HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Versionhallinta

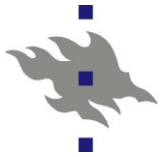
Matti Paksula ja Esko Luontola

Syksy 2009

Tietojenkäsittelytieteen laitos
Kuje Research Group

Kurssin rakenne

- 2 luentoa
  - Versionhallinta ja keskitetty malli (Subversion)
  - Hajautettu malli (Git)
- 2 harjoitukset
  - Ensimmäisissä teemana Subversion
  - Toisella kerralla Git
  - Alkavat ensi viikolla
- Harjoitustyö
  - Kaksi osaa: Subversion ja Git
  - Tällä pääsee kurssin läpi, ei siis tenttiä



Versionhallinta

- Mikä kurssi on?
  - Katsaus versionhallintaan tekniikkana, jossa opetellaan hieman enemmän, kuin pelkkä “commit ja update”
  - Kurssi järjestetään nyt ensimmäisen ja viimeisen kerran
    - Materiaali sulautetaan jossain muodossa “Ohjelmistotuotanto” -kurssiin
- Miksi Git ja Subversion?
  - Miksei Mercurial, Bazaar, CVS, Darcs …

Kieli

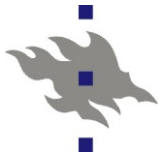
- “checkout reposta ja trunkin branchaus”
  - vs “työkopion hakeminen tietovarastosta ja trunk -hakemiston haarauttaminen”
- Koska yleisesti ei puhuta *tietovarastosta* ja *haarasta*, vaan *repositorystä* ja *branchistä*, kurssin materiaalit käyttävät tätä kamalahkoa kieltä.

Palautetta, kiitos!

- Kurssisivulla on palautelaatikko
- Normaalia kurssipalautetta arvostetaan tietysti myös

Versionhallinnan käyttötarkoituksia

- Versiointi (d'oh)
- Tiimityö
  - Tiedostojen jakaminen hankalaa, projektista on useita samanaikaisia versioita
- Varmuuskopiointi, tilan tallettaminen
  - Ilman versionhallintaa:
    - backup-gradu-20090804/
    - tomiva_versio_harjoitustyo.php
    - toimiva_versio2_harjoitustyo.php
    - Kokeilu2.java
- Historia
  - Tiedetään kuka teki muutoksen, milloin (aika) ja miksi (muutoksen kommentti)



SVN historiaa

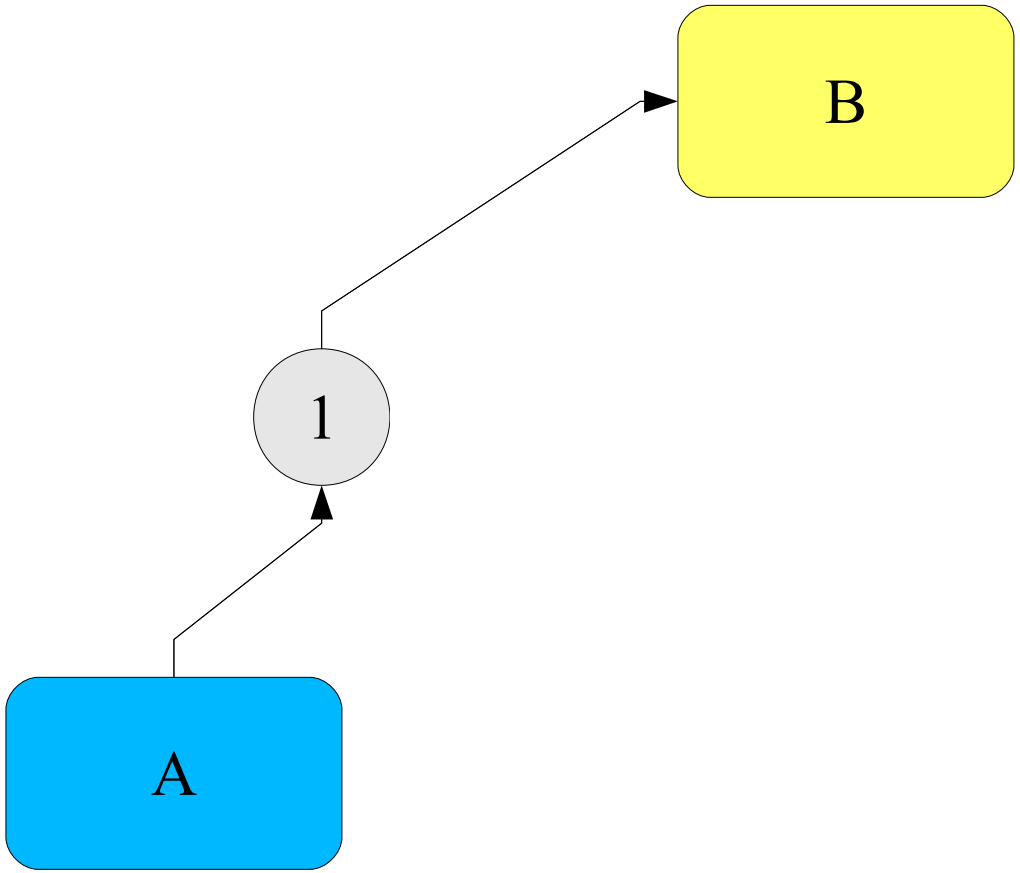
- Tarkoituksena korvata CVS
- Kehitetty 2000 lähtien

