KÄYTTÄJÄTILIAUTOMATIIKKA

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto
Opinnäytetyö
Timo Mäkelä

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

MÄKELÄ, TIMO: Käyttäjätiliautomatiikka

Ohjelmistotekniikan opinnäytetyö, 38 sivua

Kevät 2007

TIIVISTELMÄ

Lahden ammattikorkeakoulussa siirryttiin 2000-luvun alussa keskitettyyn verkkorakenteeseen, jossa koulun eri laitosten käyttäjätilit ovat yhteisessä aktiivihakemistossa. Keskittämisen myötä nousi esille mahdollisuus automatisoida käyttäjätilien ylläpito opintotoimiston Winha-opiskelijarekisterin avulla. Päijät-Hämeen Koulutuskonserni päätti tuottaa tietokantojen yhtenäistämiseen tarvittavan sovelluksen ohjelmistotekniikan opinnäytetyönä.

Opinnäytetyön tavoite oli suunnitella ja toteuttaa ajastettu automatiikka, joka päivittää Winhan syötteen mukaisesti aktiivihakemiston käyttäjätilit. Käyttäjätilien päivitykseen kuuluvat uusien käyttäjätilien luonnit, siirrot ja poistot sekä niiden ominaisuuksien, ryhmien ja kotihakemistojen ylläpitäminen.

Automatiikan toiveisiin kuului syötetiedoston suodattaminen laitos- ja ryhmäkohtaisesti, mustalista tunnuksille, joiden pääsy verkkoon halutaan estää, tulostettavat tunnus- ja sitoumuslomakkeet luoduista käyttäjätileistä sekä helppokäytöisyys ja -laajennettavuus, jotta verkon ylläpitäjät voivat tarvittaessa jatkokehittää prosessia. Työkaluksi sovittiin Microsoftin Visual Studio ja toteutuskieleksi valittiin Visual Basic .NET.

Lopputulokseksi saatiin ajastettava prosessi, jonka toimintaan voidaan vaikuttaa käyttöympäristökohtaisin asetuksin. Automatiikka täytti tavoitteet, ja se otettiin lopulta käyttöön myös koulutuskonsernin toisessa liikelaitoksessa koulutuskeskus Salpauksessa. Nykyään automatiikka huolehtii vuosittain lähes 12000 käyttäjätilistä.

Avainsanat: käyttäjätilien hallinta, hakemistopalvelut

Lahti University of Applied Sciences
Faculty of Technology

MÄKELÄ, TIMO: User Account Automation

Bachelor's thesis in software engineering, 38 pages

Spring 2007

ABSTRACT

In the early 21st century Lahti University of Applied Sciences changed over to using a centralized network hierarchy, where all user accounts of the school's separate faculties are incorporated into one Active Directory. Along the centralization there arose the possibility to automate the administration of the user accounts with the Winha student registry of the Student Services Centre. Lahti Regional Educational Consortium decided to commission the replication of the databases as a Bachelor's thesis in software engineering.

The main purpose of the thesis was to design and implement timed automation, which updates user accounts in Active Directory according to input from Winha. The updating included creation, transfer and deletion of user accounts. The work also included updating the attributes of user accounts, grouping the accounts, and managing home directories.

The extra requirements for the automation were faculty and group specific filtering of input data, a black list for the accounts that had been banned from the network, and printable forms for user credentials and agreements for created accounts. It also had to be easy to use and expandable, so network administrators could continue to evolve the process if required. Microsoft's Visual Studio was chosen as the tool and Visual Basic .NET was selected as the programming language.

The final result was a timed process which could be modified for different kinds of environments. The automation met the requirements and it was also adopted by Salpaus Further Education, another business unit of the consortium. Nowadays the automation manages almost 12 000 user accounts in a year.

Keywords: user account management, directory services

SISÄLLYS

1 JOHDANTO

Voidaan olettaa, että läpi historian lähes kaikilla ihmisillä on ollut henkilökohtaista tietoa ja tarve myös pitää ainakin osa tietämyksestään sellaisena. Tietojärjestelmien kehittyessä ja niiden käyttäjäkunnan laajentuessa on myös digitaalisen tiedon pääsyyn ollut tarve kehittää tiedon valvontamenetelmä. Jotta tietojärjestelmä voi tunnistaa käyttäjän ja näin sallia tai rajoittaa pääsyn tietyille käyttäjille tiettyyn tietoon, tarvitsee käyttäjän kertoa luotettavasti tietojärjestelmälle, kuka hän on. Tämä tiedon valvonnan paradigma on yleisesti ratkaistu tietojärjestelmäkohtaisin käyttäjätilein.

Käyttäjätili muodostuu käyttäjän henkilökohtaisesta tunnuksesta ja salasanasta, joilla käyttäjä tunnistautuu tietojärjestelmän käyttäjäksi. Kun järjestelmä saadaan tunnistamaan käyttäjä käyttäjätilin avulla, voidaan järjestelmässä talletettavalle tiedolle määrittää käyttäjätili- tai -ryhmäkohtaiset oikeudet järjestelmässä talletettavan tiedon saatavuudelle. Lisäksi tunnistautuneelle käyttäjälle tarjotaan usein järjestelmästä oma tietovarasto henkilökohtaisen tiedon varastointiin.

Tietojärjestelmän ja sen käyttäjäkunnan laajetessa käyttäjätilien ylläpitäminen käy luonnollisesti hankalaksi. Tällaisten laajojen käyttäjätilikantojen hallinnointiin onkin kehitetty useita erilaisia menetelmiä tilien luontien, voimassaolojen ja poistamisten hallinnointiin. Ehkä yleisimmin hyödynnetty hallinnointia yksinkertaistava menetelmä on tunnusten ryhmittäminen, jolloin tunnuksia ei tarvitse käsitellä yksittäin. Lisäksi tunnuskantojen ylläpitoa ovat helpottaneet erilaiset järjestelmien ylläpitäjien omiin tarpeisiinsa kirjoittamat komentojonot, joilla käyttäjäkantojen päivitysrutiineita on automatisoitu.

