LeMill-alustan modifiointi Hämeen ammattikorkeakoulun tarpeisiin

Case Hämeen ammattikorkeakoulu, oppimateriaalipankki

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, 3.6.2011

Okko Pylsy

HAMK
HÄMEEN AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

Tietojenkäsittelyn koulutusohjelma
Hämeenlinna

Työn nimi	LeMill-alustan modifiointi Hämeen ammattikorkeakoulun tarpeisiin Case Hämeen ammattikorkeakoulu, oppimateriaalipankki
Tekijä	Okko Pylsy
Ohjaava opettaja	Lasse Seppänen
Hyväksytty	_____._____.2011
Hyväksyjä	

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Visamäki
Tietojenkäsittelyn koulutusohjelma
Systeemityö

Tekijä Okko Pylsy **Vuosi** 2011

Työn nimi LeMill-alustan modifiointi Hämeen ammattikorkeakoulun tarpeisiin
Case Hämeen ammattikorkeakoulu, oppimateriaalipankki

TIIVISTELMÄ

Työn tarkoitus oli selvittää, minkälaisia muutoksia LeMill-oppimisalustaan tarvitsee tehdä, jotta sitä voitaisiin käyttää materiaalipankkina Hämeen ammattikorkeakoulussa sekä miten nämä muutokset voitaisiin toteuttaa. Työn tilaajana toimi Hämeen ammattikorkeakoulun tietojenkäsittelyn koulutusohjelman yliopettaja Lasse Seppänen. Työ pohjautuu kesällä 2010 tehtyyn Summer ICT -projektiin, jossa selvitettiin LeMill-alustan soveltumista oppimateriaalin tallentamiseen, säilömiseen sekä itse alustan muokattavuuteen. Työssä käytettiin Python-ohjelmointikieltä sekä Zope-sovelluspalvelimen scriptauskieliä TAL, TALES ja METAL.

Työssä vastattiin seuraaviin kysymyksiin: Minkälaisia ominaisuuksia LeMill-alustaan pitäisi lisätä ja kuinka LeMill-alustaan lisätään ominaisuuksia?

Ensimmäiseen tutkimuskysymykseen selvitettiin vastaus haastatteluilla. Haastateltavina toimi joukko Hämeen ammattikorkeakoulun opettajia eri opintolinjoilta. Haastatteluiden tuloksista valittiin toteutettavat muokkaukset ja nämä ohjelmoitiin LeMill-alustaan. Tällä vastattiin jälkimmäiseen tutkimuskysymykseen.

Haastattelussa ilmikäyneitä, mutta työstä ulos jääneitä ominaisuus- ja kehittämisideoita esitellään ja näiden toteuttamismahdollisuuksia pohdittiin jatkokehityssuunnitelmassa. Teoriaosuudessa esitellään, mitä ovat oppimisalustat, oppimateriaalipankit sekä mitä eroa näillä kahdella on. Työssä esitellään myös avoin lähdekoodi, sen historia sekä avoimen lähdekoodin lisenssejä. Työ esittelee Zope-sovelluspalvelimen, Plone-sisällönhallintajärjestelmän ja LeMill-oppimisalustan.

Avainsanat oppimisalusta, oppimateriaalipankki, scriptaus, ohjelmointi, avoin lähdekoodi

Sivut 27 s.

Visamäki
Degree Programme in Business Information Technology

Author Okko Pylsy **Year** 2011

Subject of Bachelor's thesis Modification of LeMill-platform for HAMK University of Applied Sciences

ABSTRACT

The purpose of the thesis is to find out what kind of modifications are needed for the LeMill learning material bank so it can be used as a learning material bank for HAMK University of Applied Sciences. It also includes the actual implementation of these modifications where they are applicable.

The subject was suggested by Lasse Seppänen and it is based on Summer ITC –project which took place in summer 2010. This work continues and builds upon that project. The project researched how modifiable the LeMill platform is and how easily you could modify it.

This thesis uses the Python-programming language, the Zope application server scripting languages TAL, METAL and TALES. Also some http markup language is used. This work answers two research questions: What kind of features should be implemented into the LeMill platform? The second question is: How to implement features into the LeMill platform?

The first question is answered with interviews. Interviewees are going to be teachers from different curricula in HAMK University of Applied sciences. The second question is answered with implementing some results of the interviews. The thesis also explains what learning platforms and learning material banks are. It also explains the differences between the two. The thesis finds out what open source is, what its history is and introduces common open source licenses. The thesis also introduces the Zope application server, the Plone content management system and the LeMill learning material bank.

Keywords learning platform, learning material bank, programming, scripting, open source

Pages 27 p.

SISÄLLYS

Sanasto

Sovelluspalvelin	Verkossa sijaitseva palvelin, joka ajaa erilaisia ohjelmia käyttäjän ottaessa yhteyttä palvelimeen. Näitä voi olla esimerkiksi jokin sisällön hallintaohjelmisto tai asiakasrekisteri.
Sisällönhallintajärjestelmä	Sovellus, jonka avulla voidaan tehdä ja julkaista esimerkiksi nettisivuja helpommin. Järjestelmä hoitaa korkean tason koodin ja käyttäjä tuottaa sisällön.
Konekielinen tiedosto	Binäärikoodina oleva tiedosto
Binäärijärjestelmä	Lukujärjestelmä
Käyttöjärjestelmä muiden	Ohjelmisto, joka hoitaa kommunikaation käyttäjän sekä ohjelmien ja laitteiston välillä.
Metadata	Kuvaus tiedosta. Esimerkiksi jonkin työn julkaisija ja julkaisuvuosi ovat metadataa.
XML	Metadatan kuvauskieli. XML-tiedosto on selkokielisesti kirjoitettu ja sitä käytetään kuvaamaan tietoa.
Public domain	Termi jota käytetään yleiseen käyttöön tarkoitetuista teoksista. Näitä saa vapaasti muokata ja uudelleen käyttää haluamallaan tavalla.

1 JOHDANTO

Erilaisten tietoverkkojen yleistyessä ovat myös oppimateriaalit siirtyneet verkkoon. Tämän johdosta oppimateriaaleista on tullut sähköisiä verkkoresursseja oppimateriaalipankeissa. Tästä on ollut se hyöty, että materiaalit ovat helpommin jokaisen saatavilla ja ne ovat myös helposti muokattavissa ja ylläpidettävissä.

Hämeen ammattikorkeakoulu (käytetään tästä edespäin lyhennettä HAMK) tarvitsee paikan säilyttää oppimateriaalia. Sitä varten on otettu kokeiltavaksi LeMill-alusta, jota muokataan tarvittavaan käyttöön. Koska LeMill on suunniteltu avoimeksi oppimisalustatyyppiseksi alustaksi, joudutaan tätä muokkaamaan vastaamaan HAMKin tarpeita.

HAMKin tarve erilliselle materiaalipankille nousi esiin, koska Moodle poistaa vanhat kurssit kahden vuoden jälkeen. Tällöin näiden kurssien materiaalit häviävät. Tämän takia erillinen materiaalien säilytyspaikka on tarpeen. LeMill-alusta nousi esiin helpon käytettävyytensä ansiosta.

Opinnäytetyön aiheena on selvittää, minkälaisia muokkauksia LeMill-alustaan tarvitsee tehdä sekä toteuttaa toteuttamiskelpoiset muutokset. Opinnäytetyö on pohjautuu Summer ICT –projektille, joka tehtiin kesällä 2010. Tässä projektissa selvitettiin, voiko LeMill-alustaa käyttää materiaalin säilyttämiseen ja kuinka työlästä sen muokkaaminen on. Tämän takia minulla on jo kohtuullisesti kokemusta LeMill-alustasta ja sen muokkaamisesta.

Kaikki tässä työssä käytettävät ohjelmat ovat avoimen lähdekoodin ohjelmia. Tämän johdosta työssä esitellään avoimen lähdekoodin historia ja periaatteet.

Opinnäytetyössä haetaan vastauksia kahteen kysymykseen. Minkälaisia ominaisuuksia LeMill-alustaan pitäisi lisätä? Kuinka LeMill-alustaan lisätään ominaisuuksia? Ensimmäiseen vastataan haastattelemalla HAMKin organisaatiossa työskenteleviä ihmisiä. Lähinnä eri alojen opettajia, mutta haastattelupyyntö lähetetään myös IT-puolen ihmisille. Tällä selvitetään, minkälaisia muutoksia LeMill-alustaan tarvitsee tehdä. Toiseen kysymykseen vastataan kokoamalla haastattelut yhteen ja päätetään haastatteluiden tulosten perusteella toteuttamiskelpoisimmat muokkaukset. Lopuksi nämä muutokset ohjelmoidaan LeMill-alustaan.

Tutkimusmenetelminä käytetään ensin haastatteluita, joilla selvitetään tarvittavat ominaisuudet. Haastatteluiden jälkeen aloitetaan kehittämisprojekti, jonka aikana lisätään LeMill-alustaan ominaisuuksia. Kehittämisprojektissa esiin nousseista kysymyksistä ja edelleen kehittämisideoista raportoidaan mahdollisimman tarkasti.

2 OPPIMATERIAALIPANKKI

Tässä luvussa kerrotaan, mitä ovat oppimisalustat sekä oppimateriaalipankit. Luku esittelee myös oppimisaihiot ja niiden käytön. Luvussa esitellään myös LeMill-oppimateriaalialusta.

2.1 Oppimisalusta vai oppimateriaalipankki

Nykyään puhutaan paljon oppimisalustoista ja oppimateriaalipankeista. Tämä luku kertoo miten ja miksi näitä on alettu käyttää ja miten ne eroavat toisistaan.

Oppimisalustat ovat verkkosivustoja, jotka sisältävät oppimateriaalia, mahdollisuuden tehtävien tekemiseen ja palauttamiseen sekä mahdollisuuden interaktiivisuuteen käyttäjien välillä, olivat he sitten oppilaita tai opettajia. Alustoihin kuuluu myös mahdollisuus hallita sisältöä ja yksittäisiä kursseja sekä tiedot siitä kuka saa käyttää aineistoa. (Manninen 2005, 11.) Interaktiivisuudella tarkoitetaan mahdollisuutta palauttaa tehtäviä, keskusteluita tai reaaliaikaista chattia.

Hyvä esimerkki tällaisesta on HAMKissakin käytössä oleva Moodle. Moodle tarjoaa mahdollisuuden suunnitella kursseja ja pitää esillä kurssiin liittyvää materiaalia. Moodleen luotuihin kursseihin voi luoda tehtäviä ja näihin voi tehtäväkohtaisesti palauttaa vastauksen opettajan haluamalla tavalla. Moodlessa on myös mahdollisuus käydä keskusteluita kurssin osallistujien kesken.

Oppimisalustoilla tulisi myös olla hyvä tuki laajentamiselle. Pelkät edellä luetellut ominaisuudet eivät välttämättä riitä oppimisalustaa käyttävälle instituutiolle. Tämän johdosta tulisi oppimisalustaan olla jo olemassa tai siihen tulee olla helppo tehdä itse lisää ominaisuuksia. Mikäli oppimisalusta on tuotettu avoimella lähdekoodilla, tämä on yleensä helppoa ja tämän takia sille tehdään tarvittaessa lisäominaisuuksia käyttäjien toimesta kun nämä niitä tarvitsevat. Mikäli oppimisalusta on suljettu, olisi silloin sitä tuottavan tahon hyvä tehdä siihen helppokäyttöinen ohjelmointirajapinta, jolloin lisäosien tuottamiseen ei tarvitsisi alkuperäisen tuottajan resursseja, vaan lisäosat voisi ohjelmoida käyttämään tuota rajapintaa. Tällä tavalla saataisiin pidettyä oppimisalustan lähdekoodi suljettuna, mutta lisäosatuotanto hajautettuna.