- “Subversion - CVS done right.
  There is no way to do CVS right.”
  – Linus Torvalds
- Kuitenkin hyvin yleisesti käytetty ja “ihan ok”

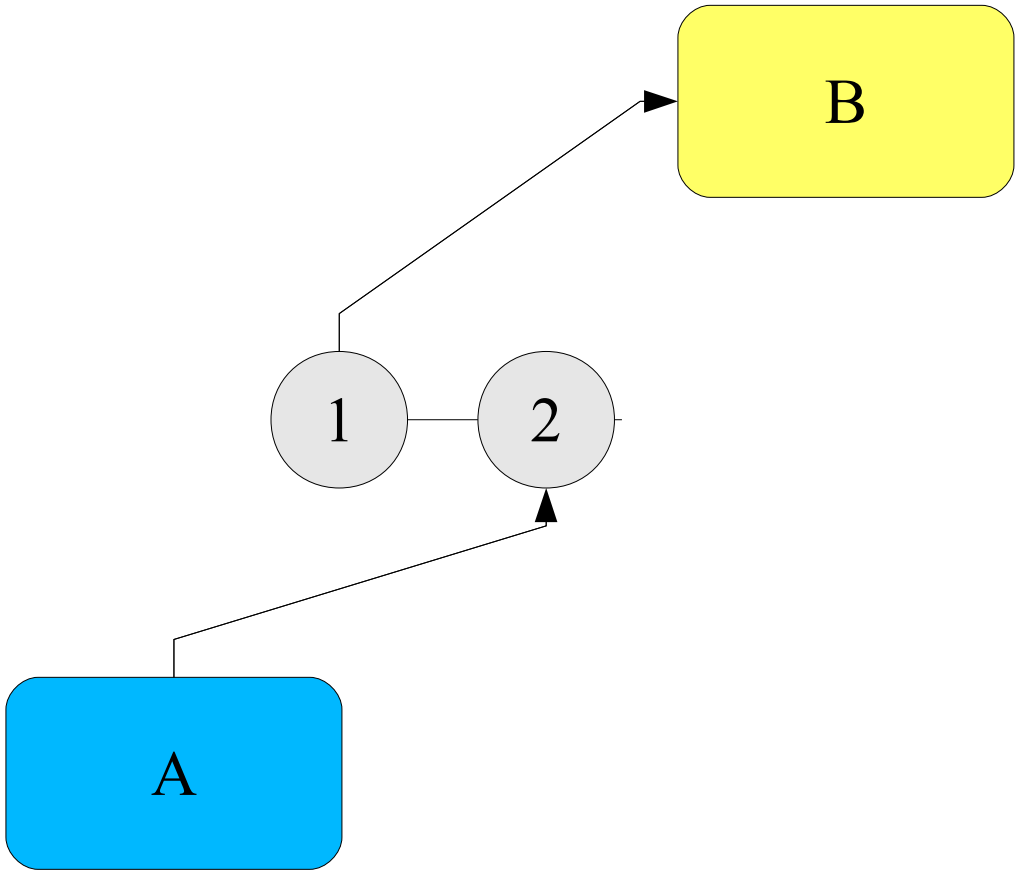
Käsitteitä

- repository
  - keskitetty paikka projektin tiedostoille
  - hyvä suomennos voisi olla tietovarasto..
- revision
  - Tietty tila repositoryssä pidetyssä projektissa
  - r1 → r2 → r3
    - Mikä tahansa tila voidaan palauttaa tai vertailla kahden eri tilan välisiä muutoksia
- checkout
  - Haetaan repositorystä (yleensä viimeinen) tila omaan..
- ..working copy, työkopio
  - Paikallinen kopio projektin tiedostoista

