Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Digitaalinen media
Toni Kettunen

Opinnäytetyö

Versionhallintajärjestelmä eEDUn Moodle-tarpeisiin

Työn ohjaaja Paula Hietala FL, Lehtori, tieto- ja viestintäteknologia
Työn tilaaja TAMK, Opetusteknologiakeskus eEDU, Jussi Hannunen
Tampere 12/2009

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma, Digitaalinen media

Tekijä	Toni Kettunen
Työn nimi	Versionhallintajärjestelmä eEDUn Moodle-tarpeisiin
Sivumäärä	42
Valmistumisaika	Joulukuu 2009
Työn ohjaaja	Paula Hietala
Työn tilaaja	TAMK, Opetusteknologiakeskus eEDU

TIIVISTELMÄ

Ohjelmistokehitys muuttuu jatkuvasti monimutkaisemmaksi projektien ja niihin osallistuvien ihmisten määrän kasvaessa. Ohjelmistojen koko ja tiedostojen määrä saattaa olla niin suuri, ettei niiden hallitseminen ole mahdollista ilman sopivaa työkalua. Myös muilla aloilla, ohjelmistoalan lisäksi, saattaa olla tarpeen ottaa käyttöön versionhallintajärjestelmä sähköisessä muodossa oleville tiedostoille. Versionhallinta mahdollistaa useiden henkilöiden yhteistyön tiedostojen parissa, tiedostojen palauttamisen vanhempaan versioon, eri versioiden yhdistämisen ym.

Opinnäytetyö tehtiin TAMKin opetusteknologiakeskus eEDUlle. Tarkoituksena oli selvittää, olisiko versionhallintajärjestelmä sopiva ratkaisu eEDUn ylläpitämän Moodle-oppimisympäristön hallintaan, koska etenkin päivitysten aiheuttama työmäärä oli tullut todella suureksi. TAMKin Moodlen lisäksi eEDUn on tulevaisuudessa tarkoitus ylläpitää myös muille tahoille räätälöityjä oppimisympäristöjä.

Aluksi tarkoituksena oli tutustua eri versionhallintajärjestelmiin ja valita näistä eEDUlle sopivin. Tavoitteena oli myös suunnitella miten versionhallinta toteutettaisiin eEDUn tapauksessa sekä luoda selkeät toimintatavat sen käyttöön. Tämän työn ei kuitenkaan ole tarkoitus olla minkään versionhallintaohjelman käyttöopas.

Aiheeseen ja eri ohjelmiin tutustumisen jälkeen, versionhallinta päätettiin käytännössä toteuttaa Subversion-ohjelmalla. Ohjelmaa testattiin mahdollisimman samanlaisessa ympäristössä ja samalla tavalla kuin sitä todellisuudessakin tultaisiin käyttämään. Myös testatut ominaisuudet valittiin käytäntöön perustuen.

Versionhallinta todettiin tutkimusten ja käytännön testien perusteella eEDUlle toimivaksi ja todella tärkeäksi tulevaisuutta ajatellen. Tämän työn tuloksia hyväksikäyttäen eEDUn on mahdollista huomattavasti vähentää Moodlen ylläpidosta ja päivittämisestä aiheutuvaa työmäärää. Versionhallintajärjestelmä tekee myös muille tahoille räätälöitävien Moodle-julkaisujen hallitsemisesta paljon selkeämpää ja helpompaa.

Avainsanat versionhallintajärjestelmä, Moodle, Subversion, ohjelmistokehitysprojekti, koodinhallinta

TAMK University of Applied Sciences
Business Information Systems, Digital media

Writer	Toni Kettunen
Thesis	Version Control System for eEDU's Moodle Needs
Pages	42
Graduation time	December 2009
Thesis Supervisor	Paula Hietala
Co-operating Company	TAMK, Education Technology Centre eEDU

ABSTRACT

Software development is constantly getting more complex as the size of the projects and the number of participating people grows. The program sizes may be so big that it is impossible to maintain them without a proper tool. Also in other fields, besides software development, it may be necessary to control the electronic files with a version control system. Version control enables co-operation of many people with the same files, returning the files to a previous state, merging of different versions etc.

This thesis was made for the Education Technology Centre eEDU of TAMK. The goal was to find out if a version control system would be a suitable solution for eEDU for maintaining the virtual learning environment Moodle, because the workload of especially updating Moodle had become so big. In addition to TAMK's Moodle, eEDU is supposed to maintain customized learning environments for other parties in the future.

The first objective was to get to know various version control systems and to choose the most suitable for eEDU. Another goal was to plan how version control would be implemented for eEDU and to create simple methods for using it. However, this thesis is not supposed to be a user's manual for any version control program.

After getting to know the whole subject and different programs, version control was decided to be implemented with a program called Subversion. The program was tested in an environment, and with methods, as similar to practice as possible. Also the tested features were selected to represent common practices.

Based on research and hands-on experience, version control was found to be extremely important for eEDU also considering the future. Taking advantage of the results of this thesis, eEDU can considerably cut down the workload of maintaining and updating Moodle. Version control system also makes controlling the customized Moodle-releases for other parties much clearer and easier.

Keywords version control system, Moodle, Subversion, software development project, code control

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Digitaalinen media

Sisällysluettelo

1 Johdanto

Ohjelmistokehityksessä siirrytään jatkuvasti laajempiin ja monimutkaisempiin projektikokonaisuuksiin, joten myös projektiin osallistuvien henkilöiden määrä kasvaa. Koodin hallinta tulee siis monimutkaisemmaksi, koska monet ihmiset tekevät siihen muutoksia. Kaikilla täytyisi kuitenkin olla ajan tasalla oleva versio koodista ja päällekkäistä työtä täytyisi välttää. Tätä ongelmaa helpottamaan on kehitetty aivan erityisiä ohjelmia, joita sanotaan versionhallintajärjestelmiksi.

Versionhallintajärjestelmiä on mahdollista käyttää niiden monipuolisten ominaisuuksien ansiosta myös hieman eri tarkoituksiin, kuin ratkaisemaan edellä mainittua yhteistyön ongelmaa. Tässä työssä ei ole tarkoitus rakentaa järjestelmää, joka hallitsee useiden henkilöiden tekemiä muutoksia tietoverkon yli, vaan ennemminkin yhteen pisteeseen keskitetty ratkaisu.

Ohjelmisto, jota tässä työssä on tarkoitus versionhallintajärjestelmällä hallita, on Tampereen ammattikorkeakoulun (TAMK) oppimisympäristö Moodle. Moodle koostuu tuhansista tekstitiedostoista, joiden omaan tarpeeseen räätälöinti ja uuteen versioon päivittäminen on tullut toimeksiantajalla niin työlääksi, että tarvitaan versionhallintajärjestelmä. Toimeksiantajan on myös tarkoitus tulevaisuudessa ylläpitää TAMKin lisäksi monen muun tahon käyttöön yksilöllisesti räätälöityjä Moodle-oppimisympäristöjä. Versionhallinta on myös tähän tarkoitukseen todella hyvä työkalu. Luvussa kolme on kerrottu enemmän oppimisympäristöstä ja sen ominaispiirteistä Tampereen ammattikorkeakoulussa.

Tämän työn ensimmäisenä tavoitteena oli tutkia käytetyimpiä versionhallintajärjestelmiä ja valita näistä sopivin toimeksiantajan tarpeisiin. Luvussa neljä on kerrottu versionhallinnasta yleisesti sekä tarkemmin kolmesta suositusta ohjelmasta. Eri ohjelmien ominaispiirteisiin, eroihin ja toiminnallisuuksiin perehtymisen jälkeen on valittu ohjelma, jolla versionhallinta kannattaisi oppimisympäristöön toteuttaa.

Toisena tavoitteena oli siis suunnitella versionhallinnan toteutus toimeksiantajan oppimisympäristöön. Luvussa viisi on kuvattu esimerkkien avulla miten järjestelmä käytännössä toteutettaisiin valitulla ohjelmalla. Luvussa kuusi on tehty johtopäätöksiä ja arvioitu työn onnistumista sekä järjestelmän kehittämismahdollisuuksia. Liitteeseen 1 on listattu tässä työssä esiintyvät versionhallintaan ja oppimisympäristöihin liittyvät termit ja niiden selitykset.

2 Lähtökohdat

Opinnäytetyön toimeksiantaja on TAMKin opetusteknologiakeskus eEDU. Opinnäytetyön aihepiiri liittyy ohjelmistoon tehtyjen muutosten ja oman koodin hallintaan versionhallintajärjestelmällä. Vaikka versionhallintajärjestelmillä yleensä ymmärretään ohjelmistojen lähdekoodin, tai minkä tahansa suuren tiedostojoukon, muokkaamista hajautetusti tietoverkossa niin, että kaikkien muokkaajien työkopiot pysyvät ajan tasalla, on versionhallintajärjestelmiä mahdollista soveltaa myös muihin tarkoituksiin. eEDUlla koodia hallitsee vain yksi henkilö, mutta koodista on tarkoitus tuottaa monta erilaista räätälöityä versiota, joten tässä työssä selvitetään versionhallintajärjestelmän soveltuvuus tällaiseen tilanteeseen.

Työn tekeminen alkaa tutustumalla taustamateriaaliin ja aiheeseen yleisesti. Subversionista ja versionhallintajärjestelmistä yleisesti kertova, noin 400 sivuinen kirja (Version Control with Subversion For Subversion 1.6 2008) on pääasiallinen tiedon lähde alussa. Myös muihin suosituimpiin versionhallintajärjestelmiin on tarkoitus tutustua lähinnä netistä löytyvän materiaalin pohjalta. Myös koulun kirjastosta löytyviin muihin versionhallintaa käsitteleviin kirjoihin on tarkoitus tutustua. Materiaalin läpikäynnin jälkeen (osaksi samaan aikaan) on tarkoitus tehdä suunnitelma siitä, miten järjestelmä käytännössä toteutetaan. Lopuksi opittua sovelletaan käytännössä ja kokeillaan, toimiiko järjestelmä toivotulla tavalla.

2.1 Toimeksiantaja

Opetusteknologiakeskus eEDU perustettiin 2001 hallitsemaan ja kehittämään verkkopohjaista opetusta ja oppimista TAMKissa. eEDU avustaa ja tukee opettajia oppimisympäristön käytössä ja vastaa verkko-opetuksen kouluttamisesta TAMKin henkilökunnalle. Pääasiallinen oppimisympäristö, jonka parissa eEDU toimii, on Moodle. eEDU on osa Tampereen ammatillista opettajakorkeakoulua (TAOKK). (eEDU -esitys 2008, Opetusteknologiakeskus eEDU 2007.)