Oppimateriaalipankeille syntyi tarve, kun erilaiset dokumentit ja muut oppimiseen käytettävät välineet digitalisoituvat. Alkujaan nämä tiedostot säilytettiin jollain palvelimella ja niihin käsiksi pääsy oli vaivalloista. Nykyään oppimateriaalipankit ovat keskittyneet helppokäyttöisyyteen ja ylläpidettävyyteen. Tällä tarkoitetaan sitä, että etsimänsä tiedon löytää helposti sekä tiedon helppoa ylläpidettävyyttä. Nykyiset oppimateriaalipankit ovat verkkopalvelimia, joissa säilytetään oppimateriaaleja. Ohjelmalle on tehty suoraviivainen käyttöliittymä ja mahdollisimman monipuoliset hakumahdollisuudet. Näiden pankkien käyttäjäkunta koostuu yleensä jostain suljetusta instituutioista, kuten opistot. Esimerkkejä oppimateriaalipankeista

ovat mm. Merlot (Merlot), Maricopa Center for Learning and Instruction (http://mcli.maricopa.edu/) ja Ariadne (http://www.ariadne-eu.org/index.php). Oppimateriaalipankkien tulisi olla mahdollisimman pelkistettyjä. Tämän takia ne on hyvä erottaa oppimisalustoista. Tästä seuraa hieman enemmän ylläpidollista toimintaa, mutta kun materiaali ja tehtävienteko on eritelty, selkeytyy materiaalin hallinnointi. Otetaan esimerkiksi jokin kurssi jollain oppimisalustalla. Opettaja tekee sinne kurssin ja sinne tehtäviä sekä materiaalia. Nyt käykin niin, että hän pitää samaa kurssia kahdelle eri ryhmälle samaan aikaan. Hän voisi laitaa kaikki käyttäjät samaan paikkaan. Hän voisi myös tehdä kummallekin ryhmälle omat tilansa. Tästä tulisi valtavasti lisätyötä, mikäli kurssilla on paljon aineistoa. Tätä työtä voidaan vähentää, mikäli käytetään materiaalipankkia. Tällöin opettajan ei tarvitse kopioida jokaiselle kurssille aineistoa, vaan hän voi linkata materiaalipankissa sijaitsevaan aineistoon.

Oppimateriaalipankkien hyviä puolia ovat myös materiaalin hyvä ylläpidettävyys. Kun aineisto sijaitsee keskitetysti materiaalipankissa, niin siellä olevaa yhtä aineistoa muokkaamalla sen tiedot päivittyvät kaikkialla, missä tätä aineistoa käytetään. Tästä seuraa resurssien säästöä monellakin tapaa. Ensimmäinen säästö on edellä mainittu ylläpidon parantuvuus. Myös palvelintilaa säästyy, koska materiaalista ei ole monia kopiota monessa eri paikassa. Kolmas säästö on aika, jota ei tarvitse enää käyttää päivittämään materiaalia monessa paikassa.

2.2 Oppimisaihio

Kun tietokoneet ja erityisesti tietoliikenneverkot yleistyivät, haluttiin oppimateriaali siirtää käytettäväksi verkkoon. Tästä syntyisi säästöä, koska materiaali olisi helpommin saatavilla ja se olisi helpommin ylläpidettävää. Aluksi oppimisaihiot olivat yksinkertaisesti verkkodokumentteja, mutta materiaali määrän kasvaessa ja monimutkaistuessa alettiin nähdä tarve jäsentää niiden sisältämä tieto paremmin.

Verkkomateriaaleista puhuttaessa tarkoitetaan yleensä oppimista edistävistä oppimisaihioista (eng. Learning Objects). Oppimisaihiot voivat olla muutaman lauseen pituisista teksteistä kokonaisiin kurssikokonaisuuksiin, jotka sisältävät dokumentteja, ääni- ja kuvatiedostoja sekä erilaisia tehtäviä. Oppimisaihio voisi siis yksinkertaisimmillaan olla yksi tekstitiedosto ja monimutkaisimmillaan laaja sivustokokonaisuus. HAMKin käyttämässä Moodlessa yksi kurssi voi yksinään olla oppimisaihio. Oppimisaihiot ovat itsessään kokonaisuuksia, jotka kertovat jonkin asian. Tämän takia niistä voidaan koota erilaisia kokonaisuuksia kuten legopalikoista. Esimerkiksi aihio voisi kertoa, kuinka tehdään yksinkertainen verkkosivu. Tätä aihiota voisi käyttää myös osana aihiota, joka kertoo, kuinka luodaan verkkosovellus. Tällöin se olisi osa suurempaa kokonaisuutta, mutta sisältäisi edelleen saman tiedon sekä sitä voitaisiin edelleen hyödyntää muokkaamattomana muuallakin.

Kun kaupalliset tahot kiinnostuivat luomaan oppimisalustoja, ne olivat tyypillisesti epäyhteensopivia keskenään. Tämä nosti esiin tarpeen saada erilaiset verkkosisällöt toimimaan erilaisissa alustoissa ilman aikaa vievää muokkausta. Tämä helpottaisi oppimateriaalialustasta toiseen siirtymistä ja saisi eri alustoilla tehdyt aihiot toimimaan muilla alustoilla. Tätä varten on kehitetty erilaisia tapoja kertoa, mitä aihio sisältää. Koska itse aihio sisältää itse tiedon, piti kehittää jonkinlainen tiivistelmä tai metatieto aihion sisällöstä. Nykyään on käytössä useita erilaisia standardeja kuvaamaan näitä aihioita.

Yksi tätä varten kehitetyistä metatietotavoista on SCORM (Shareable Content Object Reference Model). Se on joukko määrityksiä ja standardeja digitaalisille oppimateriaaleille. Sen kehitys alkoi Yhdysvalloissa Advanced Distributed Learning Initiative (ADL) eli kehittyneen jaellun tiedon aloiteena. Tällä aloitteella haluttiin luoda profiili standardeista, joita käytettäisiin digitaaliselle tiedolle. (SCORM) Ensimmäinen SCORMin versio oli vain testi, jolla haluttiin selvittää mahdollisia puutteita. Tästä julkaisusta saadut tiedot auttoivat kehittämään seuraavat versiot. Uusin versio SCORMista on SCORM 2004. SCORMin kehityksestä vastaa ADL, www.aldnet.gov. Toinen käytetty metatietostandardi on Dublincore. Se on oikeammin joukko erilaisia standardeja, joita käytetään kuvaamaan erilaisia tietoja. Dublin Corea ylläpitää ja kehittää Duplin Core Metadata Initiative.

Oppimateriaalipankkien aihioille merkataan aihiota luotaessa metatietoa. Tämä metatieto kertoo, mitä ja minkälaista tietoa aihio sisältää. Oppimisaihion metadataa kutsutaan LOMiksi (Learning Objects Metadata) ja se on IEEE:N standardi. LOM on tyypillisesti kirjoitettu XML-muotoon. Tämän johdosta se on helposti luettavissa ja mahdollisimman yhteensopiva. LOMin avulla tietoa voidaan etsiä helpommin, eikä dataa tarvitse seuloa joka kerralla hakua suorittaessa vaan haku voidaan kohdistaa metadataan. Metadata voi yksinkertaisimmillaan olla aihion nimi, mutta yleensä metadata kertoo ainakin tekijät, aihion luontiajan, joukon avainsanoja ja minkälaista materiaalia aihio sisältää. Tämä metadata voi myös sijaita eri palvelimella kuin itse oppimisaihiot, mikä säästää palvelinresursseja varsinkin silloin, kun palvelimella on paljon käyttäjiä. Tämä säästö johtuu suuressa osin erilaisista hakutoiminnoista. Kun käyttäjien suorittamat haut osoitetaan metatietoihin, jotka on sijoitettu erilliselle palvelimelle, ei se palvelin, jolla materiaali sijaitsee, rasitu hakujen johdosta.

2.3 LeMill-oppimateriaalipankki

LeMill ei ole oppimisalusta siinä mielessä miten oppimisalustat yleensä mielletään. “LeMill on verkkoyhteisö oppimisresurssien löytämiseen, tuottamiseen ja jakamiseen. Sitä kehittää LeMill-tiimi, jota johtaa Learning Environments Research Group of Media Lab Helsingin Taideteollisesta korkeakoulusta. LeMillin kehitys aloitettiin 2005–2008 CALIBRATE-projektissa.

http://calibrate.eun.org/ww/en/pub/calibrate_project/home_page.htm" (LeMill-About, 2010).

LeMillin osoite on www.lemill.net. Se sisältää tällä hetkellä noin 24000 oppimisresurssia 27 eri kielellä. Eri maiden sisällön tuottajat, LeMillin tapauksessa yleensä opettajat, käyttävät LeMilliä hieman eri tavoin. Tämä johtuu Tarmo Toikkasen mukaan siitä, että tietyt sanat ymmärretään eri tavoin eri maissa. Tämä ei kuitenkaan ole suuri huoli. Tärkeintä on, että alustaa käytetään ja tieto on vapaata. Suuri osa LeMill-alusta sisällöstä on julkaistu Creative Commons Sharealike -lisenssin alaisuudessa. Tällä suojataan alkuperäisen luojan oikeudet sekä annetaan muille oikeus jakaa ja uudelleen käyttää tietoa (Creative Commons, 2011). LeMillissä on panostettu käytettävyyteen ja tähän viittaa esimerkiksi se, että metatiedon hallinta on hyvin minimaalista. Sivua tehtäessä valitaan valmiiseen pohjaan erilaisista vaihtoehdoista aihiota kuvaavat kohdat ja LeMill hoitaa tämän jälkeen metatiedon luomisen. LeMillin luoma metatieto on SCORM-yhteensopiva. LeMill perustuu avoimuuteen ja uudelleenkäytettävyyteen ja tämän takia LeMill-alusta ei hyväksy monenlaisia tiedostoja järjestelmäänsä. Tämä on yksi modifioinnin kohde tässä työssä.

3 AVOIN LÄHDEKOODI

Tämä luku kertoo mitä on avoin lähdekoodi. Ensin käydään lävitse mikä on lähdekoodi ja tämän jälkeen syvennytään avoimen lähdekoodin historiaan. Avoimen lähdekoodin juuret ovat löytyvät -60 ja -70-lukujen Yhdysvaltain yliopistomaailmasta. Tällöin Yhdysvaltain valtio kehotti tukemiaan opistoja kehittämään tapoja hyödyntää hajautettuja järjestelmiä. (Gutschmidt 2003, 15). Avoimella lähdekoodilla tarkoitetaan ohjelmistoja, joiden lähdekoodi on saatavilla muillekin kuin ohjelman luojalle. Luvun lopussa esitellään yleisimmät avoimen lähdekoodin lisenssit ja kerrotaan, mitä niiden alaisuudessa saa sekä ei saa tehdä.

3.1 Lähdekoodi

Lähdekoodi on tietokoneohjelman alkuperäinen kirjoitusasu. Se on kirjoitettu tavallisina merkkeinä ja luettavissa sekä ymmärrettävissä ilman erillistä toimintaa. (Source Code Definition, 2003.)

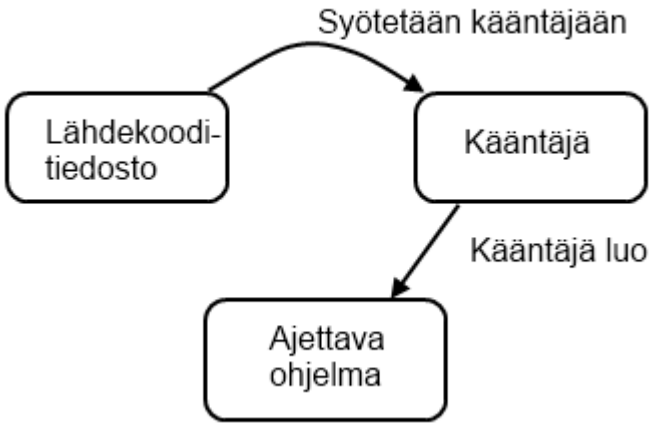
Kun ohjelmoija kirjoittaa tietokoneella ajettavan ohjelman, hän ei itse asiassa kirjoita sitä osaa, joka loppukäyttäjälle näkyy. Ohjelmoija kirjoittaa lähdekoodin, joka sitten käännetään tulkin tai kääntäjän avulla tietokoneen ymmärtämään muotoon. Tämän voi ymmärtää paremmin, kun ajattelee elokuvan tuotantoa. Elokuvan on kirjoittanut käsikirjoittaja. Mutta hän ei kirjoita sitä, mikä näkyy, kun katsot elokuvaa elokuvateatterissa tai kotona televisiosta. Käsikirjoittaja kirjoittaa tarinan ja tämän tarinan perusteella kirjoitetaan käsikirjoitus. Itse elokuva tehdään käsikirjoituksen perusteella. Näin voidaan myös ajatella ohjelmistojen teko. Ensin suunnitellaan ohjel-

ma, sitten suunnitelman pohjalta kirjoitetaan lähdekoodi ja tämän jälkeen lähdekoodi käännetään tietokoneen ymmärtämään muotoon. Nämä lähdekoodit ovat usein yhtiösalaisuuksia, koska ne kertovat, kuinka jokin ohjelma toimii ja lähdekoodin voi kääntää suoraan ohjelmaksi. Niitä suojataan tekijänoikeuksilla ja ohjelmistopatenteilla. Tätä vastaan kehittyi vapaa lähdekoodi. Se mahdollisti ohjelmistojen kehityksen erilaisin tavoin, kuten seuraavissa luvuissa kerrotaan. Tästä hyvänä esimerkkinä voi pitää Microsoftin DOS- ja Windows-käyttöjärjestelmiä. Kummatkin ovat suljetun lähdekoodin ohjelmistoja ja lähdekoodia tai oikeastaan niiden ideoita ja koodaustapoja joita puolustetaan oikeudessa. Niiden lähdekoodit ovat siis Microsoftin omaisuutta, eikä niitä anneta kuin harvoille yhteistyökumppaneille ja silloinkin tiukkojen sopimusten kanssa.