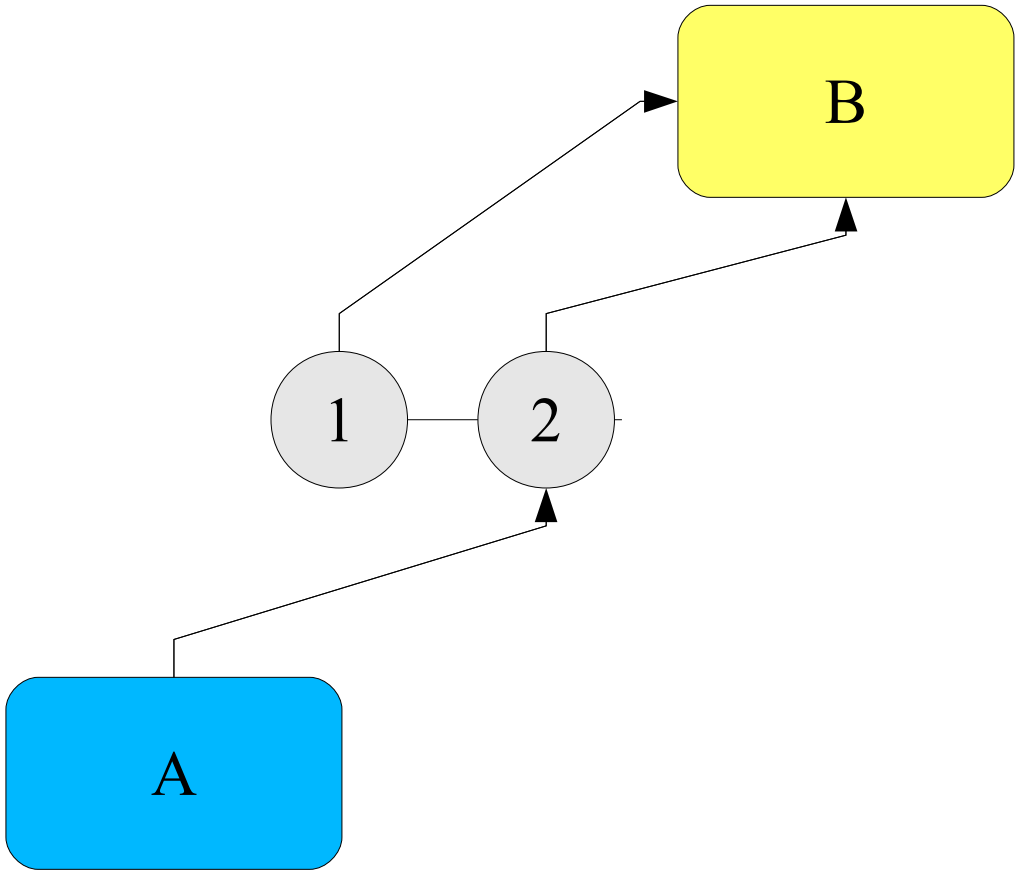
A:n joukko tiedostoja muodostaa revision 1, jonka B hakee itselleen.



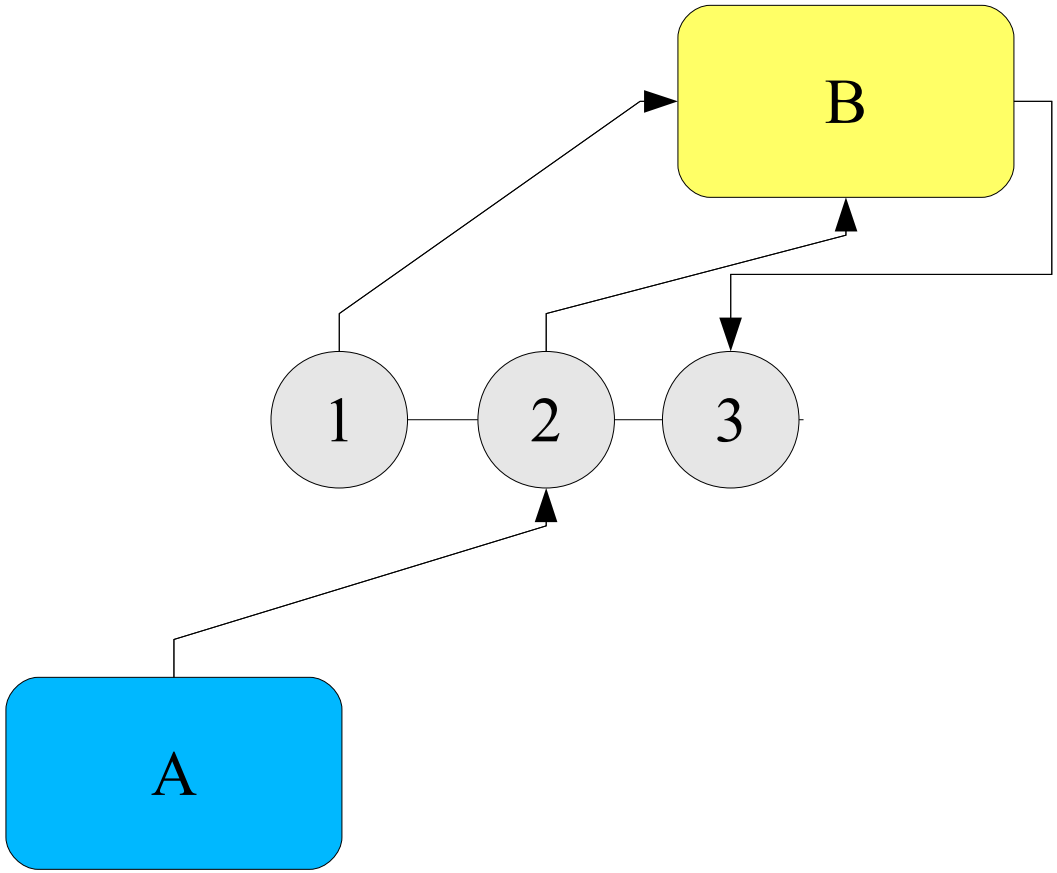
A tekee muutoksia ja lähettää tämän repositoryyn. Joukko muodostaa repositoryssä revision 2