Muita eEDUn toimia ovat muun muassa

- verkkoympäristöjen ja -materiaalien käytettävyyssuunnittelu ja testaus
- verkkopohjaisen oppimisprosessin suunnittelu
- videoneuvottelujen kehittäminen ja käytön opettaminen
- verkkomateriaalien visuaalinen suunnittelu
- verkkopohjaisen oppimissisällön tarjoaminen
- konsultointi tieto- ja viestintätekniikan opetuskäytössä myös muille oppilaitoksille sekä yrityksille
- aktiivinen osallistuminen kansallisiin ja kansainvälisiin tapahtumiin ja projekteihin
- muiden asiantuntija- ja projektitehtävien käynnistäminen ja koordinointi.

Opetusteknologiakeskus vastaa myös tutkimus- ja kehitystyöstä omalla alallaan. Nykyisiä tutkimusalueita ovat uudet teknologiat ja opetustavat verkko-oppimisessa, kuten esimerkiksi ePortfoliot, sosiaalinen verkkoympäristö ja videokonferenssit. (eEDU -esitys 2008, Opetusteknologiakeskus eEDU 2007.)

2.2 Tarve

Opetusteknologiakeskuksessa on ajan oloon syntynyt selvä tarve versionhallintajärjestelmän käytölle. Oppimisympäristö Moodleen lisätyn oman koodin määrä on kasvanut niin suureksi, että sen hallitseminen käsin on tullut päivitysten yhteydessä todella työlääksi. Oman koodin määrä tulee myös jatkuvasti kasvamaan entisestään. Tästä syystä eEDUssa on päätetty etsiä sopiva työkalu päivitysten hoitamiseen.

eEDUssa ei kuitenkaan ole ehtinyt syntyä niin sanottua koodikaaosta omien koodin räätälöintien suhteen. Tämä johtuu suurimmaksi osaksi siitä, että oman koodin määrää on koodin hallinnan puuttumisen takia jouduttu jopa karsimaan, eikä kaikkia hyödyllisiä lisäominaisuuksia ole voitu ottaa koodiin mukaan. Suurin syy versionhallinnan tarpeelle onkin siis kehityksen esteen poistaminen.

Toinen selkeä syy versionhallintajärjestelmän käyttöönottoon on Moodlen eri versioiden keskitetty hallinta. Tulevaisuudessa eEDUn on hallittava Moodle-koodin useita eri versioita, koska eEDUn on tarkoitus vastata TAMKin lisäksi myös monen muun tahon Moodle-koodin räätälöinnistä ja ajan tasalla pitämisestä.

Versionhallintajärjestelmä toimii myös aikakoneen tavoin, mikä mahdollistaa tiedoston tai kokonaisen sovelluksen palauttamisen johonkin tiettyyn vanhempaan versioon. Ongelmatilanteissa tämäkin on todella tärkeä ominaisuus. eEDUn olisi mahdollista palauttaa koko oppimisympäristö aiempaan tilanteeseen, jos esimerkiksi jokin tarvittava ominaisuus ei toimikaan uudemmassa versiossa. Myös monien ominaisuuksien ja toiminnallisuuksien tutkimisessa vanhempiin versioihin käsiksi pääsemisestä on monesti hyötyä.

2.3 Tavoite

Opinnäytetyön tarkoituksena on selvittää, onko versionhallintajärjestelmä sopiva eEDUn tarpeisiin. Toimeksiantaja on ehdottanut versionhallintajärjestelmän rakentamista Subversionin pohjalta, mutta toive on myös ollut, että selvitetään onko se paras mahdollinen ratkaisu. Myös muihin yleisimpiin versionhallintajärjestelmiin siis tutustutaan ja arvioidaan niiden soveltuvuutta eEDUlle.

Tarkoituksena on myös suunnitella Moodle-oppimisympäristön koodinhallinta tarkoitukseen parhaiten sopivalla ohjelmalla. Tavoitteena on vakiinnuttaa käytäntö Moodle-koodin hallintaan ja päivityksiin niin, että sen aiheuttama työmäärä on mahdollisimman pieni. Opinnäytetyön tärkeimpänä tavoitteena on minimoida oppimisympäristö Moodlen päivityksistä aiheutuvat virheet ja downtime.

3 Moodle

Moodle on virtuaalinen oppimisympäristö, jolla kouluttajat voivat luoda verkkoon monipuolisia interaktiivisia sivustoja opiskeluun. Oppimisympäristössä on työvälineitä muun muassa sisällöntuottamiseen, vuorovaikutukseen ja materiaalin jakamiseen. Moodlea onkin sanottu maailman monipuolisimmaksi oppimisalustaksi. Moodlea kutsutaan myös kurssin hallintajärjestelmäksi (Course Management System - CMS) tai oppimisen hallintajärjestelmäksi (Learning Management System - LMS). Moodle on ilmainen, GNU-lisenssillä julkaistava, maailmanlaajuinen kehitysprojekti. (Moodle community 2009.)

Alun perin Moodlen kehitti Martin Dougiamas Australian Curtin yliopistossa. Moodle-nimi on akronyymi sanoista “Modular Object-Oriented Dynamic Learning Environment”. Alussa Moodlen ensimmäinen kirjain tuli kuitenkin alkuperäisen kehittäjän etunimestä “Martin’s”. (Moodle 2009.)

Moodle on pääasiassa PHP- ja SQL-kieliin perustuva open source -ohjelmisto, joka toimii Windows- ja Mac-käyttöjärjestelmien lisäksi myös monissa Linux-versioissa sekä missä tahansa muussa järjestelmässä, joka tukee PHPta ja tietokantoja. Modulaarisen rakenteen takia käyttäjät voivat kehittää Moodleen itse lisää toiminnallisuuksia. Moodlen kehitystä ovat avustaneet open source -ohjelmoijat, mistä syystä sen kehitys ja virheiden korjaaminen on ollut todella nopeaa. (Moodle community 2009, Moodle 2009.)

Syyskuuhun 2009 mennessä rekisteröityjä Moodle-sivustoja oli maailmanlaajuisesti käytössä 44 256 kappaletta 206 eri maassa. Näillä sivustoilla oli yhteensä 29 360 076 käyttäjää ja 2 883 604 kurssia. Uusia Moodle-sivustojen rekisteröintejä tuli syksyllä 2009 melkein 2 000 kappaletta kuukaudessa ja Moodle-koodia ladattiin noin 85 000 kertaa kuussa. Eniten Moodle oli käytössä Pohjois- ja Etelä-Amerikassa, Keski-Euroopassa ja Australiassa. Käytetyin Moodle-versio oli 1.9.x, joka edusti noin puolta kaikista rekisteröinneistä. (Moodle statistics 2009.)

Moodle 1.1 valittiin TAMKin oppimisympäristöksi vuonna 2003 (Opetusteknologiakeskus eEDU -esitys 2008). Lokakuussa 2009 käytössä oli versio Moodle 1.9.3+ (Build: 20081217). Tuolloin TAMKissa rekisteröityneitä käyttäjiä oli yli 14 300 ja kursseja yli 2 900 kappaletta. (moodle - Learning… 2009.)

Tavoitteena on julkaista Moodlen 2.0 betaversio vuoden 2009 loppuun mennessä. Tämän jälkeen on useiden kuukausien testausjakso, jonka lopuksi on tarkoitus julkaista vakaa Moodle 2.0 versio. Uusi 2.0 versio sisältää valtavan määrän parannuksia ja uusia ominaisuuksia muun muassa tiedostojen käsittelyyn, kurssien organisointiin, aktiviteetti-moduuleihin, hallinnollisiin ominaisuuksiin jne. Myös käyttäjien järjestelmän vaatimukset nousevat version 2.0 myötä mm. viimeisimpien PHP-ominaisuuksien hyödyntämisen takia. (Roadmap 2009.)

4 Versionhallinta

Versionhallinnalla tarkoitetaan dokumenttien, ohjelmien tai muun tietokoneelle tallennetun tiedon muutosten hallintaa. Versionhallintajärjestelmien kehitys on alun perin lähtenyt tarpeesta palata tiedoston tai suunnitelman aiempaan versioon, kun kehitys on johtanut umpikujaan. Myös ohjelmistoprojektien räjähdysmäinen kasvu 1960-luvulla vauhditti versionhallinnan kehitystä. Yleisimmin versionhallintaa käytetään ohjelmistokehityksessä, jossa ryhmä ihmisiä työskentelee samojen tiedostojen parissa. Ohjelmistonkehitysprojektissa voidaan versionhallintajärjestelmällä hallita dokumentaatio- ja konfiguraatiotiedostoja sekä itse lähdekoodia. (Revision control 2009, Ohjelmiston versiohallinta 2009.)

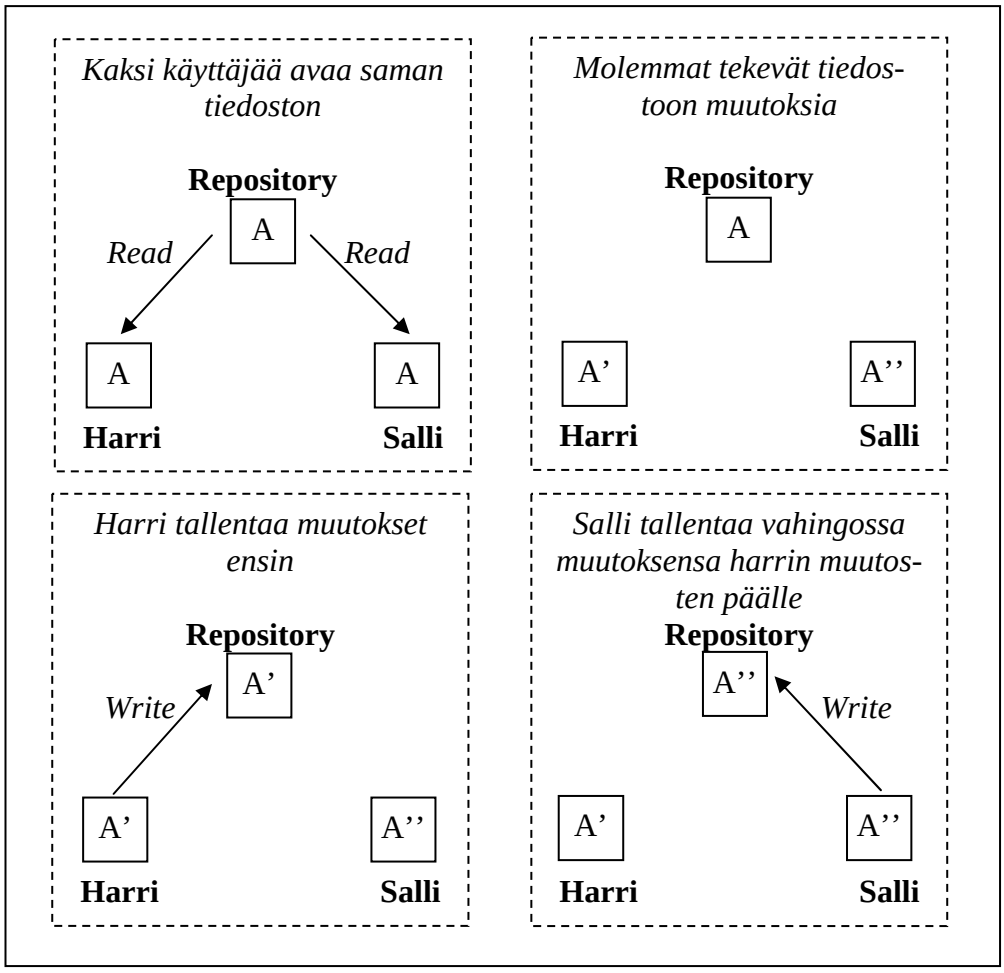
Yleensä versionhallintajärjestelmät ovat erillisiä ohjelmistoja, mutta versionhallinta on myös upotettu monentyyppisiin ohjelmiin, kuten tekstinkäsittelyyn, taulukkolaskentaan sekä sisällönhallintajärjestelmiin. Versionhallinta on myös todella tärkeä osa Wiki-ohjelmistoja. (Revision control 2009.)