3.1.1 Käännettävä lähdekoodi

Kun lähdekoodi on kirjoitettu, on kaksi tapaa suorittaa se. Ensimmäinen tapa on kääntäminen. Tällöin lähdekoodi käsitellään kääntäjällä, joka nimensä mukaisesti kääntää sen konekieliseksi tiedostoksi. Mikäli tiedosto on tarkoitettu ajettavaksi, voidaan tämän tiedoston avulla käynnistää ohjelma. Käännettyjä tiedostoja ei kuitenkaan voida enää palauttaa lähdekoodiksi, vaan tiedoston muokkaus tapahtuu muokkaamalla lähdekoodi-tiedostoa ja tämän jälkeen tiedosto pitää kääntää uudelleen. Konekielinen tiedosto voidaan kuitenkin joissain määrin palauttaa lähdekoodiksi takaisinmallinnuksella, englanniksi termi Reverse Engineering. Tämä tapahtuu tutkimalla, miten ohjelma toimii suorituksensa aikana. Takaisinmallinnus ei kuitenkaan luo samanlaista lähdekoodia kuin mitä oli alkuperäinen lähdekoodi.

Ajatellaan vaikka aiemmin esitettyä elokuvaesimerkkiä. Kun elokuva on valmis, kirjoittaa satunnainen katsoja näkemänsä ylös paperille. Vaikka tämä olisi kuinka tarkka kertomus tahansa, se tuskin olisi sanasta sanaan sama kuin oli alkuperäinen käsikirjoitus. Eri ohjelmointikielet tarvitsevat myös oman kielensä kääntäjäohjelmiston.

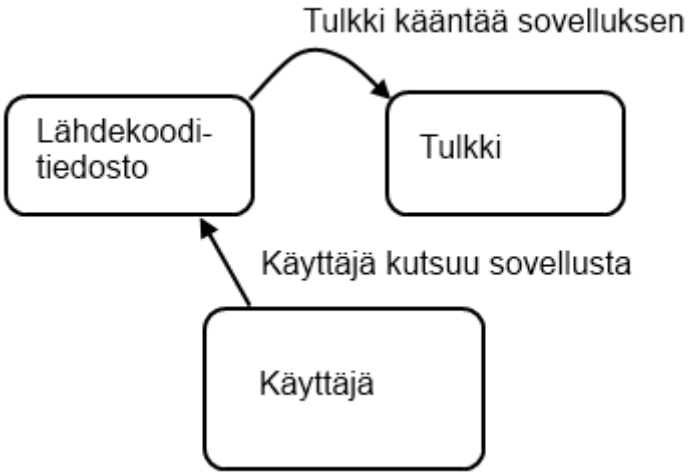


Kuva 1 Käännettävän lähdekoodin ja kääntäjän toiminta

3.1.2 Tulkattava lähdekoodi

Toinen tekniikka, jota käytetään lähdekoodin suorittamiseen, on tulkkaus. Tulkattavien kielien lähdekooditiedostot pysyvät koko elämänsä ajan selkokielisinä. Kun käyttäjä haluaa ajaa tällaisen tiedoston, hän käyttää siihen tulkkia. Tulkki on oma ohjelmansa, joka kääntää tarvittaessa lähdekooditiedoston konekieliseksi. Tästä on se etu, että ohjelmaan on helpompi tehdä muokkauksia, kun tiedostoa ei tarvitse erikseen kääntää. Tulkatut kielet eivät kuitenkaan ole aivan niin nopeita kuin käännetyt. Tämä johtuu käännösprosessista joka suoritetaan ajon aikana sekä joka kerta uudelleen, kun tiedosto halutaan ajaa.

Molemmilla tekniikoilla on hyvät ja huonot puolensa. Kääntäminen on hidasta, mutta saa aikaan nopeasti toimivan lopputuotteen. Sen virheiden korjaaminen ja uusien ominaisuuksien lisääminen sekä testaaminen on kuitenkin hidasta, koska lähdekoodin joutuu aina kääntämään uudelleen ajettavaksi ohjelmaksi. Mitä laajempi ohjelma on, sitä hitaampi prosessi on kyseessä.



Kuva 2 Tulkattavan koodin toiminta

3.2 Avoin lähdekoodi

Avoimen lähdekoodin juuret ovat löytyvät -60 ja -70-lukujen Yhdysvaltain yliopistomaailmasta. Tällöin Yhdysvaltain valtio kehotti tukemiaan yliopistoja kehittää tapoja hyödyntää hajautettuja järjestelmiä. (Gutschmidt 2003, 15.) Avoimella lähdekoodilla tarkoitetaan ohjelmistoja, joiden lähdekoodi on saatavilla muillekin kuin ohjelman luojalle. Tässä luvussa esitellään avoimen lähdekoodin avaintapahtumia. Nykyään useat tahot käyttävät avoimen lähdekoodin lähestymistapaa ohjelmistoihinsa. Tällä he saavat suuret joukot testaamaan ja muokkaamaan tuotettansa. Myös pienet harrastelijaprojektit saavat hyötyä avoimesta lähdekoodista. Kuitenkin aina käytettäessä avointa lähdekoodia tulee ottaa selvää lähdekoodin lisensseistä ja niiden rajoituksista.

3.2.1 GNU-projekti

Vuonna 1984 Richard Stallman perusti Gnu-projektin. GNU on rekursiivinen lyhenne sanoista GNU's not Unix eli GNU ei ole Unix. Gnu-projektin tarkoituksena oli tehdä ohjelmistoista vapaita ja avoimia. Stallmanin mielestä ohjelmistojen ja näiden lähdekoodin pitäisi olla vapaasti saatavilla, koska muuten harvalukuinen joukko vaikutusvaltaisia ihmisiä hallitsi tietojenkäsittelyä ja tietotekniikkaa. Stallman ei kuitenkaan tarkoittanut ohjelmistojen olevan ilmaisia vaan vapaita. Tämä aiheutti sekaannusta, koska Stallman käytti sanaa “free software”, jolla on englanniksi monta tarkoitusta. Suomeksi tällä ei ole sekoittavaa vaikutusta, koska vapaa ja ilmainen tarkoittavat erilaisia asioita.

Vuonna 1985 Stallman perusti Free Software Foundationin tukemaan GNU:n kehittämistä. Kun Gnu-projekti oli edennyt, Stallman tarvitsi jonkin tavan suojata luomuksensa väärinkäytöltä. Hän loi copyleftin periaatteet, koska public domain oli liian salliva julkaisumuoto ja tällöin GNU:ta olisi voinut kohdata liian helppo väärinkäyttö. Joku olisi vain voinut julkaista sen toisella nimellä tai väittää sen tekijäoikeuden kuuluvan hänelle. Myöskään tiukat tekijänoikeudet, copyright, eivät Stallmania kiinnostaneet, koska tämä olisi tuhonnut hänen avoimuuden periaatteensa. Niinpä Stallman loi GNU General Public License (GNU GPL) ja julkaisi GNU:n sen alaisuudessa. Ensimmäinen GNU GPL versio julkaistiin 1989, toinen 1991 ja kolmas 2007, joka on tällä hetkelläkin käytössä. Vuonna 1991 GNU:hun yhdistettiin Linus Torvaldsin luoman Linux-ydin, jolloin saivat alkunsa GNU-Linux -käyttöjärjestelmät. GNU-projekti kehittää myös omaa ydintään nimeltään HURD. Siitä ei kuitenkaan ole olemassa vakaata julkaisuversiota.

3.2.2 Berkeley Software Distribution

Berkeleyn yliopisto Kaliforniassa sai käyttöönsä UNIX-käyttöjärjestelmän vuonna 1974 AT&T:n omistamalta Bell Labsilta.. Seuraavien vuosien ajan tahot kehittivät yhteistyössä UNIXia. Vuoteen 1977 mennessä oli kuitenkin syntynyt kaksi erilaista kehityshaaraa: Bell Labsin UNIX ja Berkeley Software Distribution (BSD). BSD:tä jaettiin yliopistojen kesken ympäri maailmaa sillä ehdolla, että käyttävä taho ostaisi siihen käyttölisenssin AT&T:ltä.

Vuoden 1978 jälkeen AT&T halusi kuitenkin paremmin kaupallistaa UNIXin. Tämän johdosta Bell Labs ei enää ollut keskeisessä osassa UNIXin kehittämisessä. Tästä syystä perustettiin Berkeley Computer Systems Research Group (CSRG) joka pystyisi koordinoimaan ja kehittämää tulevia Unix julkaisuja. 1980-luvun alkupuolella CSRG lisäsi Unixiin ARPANet protokollan (TCP/IP). Alkujaan lisäys tapahtui BSD:hen, mutta lisättiin myöhemmin AT&T:n Unixiinkin. Koska AT&T lisenssimaksu oli 50 000 dollaria, oli se monille liian kallis. Tämän takia Berkeleytä pyydettiin julkaisemaan ARPANet protokollan koodi ja työkalut niin, ettei käyttäjän tarvitsisi maksaa AT&T:lle käyttölisenssistä. Koska nämä oli kokonaisuudessaan kehitetty Berkeleyssä tai BSD:n kehittäjien toimesta, niiden julkaisu oli mahdollista.

Vuonna 1989 julkaistiin Berkeleyn toimesta ensimmäinen vapaasti jaeltava lähdekoodi nimellä Networking Release 1. Networking Release 1. suosion huomattuaan eräs CSRG:n jäsen, Keith Bostic, alkoi miettiä, olisiko mahdollista julkaista laajempi kokoelma BSD:n koodia. Hänelle kuitenkin ilmoitettiin, että tämä vaatisi satojen tiedostojen uudelleen kirjoittamista, jotta nämä eivät sisältäisi AT&T:n omistamaa koodia. Bostic käytti tähän muutostyöhön uudenlaista tapaa. Käyttäen hyväksi ARPANetiä, hän sai joukon innokkaita ohjelmoijia kiinnostumaan ohjelmoimaan tarvittavat työkalut. Minkäänlaista korvausta ohjelmoijat eivät työstään saisi, ainoas-

taan heidän nimensä lisättäisiin siihen työkaluun, jonka parissa he tekivät töitä.
Tämä malli toimi hyvin ja alle kahdessa vuodessa suurin osa tarvittavista tiedostoista oli kirjoitettu uudelleen ja vuonna 1991 julkaistiin Networking Release 2. Se oli lähes valmis käyttöjärjestelmä. Vain muutama ytimen tiedosto puuttui. Bill Jolitz, CSRG:n jäsen hänkin, lisäsi Networking Release kahteen joitakin omia tiedostojaan ja julkaisi tämän nimellä 386BSD. Lähdekoodin hän laittoi kaikkien saataville FTP-palvelimelle.