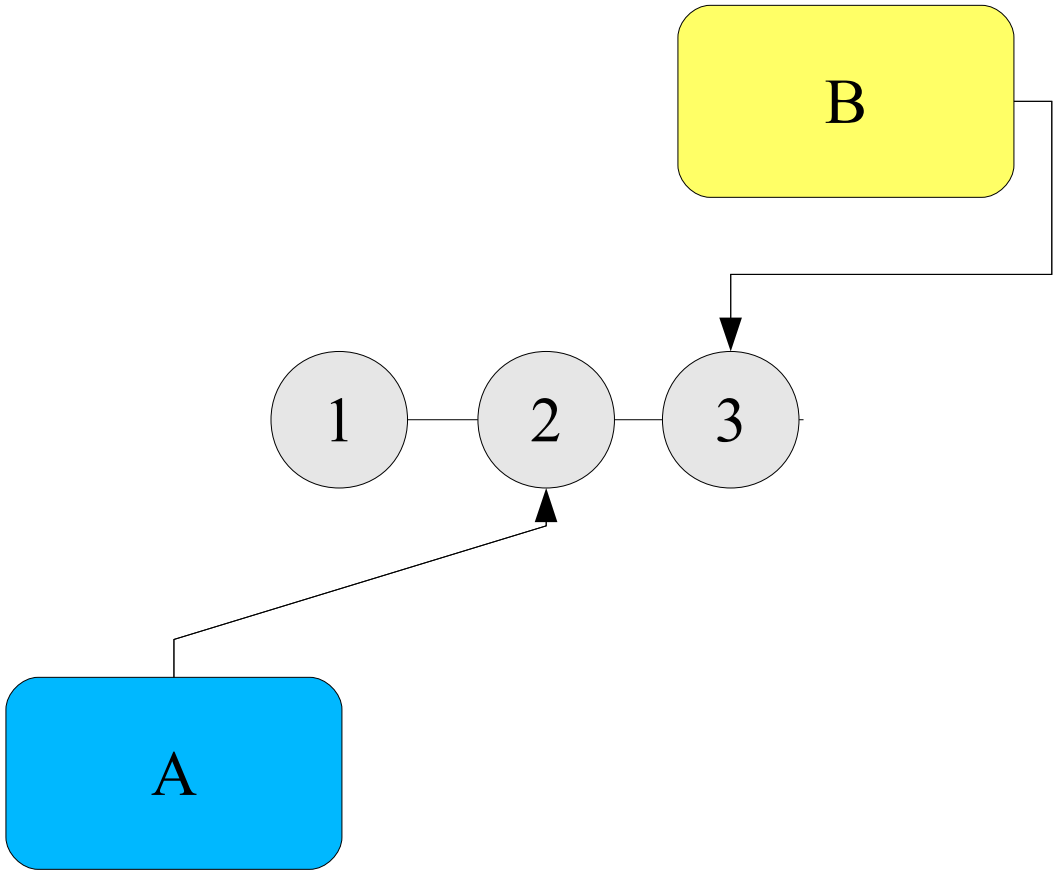
B päivittää oman työkopionsa hakemalla “A:n” muutokset revisiosta 2



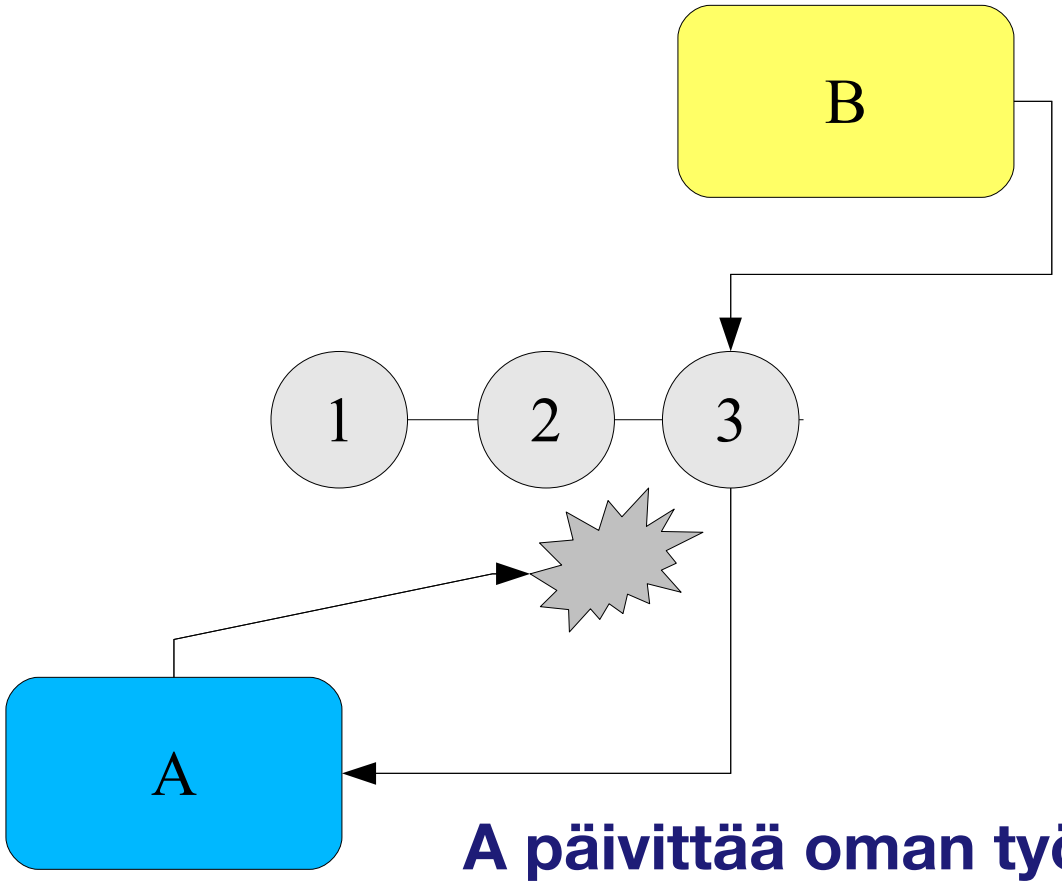
B lähettää omat muutoksensa (johon on yhdistetty myös revisio 2:n tila) revisioksi 3