Myös Lahden ammattikorkeakoulussa on perinteisesti luotu, ylläpidetty ja poistettu käyttäjätilit manuaalisesti tai yksinkertaisten ryhmäkohtaisten komentojonojen avulla. Jokaisella laitoksella on ollut oma käytäntönsä ylläpidon suhteen, ja käyttäjätietokannat ovat olleet erillään toisistaan eikä kaikilla laitoksilla ole edes ollut käyttäjätilejä. Keskimäärin ammattikorkeakoulussa opiskelee vuosittain lähes 4000 opiskelijaa, joille kaikille on otettu tavoitteeksi

antaa oma käyttäjätunnus ammattikorkeakoulun yhtenäistettyyn tietoverkkoon. (Päijät-Hämeen koulutuskonserni 2004)

Uusien järjestelmien käyttöönoton yhteydessä ja opiskelijoiden määrää huomioidessa on noussut esille ajatus ammattikorkeakoulun käyttäjätilien keskittämisestä, automatisoinnista ja yhtenäistämisestä opintotoimiston opiskelijarekisterin kanssa. Ideasta kiinnostunut Päijät-Hämeen koulutuskonserni onkin päättänyt yhdistää ammattikorkeakoulun eri laitokset yhteiseen verkkorakenteeseen, jossa käyttäjätilien hallinnointi voidaan hoitaa keskitetysti sekä tuottaa käyttäjätilien ylläpidosta huolehtiva sovellus ohjelmistotekniikan opinnäytetyönä.

Opinnäytetyön tavoitteena on toteuttaa käyttäjätiliautomatiikka, jonka tehtävä on ylläpitää Lahden ammattikorkeakoulun keskitettyä käyttäjätilien *aktiivihakemistoa* Päijät-Hämeen koulutuskonsernin opintotoimiston opiskelijarekisterin syötetiedoston mukaisesti. Opiskelijarekisterinä Lahden ammattikorkeakoulussa käytetään *Winha*-opintotietokantaa, jota opintotoimisto ylläpitää kaikista Lahden ammattikorkeakoulun opiskelijoista.

Itse työn kannalta *aktiivihakemisto* näyttelee suurta osaa, joten työssä tutustutaan ensin sen teoriaan ja yleensä hakemistopalveluihin, sekä niiden kehityksiin johtaneisiin tekijöihin, tarpeisiin ja tavoitteisiin. Samalla nostetaan esille automatiikan kannalta oleelliset tietomallit, eli käyttäjätilit ja niiden ominaisuudet sekä rooli hakemistopalveluissa. Lisäksi tutustutaan *aktiivihakemiston* tarjoamiin ohjelmointirajapintoihin, joista valitaan samalla ylläpidon käyttöön mahdollisimman yksinkertaiset. Yksi ylläpidon toive on, että se voi muokata ajan saatossa opinnäytetyönä toteutettua prosessia, joten ohjelmointityökalujen valinnassa on huomioitava helppokäyttöisyys ja laajennettavuus.

Seuraavaksi esitellään varsinainen automatiikka. Aluksi käsitellään syötteet, väliaika- ja vertailutiedostot sekä tulosteet. Seuraavaksi syvennytään prosessin rakenteeseen ja parametrointiin. Lopuksi käsitellään prosessin käyttöönottoa sekä siinä huomioitavia tekijöitä.

Viimeisessä luvussa pohditaan työn tuloksia ja verrataan niitä tavoitteisiin. Lisäksi arvioidaan automatiikan tuomaa todellista hyötyä ylläpidolle ja koulutuskonsernille sekä arvioidaan käyttäjätiliautomatiikan tulevaisuutta.

2 HAKEMISTOPALVELUT

2.1 Hakemistopalvelujen taustaa

Hakemistopalvelut ovat erikoistuneita tietovarastoja, jossa tieto tallennetaan hierarkiseen puurakenteeseen. Rakenteella on yksi juuri, joka toimii ylimmäisenä säiliönä hakemistoon talletettavalle tiedolle. Säiliöllä tarkeitetaan hakemistopalvelun solmukohtia, jotka voivat sisältää skeemaluokkien mukaisia skeemaolioita eli puun lehtiä tai alihakemistoja eli uusia säiliöitä. Perusajatukselta hakemistopalvelu muistuttaa käyttöjärjestelmän tiedostojärjestelmää.

Hakemistoajatuksen pohjalta kehitettiin vuosien 1984 ja 1994 välillä X.500-hakemistostandardi, jonka suurena tausta-ajatuksena oli sijoittaa kaikki tieto yhteen paikkaan kaikkien saataville. Näin käyttäjät ympäri maailmaa olisivat yhteydessä toisiinsa yhden ja saman hakemiston avulla, ja he voisivat oppia toisiltaan ja kommunikoida vapaasti keskenään. (Boswell 2003.)

Hakemistostandardin lisäksi tarvittiin yhtenäinen menetelmä, jolla hakemistorakenteen tietosisältöä käsitellään. Tähän tarkoitukseen kehitettiin DAP-standardiprotokolla (*Directory Access Protocol*), joka määrittelee hakemistojen tiedon käsittelyyn vaadittavat operaatiot ja näiden syötteet. (Robinson 1999, 15.)

X.500-hakemisto oli kuitenkin liian monimutkainen monipuolisine käytäntöineen, joten se ei ehtinyt saavuttaa laajaa suosiota ennen kuin 1990-luvun alkupuoliskolla Michiganin yliopistossa kehitettiin koulun omiin tarpeisiin ensimmäinen versio kevyemmästä vaihtoehdosta LDAP-protokollasta (*Lightweight Directory Access Protocol*). (Boswell 2003.)

Kuitenkin vasta LDAP2 oli ensimmäinen merkittävä DAP-protokollan toteutusversio. LDAP2 oli täysin yhteensopiva edeltävän X.500-standardin kanssa, mutta siitä oli jätetty pois useita harvoin käytettyjä operaatioita. Tuettuja

operaatioita olivat vain tarpeellisimmat kuten hakemistorakenteen selaaminen sekä skeeman mukaisten tietueiden etsiminen ja muokkaaminen. (Robinson 1999, 15.)

LDAP2 onnistui levittämään ajatusta hakemistopalveluista, mutta pian se huomattiin liian rajoittuneeksi, ja protokollasta määriteltiin uusi versio vuonna 1997. Merkittävin kehitys uudessa LDAP3 versiossa oli yleismaailmallinen UNICODE- merkistö. LDAP3 hyväksyttiin laajalti yleiseksi standardiksi, minkä johdosta myös Microsoftin *aktiivihakemisto* (*Active Directory*) tukee sitä. (Robinson 1999, 15.)

2.2 Hakemistojen periaatteet

Hakemistot ovat periaatteiltaan hyvin yhteneviä tietokantojen kanssa, mutta ne ovat usein käyttötarkoitukseltaan erikoistuneempia. Hyvä esimerkki tietohakemiston erikoistapauksesta on käyttäjätilihakemisto, jota myös tässä opinnäytetyössä käsitellään.