Koska Jolitzilla ei ollut tapana lisätä muiden tekemiä korjauksia tai vaihtoehtoisesti omia korjauksiaan, joukko 386BSD:n käyttäjiä perusti NetBSD Groupin koordinoimaan järjestelmän kehitystä. Tätä järjestelmää kutsuttiin NetBSD:ksi. NetBSD:n tavoitteena oli saada BSD toimimaan mahdollisimman monella erilaisella alustalla. Tämä aiheutti sen, että järjestelmä ei ollut kovin vakaa. Toinen ryhmä kehitti samalla NetBSD:tä Intelin x86 arkkitehtuurin parissa. Tämä ryhmä kutsui työtään FreeBSD-projektiksi. Siitä tuli nopeasti suosituin BSD-jakelu, koska sen jakelualustana käytettiin Cd-rom-levyjä ja sen asennus oli helppoa. FreeBSD sisälsi myös Linux-emulointiohjelmiston, jolloin Linux-ohjelmia voitiin ajaa BSD:ssä. Tämä tarjosi valtavan kirjaston erilaisia ohjelmistoja ja työkaluja FreeBSD:n käyttäjille. Samaan aikaan kun 386BSD:tä kehitettiin, alkoi Berkeley Software Design inc. (BSDI), markkinoida kaupallista versiota BSD:stä. Alun perin tämä tunnettiin nimellä BSD/386, mutta vaihtui myöhemmin BSD/OS:ksi. Tämä mahdollistettiin kirjoittamalla ytimen tiedostot uudestaan. Kaupallistumisen johdosta AT&T:n tytäryhtiö, Unix System Laboratories, haastoi BSDI:n oikeuteen, koska tämä markkinoi tuotetta, joka miellettiin Unixiksi. Vuonna 1994 syyte sovittiin oikeuden ulkopuolella. Tämän johdosta BSD:n uusimman lähdekoodin jakelu sallittiin ja sen nimeksi tuli 4.4BSD-Lite. Sovittelun johdosta BSD:tä ei voitu enää myöskään kutsua Unixiksi.

3.2.3 Linux

Linuxin kehittämisen aloitti Linus Torvalds vuonna 1990 omiin tarpeisiinsa. Hän oli tyytymätön koulussa käyttämäänsä Minix-järjestelmän toimintaan ja muokattavuuteen. Tämän takia hän alkoi kirjoittaa omaa käyttöjärjestelmäänsä. Nimi Linux tulee sanoista Linus ja Unix. Hyvin pian ensimmäisten versioiden jälkeen Linux keräsi ympärilleen joukon innokkaita ohjelmoijia ja Linuxin kehitys lähti huimaan nousuun. Linux oli kevyt ja tarjosi ohjelmoijille haasteita ja vapautta mitä kilpailevat tuotteet eivät tehneet. Linuxin kehitys tapahtui suurelta osin hajautetusti verkossa. Kehittäjät lähettivät oman koodinsa Torvaldsille, joka piti Linuxin ytimen kehittämisen käsissään. Koska Linux oli vain käyttöjärjestelmän ydin, joka piti kaiken koossa ja välitti eri osien viestejä toisilleen, se käytti Richard Stallmanin Gnu-projektissa kehittämiä työkaluja ympärillään ollakseen täysiverinen käyttöjärjestelmä. Tämän takia suurin osa Linuxeista julkaistaan Gnu-osien vaatiman GP-lisenssin alaisuudessa ja tämän takia Stallman haluaa, että Linuxia kutsutaan nimellä GNU-Linux.

Ensimmäinen kaupallinen Linux-jakelu ilmestyi 1992. Se oli Kalifornialaisen Yggdrasil Computingin käsialaa. Nykyään suosittuja jakeluita ovat Debian, http://www.debian.org/, Ubuntu, http://www.ubuntu.com/, Fedora Project, http://fedoraproject.org/, Mint, http://linuxmint.com/ sekä OpenSUSE, http://www.opensuse.org/ (Distrowatch, 2011).

3.3 Avoimen ja vapaan lähdekoodin lisenssit

Tässä luvussa esitellään yleisimmät vapaan ja avoimen lähdekoodin lisenssit ja kerrotaan, mitä nämä lisenssit antavat tehdä ja mitä eivät. Lisensseillä suojataan lähdekoodin tekijäoikeudet.

3.3.1 General Public License

General Public License on yksi yleisimmistä vapaan lähdekoodin lisensseistä. Sen kirjoitti Richard Stallman vuonna 1989 omiin tarpeisiinsa Gnu-projektissa. Stallman päätyi kirjoittamaan oman lisenssinsä, koska tavanomainen tekijänoikeus oli hänen mielestään liian tiukka ja julkaisu Public Domainiin, yleiseen jakeluun, taas liian väljä, eikä olisi suojannut täten tarpeeksi. Samalla Stallman kehitti Copyleftin käsitteen. Copyleft tarkoittaa sitä että ohjelman lähdekoodi on vapaassa jakelussa ja siitä muokattujen ohjelmien lähdekoodin pitää olla myös vapaasti saatavilla (Copyleft, 2010). GPL:stä on olemassa kolme versiota, joista uusin GPL v3 julkaistiin vuonna 2007. Tämän uudelleenkirjoituksen tarkoituksena on ollut saada GPL vastaamaan kehitystä.

GPL v3 -lisenssin alaisuudessa julkaistu ohjelma takaa luojalle tekijänoikeuden. Se myös antaa vapaan ohjelman käyttäjälle luvan muokata ja levittää ohjelmaa eteenpäin. Tämä voidaan tehdä ilmaiseksi tai maksua vastaan. Levittäjän on kuitenkin sisällytettävä kopio lisenssistä edelleen levittämäänsä ohjelmaan ja toimitettava pyynnöstä lähdekoodi sitä haluaville, mikäli lähdekoodi ei ole mukana ohjelman levityspaketissa. Mikäli ohjelman muokkaaja haluaa levittää edelleen muokkaamaansa ohjelmaa, pitää hänen ilmoittaa, kuka on muokannut ohjelmaa ja milloin. Ohjelma pitää lisensoida samalla GPL-versiolla kuin se ohjelma, josta muokattu ohjelma on luotu. Tämä takaa vapauden jatkuvuuden.

GPL v3 määrää myös, että ohjelman julkaisija ei saa käyttää oikeudellisia keinoja estääkseen käyttäjiä kiertämästä ohjelman teknisiä suojakeinoja. GPL v3 antaa ohjelman käyttäjille rojaltivapaan oikeuden käyttää ohjelmassa käytettäviä alkuperäisiä patentteja. (GPL, 2011.)

GPL-lisenssistä on olemassa myös ohjelman muokkaajille sallivampi versio. Tämä tunnetaan nimellä Lesser GPL. LGPL lisää muutamia oikeuksia, joista tärkein on ohjelman osan julkaisu niin, ettei haluamansa osan lähdekoodia tarvitse julkaista. Sen osan lähdekoodi, joka on ohjelman alkupe-

räisen tai edellisen muokkaajan ohjelmoima, pitää julkaista,mutta se osa, joka uuden osan ohjelmoijan itsensä kirjoittama, ei tarvitse julkaista (LPGL, 2011). GPL:n idea on hyvä, mutta se rajoittaa kaupallista toimintaa estämällä suljettujen osien käytön yhdessä GPL-lisensoidun ohjelman osan kanssa. Kaupalliseen tuotantoon ohjelmia tekevät tahot haluavat yleensä suojata lähdekoodinsa ja täten GPL-lisensoidun osan hyötykäyttö kaupallisesti on rajattua. Tätä rajoittuneisuutta LPGL vähentää hieman. Se on kuitenkin alkuperäisen GPL -lisenssin idean ja ennen kaikkea vapaiden ohjelmistojen vastainen.

3.3.2 BSD-lisenssi

BSD lisenssi on yksi vapaimmista avoimen lähdekoodin lisensseistä. Yleisin käytössä oleva BSD-lisenssi on muokattu BSD-lisenssi. Tämä eroaa alkuperäisestä lisenssistä siten, että se ei vaadi ohjelman uudelleen julkaisijaa ilmoittamaan, että ohjelmassa käytetään Berkeleyssä kehitettyä koodia. Alun perin tämä oli lisätty lisenssiin sen takia, ettei kukaan käytä Kalifornian Berkeleyn nimeä mainostaessaan omaa tuotettaan.

1. Alkuperäinen BSD-lisenssi sisältää tiedon alkuperäisestä omistajasta (ohjelman luoja), vuosi jolloin alkuperäinen ohjelma on kirjoitettu sekä seuraavat kolme pykälää:

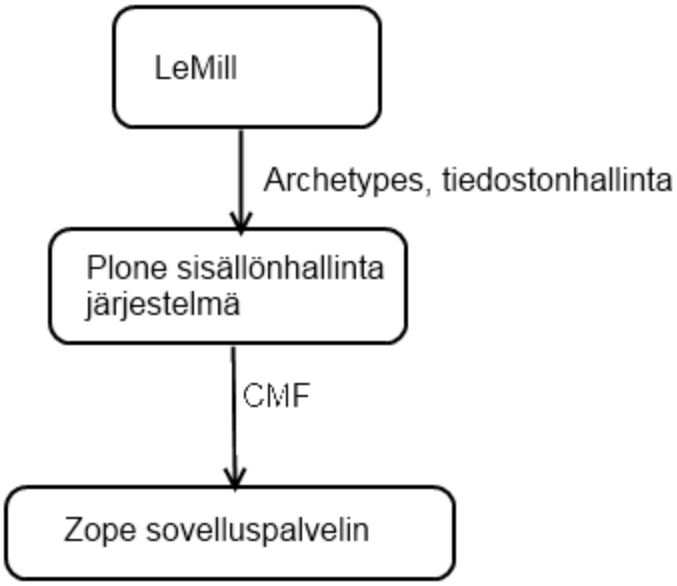
2. Lähdekoodin uudelleenjakelun täytyy sisältää yllä olevat tekijänoikeustiedot, lista näistä pykälistä sekä pykäliä seuraava vastuuvapauslauseke.
Binäärimuotoisen uudelleenjakelun tulee jäljentää yllä oleva tekijänoikeustiedot, lista näistä pykälistä ohjelman dokumentaatiossa tai muussa ohjelman mukana toimitettavassa materiaalissa.

3. Alkuperäisen ohjelman luojan organisaation nimeä tai ohjelmaan aikaisemmin jotain tuottaneiden tahojen nimeä ei saa käyttää tästä ohjelmasta jalostetun ohjelman markkinointiin ilman kirjallista lupaa asianomaisilta.

Alkuperäisen BSD-lisenssin voi käydä lukemassa osoitteessa http://www.opensource.org/licenses/bsd-license.php ja siitä löytyy tuo vastuuvapautuslauseke englanniksi. Muokattu BSD-lisenssi määrää, että uudelleen julkaisuun on sisällytettävä kopio lisenssistä, oli tämä sitten lähdekoodi-muotoinen tai konekielinen julkaisu. Alkuperäisen lisenssin kolmas, markkinointipykälä, poistettiin muokatusta lisenssistä. Näin se saatiin yhteensopivaksi MIT-lisenssin kanssa. (BSD lisenssi, 2011.) MIT-lisenssi on vapaa lisenssi, joka käytännössä tekee ohjelmasta public domain -ohjelman. BSD-lisenssi antaa käyttäjän tuottaa oman ohjelmiston lähdekoodin pohjalta. Tämä pätee myös omisteisiin ohjelmiin, eikä tällöin ohjelman lähdekoodia tarvitse jakaa. Tämä on suurin ero verrattuna GPL-lisenssiin. BSD-lisenssillä julkaistun ohjelman lähdekoodista voi myös luoda ohjelman, joka käyttää jotakin muuta lisenssiä kuin BSD-lisenssiä. Esimerkiksi Microsoft on käyttänyt TCP/IP koodia hyväkseen useassa Windows-käyttöjärjestelmässään. (Bretthauer, 2001.)

4 LEMILL-ALUSTAN OHJELMISTOYMPÄRISTÖ

LeMill ei toimi itsekseen vaan vaatii seurakseen useita eri ohjelmistoja sovelluspalvelimesta sisällönhallintajärjestelmään. Kuvassa 1. on esitetty tarkemmin tätä suhdetta. Tässä luvussa esitellään käytettävät ohjelmistot ja paneudutaan asioihin teknisemmältä kannalta.



Kuva 3 LeMillin yhteys Plonen kautta Zope-sovelluspalvelimeen. Nuolten vieressä tekstit kertovat osia, joissa ylemmän tason ohjelma käyttää alemman tason ohjelman osia.