A tekee omassa työkopiossaan muutoksia, jotka ovat ristiriidassa revision 3 kanssa

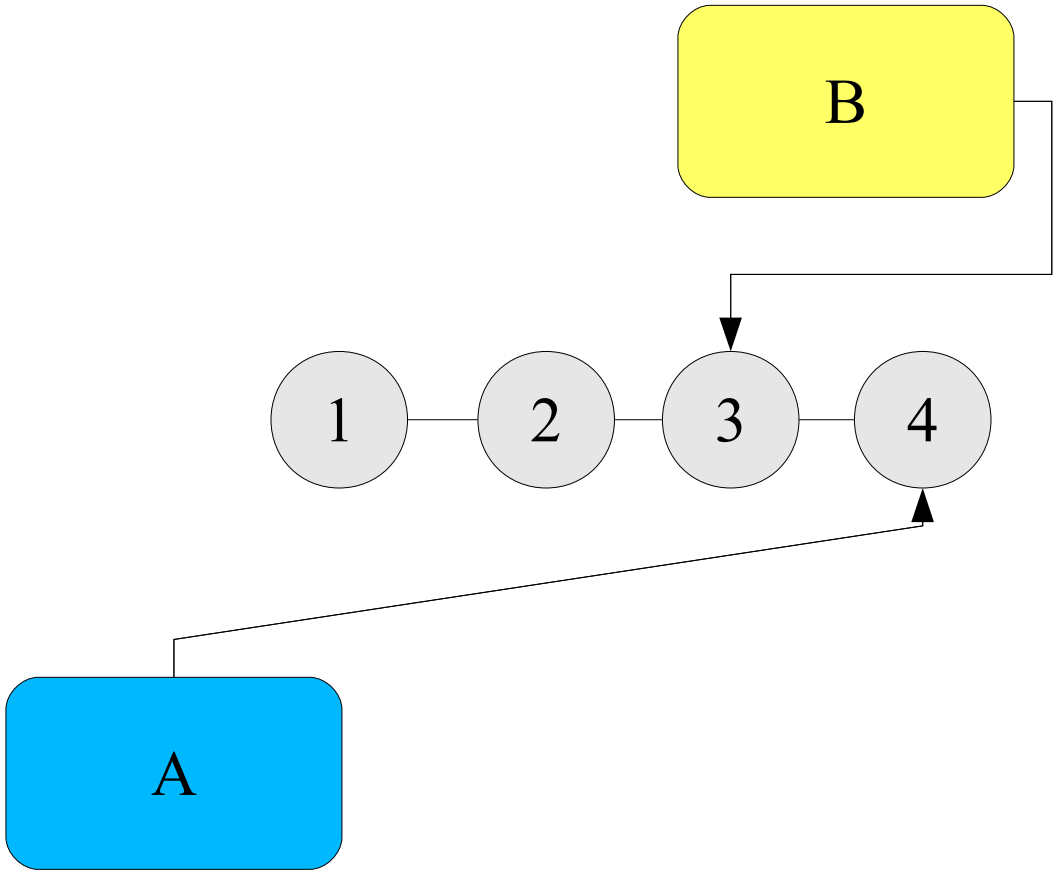


A yrittää luoda “revision 3”, mutta repository hylkää muutokset, sillä huomaa, ettei A ole ajantasalla



A päivittää oman työkopionsa 3, selvittää ristiriidat

Ristiriidaton joukko tiedostoja hyväksytään revisioksi 4



Asiakasohjelmat

- svn -komentorivityökalu
  - Tällä kurssilla kaikki tehdään tällä. On tärkeää oppia perusteet ilman hiirtä ja dialogeja.
- Elämää helpottavat graafiset työkalut, kuten:
  - TortoiseSVN (windows)
  - SmartSVN (kaikki)
  - Subclipse (Eclipsen plugin)

HELP

- Luennolla ja harjoituksissa ei millään ehditä kaikkea

  $ svn <komento> --help

- Hyvä on-line kirja: http://svnbook.red-bean.com/

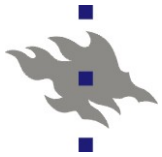
Repositoryn perustaminen

- Useimmissa tapauksissa repository on jo olemassa
- svnadmin

  $ svnadmin create <nimi>
  - Luo sisäisen repositoryn hakemistorakenteen, jota ei kannata muokata suoraan (vaan käyttää työkaluja)
- Tämän jälkeen on hyvä vielä miettiä, miten tämä on varmuuskopioitu (mutta siihen ei mennä tällä kurssila)

Käyttö

- SVN käsittelee repositoryä aina URL:n kautta
  - **file://** -protokolla jos pääsy paikallisesti
  - svn:// -protokolla
  - http:llä (WebDAV), mutta tähän tarvitaan jokin web-serveri, kuten Apache.
  - **svn+ssh://** -protokollalla, tällöin tarvitaan pääsy koneeseen, jossa repository sijaitsee.