Versionhallintajärjestelmällä useat henkilöt pystyvät siis samaan aikaan kehittämään samaa ohjelmistoa, tai mitä vain tiedostojen joukkoa, tallentamalla omia muutoksiaan verkossa olevaan säilytyspaikkaan, repositoryyn, jossa tiedostot sijaitsevat. Kuviossa 1 on suositeltava repositoryn rakenne, jota tässäkin työssä on käytetty. Projektilla on siis kolme pääkansiota (branches, tags ja trunk), joista trunk on projektin kehityksen päälinja, jossa muokattavat tiedostot yleensä sijaitsevat projektin alkaessa.

```
my_project_repository
      -branches
      -tags
      -trunk
            -file1.txt
            -file2.php
            -file3.html
            -...
```

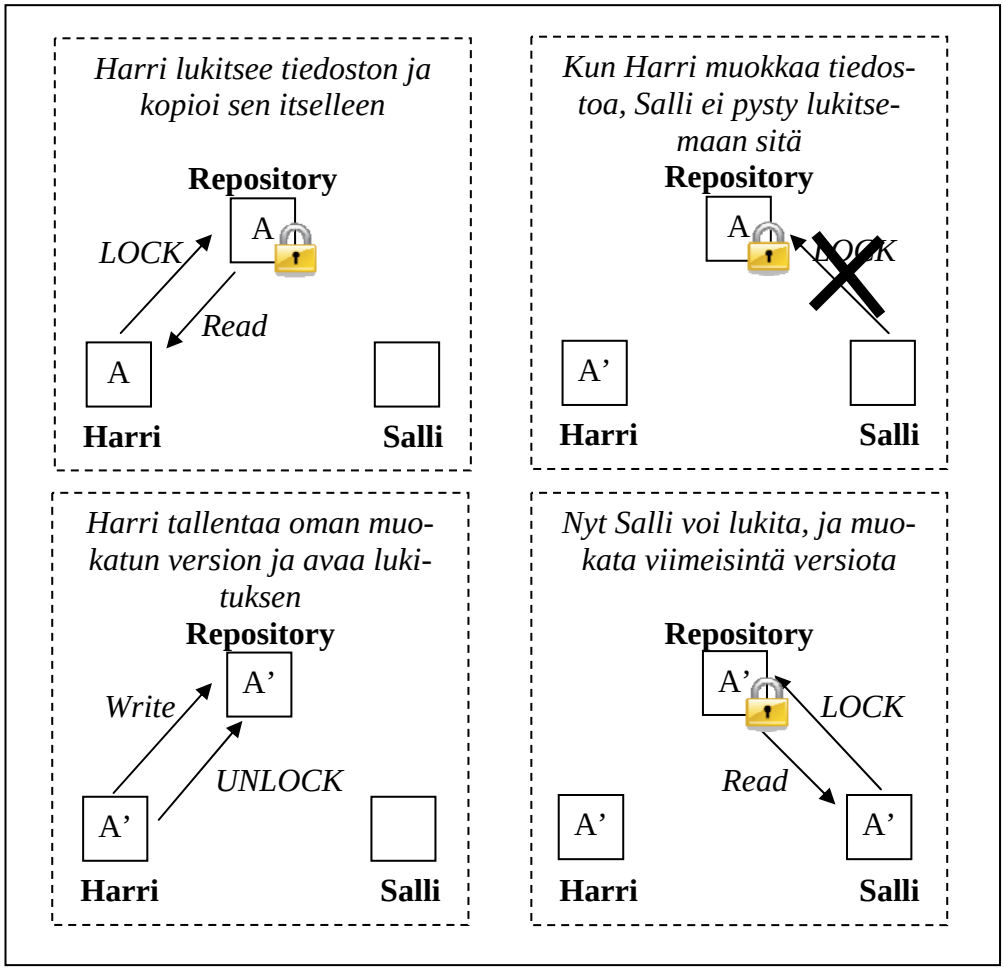
Kuvio 1. Repositoryn rakenne

Repository ei kuitenkaan ole tavallinen tiedostopalvelin, joka ainoastaan majoittaa hakemistot ja tiedostot, joihin käyttäjät tekevät muutoksia. Repository myös muistaa jokaisen muutoksen sen sisältämiin tiedostoihin tai hakemistorakenteisiin. Tavallisen palvelimen ongelmaksi tulee omien muutosten tallettaminen toisen tekemien muutosten päälle. Kuviossa 2 on kuvattu tätä tilannetta, jossa toisen henkilön työ menee täysin hukkaan. (Collins-Sussman, Fitzpatrick, & Pilato 2008, 1-2.)



Kuvio 2. Tavallisen tiedostopalvelimen ongelma (mukaillen Collins-Sussman ym. 2008, 2)

Edellä mainittuun ongelmaan yksi ratkaisu on lukitse-muokkaa-avaa -malli, jota monet versionhallintajärjestelmät käyttävät. Tässä mallissa käyttäjän täytyy lukita tiedosto ennen kuin voi tehdä siihen muutoksia. Kun tiedosto on lukittu, muut käyttäjät eivät voi lukita sitä, eivätkä siis tehdä siihen muutoksiakaan. Kun ensimmäinen käyttäjä on tallentanut muutoksensa tiedostoon ja avaa lukituksen, voi toinen käyttäjä lukita sen ja alkaa muokata sitä. Kuviossa 3 on kuvattu tämän mallin toimintaa. (Collins-Sussman ym. 2008, 2.)

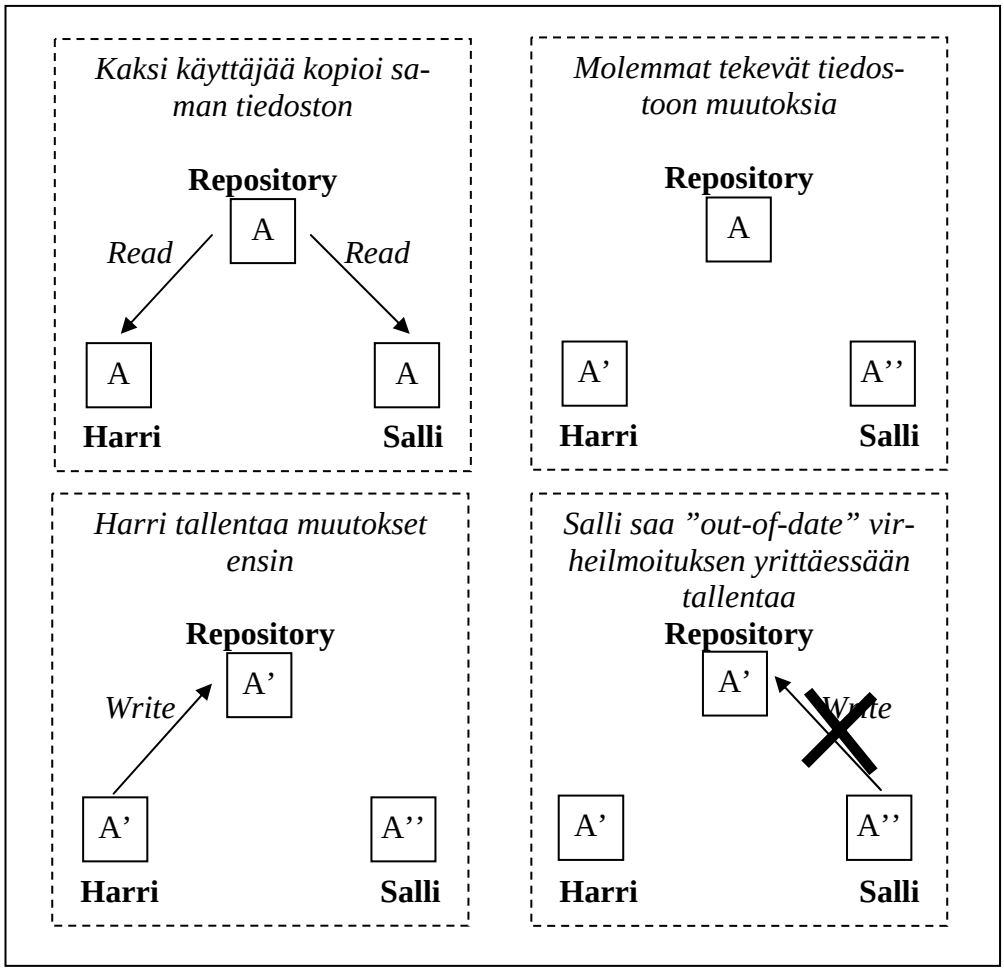


Kuvio 3. Lukitse-muokkaa-avaa -malli (mukaillen Collins-Sussman ym. 2008, 3)

Lukitse-muokkaa-avaa -mallissa on kuitenkin useita huonoja puolia. Ensinnäkin lukituksen voi unohtaa päälle, jolloin muut käyttäjät eivät pääse tekemään muutoksia tiedostoon. Käyttäjät voisivat myös aivan hyvin editoida samaan aikaan tiedoston eri osia ilman että muutosten yhdistämisestä tulisi vaikeaa. Tässä mallissa se ei kuitenkaan ole mahdollista. On myös mahdollista, että eri käyttäjät lukitsevat ja muokkaavat samaan aikaan eri tiedostoja, jotka kuitenkin ovat toisistaan riippuvaisia. Muokkausten jälkeen tiedostot eivät enää toimi yhteen, eikä lukitse-muokkaa-avaa -malli voinut mitenkään estää tätä. (Collins-Sussman ym. 2008, 3.)

Toinen (ja parempi) ratkaisu kuviossa 2 olevaan ongelmaan on niin sanottu kopioi-muokkaa-yhdistä -malli, jota käyttävät muun muassa Subversion, CVS sekä monet muut versionhallintaohjelmistot. Tässä mallissa jokainen käyttäjä ottaa repositorystä omalle koneelleen henkilökohtaisen työkopion (working copy), johon tekee sitten omat muutoksensa. Lopuksi käyttäjien henkilökohtaiset työkopiot yhdistetään (merge) uudeksi versioksi repositoryyn. (Collins-Sussman ym. 2008, 4.)