4.1 LeMill-alustan rakenne ja asennus

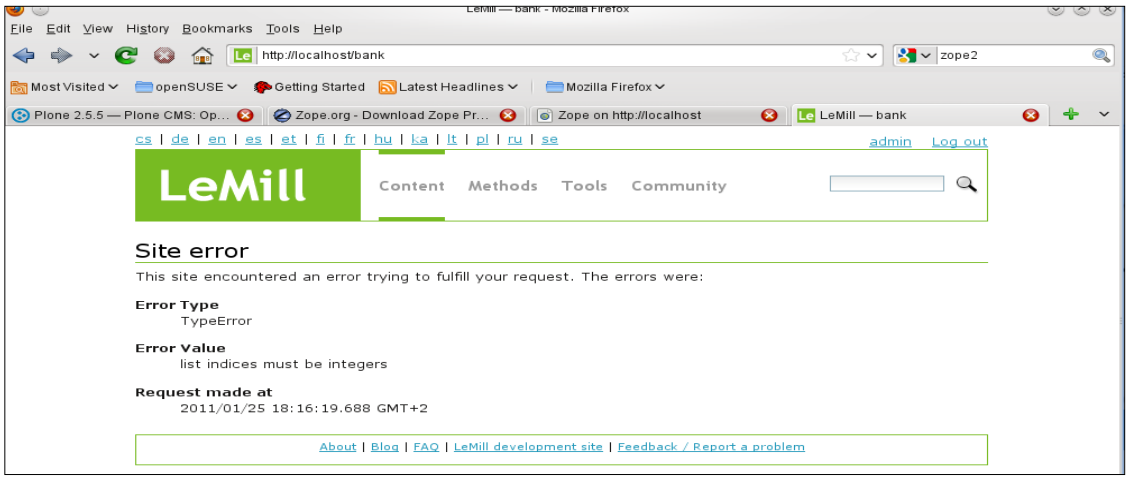
LeMill koostuu neljästä pääosiosta. Niitä ovat sisältö, metodit, työkalut ja yhteisöt. Sisältö sisältää verkkosivustoja, tehtäväsivustoja, luentosuunnitelmia ja PILOT-videoita. PILOT-videot ovat flash-videoita, joihin LeMill tarjoaa yksinkertaisen luontieditorin. Tässä editorissa oli ainakin devbankin testiversiossa jonkinlaista toimimattomuutta. Siinä pystyy tekemään videon, mutta editointi ei toimi. Tämän takia videoiden teko piilotettiin testialustalta.

LeMillin kaksi seuraavaa pääosiota metodit ja työkalut sisältävät esimerkkejä erilaisista opetus- oppimismenetelmistä sekä esittelyitä erilaisista oppimiseen käytetyistä työkaluista. Esimerkiksi työkaluissa on esiteltynä Wikipedia.com sivusto. HAMKin käytössä nämä ovat luultavimmin turhia, koska metodeihin ja työkaluihin voi tehdä ainoastaan yksinkertaisia verkkosivuja ja nämä voi samalla vaivalla tehdä sisältöön. Miksi esitellä esimerkiksi Photoshop-kuvankäsittelyohjelmisto työkaluissa, kun sille voi

tehdä oman sivun sisältöön ja tänne laittaa ohjeita sen käyttöön. Työkalut ja metodit osioiden luomat sivut sisältävät vähemmän elementtejä kuin sisällössä luodut sivut. Esimerkiksi kuvia ei voi työkaluihin ja metodeihin laittaa. Niiden metatieto on myös rajoittuneempaa. Rakenteellisesti työkalujen ja metodien erittely muusta sisällöstä on kuitenkin ymmärrettävää.

Viimeisenä osiona ovat yhteisöt. Yhteisöissä voi luoda erilaisia ryhmiä ja näille ryhmille voi antaa oikeuden muokata materiaalia. Tämä toteutuu siten, että materiaalia luotaessa luoja voi määrittää ryhmät, joilla on oikeus muokata tätä sisältöä. Näin esimerkiksi joukko opettajia voi yhteisesti muokata jotain tiettyä aineistoa. Yhteisöissä voi myös käydä keskusteluita ryhmän sisällä. Syksyllä 2010 Tarmo Toikkanen ja Jukka Purma kävivät esittelemässä LeMill-alustaa HAMKissa ja he kertoivat, että yhteisöjä ei enää kauheasti käytetä. Tämä johtuu siitä, että keskusteluihin käytetään muita siihen paremmin soveltuvia alustoja, kuten Facebook, videokonferenssit tai muut pikaviestimet. Näin LeMill yhteisöt ovat jääneet kehityksen ulkopuolelle kehitysresurssien suuntautuessa käytettyjen ominaisuuksien kehitykseen. LeMillissä on myös mahdollisuus tehdä kokoelmia. Näihin kokoelmiin voi sisällyttää kaikkea jo luotua sisältöä. Tällä tavalla saadaan suurempia kokonaisuuksia hallittua helpommin. Kokoelmista voi myös luoda kokoelmia.

Työssä käytetään LeMill versioita 2.11 ja 3.0. LeMillin asennus ja käyttöönotto on monivaiheinen projekti. Ensin pitää ladata lemill-paketti http://lemill.org/trac/wiki/Download-sivustolta. Koska HAMKin palvelimella sijaitseva versio on niin sanottu kehitysversio (numeroltaan 2.11), pitää se kaivaa hieman syvemmältä. Esimerkiksi käy kuitenkin Download sivulla oleva uusin julkaisu. Paketin voi ladata zip- tai tar.gz-muotoisena pakettina. Tämä pitää latauksen jälkeen purkaa jollain pakettienhallintaohjelmalla. Paketissa ollut LeMill-kansio siirretään Zopen asennuskansiossa olevaan Products-kansioon. Tämän jälkeen Zope pitää käynnistää uudelleen. Seuraavaksi avataan Zopen hallintapaneeli ja oikealta ylhäältä valitaan alasvetovalikosta Plone Site ja painetaan Add. Sivulle annettava id on se alihakemisto, johon palvelimella otetaan yhteys, kun halutaan luodulle sivustolle. HAMKin testisivusto on nimetty “bank” ja tämä laitetaan virtuaalipalvelimellekin. Title näkyy selaimen yläreunassa ja sen voi muokata haluamakseen. Sivustolle voi myös antaa kuvauksen. Lopuksi vielä valitaan **Extension Profiles** valikosta **LeMill Site**. Lopuksi painetaan **Add Plone Site**. Tämän jälkeen voi selaimella mennä osoitteeseen http://palvelimen-ip/bank. Tämän voi sitten halutessa Zopessa virtual host monsterilla tai vaikka Apachen avulla aliaksilla muokata niin, että ulkoa tuleva käyttäjä selaa osoitteeseen bank.hamk.fi ja hän päätyy juuri luodulle sivulle.



Kuva 4 Puhdas LeMill asennus. Kuvan ilmoittama virhe johtuu käytetyn testiversion outoudesta. Niin kauan kuin ei ole luonut sisältöä ja laittanut sisällölle kansikuvaa, ilmoittaa LeMill virheestä.

LeMill on ohjelmoitu Python-ohjelmointikielellä ja tämän takia sen lähdekoodin muokkaaminen on helppoa, koska lähdekoodia ei tarvitse erikseen hakea mistään, vaan kaikki tarvittava tulee mukana. LeMilliä voi muokata kahdella tavalla. Ensimmäinen ja helpompi tapa on käyttää Zope-hallintapaneelia. Tämän avulla voi saa Zopen omaan tekstinmuokkaustilaan LeMillin tiedostot. Tämä on nopea tapa kokeilla muutoksia. Tämän tavan huonoin puoli on se, että tehdyt muutokset eivät tallennu alkuperäisiin lähdekooditiedostoihin vaan väliaikaisiin Zopen tiedostoihin. Tästä aiheutuu se, että mikäli LeMill halutaan siirtää toiselle palvelimelle, eivät tehdyt muutokset siirry mukana. Toinen, suositeltava, tapa on mennä LeMillin asennuskansioon ja täällä avata haluamaansa tekstieditoriin muokattavat tiedostot. Tämä on hieman työläämpi tapa, mutta samalla varmistetaan se, että muutokset siirtyvät tarvittaessa.

4.2 Plone-sisällönhallintajärjestelmä

Plone sisällönhallintajärjestelmä on CMF:n, Content Management Framework, päälle rakennettu sisällönhallintajärjestelmä. Plone luotiin alun perin helpottamaan CMF:n käyttöä vuonna 2000 Alexander Limin ja Alan Runyanin toimesta. Sittemmin siitä on kasvanut itsenäinen sisällönhallintajärjestelmä. Toukokuussa 2004 perustettiin Plone Foundation suojelemaan sekä edistämään Plonen tunnettavuutta. Plonea käyttää verkkosivustojensa luomiseen ja ylläpitämiseen ainakin Nokian QT Software, Amnesty International, Discover Magazine, Brasilian hallitus sekä yliopistot, kuten Harvard ja MIT. (Plone, 2011.)

Plonella voi tehdä tavallisia verkkosivustoja, mutta monet yritykset luovat erilaisia lisäosia Ploneen. Näillä Plonen toimintaa voi laajentaa huomattavasti. Plonen uusin versio, kirjoitusaikaan talvella 2011, on versio 4. Tässä työssä käytetään kuitenkin vanhempaa 2.5.5 versiota Plonesta, koska Le-

Mill-alusta ei tue uudempia Plonen versioita. Tämä johtuu Ploneen tehdyistä muutoksista ja niiden tuomista tarpeista muokata LeMill-alustaa. Näihin muutoksiin käytettävä aika olisi pois LeMillin omasta kehityksestä. Pikaisen testauksen pohjalta Plonella pystyy tekemään helposti erilaisia verkkosivustoja.

4.3 Zope-sovelluspalvelin ja sen käyttö

Luku esittelee Zope-sovelluspalvelimen sekä kertoo kuinka sen saa asennettua yhdessä Plone-sisällönhallintajärjestelmän kanssa. Lopuksi kerrotaan, kuinka PloneLDAP-pluginin saa asennettua ja konfiguroitua.

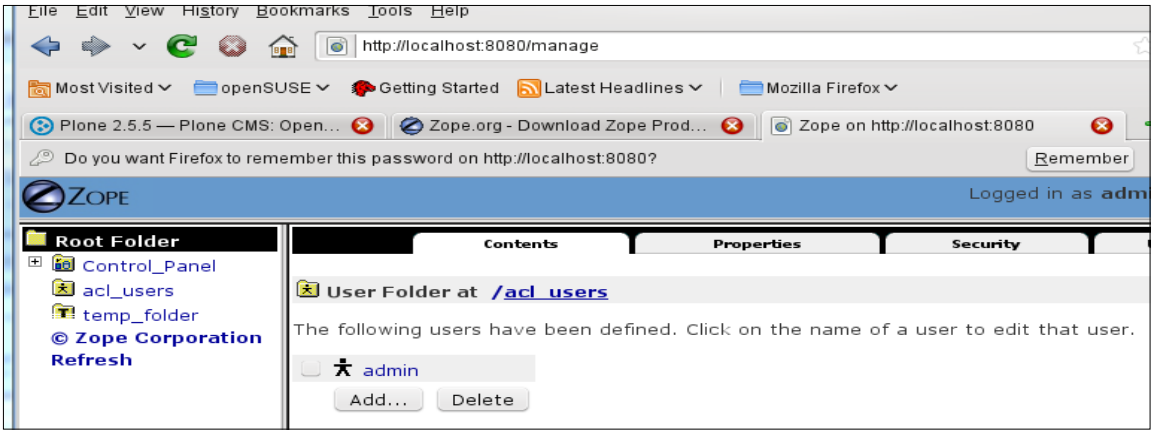
4.3.1 Zope-sovelluspalvelin

Zope sovelluspalvelin on Zope Corporationin kehittämä sovelluspalvelinohjelmisto. Digital Creations, Zope Corporationin tuolloinen nimi, julkaisi 1996 kolme avoimen lähdekoodin ohjelmistoa helpottamaan verkkojulkaisua: Bobo, Document Template ja BopoPOS. Näiden ympärille Digital Creations kehitti kaupallisen, suljetun lähdekoodin, sovelluspalvelimen nimeltä Principia. Vuonna 1998 sijoittaja nimeltä Hadar Pedhazun sai Digital Creationin julkaisemaan Principian avoimen lähdekoodin ohjelmistona.