Yksi hakemistojen tärkeimmistä ominaisuuksista on tiedon keskittäminen. Keskittämisen tärkeys nousee esille erityisesti useiden tietokoneiden verkoissa, joissa jokaiselta koneelta tarvitaan pääsy yhteiseen tietovarastoon. Koneiden keskinäiseen viestintään verkon yli hakemistot tarjoavat protokollan, joka noudattaa asiakas/palvelin-mallia. Laajassa verkkoympäristössä hakemistot voivat tarjota protokollan lisäpiirteenä tietosisällön peilauksen tai hajauttamisen useamman palvelimen välillä vähentääkseen yksittäisen palvelimen kuormaa. (LDAP-hakemistopalvelu 2004.)

Tiedon keskittäminen on myös tämän opinnäytetyön käyttäjätilihallinnan yksi tärkeimmistä edellytyksistä. Juuri keskittämisen ansiosta käyttäjätilien ylläpidettävyys voidaan pitää yksinkertaisena, kun kaikki verkkoalueen päivitykset voidaan hoitaa yhdeltä palvelimelta.

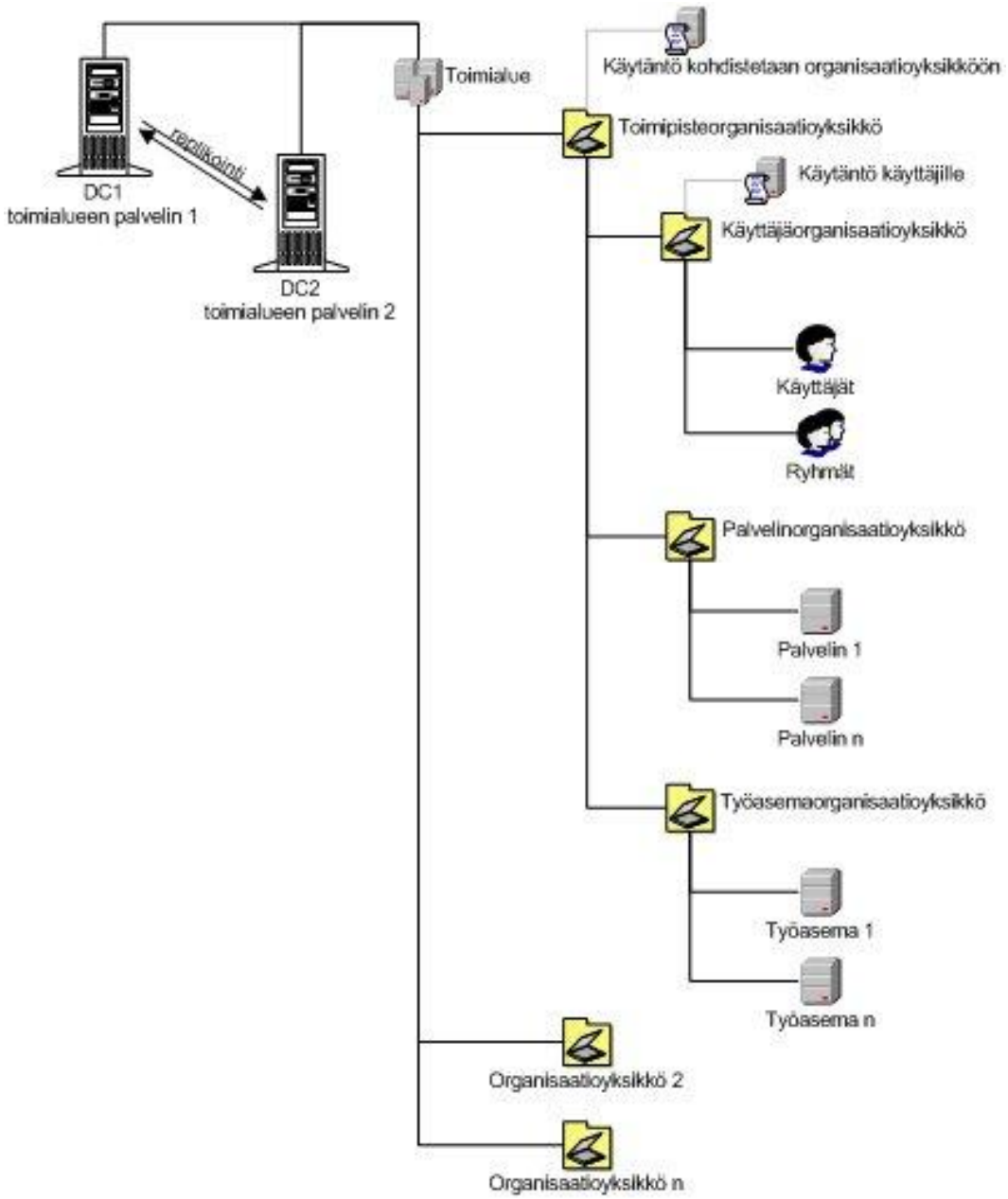
Merkittävin ero hakemistopalveluiden suunnittelussa verratuna muihin tietokantoihin on oletus, että tietoa kysytään useammin kuin muutetaan ja että päivitykset ovat tyypillisesti hyvin yksinkertaisia ilman tiukkaa tapahtumankäsittelyä. Näihin periaatteisiin nojaten hakemistopalvelut ovat optimoituja erityisesti tiedon lukemiseen, ja muutosten tulee sietää viiveellä päivittyvää informaatiota. (LDAP-hakemistopalvelu 2004.)

Käyttäjätilien hallintaa ajatellen edellä huomioitu lukuoperaatioiden priorisointi sopii erityisen kätevästi myös suuren organisaation sisäverkon tarpeisiin, jossa käyttäjien voidaan olettaa autentikoituvan työasemille huomattavasti useammin kuin heidän tunnuksia tarvitsee päivittää, lisätä tai poistaa.

2.3 Hakemistojen sisältö

2.3.1 Hakemistorakenne

Selkeän mielikuvan hakemistopalveluiden rakenteesta saa, kun ajattelee tiedostojärjestelmää, jossa hakemistot eli oksat tai säiliöt ovat kansioita ja tieto-oliot eli lehdet tiedostoja. Tässä vertauksessa on kuitenkin syytä tehdä ero hakemistosäiliön ja tiedostojärjestelmän kansion välille siinä, että myös hakemistosäiliö voi sisältää vastaavaa tietoa kuin tieto-oliot toisin kuin tiedostojärjestelmän kansiot. (Robinson 1999, 35.)