Työkopion hakeminen

- Checkout, eli työkopion hakeminen paikallisesti
  $ svn checkout file:///koko/polku/repositoryyn kohde
  - Hakee viimeisimmän version työkopioksi (working copy) hakemistoon “kohde”
  - Hakemistossa “kohde” on tällöin “.svn” -hakemisto, jossa pidetään kirjaa muutoksista ym.

- checkout etänä omalle koneelle
  $ svn checkout svn+ssh://kohdekone/polku/repo
  - Tekee saman kuin edellinen, mutta käyttäen SSH:ta.
  - Tarvitaan siis pääsy SSH:lla kohdekoneeseen, ei muuta.

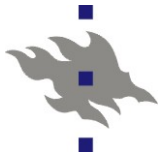
Lyhenteet

- svn checkout → co
- svn update → up
- svn status → st

$ svn help

- kaikki kommennot ja lyhenteet..

- HEAD, PREV..
  - “Lyhenteitä” revisionumeroille
    - HEAD: uusin revisio repositoryssä
    - PREV: edellinen



Repositoryn rakenne

- Työkopio, working copy
  - Kaikki muutokset, mitä tehdään paikallisesti ovat paikallisia, kunnes ne päivitetään repositoryyn (commit)
- Repository järjestetään yleensä kolmeen hakemistoon: trunk, branches ja tags
  - trunk
    - “viimeisin versio”, *päähaara*
  - branches
    - kokonaisia kopioita repositoryn tilasta, jonka tila on osittain eri, kuin esim. trunk -haara.
  - tags

    yksittäisiä repositoryn tiloja, kuten “1.0” tai “julkaisu”

Ensimmäinen commit

- tehdään muutoksia

  $ svn mkdir trunk branches tags

  - hakemistojen luontiin ja muut tiedosto-operaatiot on hyvä tehdä svn -komennon kautta

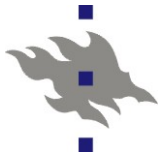
  A trunk

  A branches

  A tags

  - SVN kertoo meille, että työkopion seurantaan on lisätty (Added) nyt hakemistot
- Muutokset saadaan repositoryyn commitilla

  $ svn commit -m “Hakemistorakenne”
- SVN näyttää mitä repositoryyn lisättiin ja kertoo, millä revisionumerolla muutokset ovat repositoryssä.



Jatketaan lisäilyä

- editoidaan tiedostoa trunk/kokeilu.txt
- Kysytään SVN:ltä mikä on työkopion tila:

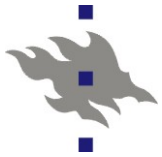
  $ svn st

  ? trunk/kokeilu.txt
  - Tiedostoa kokeilu.txt ei ole vielä lisätty *seurantaan*
- Lisätään tiedosto versionhallinnan seurattavaksi

  $ svn add trunk/kokeilu.txt
  - SVN kertoo, että tiedosto on lisätty työkopion seurantaan. (ja saman sanoo $ svn st)

  $ svn ci -m “Kokeilu”

  Commited revision edellinen+1 viestillä “Kokeilu”



Muokkaaminen ja commit

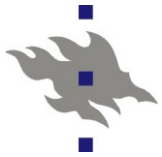
- Jos tiedostoa nyt muokataan

  $ svn st

  M trunk/kokeilu.txt

  - SVN kertoo tiedoston muuttuneen työkopiossa.
    - (ja tiedosto on seurannassa)

- $ svn ci
  - Koska viestiä ei aseteta komentorivillä, aukeaa $SVN_EDITOR, $EDITOR tai $VISUAL ympäristömuuttujissa määritelty editori.. ..tai virhe: ympäristömuuttujaa ei ole asetettu.



Commit

- Muista hyvät kommentit
  - Kerro lyhyesti ja ytimekkäästi mikä muuttui ja miksi. “Bugfix”, “korjaus” ei kerro mitään.
  - Muiden on helpompi lukea commit, kuin 200 riviä koodia.
- On parempi tehdä paljon pieniä committeja, kuin muutama iso / koko päivän työt kerralla.
- Jokaisen commitin tulisi käsitellä vain yhtä asiaa

Muiden tekemät muutokset

- Muiden tekemät muutokset saadaan päivitettäy omaan työkopioon updatella.
  $ svn update
  - SVN hakee uusimman tilan repositorystä ja kertoo mitä muutoksia tapahtui
- Mahdollisia muutoksia
  - **A**dded : tiedosto listättiin
  - **D**eleted : tiedosto poistettiin
  - Mer**G**ed : tiedoston sisältö yhdistettin automaattisesti
  - **C**onflict : sisällön muokkaaminen ei onnistunut, sillä riveissä on ristiriita.
  - **U**pdated : Tiedosto päivitettiin

Conflict - Ristiriita

- Jos SVN ei osaa yhdistää kahta muutosta toisiinsa (muutokset koskevat samoja rivejä), syntyy ristiriita, joka pitää ratkaista käsin.