Zope Corporation kehittää edelleen Zopea, mutta suurilta osin kehitys tapahtuu yhteisöpohjalta, yhteisön jäsenet lähettävät päivityksensä virheiden kokoajille. Zope on ohjelmoitu avoimen lähdekoodin Python-ohjelmointikielellä. Zopesta on olemassa kaksi versiota. Zope 2 on tällä hetkellä versionumerossa 2.11.4, joka on julkaistu 6.8.2009. Zope 2 on yleisin käytössä oleva Zopen versio. Tämä siksi, koska Plone sisällönhallintajärjestelmä käyttää Zope 2 -palvelinohjelmistoa. Zope 3 on uuden sukupolven versio Zopesta ja on helpottaa erityisesti muokattavuutta. Zope 3:n uusin vakaa versio on 3.4.0. Sen kehittäminen aloitettiin vuonna 2001 ja se jatkuu edelleen. Vuonna 2010 Zope 3 nimeksi vaihdettiin BlueBream, koska kahden yhteensopimattoman Zope version olemassaolo koettiin sekavaksi. Zopen kumpaakin versiota kehitetään yhtä aikaa, mutta suurelta osin uutta tekniikkaa kehitetään Zope 3 -ohjelmistoon. Joitakin Zope 3:n uusia tekniikoita voi kuitenkin käyttää yhdessä Zope 2:n kanssa, mutta se vaatii erillisen moduulin toimiakseen. Zope käyttää omaa oliopohjaista tietokantaansa, ZODB Zope's Object Database, mutta siihen saa helposti integroitua relaatiotietokantoja, kuten Oracle tai MySQL. Zopen sisällöntuotantoa helpottamaan kehitettiin Content Management Framework. Se on kokoelma useasta Zope-tuoteesta. Näitä ovat CMFCore, CMFDefault, DCWorkflow, CMFCalendar ja CMFTopic. Nämä ohjelmat puolestaan tarjoavat joukon erilaisia sisältötyyppejä (ContentTypes) sekä työkaluja (CMFTools).

4.3.2 Zopen asennus ja hallinta

Zopen ja aikaisemmin esitellyn Plone-sisällönhallintajärjestelmän asennus on tehty yksinkertaiseksi. Plonen sivuilta pystyy lataamaan yhdistetyn asennuspaketin niin Windows-käyttöjärjestelmille, Mac os/x:lle kuin Linuxille. Yhdistetty ei kuitenkaan tässä tapauksessa tarkoita sitä, että kaikki käyttöjärjestelmäversiot olisivat samassa paketissa vaan jokaiselle on oma pakettinsa. Windows-versio on ajettava tiedosto, jonka suorittamalla käynnistyy asennusohjelmisto. Linux-versio tulee paketoituna .tgz-muotoon. Tämän saa purettua komentokehotteessa tai terminaalissa komennolla **tar -xvf tiedosto.tgz**. Tiedoston voi myös purkaa graafisen käyttöliittymän pakettienhallintaohjelmistolla. Asennus aloitetaan Windowsissa suorittamalla asennusohjelma ja Linuxissa komennolla **sh install.sh**. Asennuksen loputtua palvelimen voi käynnistää. Tämän jälkeen palvelimeen saa yhteyden menemällä selaimella osoitteeseen http://palvelimen-ip:8080/ Zopen hallintapaneeliin pääsee käsiksi osoitteessa http://**palvelimen-ip:8080/manage** ja sisään voi kirjautua asennuksen ilmoittamilla tunnuksilla. Hallintapaneelista kannattaa mahdollisimman pian muuttaa valmiiksi luodun pääkäyttäjän tunnus

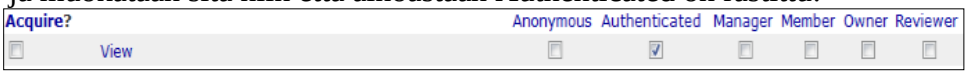


Zopen hallintapaneeli heti asennuksen jälkeen. Sovelluksia tai sivustoja ei ole vielä luotu.

4.3.3 PloneLDAP

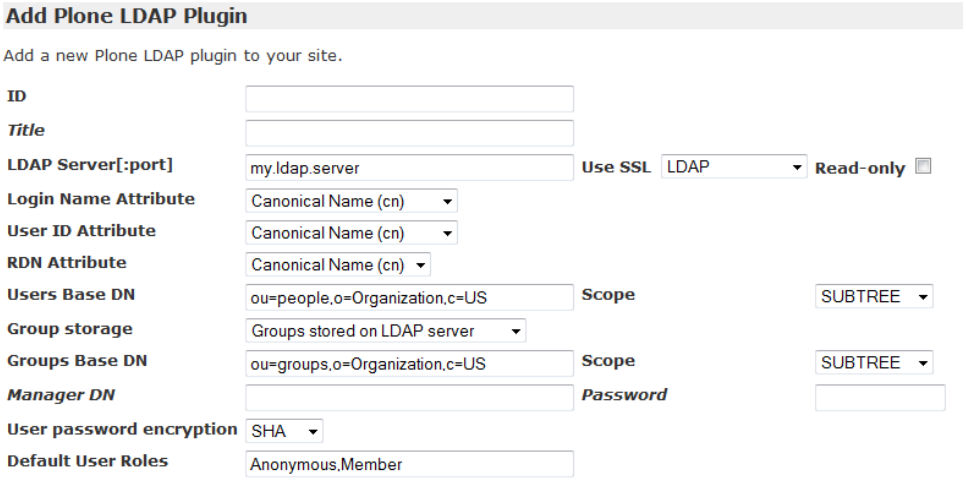
PloneLDAP-pluginia tarvitaan käyttäjien tunnistamiseen. Ilman tätä pluginia jouduttaisiin Zopeen luomaan kaikki käyttäjät ja määrittämään näille jonkinlainen rooli. PloneLDAP-pluginin avulla voidaan käyttää tunnistamisessa hyödyksi jo olemassa olevia tunnuksia. Ennen asennusta tulee huomioida, että mikäli LDAP-puusta löytyy samanniminen käyttäjä kuin Zopen hallinnasta, aiheuttaa tämä erittäin pahaa hidastelua. Esimerkiksi yleinen Admin-tunnus luultavimmin löytyy kummastakin ja jos LDAP:n asennuksen jälkeen käyttää tuota tunnusta yrittäessään Zopen hallinnoimille sivuille, hidastuu sivustojen toiminta merkittävästi.

Asennus aloitetaan hakemalla PloneLDAP-paketti osoitteesta http://plone.org/products/ploneldap/releases/1.0/PloneLDAP-bundle-1.0.tar.gz
Tämä paketti puretaan Zopen Products kansioon. Purkaminen luo kolme uutta kansiota: PloneLDAP, LDAPUserFolder ja LDAPMultiPlugins. Käynnistetään Zope nyt uudelleen. Seuraavaksi mennään selaimella Zopen hallintapaneeliin. Se löytyy osoitteesta **palvelimen_ip:8080/manage**. Täällä valitaan luotu sivusto. Tässä esimerkissä se on bank. Nyt valitaan yläreunassa keskellä oleva Security-välilehti. Täällä etsitään valinta **view** ja muokataan sitä niin että ainoastaan Authenticated on rastittu:



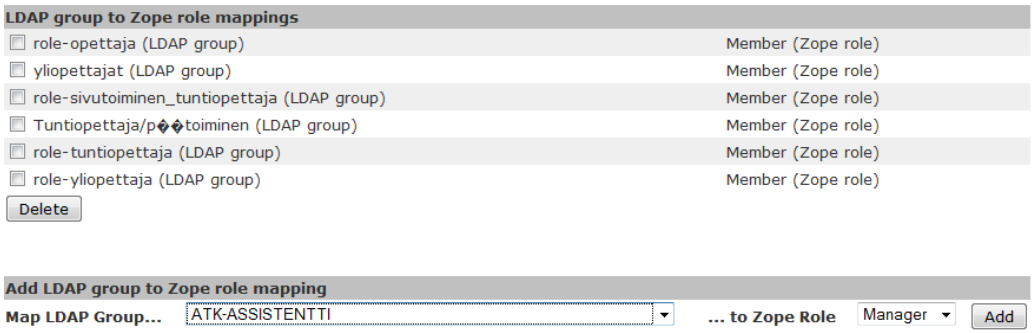
Seuraavaksi poistetaan keksien avulla tapahtuva tunnistaminen. Avataan ensin bankin paneelista acl_users. Täältä plugins ja Extraction Plugins. Poistetaan **credentials_cookie_auth** Active plugineista, jolloin aktiiviseksi jää ainoastaan **credentials_basic_auth**. Nyt mennään takaisin plugins-hakemistoon ja tehdään samalla tavalla lisäosille **Challenge Plugins**, **Reset Credentials Plugins** ja **Update Credentials Plugins**.

Tämän jälkeen mennään takaisin bank-sivun acl_users-kansioon. Oikean puolen alasvetovalikosta valitaan Plone LDAP plugin ja painetaan Add. Nyt auenneeseen sivuun syötetään tarvittavat tiedot. Nämä tiedot saa organisaation verkkohallinnoijilta ja ne on jätetty pois tästä tietoturvasyistä.



Kuva 5 PloneLDAP:n konfigurointi

Nyt voidaan asettaa käyttäjille rooleja. Kansiossa /bank/acl_users/LDAP/acl_users valitaan Groups välilehti.



Kuva 6 HAMKin devbankissa mapattuja ryhmiä.

Seuraavaksi mennään bank/acl_users ja klikataan LDAP-valintaa. Muokataan tämä siten, että aktivoidaan kaikki muut paitsi User_Adder. Tämän jälkeen tarkistetaan, että jokaisessa toiminnassa ovat lisäosat oikeassa järjestyksessä. Tämä suoritetaan avaamalla jokainen kohta ja tarkistamalla, että lisäosat ovat Active-kohdassa seuraavissa järjestyksissä:
Authentication → LDAP
Reset Credentials → credentials_basic_auth, LDAP
Group_Enumeration → LDAP, source_groups
Group_Introspection → source_groups, LDAP
Group_Management → source_groups, LDAP
Groups –> LDAP, source_groups
Properties → LDAP, mutable_properties
Role_Enumeration → LDAP
Roles → LDAP, local_roles
User_Enumeration → LDAP, source_users
User_Management → source_users, LDAP

Seuraavaksi mennään bank/acl_user/source_groups-kansioon ja valitaan active-välilehti. Varmistetaan että kaikki toiminnot ovat valittuna ja painetaan Update. Sitten mennään kansioon /bank/acl_users/portal_role_manager. Täältä valitaan myös Active-välilehti ja tarkastetaan, ettei mitään ole valittuna. Tämän jälkeen painetaan Update. Tarkista vielä /bank/acl_users/source_users-kansio. Tarkistetaan, että **User_Adder**, **User_Enumeration** **User_Introspection** ja **User_Management** ovat valittuna ja painetaan Update.

4.4 Ohjelmointikielet

Työn kaikki ohjelmat käyttävät tai on ohjelmoitu käyttämällä Python-ohjelmointikieltä. Tämä esitellään tässä luvussa. Luku esittelee myös Zopeen luodut scriptauskielet.

4.4.1 Python

Python kielen kehitti Guido von Rossum vuonna 1990. Se nimettiin Monty Pythonin Lentävä Sirkus tv-ohjelman mukaan. Nykyään sitä kehittää iso joukko vapaaehtoisia ohjelmoijia ympäri maailmaa ja kehitystä valvoo Python Software Foundation. (Lukaszewski, 2010.) Python on avoimen lähdekoodin ohjelmointikieli. Tämän ansiosta sen kehitys on joustavaa ja siihen voi itse tehdä muutoksia tarvittaessa. Python on tulkattava ohjelmointikieli. Tämä tarkoittaa sitä, että Python koodia ei erikseen tarvitse muuntaa konekieliseksi tiedostoksi, vaan käännös tapahtuu Python-tulkin avulla ohjelmaa ajettaessa. Tästä on se etu, että lähdekoodia on nopea muokata ja testata toteutusympäristössä, koska kirjoitettu ja ajettava tiedosto ovat samanlainen. Pythoniin on kuitenkin olemassa myös kääntäjiä, joilla lähdekooditiedosto voidaan kääntää konekielisiksi tiedostoiksi eri ympäristöihin. Pythonin käyttöä puoltaa myös se, että se on ”kirjoita kerran, käytä missä tahansa” -kieli. Tämä tarkoittaa sitä, että Python-koodi ei ole laitteisto- tai käyttöjärjestelmäriippuvainen vaan sama lähdekoodi toimii samalla tavalla eri järjestelmissä. Järjestelmissä pitää kuitenkin tällöin olla asennettuna järjestelmään luotu Python-kirjasto ja tulkki.

