - Bereits sehr gute Open Source Software im Bereich vorhanden



  - Geringer, öffentlicher Nutzen => geringe Beteiligung der Gemeinschaft zu erwarten

Versionskontrollsysteme



Versionskontrolle – Was ist das?

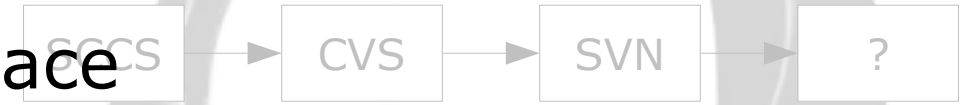
- Versionskontrolle bezeichnet das konsistente Speichern aller Eigenschaften und Versionen eines Objekts
- Anwendung vor allem in der Softwareentwicklung, aber auch andere denkbar (z.B. Verteilungsszenarien)
- Rasche Evolution im Bereich, Zugriff auf Systeme früher vor allem zentralisiert und wechselseitig, heute eher dezentralisiert und gleichzeitig (concurrent) -> schneller und flexibler

Gründe für Versionskontrolle

- Zentralisierte Verwaltung aller Dateien eines Projekts
- Tracking der Historie und von Benutzeraktionen
- Softwareentwicklung: Management paralleler Entwicklungszweige
- Paralleler Zugriff auf selbe Dateien (und späteres Zusammenführen der Änderungen)
- Im Vgl. zu klassischen Dateiservern verbrauchen VC-Systeme sowohl bei der Dateispeicherung als auch -übertragung viel weniger Platz (Deltas)

Grundlagen und Konzepte

- Product-Space
  - Sichtweise bezeichnet den einfachen, nicht versionisierten Zustand eines Produkts / eines Projekts
  - Einzelne Objekte können Beziehungen untereinander aufweisen („alle Dateien in einem Verzeichnis“ oder „benötigte Dateien für die Übersetzung / Kompilierung einer anderen Datei“)
- Version-Space
  - Definiert Objekte, deren Attribute und Deltas, die versionisiert werden sollen
  - Unterschied Logical und Practical Version-Space
  - Versionisierung erfolgt heute meist **explizit**



Grundlagen und Konzepte (II)

- Dateisystem-basiertes Repository
  - Versionisierte Objekte werden als echte Dateien im Backend abgelegt (Bsp: RCS-Format)
  - Vorteile: Einfaches Backup, Nachvollziehbarkeit
  - Nachteile: Skaliert mit zugrundeliegendem Dateisystem
- Datenbank-basiertes Repository
  - Relationale Datenbank übernimmt Speicherung
  - Vorteile: Zusätzlicher, wohldefinierter Layer für Drittapplikationen, Nutzung DB-spezifischer Optimierungen oder auch Replikation
  - Nachteile: „Hot Backups“ schwer, Synchronisierung?

Grundlagen und Konzepte (III)

- Zentralisierte Versionskontrolle
  - Ein Server, auf den mehrere Clients zugreifen
  - Vorteile: zentrale Anlaufstelle, einfache Konfiguration in kleinen Teams
  - Nachteile: „Central Point of Failure“, Skalierbarkeit, Rechteverwaltung in großen Teams aufwendig
- Dezentralisierte Versionskontrolle
  - „P2P“-Charakter, jeder Client ist auch gleichzeitig Server
  - Vorteile: eigenes Repository offline, hohe Ausfallsicherheit
  - Nachteile: „Umdenken“ erforderlich, Systeme wenig genutzt / wenig bekannt

Quelloffene Versionkontrollsysteme

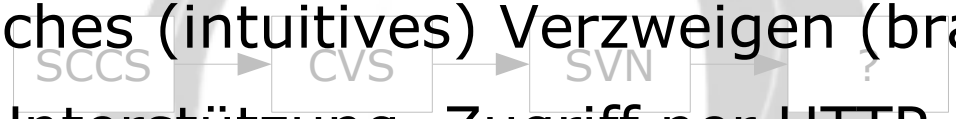
- CVS
  - „Standard"-Systeme seit vielen Jahren
  - Zentralisiert, Dateisystembasiert
  - Unterstützung vieler Drittanbieter
  - Nachteile: keine neuen Features, da Entwicklung eingestellt; es fehlt an Metadaten, Rename-Support, atomaren Aktionen uvm.
  - Sollte für neue Projekte nicht mehr genutzt werden, ggf. können ältere Repositories in andere Systeme importiert werden (CVSNT, Subversion, monotone, ...)