- Avataan tiedosto jossa ristiriita on:

  rivi, jolla ei ole konfliktia
  <<<<<<< .mine
  omassa työkopiossa oleva rivi
  =======
  revisiossa r3 oleva rivi
  >>>>>>> .r3
  rivi, jolla ei ole konfliktia

- Muokkaa konflikti siten, että lopputulos on haluttu ja poista siis <<<.., ===.. ja >>>.. -rivit

Ristiriidan merkitseminen selvitetyksi

- $ svn st

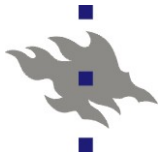
```
?      trunk/kokeilu.txt.r2 # revision 2 tila
?      trunk/kokeilu.txt.r3 # revision 3 tila
?      trunk/kokeilu.txt.mine          # työkopion tila
C      trunk/kokeilu.txt     # yhdistetty tila
```

Tiedostoa ei seurata versionhallinnassa

$ svn resolve --accept working trunk/kokeilu.txt

- Poistaa “ylimääräiset” tiedostot ja asettaa yhdistetyn työkopiossa tilan lopulliseksi tilaksi

- Vanha $ svn resolved on deprekoitu (!)



Eclipsen SVN-plugin & Conflict edit

diff - eroavaisuudet

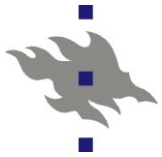
$ svn diff

```
Index: trunk/kokeilu.txt
===================================================================
--- trunk/kokeilu.txt           (revision 4)
+++ trunk/kokeilu.txt           (working copy)
@@ -1,3 +1,6 @@
 1 rivi
 2 rivi
-3 rivi
+2.1 alarivi
+2.2 alarivi
+2.3 alarivi
```

Tätä riviä ei ole työkopiossa → Eli se poistuu.

← Näitä kolmea riviä ei ole revision 4:ssä, joten ne tulevat uuteen versioon.

- diffille voi antaa useita parametrejä, kuten:
  - -r 3:23, vertaile revisioita 3 ja 23
  - -r 23 vertaile työkopiota revision 23 kanssa



Revert (ja Merge)
hups, muutosten kumoaminen

- Jos muutoksia ei ole vielä viety repositoryyn

  $ svn revert tiedosto.txt

  - Kumotaan työkopion muutokset

- Jos taas muutokset on jo viety pitää luoda uusi revisio mergen avulla, jotta historia olisi yhtenäinen kaikille.

  $ svn diff -r HEAD:*haluttu* tiedosto.txt

  - Vertaillaan uusinta (HEAD) ja haluttua, näytettään - ja + merkeillä, mitä poistetaan ja lisätään

  $ svn merge -r HEAD:*haluttu* tiedosto.txt

  - tiedosto.txt:n sisältö on nyt revisiossa *haluttu*
  - Merge toteuttaa siis diffin muutokset.

Log

- Repositoryn tai tiedoston historia: kuka, aika, muuttuneet rivimäärät, kommentti..

  $ svn log hakemisto
  - Repositoryn historia
  - Optio -v antaa enemmän tietoa

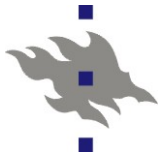
  $ svn log tiedosto.txt
  - Yksittäiset tiedoston historia

Tiedostojen poisto ja uudelleennimeäminen

- Tiedosto-operaatiot
  - Jotta versionhallinta tietää, mitä työkopiossa tapahtuu, on turvallista tehdä tiedostojen poistot, uudelleennimeämiset ym. versionhallinnan kautta:

$ svn mv tiedosto.txt uusinimi.txt

  - Lisää tiedoston uusinimi.txt, poistaa tiedoston tiedosto.txt
  - Säilyttää silti historian, uusinimi.txt:llä on tiedosto.txt:n historia!



Ignore – kaikkea ei haluta pitää versionhallinnassa

- Osa tiedostoista on työtiedostoja tai muuta, jota ei tarvitse jakaa muille.
- “Global ignore” jättää tiedostot aina huomiotta
  - ~/.subversion/config

    [miscellany]

    global-ignores = *.tmp .DS_Store
- Yleensä tieto halutaan kuitenkin säilyttää repositoryssä, repositoryn tulee kertoa, mitkä tiedostot jätetään huomiotta. Tästä lisää kohta.

Properties

- SVN:n voidaan tallettaa propertyjä, eräänlaista metatietoa.
- Propertyt ovat “avain-arvo” -pareja, jossa avain voi olla mitä tahansa.
- Muutamat varatut avaimet, kuten “svn:ignore” vaikuttavat SVN:n toimintaan.
- Propertyt asetetaan aina johonkin hakemistoon, johon ne vaikuttavat. Vaikutus ei ole rekursiivinen.