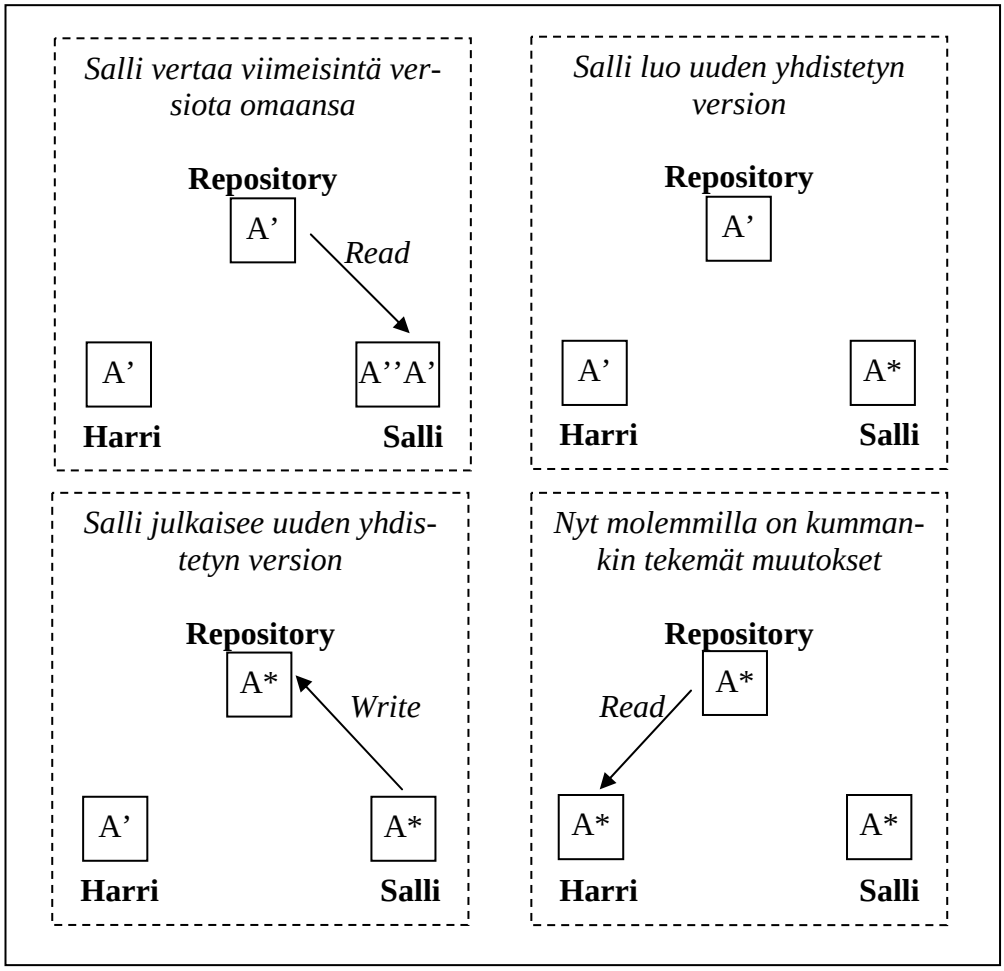
Versionhallintajärjestelmillä on siis kyky yhdistää esimerkiksi toisistaan hieman poikkeavat tekstitiedostot yhdeksi tiedostoksi, jossa mukana ovat kummankin alkuperäisen tiedoston yksilölliset tekstirivit. Versionhallintajärjestelmä auttaa tässä yhdistämisessä, mutta lopulta käyttäjä on vastuussa yhdistämisen onnistumisesta. Kuviossa 4a ja 4b on esimerkki kopioi-muokkaa-yhdistä -mallin toiminnasta. (Collins-Sussman ym. 2008, 4.)



Kuvio 4a. Kopioi-muokkaa-yhdistä -malli (mukaillen Collins-Sussman ym. 2008, 4)

Kuviossa 4a Harri ja Salli työskentelevät saman projektin parissa ja molemmat ottavat työkopion repositoryssä olevasta tiedostosta A. Molemmat tekevät siihen omia muutoksiaan ja Harri ehtii tallentamaan omat muutoksensa ensin takaisin repositoryyn. Kun Salli yrittää tallentaa hän saa virheilmoituksen ”out-of-date”, eli repositoryssä oleva tiedosto on muuttunut sen jälkeen, kun hän viimeksi otti siitä työkopion. (Collins-Sussman ym. 2008, 4.)

Sama tilanne jatkuu kuviossa 4b, jossa Salli ensin vertaa viimeisintä tiedoston versiota omaansa, ja yhdistää (merge) uudet repositoryssä olevat muutokset omaan tiedostoonsa, jos muutokset eivät ole ristiridassa (conflict). Ristiriidalla tarkoitetaan sitä, että molempien versioiden muutokset olisivat kohdistuneet tiedoston samaan kohtaan tai kohtiin. Lopuksi Salli tallentaa (commit) uuden yhdistetyn tiedoston repositoryyn, josta Harri saa sen (update). Lopputuloksena molemmilla käyttäjillä on molempien tekemät muutokset omassa työkopiossaan. (Collins-Sussman ym. 2008, 4-5.)

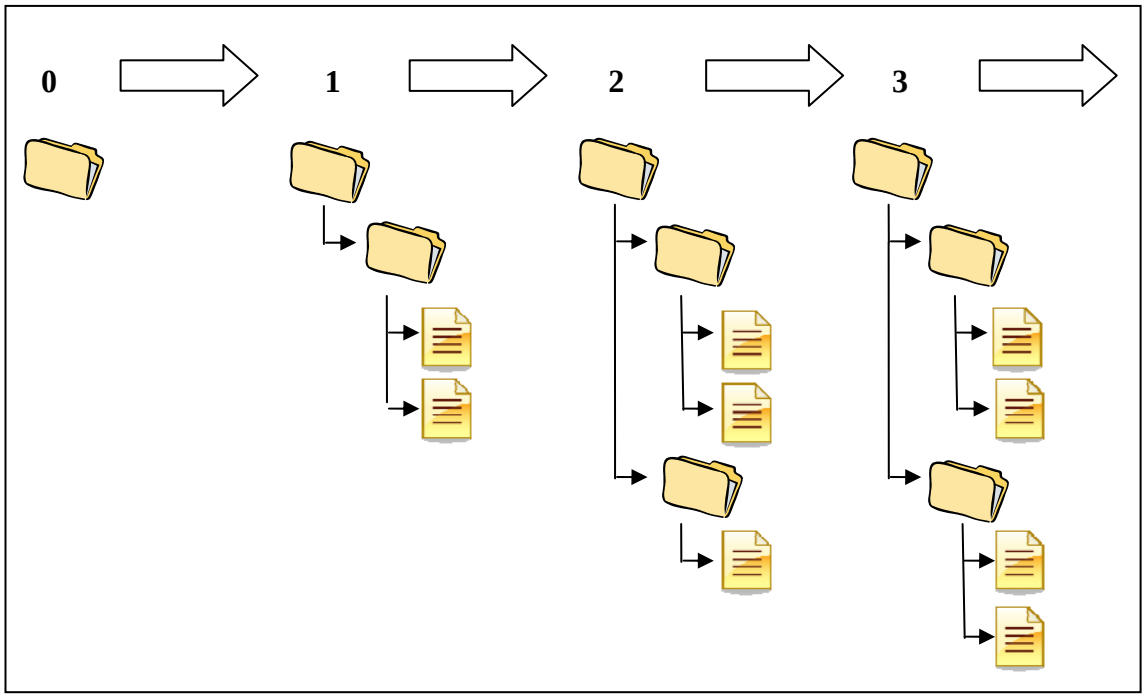


Kuvio 4b. Kopioi-muokkaa-yhdistä -malli (mukaillen Collins-Sussman ym. 2008, 5)

Jos tiedostojen muutokset ovatkin ristiriidassa, täytyy käyttäjän päättää jokaisen ristiriidan kohdalla, kumpaa versiota käyttää. Käyttäjä voi manuaalisesti verrata kumpaankin versioon tehtyjä muutoksia ja tehdä sen pohjalta päätöksensä. Ohjelmisto ei voi automaattisesti ratkaista tiedostoissa olevia ristiriitoja, vaan ne on käyttäjän itse aina ratkaistava. Edellä olevassa esimerkissä on ristiriidan sattuessa erityisen tärkeää, että Harri ja Salli keskustelevat ja tekevät yhteistyötä parhaan ratkaisun löytämiseksi. (Collins-Sussman ym. 2008, 5.)

Versionhallintajärjestelmiä voisi ajatella myös aikakoneina, joilla tiedostoja voi palauttaa siihen muotoon, missä ne ovat olleet tiettynä päivänä tai tuntina menneisyydessä, tai tutkia tiedostossa tapahtuneita muutoksia. Monesti tulee tarvetta palauttaa tiedosto entiseen muotoonsa esimerkiksi virheitä etsittäessä tai tiettyä ominaisuutta tutkittaessa. Aiemmassa versiossa saattaa olla myös suorituskyvyn tai jonkin ominaisuuden kannalta sellaisia piirteitä, joita viimeisessä versiossa ei enää ole. Myös eEDUn kannalta tämä ominaisuus on todella houkutteleva. Jos jostain syystä eteen tulee tilanne, missä olisi tarpeen palauttaa käyttöön Moodlen vanhempi versio, onnistuisi tämä hyvin kätevästi versionhallintajärjestelmän avulla. (Ohjelmiston versiohallinta 2009.)

Käytännössä kaikki repositoryyn tehdyt muutokset tallentuvat revisio-numeroiksi. Revisio-numerolla tarkoitetaan repositoryn tiedostojen ja hakemistojen tilaa tiettyjen muutosten tallennuksen (commit) jälkeen. Monissa versionhallintajärjestelmissä revisionumerot seuraavat ainoastaan yksittäisiä tiedostoja, kun taas toisissa, kuten Subversion, tallentuu myös tieto koko hakemistopuun rakenteesta. Kuviossa 5 on kuvattu repositoryn rakennetta eri revisioiden kohdalla. Juuri luodun, tyhjän repositoryn revisionumero on 0. Jokaista revisiota voidaan ajatella valokuvana sen hetkisestä repositoryn rakenteesta. (Collins-Sussman ym. 2008, 10.)



Kuvio 5. Revisiot ja repository (mukaillen Collins-Sussman ym. 2008, 10)

Jokaiseen revisioon tallentuu myös aikaleima sekä tieto muutoksen tekijästä. Eri revisioita voidaan verrata keskenään tai tiedostot voidaan palauttaa siihen tilaan, missä ne tietyssä revisiossa ovat olleet. Tietyn tyyppisten tiedostojen eri revisioita voidaan myös yhdistää (merge) keskenään. (Revision control 2009.)