KUVIO 1. Hakemistorakenne.

Kuviossa 1 hakemistopalvelu on hajautettu kahdelle toimialuepalvelimelle (*Domain Controller*). Toimialueen hakemistoja esittävät organisaatioyksiköt ja lehtiä ovat käyttäjätunnukset, palvelimet ja työasemat. Toimialueesta voidaan käyttää myös termiä metsä, joka on kokoelma itsenäisiä puita eli hakemistoja, jotka kaikki hyödyntävät yhteisiä skeema- ja asetustietoja.

2.3.2 Skeema

Hakemistopalvelut muodostuvat skeemaolioista, jotka täyttävät oman skeemaluokkansa määritykset. Skeemaoliot ovat toisin sanoen varastoja tiedolle, kuten käyttäjätileille, -ryhmille ja työasemille. Periaatteessa skeemaoliot voivat sisältää mitä vain tietoa ominaisuuksissaan, mutta yleensä hakemistopalvelut ovat erikoistuneet tietynmuotoisen informaation varastointiin.

Kaikilla tietokannoilla on muodollinen skeema, joka määrittää tietokannan sisällön. Puurakenteisessa tietokannassa skeema sisältää kaikki mahdolliset hakemistossa säilöttävät skeemaluokat ja -ominaisuudet. Toisin sanoen skeema on kokoelma sääntöjä hakemistoon talletettavalle tiedolle. Skeema talletetaan hakemistorakenteeseen aivan kuten sen mukaiset skeemaoliot. Näin mahdollistetaan hakemiston laajennettavuus ja uusien skeemojen lisääminen valmiiseen hakemistorakenteeseen.

Skeemaluokat ovat kokoelmaluokkia, jotka sisältävät säännöt kyseisen luokan ominaisuuksille. Ominaisuudet ovat yksinkertaisia nimi-arvo -pareja, joita voidaan verrata ohjelmakoodin muuttujiin. Esimerkiksi käyttäjätililuokalla on ominaisuus nimeltä *telephone*, jossa pidetään tallessa käyttäjän puhelinnumeroa. Skeemaluokkien ominaisuudet voivat olla myös moniarvoisia, jolloin sama ominaisuus voi sisältää useita arvoja. Esimerkiksi jollakin käyttäjällä voi olla useita puhelinnumeroita. (Robinson 1999, 11.)

Skeemaluokat (*classSchema*) ovat joko rakenteellisia (*stuctural*), abstrakteja (*abstract*) tai avustavia (*auxiliary*). Rakenteelliset luokat ovat tietosäiliöitä tietyn tyyppisille ilmentymille. Abstraktit luokat ovat korkean tason luokkia, joilla ei oleteta olevan ilmentymiä, mutta jotka keräävät tietyn tyyliset ominaisuudet yhteisen perittäväksi tarkoitetun luokan yhteyteen. Avustavat skeemaluokat ovat tarkoitettu laajentamaan rakenteellisia luokkia lisäämällä niille ominaisuuksia. (Kouti & Seitsonen 2002.)

Skeemaominaisuuksien (*attributeSchema*), -luokkien (*classSchema*) ja syntaksien nimeäminen ja tunnistus toimivat erilaisilla nimillä ja hierarkisilla numeroinnilla, jossa hakemistorakenteen tasot erotetaan pisteillä toisistaan. Skeema eli hakemiston ominaisuudet ja luokat muodostetaan tietystä joukosta olioita. Jokaisesta luokasta ja ominaisuudesta löytyy hakemistosta yksi ilmentymä, olio. Poikkeuksellisesti syntakseista ei ole ilmentymää hakemistoissa, vaan ne on rakennettu sisään hakemistopalvelun toteutukseen. Tämä tarkoittaa sitä, että luokkia ja attribuutteja voidaan luoda uusia ja muokata jo olemassa olevia, mutta syntaksi on kaikilla aina samanlainen. (Kouti ym. 2002.)

Edellä esitellyt skeemaluokat ja -ominaisuudet muodustuvat säännöistä, jotka ovat siinä mielessä poikkeuksellisia, ettei niitä voida laajentaa. Säännöt voidaan jakaa rakenteellisiin, sisällöllisiin ja syntaktisiin sääntöihin. Rakenteellisiin sääntöihin kuuluu skeemaluokkien tieto siitä, onko luokka säiliö vai lehti. Sisällölliset säännöt määrittävät ominaisuuksien pakollisuuden eli tiedon siitä, onko kyseinen ominaisuus kyseiselle skeemaluokalle pakollinen (*mandatory*) vai vaihtoehtoinen (*optional*). Esimerkiksi käyttäjätileillä pakollisia ominaisuuksia ovat kirjautumistunnus (*sAMAccountName*) ja vaihtoehtoisia puhelinnumero (*telephone*). Syntaktiset säännöt puolestaan rajoittavat ominaisuuksien arvoalueen, esitysmuodon sekä tiedon, onko attribuutti yksiarvoinen (*single-value*) vai moniarvoinen (*multivalue*). Muita skeemaan kuuluvia sääntöjä ovat omiinaisuuksien indeksointi, turvallisuussäädökset ja luokkien nimeämiskäytäntö. (Kouti ym. 2002.)

2.3.3 Hakemistojen lukeminen

Hakemistopalveluiden peruspiirteisiin kuuluu tiedon hajauttaminen useille palvelimille sekä tiedon saavutettavuus miltä palvelimelta hyvänsä. Juuri näiden ominaisuuksien toteuttamisessa tarvitaan yhtenevää nimeämiskäytäntöä, jonka avulla oliot voidaan helposti paikallistaa laajoistakin hakemistoista. (Boswell 2003.)

Yleinen nimeäminen toteutetaan korkeimmalla tasolla yksilöllisellä polulla (*ADsPath*), jonka avulla skeemaoliot voidaan lukea nopeasti hakemistosta. Tällainen polku on yleensä muotoa LDAP://OU=PHKK, jossa OU on kyseisen olion tyyppiä ilmaiseva termi ja tarkoittaa tässä yhteydessä organisaatioyksikköä (*OrganizationUnit*). PHKK on puolestaan organisaatioyksiköllе annettu nimi. Lisäksi tämä ensimmäisen tason organisaatioyksikkö on tietosäiliö, joka voi sisältää esimerkiksi Tekniikan laitokselle oman säiliön, jonka polku olisi muotoa LDAP://OU=TEKNIIKANLAITOS,OU=PHKK, tai käyttäjätilin, jonka polku olisi LDAP://CN=MAKEMTIM,OU=PHKK. Puurakenteen oksat erotellaan esimerkkien mukaan toisistaan pilkulla (,). Vastaavasti, jos siirrämme juuri tason käyttäjätilin Tekniikan laitoksen alle sen polku muuttuisi muotoon LDAP://CN=MAKEMTIM,OU=TEKNIIKANLAITOS,OU=PHKK. Käyttäjätilin polussa CN on lyhenne olion yleisnimestä (*Common name*). (Robinson 1999, 300.)