Quelloffene Versionskontrollsysteme (II)

- CVSNT
  - CVSNT ursprünglich als reiner Windows-Pendant zu CVS geplant, heute für fast alle Plattformen verfügbar
  - Zentralisiert, Dateisystem-basiert (zukünftig wohl mit Datenbank-Unterbau)
  - wird aktiv weiterentwickelt, wenn auch Rückwärtskompatibilität große Sprünge verhindert
  - Geeignet, wenn neues System wenig Umlernen von CVS bei den Benutzern herbeiführen soll
  - Guter, kommerzieller Support, viele Protokolle
  - Keine Unterstützung von Metadaten, Rename-Support im Alpha-Stadium

Quelloffene Versionskontrollsysteme (III)

- Subversion (SVN)
  - Komplette Neuentwicklung, Support und Sponsoring durch CollabNET
  - Nahezu gleiches Kommandoset wie CVS/CVSNT
  - Ältere Versionen DB-basiert (BerkeleyDB), neueres „FSFS"-Format dateibasiert, zentralisiertes Modell
  - Sehr einfaches (intuitives) Verzweigen (branching)
  - WebDAV-Unterstützung, Zugriff per HTTP
  - Nahezu alle modernen Features inkludiert (Metadaten, Atomare Operationen, ...), lediglich Rename-Support ist noch etwas holprig

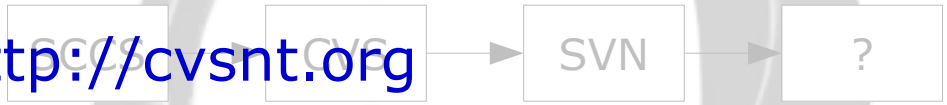


Quelloffene Versionskontrollsysteme (IV)

- monotone
  - Dezentraler Ansatz, dreigliedrige Architektur, Datenbank-Backend (SQLite)
  - Identifikation aller Entitäten über SHA-1-Summen
  - CVS-ähnliches Kommandoset
  - Volle Unterstützung von Renaming (sogar ringförmig!)
  - Leider noch im Beta-Stadium, bis jetzt kein kommerzieller Support, so gut wie keine GUI's verfügbar
  - Protokoll auf Basis von Netsync, IANA registrierter Port
  - „sauberste“ Implementierung des dezentralen Ansatzes im Vgl. zu anderen Systemen (GNU Arch)

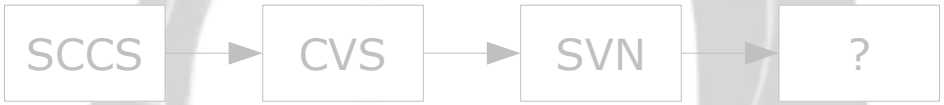
Quelloffene Versionskontrollsysteme

- Zusammenfassung
  - Für unmittelbaren Einsatz im Unternehmen am ehesten Subversion und CVSNT geeignet (Support!)
  - monotone könnte in Zukunft eine Option werden
- URLs:
  - CVS: http://ximbiot.com/cvs/wiki/
  - CVSNT: http://cvsnt.org
  - Subversion: http://subversion.tigris.org
  - monotone: http://venge.net/monotone



Vielen Dank für die Aufmerksamkeit!

Fragen?



Die Arbeit ist abrufbar unter folgender URL:

http://thomaskeller.biz/work/thesis/thesis_final.pdf