Versionhallintajärjestelmällä voi nimensä mukaisesti helposti hallita myös montaa eri versiota samasta ohjelmasta. Tämäkin ominaisuus on erityisen tärkeä tässä työssä, koska se tulee myös tulevaisuudessa olemaan keskeisessä osassa työn tilaajan versionhallintajärjestelmän tarpeessa. Käytännössä projektin eri kehityslinjat tallennetaan repositoryn branches-kansioon kuvion 6 mukaisesti. Branches-kansioon voidaan myös tallentaa ohjelmasta uusi haara, jossa kehitetään ohjelmaan jotain isompaa lisäosaa tai ominaisuutta. Myöhemmin nämä muutokset voidaan yhdistää (merge) takaisin kehityksen päälinjaan eli trunk-kansioon.

```
my_project_repository
      -branches
            -my_project_branch1
                  -file1.txt
                  -file2.php
                  -...
            -my_project_branch2
                  -file1.txt
                  -file2.php
                  -...
      -tags
      -trunk
            -file1.txt
            -file2.php
            -...
```

Kuvio 6. Repositoryn rakenne

4.1 Versionhallintajärjestelmät

Kun versionhallinnan maailmaan syvennyttiin paremmin, esiin nousivat jatkuvasti samat kolme ohjelmaa: CVS, Git ja Subversion. Seuraavassa on kerrottu lyhyesti perusasioita näistä yleisimmin käytetyistä versionhallintajärjestelmistä. Kaikki kolme ohjelmistoa toimivat Mac- ja Windows-ympäristöissä ja ovat ilmaisia.

CVS

CVS eli Concurrent Versions System (tai Concurrent Versioning System) on Dick Grunen jo 80-luvulla kehittämä versionhallintajärjestelmä, jonka nimi oli aluksi cmt. CVS kehittyi aikaisemmasta järjestelmästä nimeltä Revision Control System (RCS), joka hallitsi yksittäisiä tiedostoja, mutta ei kokonaisia projekteja. Grune kehitti ohjelman yhteistyön apuvälineeksi, koska hänellä oli ongelmia sovittaa aikataulut yhteen oppilaidensa kanssa. (Concurrent Versions System 2009.)

Yksi CVSn innovaatio oli ”branching”, eli projektin haarautuminen eri kehityslinjoiksi versionhallintajärjestelmässä. Kaikkien muiden järjestelmien haarautumistekniikat johtavat juurensa jo 1990 dokumentoidusta CVSn toteutuksesta. Nykyään CVSn kehitystä jatkaa joukko vapaaehtoisia. CVSn Windows-version kehitys on haarautunut omaksi projektikseen, jonka nimi on CVSNT. (Concurrent Versions System 2009.)

CVS on hyvinkin vajavainen nykyaikaisiin versionhallinnan vaatimuksiin nähden. CVSn kehittäjät kuitenkin väittävät, että monet ohjelman puutteista eivät ole puutteita, vaan tarkasti harkittuja ja suunniteltuja ominaisuuksia. Puutteet tai ominaisuudet, kummaksi niitä haluaakin kutsua, liittyvät muun muassa tiedostojen siirtämiseen ja uudelleen nimeämiseen, turvallisuusriskeihin, tallennusten epävarmuuteen, tiedostojen oletusmuotoihin ym. Ajan mittaan monet ovat halunneet muokata CVS-koodia ja lisätä ominaisuuksia, mikä on johtanut useisiin eri projekteihin, kuten CVSNT, EVS, OpenCVS, Subversion ym. (Concurrent Versions System 2009.)

Subversion

Subversion on CollabNet:in vuonna 2000 julkaisema versionhallintajärjestelmä, jonka oli tarkoitus olla laajalti käytetyn CVS:n seuraaja, tai oikeastaan ”parempi CVS”, jossa CVSn pahimmat puutteet ja viat on korjattu. 2000-luvun alussa monet käyttäjät alkoivatkin siirtyä CVS:stä Subversioniin. Subversion on hyvin tunnettu ja paljon käytetty open source -yhteisöissä ja vuonna 2007 Forrester Research tunnusti Subversionin johtavaksi ohjelmistoksi Standalone Software Configuration Management (SCM) -kategoriassa. (Subversion 2009, Concurrent Versions System 2009.)

Subversion-projektilla ei ole graafista käyttöliittymää, vaan sitä käytetään pääasiassa komentokehotteessa. Subversionille on kuitenkin kehitetty montakin asiakasohjelmaa (client), joissa on graafinen käyttöliittymä. Näistä suosituin Windows-käyttöjärjestelmälle on TortoiseSVN, jota käytetään suoraan Windowsin resurssienhallinnassa. TortoiseSVN, joka myös on ilmaisohjelma, voitti vuonna 2007 SourceForge.netin Community Choice Awardin kategoriassa Best Tool or Utility for Developers. (Subversion 2009, TortoiseSVN 2009.)

Subversionissa tallennukset (commits) ovat ydinoperaatioita (atomic operations). Tämä tarkoittaa sitä, että esimerkiksi verkkoyhteyden katkeaminen tallennuksen aikana ei aiheuta muutoksia repositoryyn, eli tallennuksen yhteydessä joko kaikki muutokset tallentuvat, tai sitten yksikään muutos ei tallennu. Subversion tallentaa myös uudelleen nimettyjen, kopioitujen, siirrettyjen sekä poistettujen tiedostojen sekä hakemistojen historiatiedot. (Subversion 2009.)

Tiedostojen ja hakemistojen kopiointi on ”halpaa”, eli Subversion ei varsinaisesti kopioi tiedostoa, vaan tiedosto tallennetaan linkkinä viimeisimpään siihen tehtyyn muutokseen. Tämä ominaisuus säästää todella paljon tilaa, koska tilan tarve on suhteessa tiedostoihin tehtyihin muutoksiin, eikä niinkään tiedostojen kopioiden tai revisioiden määrään. Subversionissa on myös paljon muita tärkeitä parannuksia verrattuna CVS:ään. (Subversion 2009.)

GIT

Git on Linus Torvaldsin kehittämä versionhallintajärjestelmä, joka painottaa toiminnan nopeutta. Alun perin Git suunniteltiin Linux-ytimen kehittämistä varten, koska kehittäjien käyttämä BitKeeper-versionhallintajärjestelmä muuttui maksulliseksi. Torvalds halusi jatkaa Linux-ytimen kehitystyötä hajautetulla järjestelmällä, kuten BitKeeper, mutta yksikään ilmainen järjestelmä ei vastannut Torvaldsin tarpeita, varsinkaan suorituskyvyn osalta. Gitin kehitystyö alkoi huhtikuussa 2005 ja jo kesäkuussa Linux-ytimen julkaisu 2.6.12. hallittiin Git:llä. (Git 2009.)

Vaikka Git otti paljon vaikutteita BitKeeperistä, yritti Torvalds tarkoituksellisesti välttää tavanomaisia ratkaisuja ja lopputuloksena syntyikin uniikki versionhallintajärjestelmä. Torvalds kehitti järjestelmää kesäkuun 2005 loppuun asti, jolloin hän siirsi vastuun kehitystyöstä Juloi Hamanolle, joka on edelleen projektin ylläpitäjä. (Git 2009.)

Git on siis hajautettu järjestelmä, toisin kuin keskitetyt järjestelmät CVS ja Subversion. Tämä tarkoittaa sitä, että versionhallinnan alla olevat tiedostot eivät sijaitse yhdessä repositoryssä verkkopalvelimella, johon kaikilla käyttäjillä on pääsy. Gitin tapauksessa jokainen työhakemisto projektin osallistujien koneilla on itsessään täysin toimiva repository, millä on täysi historian ja revisioiden seuraamiskyky. Versionhallinta ei ole riippuvainen verkkoyhteydestä eikä pääsystä keskuspalvelimelle. Git antaa siis kehittäjälle paikallisen kopion mukana koko projektin kehityshistorian. Kehittäjän omat muutokset tallentuvat myös historiatietoihin, jotka taas siirtyvät muille projektiin osallistuville. (Git 2009, Git homepage 2009.)

4.2 Versionhallintajärjestelmän valinta

Luvun 4.1 perusteella valittava versionhallintajärjestelmä eEDUn tarpeisiin on hyvinkin selvästi Subversion. CVS sisältää liian paljon puutteita ja alkaa jo olla hieman vanhentunut järjestelmä nykyaikaiseen versionhallintaan. RCS:ää on turha edes harkita koska se on CVS:n ”isoisä” ja monella tapaa CVS:ää huonompi valinta. Subversionin kehittäjien iskulause ”CVS done right!”, kertoo myös selvästi CVSn ja Subversionin suhteesta.

Git sopii erityisen hyvin suuriin, hajautettuihin projekteihin, kuten Linuxin kehittämiseen, mihin se oli alun perin suunniteltukin. Jo tämä lähtökohta Gitin ja Subversionin välillä saa eEDUn tapauksessa valinnan kallistumaan voimakkaasti Subversionin puolelle. Koska Moodle-koodia hallitsee vain yksi tai muutama henkilö, on Subversion keskitettynä järjestelmänä paljon järkevämpi valinta.

Subversionin puolesta puhuu myös se seikka, että Tampereen ammattikorkeakoulun tietokonekeskus avasi tätä työtä tehtäessä erillisen Subversion-palvelimen opiskelijoiden ja henkilökunnan käyttöön. Seuraavassa luvussa kuvattu toteutus tehtiin siis TAMKin Subversion-palvelimella, johon ei ole pääsyä muilla kuin ylläpitäjillä, joten tietoturvariskiä ei pitäisi olla. Palvelimen tiedostoista otetaan myös varmuuskopiot säännöllisesti, joten Moodle-koodin hallitseminen kyseisellä palvelimella pitäisi olla hyvin turvallista. (Rintala 2009.)

5 Toteutus

Toteutusta ei lähdetty turvallisuussyiden takia heti tekemään oikealle TAMKissa käytössä olevalle Moodlelle, vaan päivityksiä testattiin erillisellä Moodle-asennuksella, johon tehtiin omia muutoksia. Itse asiassa päivitystestauksia tehtiin useammastakin vanhasta Moodle-versiosta uudempaan, mutta tässä on tilan säästämiseksi kuvattu vain yksi testaus Moodlen versiosta 1.8.9 versioon 1.9.5. Testauksia tehtiin käyttäen Subversionia komentokehotteen kautta sekä graafisen käyttöliittymän sisältävää TortoiseSVN:ää.

Toteutus on kuvattu hyvinkin yksityiskohtaisesti vaihe vaiheelta. Tarkoituksena on ollut tehdyn työn kuvaamisen lisäksi myös luoda eEDUlle ohje versionhallinnan toteuttamisesta käytännössä.

Alaluvuissa 5.1 ja 5.2 on tarkasteltu esimerkkien kautta, miten mahdollisesti eEDUssa käyttöön otettava versionhallintajärjestelmä tulisi toimimaan tulevaisuudessa. Esimerkkejä on lähdetty avaamaan repositoryn rakenteen kautta, jotta nähdään miten tietyt tilanteet vaikuttavat siihen.