Hakemistohauissa tietoa voidaan suodattaa skeemaluokkien tyyppien ja ominaisuuksien mukaan. Esimerkiksi käyttäjän autentikoituessa verkkoon kyseisen tilin haku suodatetaan säiliön tyypin mukaan niin, että haetaan vain käyttäjätilioliota, jonka *sAMAccountName* on kirjautujan syöttämä käyttäjätunnus. Tämä haku palauttaa yhden käyttäjätilin, jonka salasanaa verrataan kirjautujan syöttämään salasanaan.

2.4 Aktiivihakemisto

Kun Microsoft päätti korvata kömpelön rekisteripohjaisen tunnistushallinnan kehittyneemmällä hakemistopalvelulla, se valitsi myös LDAP:in. Omassa hakemistopalvelussaan se halusi hyödyntää ESE (*Extensible Storage Engine*) ja DNS (*Domain Name System)* tekniikoita. (Boswell 2003.)

Windows 2000 Serverissä on täydellinen valikoima peruspalveluita, joiden pohjana on hakemistopalvelu nimeltä *aktiivihakemisto* (*Active Directory*). *Aktiivihakemisto* yksinkertaistaa tiedon hallintaa, parantaa turvallisuutta ja lisää

yhteistoimintamahdollisuuksia. Se on keskitetty menetelmä käyttäjien, ryhmien, suojauspalveluiden ja verkkoresurssien hallintaan. *Aktiivihakemisto* sisältää myös joukon standardoituja käyttöliittymiä, jotka mahdollistavat monenlaisten ohjelmien ja laitteiden yhteistoiminnan.

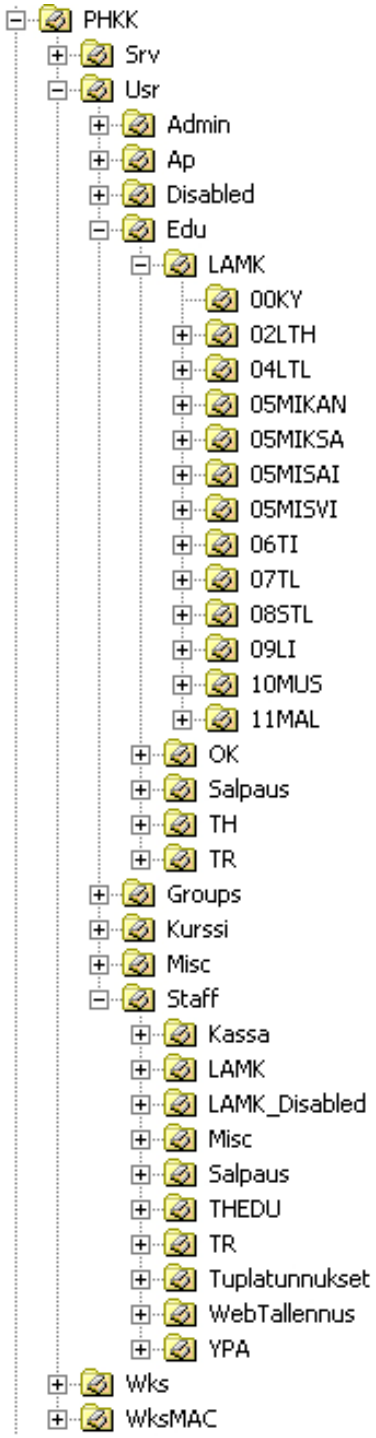
Aktiivihakemisto on *Windows Server 2000* ja *Windows Server 2003* toiminimipalvelimille toteutettu LDAP -hakemistorakenne. Se tarjoaa turvallisen, rakenteellisen ja hierarkkisen tietovaraston verkko-olioille kuten käyttäjätilit, työpisteet, tulostimet ja muut palvelut. *Aktiivihakemisto* on kehitetty erityisesti näiden olioiden tehokkaaseen hyödyntämiseen toimialueverkkoympäristössä. (Robinson 1999, 168.)

Aktiivihakemiston perusskeema sisältää 56 säiliö- ja 86 lehtiluokkaa. Skeemoja muokkaamalla on lehtiluokistakin mahdollista tehdä säiliöitä. *Aktiivihakemisto* voi sisältää jopa miljoonia olioita, joten tietokantahakujen nopeuttamiseksi tarvitaan kehittynyttä indeksointijärjestelmää. *Aktiivihakemiston* perusskeemasta indeksoituja attribuutteja löytyy 64. (Kouti ym. 2002.)

Tarkemmin *aktiivihakemiston* käyttäjätilien ominaisuuksiin on keskitytty luvussa kolme, jossa käsitellään ADSI-ohjelmointirajapintaa.

2.4.1 Aktiivihakemisto Lahden ammattikorkeakoulussa

Lahden ammattikorkeakoulussa *aktiivihakemisto* otettiin käyttöön pääosin vuoden 2002 aikana laitos kerrallaan. Kaiken kaikkiaan käyttöönotto kesti n. kolme vuotta. Koulun verkkoalueella on kaksi toimialuepalvelinta (*Domain Controller*), jotka toimivat myös välitysnimipalvelimina. Ammattikorkeakoulun *aktiivihakemiston* rakenne on muodostettu organisaatioyksiköittäin omiin itsenäisesti ylläpidettäviin haaroihin. (Känkänen 2004.)



KUVIO 2. Lahden ammattikorkeakoulun käyttäjätilikannan rakenne *aktiivihakemistossa*.

Lahden ammattikorkeakoulun käyttäjätilit ovat kaikki yhden yhteisen organisaatioyksikön (OU=Usr,OU=PHKK) alla. Tämän alle käyttäjätilit ovat jaettu omiin alipuihin. Ylläpitäjät löytyvät *Admin*-hakemistosta, muu henkilökunta

Staff-hakemistosta, opiskelijat *Edu*-hakemistosta, passiiviset tilit *Disabled*-hakemistosta ja käyttäjäryhmät *Groups*-hakemistossa. Lisäksi henkilökunta ja oppilaat ovat jaettu vielä yksikkö ja laitoskohtaisiin alisäiliöihin. Esimerkiksi Tekniikan laitoksen opiskelijat löytyvät hakemistosta OU=07TL,OU=LAMK,OU=Edu,OU=Usr,OU=PHKK.