4.4.2 TAL, TALES ja METAL

Tag Attribute Language (TAL) on Zopen sivupohjien käyttämä ominaisuuskieli, jolla sivun sisältöä voidaan muokata, toistaa tai poistaa. (The Zope Book, 27) TAL-kielellä esimerkiksi voidaan joku sivukohta olla näyttämättä tietyille käyttäjille tai muuttaa näytettävän kohdan sisältö käyttäjäkohtaisesti. Template Attribute Language Expression Syntax (TALES) kieltä käytetään syöttämään tietoa TAL ja METAL kielille. TALES-kielelle voidaan antaa lauseita, joita se käyttää muuttujissa ja näistä muuttujia voidaan käyttää muualla. TALES-kieltä voidaan käyttää myös ilman TAL- tai METAL-kieliä. The Macro Expansion Template Attribute Language (METAL) on makrojen luontikieli. Sitä voidaan käyttää yhdessä TAL tai TALES kielien kanssa tai yksinään. Makroilla voidaan jakaa jokin ominaisuus monen eri mallipohjan kanssa niin, että makro tarvitsee kirjoittaa vain yhdelle sivulle ja muissa sivuissa vain viitataan tähän makroon, jolloin se on käytössä tälläkin sivulla.

5 HAASTATTELUT JA NIIDEN TULOKSET

Tämä luku on tehty yhteistyössä Tero Mansikan kanssa. Mansikka teki työtään LeMillin ja Moodlen käytöstä ja hänen työnsä nimi on ”LeMillin käyttö oppimis- ja opetustyökaluna sekä yhteistyö Moodlen kanssa”. Mansikka teki työhönsä Webropol-kyselyn ja sen tuloksia käytetään hyväksi tässä työssä. HAMKin LeMill-alustaa kutsutaan työssä tästä eteenpäin nimellä HaMill.

5.1 Haastattelun rakenne

Haastattelu koostui LeMill-alustan esittelystä ja samalla esitetyistä kysymyksistä haastateltaville. Tilanteessa käytettiin ennen haastatteluita tehtyä kysymyslomaketta, johon oli listattu 10 kysymystä. Haastateltavien vastaukset kirjattiin ylös vapaaseen muotoon ja kerättiin yhteen, kun viimeinen haastattelu oli ohitse. Haastattelun avulla pyrittiin saamaan laadullisia sekä määrällisiä vastauksia. Pyyntöjä haastatteluun esitettiin 22 kpl ja neljä suostui pyyntöön. Aluksi käytiin läpi Lemillin toiminta ja esiteltiin sivut. Samalla haastateltavilta kysyttiin mielipiteitä ennalta mietittyihin kysymyksiin. Kaksi ensimmäistä haastattelua olivat tietojenkäsittelyn opettajille ja nämä toimivat samalla harjoitteena muiden alojen opettajia varten. Näissä kahdessa ensimmäisessä ei ollut vielä kunnollista suunnitelmaa, vaikkakin kysymyksiä oli mietitty valmiiksi. Taulukossa 1 on esiteltynä haastatellut henkilöt sekä heidän toimenkuvansa HAMKin organisaatiosta.

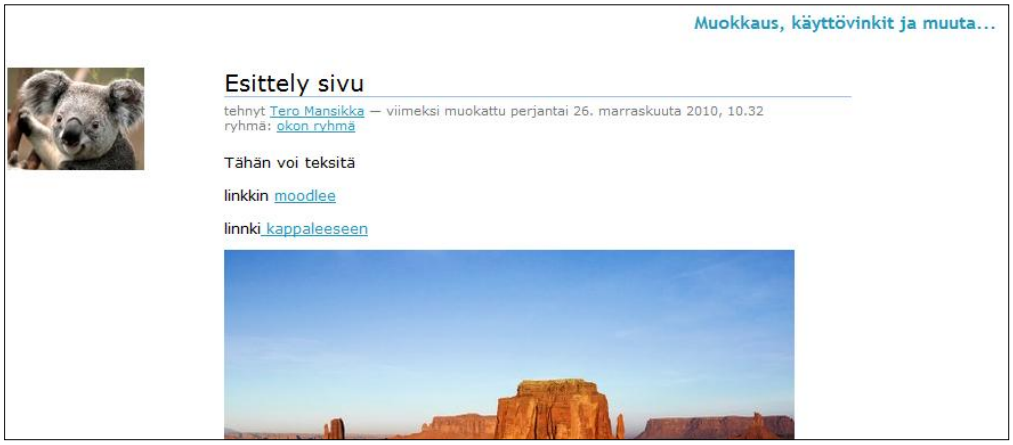
Lasse Seppänen	Yliopettaja, Tietojenkäsittely
Erkki Laine	Päätoiminen tuntiopettaja, Tietojenkäsittelyn ko
Jaana Nuuttila	Lehtori, Maaseutuelinkeinojen ko
Saija Honkala	Lehtori, Ohjaustoiminnan ko

Taulukko 1 Haastatellut opettajat

5.2 Haastatteluiden tulokset

Vaikkakin kaksi ensimmäistä haastattelua eivät sisältäneet ennalta suunniteltuja kysymyksiä vaan esiin nostettiin asioita LeMill-alustan esittelyn yhteydessä, niistä saadut tulokset eivät juurikaan eroa kysymyslistan kanssa tehdyistä. Seppänen nosti esiin materiaalin siirtämisen Data-palvelimelta LeMill-alustalle. Tähän ei kuitenkaan ole mitään nopeaa oikotietä, vaan sivut joutuu tekemään uudestaan LeMilliin. Seppäseltä ja Laineelta kysyttiin myös, tulisiko alustalla olevien linkkien johtaa oppilasnäkymään vai muokkausnäkymään. Kummankin mielestä muokkausnäkymä olisi loogisempi vaihtoehto, koska sitä luultavimmin käytetään enemmän. Muualta tulevan liikenteen voisi sitten linkittää suoraan oppilasnäkymään.

Kuva 7 Esimerkki muokkausnäkyvästä.



Kuva 8 Sama sivu kuin edellisessä kuvassa, mutta nyt oppilasnäkymästä. Huomioitavaa valikoiden puuttuminen. Yläoikealla linkki muokkausnäkymään.

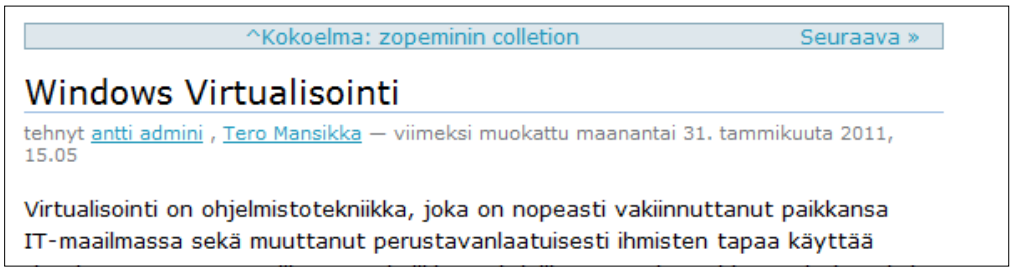
Yllä olevista esimerkkikuvista käy ilmi kahden eri näkymän erot. Oppilasnäkymä on tarkoituksella yksikertainen ja erittäin hillitty. Tällä on tarkoituksena pitää materiaalin käyttäminen selkeänä. Ainoa ero näiden sivujen osoitteessa on muokkausnäkymän osoitteen perään lisätty /view. Tämä on hieman epäloogista, koska jos opettaja haluaa lisätä esimerkiksi linkin Moodleen tekemäänsä materiaaliin, hän luultavasti kopioi sivun osoitteen selaimen osoitepalkista ja liittää sen linkiksi Moodleen. Nyt tämä linkki kuitenkin vie käyttäjän muokkausnäkymään.

Kun LeMillissä julkaisee aineistoa, pitää sille antaa jonkin näköinen kansikuva. Tämän kansikuvapakollisuuden poistoa puolsivat kaikki haastatellut henkilöt. Esimerkkeinä tuli se, että luultavasti moni opettaja, joka aineistoa tekisi ja julkaisisi, laittaisi moneen aineistoon saman kansikuvan. Tämä seuraisi sekaannusta. Kansikuvilla voisi olla käyttöä esimerkiksi siten, että tietyn kurssin tai opintolinjan kursseilla olisi sama kansikuva.

LeMillin kaksi aikaisemmin mainittua osiota, metodit ja työkalut, ihmetyttivät haastateltuja. Niille ei löytynyt lisäarvoa, mutta ei minkään näköistä haittaakaan niistä koidu. Ne jätetään tämän takia paikoilleen. Yhteisöille löydettiin käyttöä keskusteluista ja sisältöä muokkaavien ryhmien hallinnasta. Haastateltavat haluaisivat, että opettaja voisi määritellä jonkin oppilasryhmän, jolla olisi oikeus muokata jotakin tiettyä aineistoa. Tämä tulee olemaan hankalaa, koska HaMill käyttää käyttäjien tunnistamiseen erillistä Plonen lisäosaa PloneLDAP-bundle-1.0. Lisäosa on avoimen lähdekoodin ohjelmisto ja ohjelmoitu Python-ohjelmointikielellä. Tämän lisäosan ansiosta HaMill-käyttäjät voivat kirjautua HaMilliin HAMKin omilla tunnuksilla, mutta sen avulla voi antaa muutaman erillisen roolin käyttäjille. Näitä ovat manager ja member. Lisäosan avulla voi määrittää, mikäli käyttäjä on tunnistettu, eli jos käyttäjällä oli hyväksytyt tunnukset, hän saa nähdä HaMillin sisältöä. Tämän roolijaon perusteella HaMilliin on mahdollista luoda oppilaille mahdollisuus muokata resursseja.

Nimeämiskäytännöistä haastatteluissa saatiin seuraavanlaisia ehdotuksia: Kokoelmilla olisi kurssin nimi ja tunnus, ehkä opettajan nimi. Itse materiaalilla olisi materiaalin luoneen opettajan nimi ja jokin tunnus. Kokoelmien selaamiseen kaikki haastatellut halusivat lisävalikon. Tällä hetkellä kokoelmissa on ainoastaan pääsivu, josta voi navigoida sisältösivuille. Näillä sivuilla on kokoelmien kautta tultaessa yläreunassa pieni valikko, josta pääsee kokoelman edelliselle ja seuraavalla sivulle sekä takaisin kokoelman pääsivulle.

Kuva 9 Kokoelman päävalikko



Kuva 10 Windows Virtualisointi nähtynä zopemin kokoelman kautta. Yläreunassa näkyy linkit kokoelman pääsivulle ja kokoelman seuraavaan materiaaliin.

Tiedostonhallinta on LeMillissä hieman omalaatuista. Alustalle voi ladata tiedostoja ja näitä voi sitten periaatteessa käyttää eri materiaaleissa. Tiedostot on kuitenkin ladattava alustalle yksi kerrallaan ja tiedostojen ylläpito hankaloituu. Tätä edistää myös se asia, että LeMill nimeää tiedoston uudelleen, mikäli alustalta löytyy jo samanniminen tiedosto. Tällöin tiedoston nimeen liitetään ainakin päivämäärä, jolloin se on palvelimelle ladattu. Tämän johdosta tiedostojen, varsinkin useasti muokattavien tekstidokumenttien ylläpito hankaloituu, koska muokattava tiedosto olisi hyvä poistaa palvelimelta samalla, kun sinne laittaa muokatun version tiedostosta. Muuten vanhat tiedostot jäävät kuormittamaan alustan tietokantaa. Lasse Seppänen ehdotti tähän ratkaisua. Opettajille tehtäisiin verkkojakokansio ja sinne scripti, joka loisi kansion sisällöstä sivun. Tämä sivu linkitettäisiin iFramena HaMillin materiaalisivuille. Haastateltavat pitivät tätä tapaa käytännöllisempänä vaihtoehtona tiedostojen hallintaan.

Kaikki haastateltavat haluaisivat oppilaille mahdollisuuden muokata materiaalia. Tällä saataisiin interaktiivisuutta oppilaiden, opettajien ja materiaalin välille. Oppilaat voisivat esimerkiksi tehdä jollekin kurssille materiaalia sekä korjata tai muokata jonkin kurssin materiaaleja.

Seuraavaksi Mansikan tekemästä kyselystä esiin nousseita asioita koskien HaMill-alustaa. Kysyttäessä käyttöliittymän ensivaikutelmia 45 % kysymykseen vastanneista piti käyttöliittymää sekavana. Osa näistä toivoi jonkinlaista selkeyttävää ohjetta. 57 % vastanneista oli tyytyväisiä HaMillin konfigurointimahdollisuuksiin tätä kysyttäessä. Kaikki vastanneet pitivät oppimateriaalin päivitystä tärkeänä. 71 % vastanneista olisi valmis tekemään päivityksiä oppimateriaaliin.