Tarvittavien ohjelmien asennukset

Toteutus aloitettiin lataamalla ja asentamalla TortoiseSVN valmistajan sivuilta http://tortoisesvn.net/downloads. Uusin versio tuolloin oli 1.6.5, joka on rakennettu Subversionille 1.6.5 (julkaistu 22. elokuuta 2009). TortoiseSVN on Subversionin asiakasohjelma, jolla Subversionia voi käyttää suoraan Windowsin resurssienhallinnan kautta. (TortoiseSVN Project home 2009.)

TortoiseSVN ei valitettavasti sisällä oikopolkua tässä työssä hyvin oleelliseen Perl-skriptiin svn_load_dirs.pl, joten oli tarpeen asentaa myös Subversion käytettäväksi komentokehotteen kautta. Tähän tarkoitukseen asennettiin CollabNetin binääriversio 1.6.5 Subversionista (CollabNet Subversion Server and Client v1.6.5) Windowsille osotteesta http://www.collab.net/downloads/subversion/ (vaatii kirjautumisen). (Large 2008.)

Itse Perl-skripti ladattiin osoitteesta http://svn.collab.net/repos/svn/trunk/contrib/client-side/svn_load_dirs/. Tämä skripti on suunniteltu lataamaan Subversioniin useita tiedostohakemistoja kerralla uuden vendor dropin yhteydessä. Jotta skripti toimisi, asennettiin koneelle myös ActivePerl-5.10.1.1006 osoitteesta https://www.activestate.com/activeperl/downloads/. (Subversion Tools and Contrib 2009.)

Näiden esivalmistelujen jälkeen päästiin Moodlen version 1.8.9 asennukseen. Versiot 1.8.9 ja 1.9.5 ladattiin osoitteesta http://download.moodle.org/. Moodlesta julkaistaan viikoittain myös uusimmat päivitykset sisältävä versio 1.9.5+, jota suositellaan parhaana vaihtoehtona uudelle palvelimelle. Testaus päätettiin kuitenkin tehdä aiemmin mainituilla versioilla, koska ne olivat viimeisimmät virallisella numerolla julkaistut versiot. (Standard Moodle Packages 2009.)

Moodlen asennus ja räätälöintitestit

Ennen varsinaista asennusta Moodle-koodi vietiin (import) kuitenkin Subversionin repositoryyn, jotta versionhallintaan tallentui myös alkupiste, josta koodia on lähdetty muokkaamaan. Tarpeen tullen tähän alkuperäiseen koodiin voidaan palata. Liitteessä 2 on kaavio repositoryn rakenteesta ja tiivistetty ohje normaalista menettelystä uuden Moodle-version ilmestyessä. Repositoryn kaavion seuraaminen selkeyttää huomattavasti seuraavien vaiheiden seuraamista.

Moodle-koodi vietiin siis aluksi kansioon \vendor\current, että kaikille projektiin osallistuville on selvää, mistä löytyy viimeisin versio koodista. Seuraavaksi koodi kopioitiin, eli luotiin tunniste (tag) kansioon \vendor\Moodle_1.8.9. Tunnisteen luomisen tarkoituksena on tallentaa uusi versio muistiin, jos ilmenee tarvetta palata siihen myöhemmin. Periaatteessa samaan revisioon voidaan muutenkin palata, jos tiedetään revisionumero, mutta tunnisteen luominen on selkeämpi ja helpompi tapa. Tunniste kopioitiin tämän jälkeen kehityksen päälinjaan eli kansioon \trunk. Siellä koodiin oli tarkoitus tehdä tarvittavat omat muutokset ja räätälöinnit.

Tämän jälkeen koodi ”kirjattiin ulos” (checkout) repositorystä suoraan verkkopalvelimelle, jossa suoritettiin asennus. Tämä tehtiin, jotta koodia päästiin muokkaamaan oikeassa toimintaympäristössä ja muutosten vaikutus nähtiin heti. Muutokset pystyttiin näin myös lähettämään (commit) heti samasta versiosta Subversionin repositoryyn.

Moodlen asennus luo Moodlen hakemistojuureen config.php tiedoston, joka sisältää tunnukset sql-tietokantaan, polun itse Moodlen tiedostoihin palvelimella sekä muutamia muita tarpeellisia tietoja. Tämän lisäksi asennus rakentaa tarvittavat taulukot sql-tietokantaan. Testauksessa käytettiin TAMKin uutta MySql-tietokantapalvelinta osoitteessa mydb.tamk.fi.

Asennuksen jälkeen Moodleen tehtiin erilaisia muutoksia, jotta nähtäisiin pysyvätkö omat muutokset opetusympäristössä myös päivityksen jälkeen. Muutoksia tehtiin itse php-tiedostoihin, sql-tietokantaan ja moodledata-kansioon, jonne Moodle tallettaa esimerkiksi käyttäjätietoja, järjestelmään viedyt tiedostot ja asennetut kielipaketit.

Moodlen php-koodia muokattiin tiedostossa \moodle\admin\index.php sekä oletuskielen tiedostossa \moodle\lang\en_utf8\role.php. Tiedostoon index.php lisättiin useampi oma tekstirivi echo-funktiolla ja siirrettiin sivun alareunassa ollut tekijänoikeus-osa sivun yläreunaan. Tiedostoon role.php muutettiin $string['defineroles'] arvoa hieman pidemmäksi alkuperäisestä. Asennus loi uuden tiedoston config.php, jonka pitäisi myös säilyä mukana päivityksessä. Lisäksi Moodleen kansioon \moodle\theme asennettiin uusi teema, jossa oli noin 300 tiedostoa.

Tietokantaan muutettiin ylläpitäjän nimeä sekä kuvausta hieman pidemmäksi alkuperäisestä. Kuviossa 7 on sql-lause, jolla kuvauksen muuttaminen suoritettiin. Tietokantaan tallentuu myös valtava määrä muuta Moodlen tietoa. Testin yhteydessä Moodleen lisättiin kurssikategorioita, kursseja, käyttäjiä ja muutama rss-syöte kursseille. Myös nämä tiedot tallentuivat tietokantaan ja niiden olisi tarkoitus siellä myös pysyä päivityksen yhteydessä.

```
UPDATE mdl_role

SET description='Administrators can usually do anything on the site, in all courses. Basically they are almighty superhuman beings blessed with powers beyond belief.'

WHERE id='1';
```

Kuvio 7. Sql-lause

Moodledata-kansion sisältöä ei sinänsä muokattu, vaan tarkoitus oli seurata säilyvätkö sinne tallennetut tiedostot päivityksen yhteydessä. Admin-käyttäjälle lisättiin Moodle-profiiliin kuva ja oppimisympäristöön asennettiin kielipaketti fi_utf8 eli suomi. Lisäksi yksi käyttäjä lähetti kurssille tiedoston, joka tallentui Moodledata-kansioon.

Lopuksi uudet versionhallinnan pariin tuodut tiedostot, kuten config.php ja teeman tiedostot, lisättiin (add) ja muutokset lähetettiin (commit) repositoryyn. Tämä prosessi on perinteinen versionhallinnan alla olevien tiedostojen/tiedoston muokkausprosessi, joka toistuu aina muutoksia tehtäessä. Seuraavassa osassa kuvataan tilanne uuden Moodle-version ilmestyessä (Liitteen 2 kohdat 4. ja 5.).

Versioon 1.9.5 päivittäminen

Kun Moodlesta ilmestyy uusi versio, on oma räätälöity Moodlen vanhempi versio syytä päivittää. Päivittämisen periaatteena ei ole yhdistää (merge) omia räätälöintejä uusimpaan versioon, vaan paremminkin yhdistää versioiden välillä tapahtuneet muutokset omaan räätälöityyn versioon (Collins-Sussman ym. 2008, 112). Tämä saattaa aluksi kuulostaa hieman sekavalta, mutta on käytännössä suhteellisen suoraviivainen prosessi.

Aluksi käytössä ollut Moodle-versio kopioitiin talteen kansioon \tags\TAMK_Moodle_1.8.9. Seuraavaksi repositorysta kirjattiin ulos (check out) koko kansio \vendor. Sen jälkeen viimeisimmän version (\vendor\current) päälle kopioitiin käsin uusi Moodlen versio. Näin uudet, ja mahdollisesti muokatut tiedostot, korvaavat vanhemmat ja siirtyvät automaattisesti versionhallinnan alle. Tämän jälkeen on syytä poistaa tiedostot, jotka ovat vanhemmassa versiossa, mutta joita ei enää uudemmassa versiossa esiinny. Tähän on hyvä työkalu esimerkiksi WinMerge (tai joku muu tiedostojen vertailu -ohjelma), jolla näkee nopeasti käytöstä poistuneet tiedostot. TortoiseSVN:n mukana tuleva TortoiseMerge ei valitettavasti pysty vertaamaan kansioiden sisältöjä keskenään.

Uuden Moodle-version mukana hyvinkin luultavasti on uusia tiedostoja, joita ei vanhassa ollut. Nämä uudet tiedostot on syytä lisätä (add) versionhallintaan, jonka jälkeen kaikki muutokset voidaan tallentaa (commit) myös repositoryyn. Lopuksi vielä uusi versio kopioidaan (tag) kansiosta \vendor\current esimerkiksi uuden version numerolla merkittyyn kansioon \vendor\Moodle_1.9.5.

Edelliset kaksi kappaletta kuvaavat liitteen 2 kohdan 4. Samat toiminnot on mahdollista suorittaa myös perl-skriptillä svn_load_dirs.pl, joka tekee kaikki tarvittavat lisäykset, kopioinnit ym. Valitettavasti useista yrityksistä huolimatta skriptiä ei tätä työtä tehtäessä saatu kuitenkaan toimimaan, mutta toisaalta se ei koko työtä ajatellen ollut välttämätöntä.

Nyt kun Moodlen uusi versio oli tuotu mukaan versionhallintaan, voitiin siihen tehdyt muutokset viimein yhdistää (merge) omaan räätälöityyn versioon. TortoiseSVN:ssä on kolme eri yhdistämistapaa, joista tässä yhteydessä kuuluu valita kahden eri puun yhdistäminen (Merge two different trees). Kun SVN yrittää yhdistää tiedostoja, joiden sisällössä on ristiriita (conflict), kysyy ohjelma mitä käyttäjä haluaa tehdä. Vaihtoehtoina on käyttää paikallista tai repositoryssä olevaa tiedostoa, avata TortoiseMerge-ohjelma ristiriidan selvittämiseen käsin, tai jättää ristiriita ratkaisematta.

Jos käyttäjä ei ole täysin varma, mikä on oikea valinta, kannattaa ristiriita selvittää TortoiseMerge-ohjelmalla. TortoiseMerge merkkaa ristiriidat punaisella värillä, jolloin ne on helppo huomata ja valita oikea versio lopulliseen yhdistettyyn tiedostoon. Kun tiedoston kaikki ristiriidat on selvitetty, voidaan tallennuksen jälkeen TortoiseMerge sulkea. Tämän jälkeen voi kyseisen tiedoston ristiriidat merkitä TortoiseSVN:ssä ratkaistuksi (resolved). Kun kaikkien tiedostojen ristiriidat on ratkaistu, voidaan muutokset tallentaa (commit) repositoryyn.