3 HAKEMISTOJEN OHJELMOINTI

3.1 Visual Basic .NET

Windows-verkkoympäristöä ja erityisesti *aktiivihakemistoa* ohjelmoitaessa yhteensopivin vaihtoehto lienee *Microsoftin* oma kehitysympäristö *Visual Studio .NET*, jonka käytöstä tässä opinnäytetyössä sovittiin jo ennen työn suunnittelua. Lisäksi automatiikan määrittely puolsi ohjelmointikielen valinnassa yksinkertaista ja ylläpitäjien keskuudessa helposti luettavaa ja omaksuttavaa kieltä. Tästä syystä ohjelmointikieleksi valittiin Visual Basic, josta uusin versio on osa .NET -tuoteperhettä.

Visual Basic .NET on syntaksiltaan ja rajapintojen käytettävyydeltä suurimmalta osin yhtenevä kielen vanhempien versioiden kanssa. Aiemmin Visual Basicia on pidetty enemmän skriptauskielenä, joka tulkataan ajon aikaisesti, mutta uuden version myötä ohjelmat käännetään .NET binääriksi. Suurimpana muutoksena Visual Basicin .NET -versio on kehitetty oliosuuntaiseksi (*object-oriented*) ohjelmointikieliksi, joka mahdollistaa ohjelmarakenteen jakamisen luokkiin. Uudessa versiossa on myös helpotettu muistinhallintaa käyttämällä CLR-tekniikkaa (*Common Language Runtime*), joka hoitaa muistin käsittelyn ja roskien keruun. Lisäksi muutamia kielen perusosia on kirjoitettu kokonaan uusiksi. Tällaisista erityisesti mainittakoon taulukot (*array*), jotka eivät ole uuden version myötä yhteensopivia aiempiin versioihin, ja UNICODE-merkkijonot. (Pattison 2001.)

3.2 Aktiivihakemiston ohjelmointi

3.2.1 Rajapinnat ja nimiavaruudet

Aktiivihakemistoa voidaan ohjelmoida yleisellä LDAP (*Lightweigth Directory Access Protocol)* API:lla, *Microsoftin* omalla yksinkertaistelulla ADSI (*Active*

Directory Services Interface) API:lla tai vaihtoehtoisesti hyödyntäen *System.DirectoryServices* nimiavaruutta, joka hyödyntää ADSI-rajapintaa. Koska näistä vaihtoehdoista yksinkertaisin riittävä vaihtoehto *aktiivihakemiston* ohjelmointiin on *System.DirectoryServices* nimiavaruus, valittiin se tämän työn tarpeisiin.

Koska *System.DirectoryServices* nimiavaruus hyödyntää ADSI-rajapintaa, perehdytään ensin siihen. ADSI on kokoelma dynaamisesti linkitettäviä kirjastoja (DLL), joilla voidaan ohjelmallisesti muokata *aktiivihakemiston* olioita, kuten tulostimia, verkon tietokoneita ja käyttäjätilejä. Vaikka ADSI on ensisijaisesti suunniteltu ja toteutettu *aktiivihakemiston* ohjelmointiin, voidaan sitä käyttää myös muiden hakemistopalvelujen kanssa. (ADSI Architecture 2000; Robinson 1999, 33.)

System.DirectoryServices tarjoaa *aktiivihakemiston* ohjelmointiin vain kaksi luokkaa: *DirectoryEntry* ja *DirectorySearcher*. *DirectoryEntry* on säieturvallinen yleisluokka *aktiivihakemiston* olioiden käsittelyyn, mutta sitä voidaan käyttää vain rajoitetusti skeemaolioiden kanssa. *DirectorySearcher* on puolestaan yleisluokka eri olioiden etsimiseen *aktiivihakemistostasta*.

3.2.2 Aktiivihakemistosta etsiminen

DirectorySearcher on tarkoitettu olioiden etsimiseen *aktiivihakemiston* puurakenteesta. Tätä käytettäessä määritellään ensin olio, jota etsitään sekä mahdolliset ominaisuudet, joiden mukaan etsittäviä olioita halutaan ottaa haun tuloksiin mukaan tai poissulkea. Lisäksi hauille voidaan määrittää rajoitteita, kuten aika, joka etsimiseen enimmillään käytetään tai suurin palautettavien *DirectoryEntry*-olioiden määrä. Löydetyt tulokset voidaan tallettaa paikalliselle koneelle väliaikaismuistiin, jolloin oliota voidaan muokata paikallisesti ja päivittää *aktiivihakemistoon* myöhemmin *CommitChanges*- funktiolla. (Visual Studio .NET dokumentaatio 2004.)

Haut jaetaan kolmeen loogiseen elementtiin: aloituspaikkaan (*DN base*), hakuehtoihin (*Filter*) ja rajoitusalueeseen (*Scope*).

Aloituspaikalla (*DN base*) tarkoitetaan ylimmäistä säiliötä, jonka alta haluttua skeemaoliota yritetään paikallistaa. Mikäli *aktiivihakemiston* rakenne on suunniteltu järkevästi, voidaan hyvän haun aloituspaikan valinnalla nopeuttaa hakua hyvinkin paljon, koska koko hakemistopuuta ei tarvitse etsiä halutun olion löytämiseksi.

Hakuehdoilla (*Filter*) rajataan haun etsimiä olioita luokan ja ominaisuuksien mukaan. Hakuehtojen ominaisuuksia suodatettaessa kannattaa suosia yksiarvoisia attribuutteja moniarvoisten sijaan, koska yksiarvoiset kentät ovat nopeampia tarkistaa kuin moniarvoiset. Samasta syystä *Microsoft* suosittelee rajaamaan haettavia luokkia *objectCategory*-määrityksellä *ojectClass*-määrityksen sijaan. (Robinson 1999, 304 ja 332.)

Haun rajoitusalueella (*Scope*) kerrotaan haulle, kuinka syvältä hakemistopuun aloituspaikan alisäiliöistä haku etsii haluttua oliota. Rajoitusalueen vaihtoehtoja ovat koko *aktiivihakemisto* (*base*), yksittäinen puun oksa eli säiliö (*OneLevel*) tai aloituspaikasta katsottuna hakemiston alipuu (*Subtree*). Esimerkiksi, jos tiedetään, että etsityn olion on oltava tietyssä säiliössä tai sitä ei löydy lainkaan *aktiivihakemistosta*, voidaan haku rajoittaa yhdelle tasolle (*OneLevel*) haun nopeuttamiseksi.