Ignore

$ svn propset svn:ignore "*.log" logs

- Asettaa kaikki “*.log” -tiedostot logs -hakemistossa “ignoreen”

$ svn propget svn:ignore logs

- Näyttää logs -hakemiston svn:ignore -propertyn sisällön

- propertyn asetus pitää lisätä versionhallintaan, tällöin muutos näkyy hakemistossa, johon property on asetettu.
- Huom! Jos tiedosto on jo lisätty versionhallintaan, niin svn:ignore ei vaikuta niihin.
  - Tiedosto tulee ensin poistaa versionhallinnasta.

Branches & tags

- Ohjelmistoa kehitetään jatkuvasti ja ajoittain kehitys haarautuu
  - Uusi ominaisuus
  - Kokeilut
  - Toinen versio ohjelmasta
- Branch
  - Erillinen kehityshaara
- Tag
  - Repositoryn tietyn hetken tila “versio-2.2”

Tagit

- Tila muuttuu jatkuvasti kehityksen aikana, miten tiedetään mikä tila oli mikäkin?
- “Revisio 273”:n sijasta voidaan käyttää tagia “julkaisu_1.0”

- Tagi luodaan käyttämällä copy -komentoa. SVN luo tällöin linkin hakemiston tai tiedoston revisioon.

  $ svn copy trunk tags/julkaisu_1.0

  - Tämänhetkinen trunk -hakemiston tila on nyt “tägätty” hakemistoon tags/julkaisu_1.0
  - Jos kopio halutaan luoda jostain aikaisemmasta revisiosta, voidaan se määritellä -r # -parametrillä.

Branch

- Joskus ohjelman kehitys on järkevää hajauttaa omaan haaraan
- Branch luodaan käyttämällä copy -komentoa. SVN luo tällöin linkin hakemiston tai tiedoston revisioon.

  $ svn copy trunk branches/uusi_ominaisuus
  - Tämänhetkinen trunk -hakemiston tila on nyt “branchattu” hakemistoon branches/uusi_ominaisuus
  - Jos kopio halutaan luoda aikaisemmasta revisiosta, voidaan se määritellä -r # -parametrillä.
  - Huom: “branches” hakemiston tulee olla seurannassa

Havaitaan

- SVN:ssä on vain käsitteet “tag” ja “branch”. Yhtä oikea käsite olisi “heppa”. Tämäkin voitaisiin luoda käyttäen copy -komentoa.
- Molemmat ovat vain hakemistoja.

- Kuitenkin tag ja branch ovat hyviä ratkaisuja oikeisiin ongelmiin.

Switch

- Työkopion sijaintia repositoryssä voi vaihtaa. Tämä on kätevää esim. branchien kanssa työskennellessä.

  $ svn switch <REPO-URL>
  - Vaihtaa työkopion käyttämän “hakemiston repositoryssä”

  $ svn switch svn+ssh://..../branches/uusi_branch
  - Koko työkopio on nyt vaihdettu

  $ svn switch svn+ssh://.../trunk
  - Takaisin trunk -hakemistossa

Merge – muutosten yhdistäminen

- Merge yhdistää kaksi hakemistoa (tai tiedostoa)
  - Ajaa diffin ja tuottaa yhdistetyn tilan työkopioon.

  $ svn diff -r

  $ svn merge <REPO_URL> <kohde>

- Branchin yhdistäminen trunk -hakemistoon:

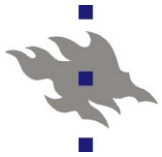
  $ cd trunk

  $ svn merge -r ##:HEAD ../branches/haara

  - Jossa ## on revisio, jossa on ensimmäinen yhdistettävä muutostila (esim branchin luonti)
  - Luontirevision saa selville:

    $ svn log –stop-on-copy branches/haara

    - Eli pysäytetään logi kun “copy” suoritetaan



Branch merge - best practices

- Luo branch
  - Commitoi branch **heti**
    - Ota revisionumero ylös (tai katso svn log -kommennolla)
- Tee muutoksia branchiin
  - Commit, kun valmista
- Vaihda trunk -hakemistoon
  - Aja merge -r <revisio-jossa-branch-luotiin>:HEAD
  - Tällöin mergeen tulee kaikki branchiin tehdyt muutokset.

$ svn diff -r **##**:HEAD --new=branches/*haara* –old=*trunk*

  - Eroavaisuudet *haaran* ja *trunk* välillä

Muuta

- Versionhallinnan versio-ongelmat

```
$ svn --version
```

  - Joskus asennettu versio SVN:stä saattaa olla uudempi tai vanhempi eri koneilla (erityisesti yliopiston eri palvelimilla)
  - Jos näin käy, on helpointa ottaa uusi työkopio uudella versiolla..
  - Laitoksella on SVN:stä asennettuna 1.4.x ja 1.5.x riippuen koneesta :(

Hooks

- SVN osaa reagoida tapahtumiin, nämä on listattu hakemistossa <repository>/hooks
- Yksi tapahtuma on post-commit, joka ajetaan jokaisen commitin jälkeen repositoryssä.
  - Tätä voi hyödyntää esim:
    - Yksikkötestien ajamisessa automaattisesti
    - Päivittämään www-sivut repositorystä
    - Sähköposti-ilmoitukset, RSS-syöte..