Kun päivittämisen jälkeen Moodleen mentiin selaimella, alkoi automaattisesti Moodlen ja palvelimen päivittäminen uuteen versioon. Päivittämisen jälkeen käytössä oli uusi Moodle 1.9.5, jossa olivat edelleen mukana omat räätälöinnit. Nopean tutkimisen jälkeen huomattiin, että itse tehdyt muutokset ja lisäykset olivat säilyneet koko prosessin läpi php-tiedostoissa, sql-tietokannassa ja moodledata-kansiossa. Myös itse lisätty teema ja kielipaketti olivat vielä mukana. Kaikesta päätellen päivittäminen oli sujunut onnistuneesti. Lopuksi uusi, toimiva ja räätälöity Moodle kopioitiin (tag) talteen kansioon \tags\TAMK_Moodle_1.9.5.

5.1 Case 1 - uusi räätälöity julkaisu

Case 1 kuvaa tilannetta, jossa eEDUn hallittavana on jo ollut kolme eri Moodle-julkaisua, ja mukaan tulee neljäs. Alkutilanteessa, joka on kuvattu kuviossa 8, on eEDUssa ollut hallittavana erilliset TAMKin ja PIRAMKin Moodlet sekä uuden yhdistyneen TAMKin yhteinen Moodle-julkaisu elearning.tamk.fi. Tässä esimerkissä oletetaan, ettei jo ennestään käytössä olleita Moodle-julkaisuja ole ehditty vielä kertaakaan päivittää ensimmäisistä versionhallintaan otetuista versioista 1.9. Tässä mallissa ei myöskään ole varsinaista kehityksen päälinjaa, koska kaikki Moodlen eri julkaisut ovat samanarvoisia.

```
Moodle repository
      -branches
            -TAMK
            -PIRAMK
            -elearning.tamk.fi
      -tags
            -TAMK_Moodle_1.9
            -PIRAMK_Moodle_1.9
            -elearning.tamk.fi_Moodle_1.9
      -trunk
      -vendor
            -current
            -Moodle_1.9
```

Kuvio 8. Case 1 alkutilanne

Kun Moodlesta halutaan tehdä uusi julkaisu esimerkiksi Tampereen kaupungin käyttöön, kopioidaan Moodlen viimeisin versio kansiosta /vendor/current uudeksi haaraksi (branch) kansioon /branches/Tampere. Tähän uuteen haaraan tehdään sen jälkeen kaikki tarvittavat räätälöinnit. Jos halutaan hyödyntää johonkin toiseen julkaisuun tehtyjä räätälöintejä, voidaan uusi haara kopioida myös jostain toisesta jo valmiista julkaisusta, esimerkiksi /tags/TAMK_Moodle_1.9. Tällöin uuden haaran räätälöinnistä päästään mahdollisesti vähemmällä työllä.

Otettiinpa kopio mistä tahansa, näyttää repository lopuksi kuvion 9 mukaiselta. Kansioon /branches on siis tullut uusi haara /Tampere, jossa jatketaan kyseisen julkaisun kehittämistä edelleen. Kansiossa /tags/Tampere_Moodle_1.9 on kopio käytössä olevasta, valmiiksi räätälöidystä julkaisusta.

```
Moodle repository
      -branches
            -TAMK
            -PIRAMK
            -elearning.tamk.fi
            -Tampere
      -tags
            -TAMK_Moodle_1.9
            -PIRAMK_Moodle_1.9
            -elearning.tamk.fi_Moodle_1.9
            -Tampere_Moodle_1.9
      -trunk
      -vendor
            -current
            -Moodle_1.9
```

Kuvio 9. Case 1 lopputilanne

5.2 Case 2 - uuteen versioon päivittäminen

Case 2 kuvaa tilannetta, jossa eEDUn hallittavana on ollut neljä edellisen esimerkin Moodle-julkaisua, jotka kaikki halutaan päivittää uuteen versioon 2.0. Tätä lähtötilannetta, joka on sama kuin Case 1 lopputilanne, kuvaa edellä ollut kuvio 9. Uuden Moodle 2.0:n ilmestyttyä tuodaan sen koodi kansioon /vendor/current luvussa 5 sekä liitteessä 2 kuvatulla tavalla. Tämän jälkeen sen hetkinen versio koodista kopioidaan (tag) talteen kansioon /vendor/Moodle_2.0.

Seuraavaksi yhdistetään (merge) muutokset Moodlen version 1.9 ja 2.0 välillä kaikkiin eri Moodle-julkaisuihin kansiossa /branches. Tämä tehdään luvun 5 sekä liitteen 2 kohdan 5 mukaisesti. Muutosten yhdistäminen täytyy tehdä erikseen jokaiseen julkaisuun. Kaikkien julkaisujen uudet päivitetyt, ja toimiviksi todetut, versiot kopioidaan talteen kansioon /tags. Kuviossa 10 on lopputilanne Moodle 2.0:aan päivittämisen jälkeen.

```
Moodle repository
      -branches
            -TAMK
            -PIRAMK
            -elearning.tamk.fi
            -Tampere
      -tags
            -TAMK_Moodle_1.9
            -TAMK_Moodle_2.0
            -PIRAMK_Moodle_1.9
            -PIRAMK_Moodle_2.0
            -elearning.tamk.fi_Moodle_1.9
            -elearning.tamk.fi_Moodle_2.0
            -Tampere_Moodle_1.9
            -Tampere_Moodle_2.0
      -trunk
      -vendor
            -current
            -Moodle_1.9
            -Moodle_2.0
```

Kuvio 10. Case 2 lopputilanne

6 Päätelmät

Tämän työn lähtökohtana oli tutustua eri versionhallintajärjestelmiin sekä eEDUn toiminnan ominaispiirteisiin, ja tätä tietoa käyttäen päättää, olisiko versionhallintajärjestelmä sopiva työkalu eEDUn tarpeisiin. Hyvin pian työn alettua kävi selväksi, että versionhallinta kannattaa toteuttaa Moodle-oppimisympäristölle.

Tavoitteena oli myös löytää paras mahdollinen versionhallintajärjestelmä eEDUn käyttöön. Useisiin järjestelmiin tutustumisen ja niiden soveltuvuuden arvioinnin jälkeen hyvin selväksi valinnaksi muodostui Subversion, jolla työn varsinainen käytännön osuus toteutettiin. Ohjelmaa testattiin käytännössä mahdollisimman samanlaisessa ympäristössä, kuin koodin hallintaa tultaisiin todellisuudessakin toteuttamaan.

Työn tavoitteena oli myös suunnitella Moodle-koodin hallinta mahdollisimman helpoksi, jolloin myös päivitysten aiheuttamat virheet ja downtime vähenisivät. Myös tässä tavoitteessa onnistuttiin, pienistä vastoinkäymisistä huolimatta, testien perusteella hienosti. Moodlen päivityksiin, räätälöinteihin ja eri versioiden hallintaan luotiin selkeät käytännöt, joiden avulla työmäärä pysyy mahdollisimman pienenä.

Luvuissa 6.1 ja 6.2 on pohdittu työn onnistumista, tuloksia, hyödyllisyyttä ja kehittämismahdollisuuksia.

6.1 Johtopäätökset

Pääosin työ onnistui hyvin, koska kaikkiin tavoitteisiin päästiin. Ainoa suurempi vastoinkäyminen tuli Perl-skriptin svn_load_dirs.pl kanssa. Useista yrityksistä huolimatta skriptiä ei saatu toimimaan, eikä ongelmaan löytynyt syytä saati sitten ratkaisua. Koska skripti ei toiminut, tuli työn aikana myös asennettua ActivePerl täysin turhaan. Subversionia ei olisi skriptin toimimattomuuden takia myöskään tarvinnut käyttää komentokehotteen kautta, joten myös CollabNetin valmistaman Subversionin binääriversion olisi voinut jättää asentamatta.

Työn konkreettinen merkitys on eEDUlle varmasti suuri tulevaisuutta ajatellen. Moodle-päivitykset ovat ennen olleet todella työläitä omien räätälöintien uuteen versioon siirtämisen takia. Koska räätälöintien siirto on ollut käsityötä, on oman koodin määrää yritetty pitää mahdollisimman pienenä koodikaaoksen välttämiseksi. Tämä on johtanut kehityksen todella hitaaseen etenemiseen, ellei lähes pysähtymiseen. Kun versionhallinta otetaan käyttöön, poistuu tämä kehityksen este, eikä koodia tarvitse enää karsia. Näin oppimisympäristö kehittyy nopeammin paremmaksi, toimivammaksi ja uniikimmaksi.

Myös useiden eri Moodle-julkaisujen hallinnointi ilman versionhallintajärjestelmää olisi tullut olemaan varmasti todella hankalaa ja sekavaa. Versionhallinnan käyttöönotto luo tässäkin mielessä kestävän kasvupohjan tulevaisuudelle.

Koko tietotekniikan ala on niin nopeasti muuttuvaa ja kehittyvää, etteivät tämän työn tulokset ole kovinkaan pitkään ajan tasalla tai käyttökelpoisia. Siksi kaikki työstä irti saatava hyöty kannattaakin ottaa käyttöön mahdollisimman pian. Työn tulosten soveltuvuus eEDUn ulkopuolelle saattaa olla melko vähäistä. Monilla yrityksillä tai toimijoilla on niin erilaiset tarpeet ja vaatimukset versionhallinnan suhteen, että tilanne täytyy arvioida joka tapauksessa erikseen jo valittavasta versionhallintajärjestelmästä lähtien. Työn tuloksista voivat kuitenkin muutkin tahot nähdä selvästi versionhallinnan edut, hyödyt ja tärkeyden.

6.2 Kehittämismahdollisuudet

Ensimmäinen askel eEDUlle on tietenkin versionhallinnan ottaminen käyttöön tässä työssä esitetyllä tavalla. Tähän on hyvä mahdollisuus esimerkiksi kun Tampereen ammattikorkeakoulu ja Pirkanmaan ammattikorkeakoulu yhdistyvät vuoden 2010 alussa. Koulujen sisäiset tietojärjestelmät kokevat suuria muutoksia ja Moodlen päivittäminen sekä versionhallinnan pariin ottaminen on myös hyvä tehdä samalla. Uudelle organisaatiolle on helppo luoda uudet käytännöt myös Moodlen ylläpitämiseen ja päivittämiseen.

Kun versionhallinta on saatu hyvin lähtemään käyntiin, voidaan miettiä kehittämismahdollisuuksia edelleen. Yksi mahdollinen, tässäkin työssä esiin tullut, kehitysmahdollisuus on svn_load_dirs.pl -skriptin toimintaan saattaminen. Skriptistä saatava hyöty ei ole valtava, mutta silläkin saataisiin päivitysprosessia hieman yksinkertaisemmaksi ja nopeammaksi.