3.2.3 Hakemisto-olioiden luonti

Uusien olioiden luonnissa *aktiivihakemistossa* on käytettävä ADSIN lapsiluokkia. Käyttäjätilin yhteydessä riittää, että luodaan uusi yleinen skeemaolio *IADsContainer*. Luonnon yhteydessä oliolle tarvitsee määrittää säiliö- eli äitiluokka, jonka alle se hakemiston puurakenteeseen luodaan sekä vähintään yksi skeemaluokka, jonka säännöt olio täyttää. (Visual Studio .NET dokumentaatio 2004.)

Käyttäjätilien lisäys onnistuu luomalla käyttäjätiliolion verkkoalueen *domain containeriin*. Käyttäjät voidaan luoda joko toimialueen juureen, organisaatioyksikköön tai mihin tahansa muuhun hakemiston säiliöolioon. Käyttäjää luotaessa tilille on asetettava olion yleinen nimi (*commonName*) ja yksilöllinen tunnus (*sAMAccountName*). Lisäksi tilille on hyvä määritellä muita ominaisuuksia, kuten käyttöikä (*accountExpires*), skeeman määritykset (*objectCategory*), nimi (*name*) ja ryhmä (*memberOf*). Lisäksi *DirectoryEntry* on muokattava vielä käyttökuntoon lisäämällä sille salasana ja käyttöoikeudet (*userAccountControl*). (Creating a User 2004.)

4 KÄYTTÄJÄTILIEN AUTOMATISOINTI

4.1 Automatiikan vaatimukset

4.1.1 Automatiikan ympäristö

Automatiikan verkkoympäristönä toimii Lahden ammattikorkeakoulun *Microsoft Windows*-verkko, jonka käyttäjätilejä ylläpidetään *aktiivihakemistossa*. *Aktiivihakemisto*-palvelimina toimii kaksi *Windows Server 2003* toimialuepalvelinta (*Domain Controller*) ja *Windows 2000* ja *XP* -käyttöjärjestelmillä varustettuja työasemia on n. 1700. Verkkoa käyttää n. 5000 opiskelijaa ja n. 400 henkilökunnan jäsentä. Toimipisteitä on 12 ja ylläpitäjiä n. 20. (Känkänen 2004.)

Opintotoimiston *Winha*-järjestelmä lähettää joka yö automatiikalle viimeisimmän tulosteen omasta tietokannastaan. Automatiikka puolestaan kirjoittaa omat tulosteensa ajokansionsa alle, josta Unix-sähköpostipalvelin noutaa oman syötteensä ja josta opintotoimiston henkilökunta noutaa sitoumus- ja tunnuslomakkeet.

4.1.2 Laatuvaatimukset

Laadullisina tavoitteina pyritään luomaan prosessista joustava, monipuolinen ja helposti ylläpidettävä, jotta pienten muutosten tekeminen ja automatiikan käyttöönotto muissa vastaavissa verkoissa olisi vaivatonta.

Helpolla ylläpidettävyydellä tarkoitetaan ylläpidon näkökulmasta mahdollisimman läpinäkyvää prosessia, jonka voidaan olettaa käsittelevän opiskelijatunnukset automaattisesti, niin ettei ylläpidon tarvitse käsitellä niitä käsin. Käyttäjätilien oletetaan siis muodustuvan, päivittyvän ja poistuvan automaattisesti *aktiivihakemistosta* syötteen mukaisesti.

Joustavalla parametroinnilla puolestaan halutaan vaikuttaa prosessin toimintaan mahdollisimman korkealla tasolla. Parametroitavia asioita ovat *aktiivihakemiston* hakemistorakenne, prosessin ympäristö kuten syöte- ja tulostehakemistot ja -tiedostot, prosessin huolehtimat laitokset sekä näiden kotihakemistot sekä automatiikan ulkopuolelle jätettävät opiskelijaryhmät.

4.1.3 Muut ominaisuudet

Muita automatiikalta toivottuja ominaisuuksia ovat erillinen mustalista ja tulostettavien sitoutumislomakkeiden muodostaminen sekä käyttäjätilien muutossyötteen kirjoittaminen sähköpostipalvelimelle.

Sitoutumislomakkeet ovat opintotoimiston tulosteita, joihin kerätään oppilaan suostumus koulun verkon käyttöehtoihin tunnuksen luovuttamisen yhteydessä. Opintotoimisto arkistoi allekirjoitetut tulosteet.

Mustalista on lista opiskelijanumeroita, joiden käyttäjätilit pidetään a*ktiivihakemistossa* passiivisina. Tällaisia tapauksia ovat esimerkiksi oppilaat, jotka eivät suostu tunnusten käyttöehtoihin tai joiden tunnuksilta halutaan väärinkäytöksistä tai muista syistä johtuen estää kirjautuminen koulun verkkoon.

Lisäksi lisätyistä ja poistetuista tunnuksista kirjoitetaan erillistä tulostetta, joka toimitetaan sähköpostitunnuksia varten *Unix*-sähköpostipalvelimelle, jotta opiskelijan *aktiivihakemiston* käyttäjätili ja sähköpostitili ovat yhteneviä.

4.2 Syötteet ja tulosteet

4.2.1 *Winhan* CSV-syöte

Winhan käyttäjätietokanta on LDAP- hakemistopalvelu, josta ajetaan ajastetusti joka aamu Lahden ammattikorkeakoulun opiskelijat CSV (*Comma Separated*

Value) muotoiseen tiedostoon. Tiedosto lähetetään ajastetusti *automatiikalle* uudeksi syötteeksi.

CSV on yleisesti käytetty esitysmuoto, jossa yhtä oliota koskeva tieto esitetään yhdellä rivillä niin, että sen ominaisuudet ovat eroteltu pilkuin tai puolipistein. CSV-muotoa käytettäessä arvojen esittämisjärjestys dokumentoidaan erikseen. Automatiikka käyttää WINHAPRO 4.5 LDAP käyttöohjetta tiedostomuodon lukemiseen. *Winhan* syötteessä kukin rivi vastaa yhtä roolia. Roolilla on kaiken kaikkiaan 28 ominaisuutta, joista osa on moniarvoisia. Moniarvoisten ominaisuuksien arvot erotellaan omassa CSV-kentässään kaksoispisteellä (:). Alla esimerkki syöterivistä:

```
01XXXXX;;;;07TIE01;07DIILI06:07HOHJ0103:H07TIE1Y01:H07TIEY202;YA;2
061;99999;1;;1;07TL;1;Mäkelä;Timo-Tapio
Petteri;044XXXXXXX;Tapparakatu X B 29;LAHTI;15700;FIN;timo-
tapio.makela@lpt.fi;;1;;;Timo;;270980-
125B;;20070531;;Tietotekniikan
koulutusohjelma;12;0;LUK;127024af70a6d5c6705abd18df0f5a0d;20034;;
```

KUVIO 3. Syötetiedoston esimerkkirivi.