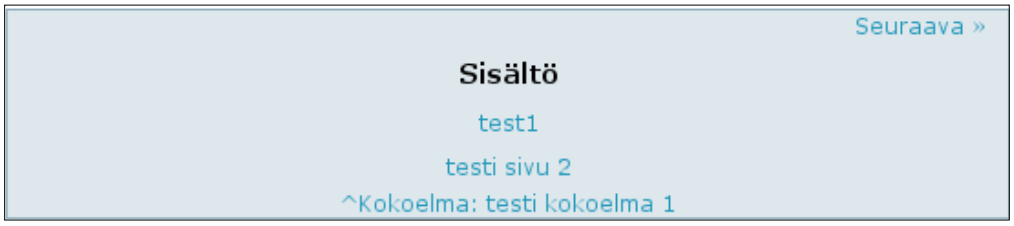
6 MUUTOKSIEN TOTEUTUS

Luvussa kerrotaan, kuinka mahdolliset muutokset voidaan toteuttaa sekä esitetään jatkokehitysmahdollisuuksia.

6.1 Toteutus

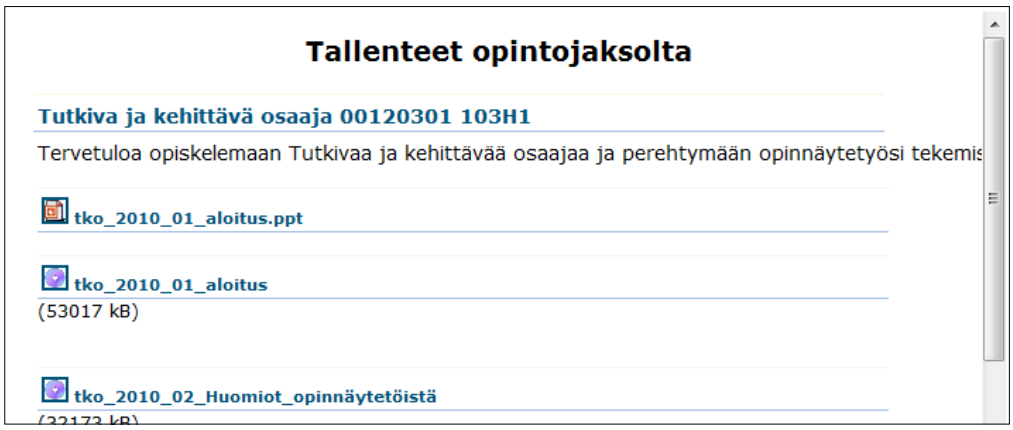
Testialustana oleva devbank.dev.hamk.fi:8080-palvelin oli jo aluksi hieman sekaisessa kunnossa. Tämä johtui siinä tehdystä testailuista ja käytännön kokeiluista. Tämän takia testiympäristöksi luotiin uusi virtuaalikone ja siihen siirrettiin LeMill devbank-palvelimelta. Täällä saisi huoletta tehdä ja kokeilla muutoksia ja niiden vaikutuksia alustaan. Asennuksen jälkeen tarvitsi korvata osa koodista Zopen hallintapaneelin kautta tehtyjen muutoksien takia. Tämä täytyy tehdä, koska jos LeMillin tiedostoja muokkaa Zopen hallintapaneelin avulla, ei tämä tee muutoksia alkuperäisiin tiedostoihin. Kun kaikki uudessa versiossa toimivat muokkaukset oli siirretty, pystyi muokkauksen aloittamaan.

Kansikuvan pakollisuus poistettiin muokkaamalla skins/lemill-kansiossa tiedostoa script_changeCoverImage.cpy. Täällä muokattiin kommenteiksi kaksi ensimmäistä ”if not” lauseketta. Haastateltavien haluaman listauksen kokoelmien sisällöstä sisältösivuille saa muokkaamalla luotua muokkaamalla collection_navigation.cpt-tiedostoa. Tämä tiedosto luo vanhan navigointipalkin, jossa on linkit edelliseen, seuraavaan ja kokoelman etusivulle. Tiedostoa muokkaamalla hakemisto tulostuu sivulle kaksi kertaa, kerran sivun alkuun ja kerran loppuun.



Kuva 11 Kokoelman sisältö navigointipalkissa

LeMilliin saa jo upotettua esimerkiksi Youtube-videoita. Tätä samaa ominaisuutta käyttämällä saa myös upotettua aiemmin mainitun tiedostojenlukuscriptin. Upotus-valinta luo sivulle iFrame-sisältöä. Tällöin scriptin ominaisuudet pitää kuitenkin itse syöttää. Tällä tavalla luodut tiedostolistaukset eivät kuitenkaan saa kuitenkaan olla 700 pikseliä leveitä. Jos näin kuitenkin käy, pitkistä tiedoston nimistä johtuen, ilmestyy tiedostonselausikkunaan sivuttainen vierityspalkki.



Scriptin luoma iFrame-esimerkki sisältösivulta.

6.2 Jatkokehitys

Ensimmäisenä jatkokehitys ideana on roolien jako. Tätä varten pitäisi muokata PloneLDAP-pluginia. Tällä hetkellä se toimii siten, että se lukee HAMKin LDAP-puusta ryhmiä ja näiden ryhmien perusteella jakaa oikeuksia. Eli opettajat saavat member-tason jäsenyyden, koska kuuluvat ennalta määrättyihin ryhmiin. Administraattorit saavat samalla tavalla roolinsa. Oppilaille ei saa asetettua roolia, koska PloneLDAP-plugi ei hyväksy Zopeen itse luotuja rooleja vaan siinä on nuo kaksi vakioroolia, jäsenet ja pääkäyttäjät. Tämä aiheuttaa sen, että yksittäisille oppilaille on mahdotonta antaa oikeuksia muokata sivustoa.

Kokoelmien sisällön listaus sisältösivuille tarvitsisi kokonaan oman erillisen navigointipalkkinsa. Tämän voisi toteuttaa siten, että tekisi uuden navigointitiedoston ja tänne laittaisi ainoastaan hakemistolistauksen. Tämän jälkeen täytyy löytää tiedosto, jonka avulla LeMill luo tuon kokoelmanäkymän tiedostoista ja sijoittaa sinne kutsu tuohon uuteen navigointitiedostoon ja suorittaa navigointipalkin asemointi paikalleen. Pitää myös miettiä, mitä sisältöä tuossa palkissa näyttää. Laitetaanko siihen sisältö, menetelmät, työkalut ja kokoelmat, joita kokoelma sisältää vai vain sisältö?

Sivuston ulkoasua pitää parantaa. Pitää myös miettiä, tehdäänkö sivustosta graafisesti yhteneväinen Oskarin, HAMKin oman portaalin, vai HAMKin käyttämän Moodlen kanssa. Tämän jälkeen muokkaus suoritetaan normaalina html- ja css-ohjelmointina.

Tiedostojen hallinnasta pitää myös huolehtia. Tiedostoscriptille pitää tehdä oma painikkeensa, joka määrittää ominaisuudet itsestään. Tällöin käyttäjän ei tarvitsisi kuin syöttää linkki oikeassa paikassa sijaitsevaan scriptitiedostoon. Pitää kuitenkin miettiä, miten tätä tullaan hallitsemaan. Onko käyttäjillä verkkojaossa joku kansio ja miten kansion sisällä jako-oikeudet periytyvät. Ideaalitilanteessa käyttäjä voi vain luoda uuden kansio ja tänne kopioida tiedostoscriptin.

Kaikki LeMillin tiedostot pitäisi myös dokumentoida järjestelmällisesti jatkokehitystä varten. Tämän voi toteuttaa esimerkiksi siten, että luetteloi kaikki LeMillin tiedostot ja jokaisen kohdalle sitten selvityksen siitä mitä tämä tiedosto tekee. Näin säästettäisiin, aikaa kun seuraavan kerran tarvitsee tehdä uusia muokkauksia.

7 YHTEENVETO

Työssä vastattiin sen esittämiin kysymyksiin. Työn aikana saatiin selville, minkälaisia ominaisuuksia HaMilliin pitäisi lisätä ja miten niitä siihen voidaan lisätä. Työhön saatiin myös lisäarvoa Mansikan tekemistä Webropol-kyselyistä. Haastateltavia olisi voinut olla enemmän kuin neljä ja tämän takia lisättäviä ominaisuuksia ei kovinkaan paljoa ole. Useampi haastateltava olisi myös selkeyttänyt eritoten sivujen nimeämiskäytäntöä. Myös tiedostojen hallintaan olisi ollut mukava saada useampi vastaus. Tämä kuitenkin on sellainen asia, joka muovautuu vasta kun alustaa käytetään enemmän ja toivon mukaan jonkinlainen testauskäytäntö suoritetaan jossain vaiheessa. Se myös paljastaisi enemmän järjestelmän puutteitta ja sitä, mitä siltä vaaditaan sekä mitä sille pitäisi tehdä.

Käytännön osuudessa tehtiin muutoksia HaMill-alustaan ja selvitettiin, miten sen muokkaus onnistuu. Esiin nousi miete siitä, kuinka paljon sitä pitäisi muokata, että se vastaisi HAMKin tarpeita. HAMK tarvitsee materiaalipankin ja LeMill valittiin helppokäyttöisyyden perusteella. Sivustojen luonti on helppoa ja metadatan luonti selkeää. Muokkaaminen on kuitenkin kohtuullisen helppoa, kun ensin löytää oikean paikan sille mitä haluaa tehdä. Tätä asiaa vaikeuttaa dokumentaation puute. Tähän asiaan otettiin kantaa myös jatkokehitysosiossa, jossa esiteltiin idea tiedostojen dokumentoinnista.

Opinnäytetyötä tehdessä opin oppimisalustoista sen, mitä ne ovat, mihin niitä käytetään sekä miksi oppimisalustoja on alettu käyttää. Opin myös suunnittelemaan haastattelutilanteita ja miten haastatteluita voi hyödyntää tiedonkeräämisessä. Opinnäytetyöprosessin aikana minulle selvisi myös jo kirjoitetun tekstin muokkauksen tärkeys lukijoita kohtaan. Työ antoi minulle uusia tapoja lähestyä erilaisia asioita, koska niistä piti pystyä kertomaan jollekin, joka ei välttämättä ole perehtynyt siihen alaan, jota työ koski. Tämä tuli hyvin esille haastatteluiden aikana, koska silloin piti osata selittää asioita ihmisille, jotka eivät tienneet tietotekniikasta kovin paljoa.

LÄHTEET

Aspeli, Martin, “Plone: A Model of mature open source project” (2004/05), http://www.martinaspeli.net/publications/Plone%20-%20A%20Model%20of%20a%20Mature%20Open%20Source%20Project.pdf/view

Bretthauer David, "Open Source Software: A History" (2001). *UConn Libraries Published Works.*Paper http://digitalcommons.uconn.edu/libr_pubs/7

BSD lisenssi, viitattu 19.1.2011, http://www.opensource.org/licenses/bsd-license.php

Copyleft, viitattu 11.10.2010, ”What is Copyleft”, http://www.gnu.org/copyleft/

Creative Commons, viitattu 19.1.2011, http://creativecommons.org/licenses/by-sa/2.5/

Distrowatch, viitattu 19.1.2011, http://distrowatch.com/

GNU HURD, viitattu 19.10.2010, http://www.gnu.org/software/hurd/hurd.html

GPL, viitattu 11.10.2010, ”General Public License version 3”, http://www.gnu.org/licenses/gpl.html

Gutschmidt Tom, “Game progaramming with Python, Lua and Ruby”, 2003

History of Zope, viitattu 18.10.2010, http://zope2.zope.org/about-zope-2/the-history-of-zope

LeMill-About, viitattu 20.10.2010 http://lemill.net/content/webpages/about-lemill

LPGL, viitattu 19.1.2011, http://www.gnu.org/copyleft/lesser.html

Lukaszewski, Al , ”A Brief history of Python”, viitattu 2.11.2010, http://python.about.com/od/gettingstarted/ss/whatispython_2.htm

Matikainen, Hannu, “Verkko-oppimateriaalien uudelleenkäytettävyys ja siirrettävyys oppialustalta toiselle” (2005)https://www.doria.fi/bitstream/handle/10024/30169/TMP.objres.129.pdf?sequence=1

Plone, viitattu 19.1.2011, www.plone.org

SCORM, viitattu 19.1.2011, http://www.ostyn.com/standards/docs/HistoryOfSCORM.htm
Source Code Definition, viitattu 5.11.2010 http://www.linfo.org/source_code.html

The Zope Book, viitattu 11.10.2010, http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition

Zope, viitattu 17.10.2010, http://www.zope.org/WhatIsZope