Tämän työn käytännön testailua tehtiin ensin paikallisessa, omalla koneella sijaitsevassa repositoryssä. Silloin oli mielessä, että yksi selkeä eEDUn versionhallinnan kehitysmahdollisuus on siirtää repository verkkopalvelimelle, jotta myös muut mahdollisesti tulevaisuudessa Moodlen kehittämiseen osallistuvat tahot pääsevät kiinni koodiin. Tämä verkkoon siirtyminen toteutui kuitenkin jo työn aikana, kun TAMK perusti oman Subversion-palvelimen.

Versionhallinnan maailmaa kannattaa seurata tulevaisuudessa, etteivät uudet kehittämismahdollisuudet ja parannukset jää huomaamatta. On myös tärkeää muistaa päivittää versionhallintaohjelmistoa tasaisin väliajoin, jotta varmistetaan järjestelmän paras mahdollinen toiminta ja suorituskyky. Kun versionhallinta on otettu eEDUssa käyttöön, kannattaa miettiä voisiko sitä hyödyntää Moodlen lisäksi myös muissa projekteissa. Versionhallintaohjelmistot soveltuvat hyvin monenlaisiin tarpeisiin ja hyödyt ovat huomattavia.

Lähteet

Collins-Sussman, Ben & Fitzpatrick, Brian W. & Pilato, C. Michael 2008. Version Control with Subversion For Subversion 1.6 (Compiled from r3543). California: Creative Commons.

Concurrent Versions System 2009. [online] [viitattu 26.10.2009]. http://en.wikipedia.org/wiki/Concurrent_Versions_System

eEDU -esitys 2008. [online] [viitattu 20.10.2009] Saatavissa vain TAMKin sisäisessä verkossa. https://intra.tamk.fi/asiakirja/18009, eLearning_at_TAMK_2008.pdf

Git 2009. [online] [viitattu 27.10.2009]. http://en.wikipedia.org/wiki/Git_(software)

Git homepage 2009. [online] [viitattu 27.10.2009]. http://git-scm.com/

Large, Simon 2008. TSVN and Vendor Branch (or how to use svn_load_dirs.pl with TSVN). [online] [viitattu 14.9.2009]. http://svn.haxx.se/tsvnusers/archive-2008-03/0213.shtml

Moodle. [online] [viitattu 16.10.2009]. http://en.wikipedia.org/wiki/Moodle

moodle - Learning Environment of TAMK. [online] [viitattu 16.10.2009]. http://moodle.tamk.fi/

Moodle community. [online] [viitattu 16.10.2009]. http://moodle.org/

Moodle statistics. [online] [viitattu 16.10.2009]. http://moodle.org/stats/

Ohjelmiston versiohallinta 2009. [online] [viitattu 27.10.2009].
http://fi.wikipedia.org/wiki/Ohjelmiston_versiohallinta

Opetusteknologiakeskus eEDU 2007. [online] [viitattu 20.10.2009].
http://www.tamk.fi/fi/WWWYRIT/opetusteknologia.html

Opetusteknologiakeskus eEDU -esitys 2008. [online] [viitattu 20.10.2009] Saatavissa vain TAMKin sisäisessä verkossa.
https://intra.tamk.fi/asiakirja/18009, Tukipalvelut_eEDU_2008.pdf

Revision control 2009. [online] [viitattu 26.10.2009].
http://en.wikipedia.org/wiki/Revision_control

Rintala, Matti 2009. versionhallinta Subversionilla.
S-posti. matti.rintala@tamk.fi. Tulostettu 13.10.2009.

Roadmap 2009. [online] [viitattu 16.10.2009].
http://docs.moodle.org/en/Roadmap

Standard Moodle Packages 2009. [online] [viitattu 24.9.2009].
http://download.moodle.org/

Subversion 2009. [online] [viitattu 26.10.2009].
http://en.wikipedia.org/wiki/Subversion_(software)

Subversion Tools and Contrib 2009. [online] [viitattu 14.9.2009].
http://subversion.tigris.org/tools_contrib.html#svn_load_dirs_pl

TortoiseSVN 2009. [online] [viitattu 26.10.2009].
http://en.wikipedia.org/wiki/Tortoisesvn

TortoiseSVN Project home 2009. [online] [viitattu 14.9.2009].
http://tortoisesvn.tigris.org/

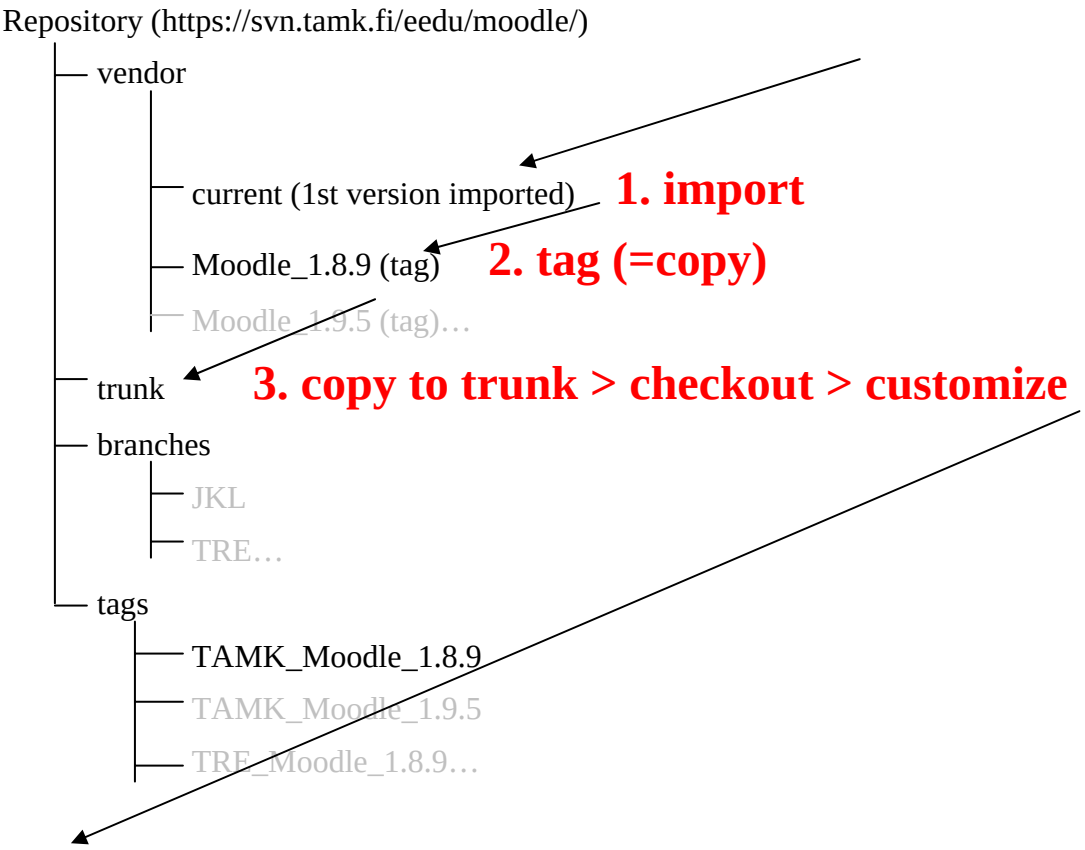
Liite 1 Versionhallinnan sanastoa

Tähän liitteeseen on listattu tässä työssä esiintyviä, versionhallintaan ja oppimisympäristöihin liittyviä, termejä ja niiden selityksiä.

ActivePerl	Perl-kielen jakelu
Add	lisätä tiedosto/tiedostoja versionhallintajärjestelmään
Atomic operation	ydinoperaatio; joko kaikki muutokset tallentuvat, tai sitten yksikään muutos ei tallennu
BitKeeper	maksullinen versionhallintajärjestelmä
Branch	kehityksen muista riippumaton haara
Branching	projektin haarautuminen eri kehityslinjoiksi
Checkout	koodin ulos kirjaaminen versionhallintajärjestelmästä
Client	asiakasohjelma
Commit	muutosten tallentaminen repositoryyn
Conflict	ristiriita tiedostojen muutosten välillä
CVS	Concurrent Versions System, ilmainen versionhallintajärjestelmä
CVSNT	CVSn Windows version kehitysprojekti
Downtime	ajanjakso jolloin järjestelmä ei ole käytettävissä
EVS	CVS:stä haarautunut kehitysprojekti
Git	ilmainen versionhallintajärjestelmä
Hajautettu järjestelmä	versionhallinnan parissa olevat tiedostot eivät sijaitse yhdessä repositoryssä
Import	koodin tuonti repositoryyn
Keskitetty järjestelmä	versionhallinnan parissa olevat tiedostot sijaitsevat yhdessä repositoryssä

Merge	tiedostojen eri versioiden yhdistäminen
Moodle	ilmainen virtuaalinen oppimisympäristö
OpenCVS	CVS:stä haarautunut kehitysprojekti
PHP	Hypertext Preprocessor, ohjelmointikieli
RCS	Revision Control System, vanha versionhallintajärjestelmä
Repository	varastokirjasto jossa versionhallinnan alla olevat tiedostot sijaitsevat
Resolve	ratkaista tiedostojen yhdistämisen ristiriita
Revisio	revisionumerolla tarkoitetaan repositoryn tiedostojen ja hakemistojen tilaa tiettynä aikana
SQL	Structured Query Language, ohjelmointikieli
Subversion	ilmainen versionhallintajärjestelmä
svn_load_dirs.pl	Perl-skripti joka on suunniteltu lataamaan Subversioniin useita tiedostohakemistoja kerralla
Tag	tunniste
TortoiseMerge	TortoiseSVN:n mukana tuleva tiedostojen vertailu -ohjelma
TortoiseSVN	graafisen käyttöliittymän sisältävä Subversionin asiakas-ohjelma
Trunk	runko, projektin kehityksen päälinja
Update	repositoryn muutosten tallentaminen omaan työkopioon
Vendor	ohjelmiston toimittaja/valmistaja/myyjä
Vendor drop	ohjelmiston valmistajan julkaisema uusi versio
WinMerge	tiedostojen vertailu -ohjelma
Working copy	henkilökohtainen työkopio repositoryn tiedostoista

Liite 2 General vendor branch management procedure



- tag customized TAMK_Moodle_1.8.9 to /tags

- *1.9.5 comes out*

4. (= svn_load_dirs.pl perl script, use this if vendor drop has many tree changes; no shortcut in TortoiseSVN)

- check out /vendor

- copy 1.9.5 on top \vendor\current (by hand in my computer)

- svn add new files

- svn delete any files present in 1.8.9 but not in 1.9.5 (easier with some comparison tool like e.g. WinMerge / or just leave them in unused obscurity)

- commit \vendor\current

- tag the new vendor drop as 1.9.5 (= copy from /vendor/current)

5.

- merge the differences between tag 1.8.9 and /vendor/current into working copy (\trunk)

- resolve all conflicts between 1.8.9/1.9.5 changes and own changes

- commit \trunk (message: “Merging 1.9.5 into the main development branch)

- tag customized TAMK_Moodle_1.9.5 to /tags