4.2.3 Tuloste sähköpostipalvelimelle

Automatiikka muodostaa luomistaan tunnuksista syötteenkaltaisen CSV-tiedoston, jonka mukaisesti sähköpostitunnukset uusille opiskelijoille luodaan. Tiedostossa siirtyy tärkeimpinä kenttinä oppilaan tunnus ja salasana *aktiivihakemistossa*, joiden mukaan uusille sähköpostitunnuksille asetetaan sama tunnus ja salasana sähköpostiin.

Luotujen tunnusten lisäksi sähköpostipalvelimelle välitetään tieto myös poistetuista tunnuksista.

4.2.4 Tunnus- ja sitoumuslomakkeet

Automatiikka muodostaa luomistaan tunnuksista laitoskohtaiset tunnuslomakkeet sekä ryhmäkohtaiset sitoumuslomakkeet. Kummatkin lomakkeet kirjoitetaan

RTF- muotoon (*Rich Text Format*), ja ne kirjoitetaan automatiikan ajokansion alle omaan alihakemistorakenteeseen, johon opintotoimistolla on lukuoikeus.

Tunnuslomakkeet ovat yksinkertaisia tiedostoja, joissa aina yhdelle sivulle tulostetaan yhden luodun tunnuksen luovutustiedot eli käyttäjätunnus ja salasana kyseiselle oppilaalle. Näin tunnuslomakkeet on helppo tulostaa yhtenä dokumenttina ja jakaa sivuttain oppilaille.

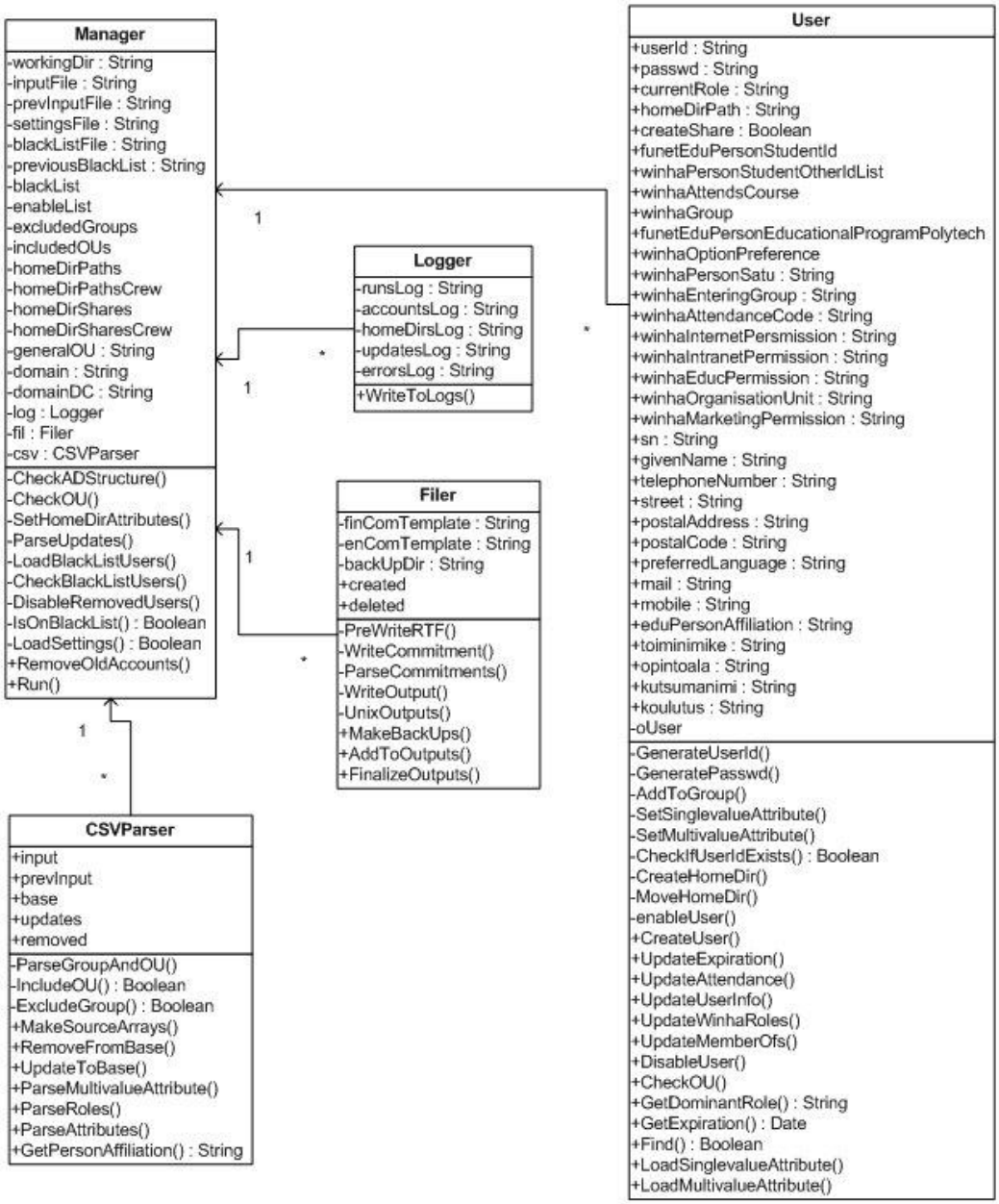
Sitoumuslomakkeille kirjoitetaan tunnuksen luovuttamisen yhteydessä allekirjoitettavat ehdot. Ehdot luetaan erillisestä tekstitiedostosta, jotta niiden päivittäminen olisi yksinkertaista. Sitoumuslomake on saapumisryhmä- ja asiointikielikohtainen arkistoinnin helpottamiseksi. Sitoutumislomakkeita kirjoitetaan kielillä suomi ja englanti.

4.2.5 Mustalista

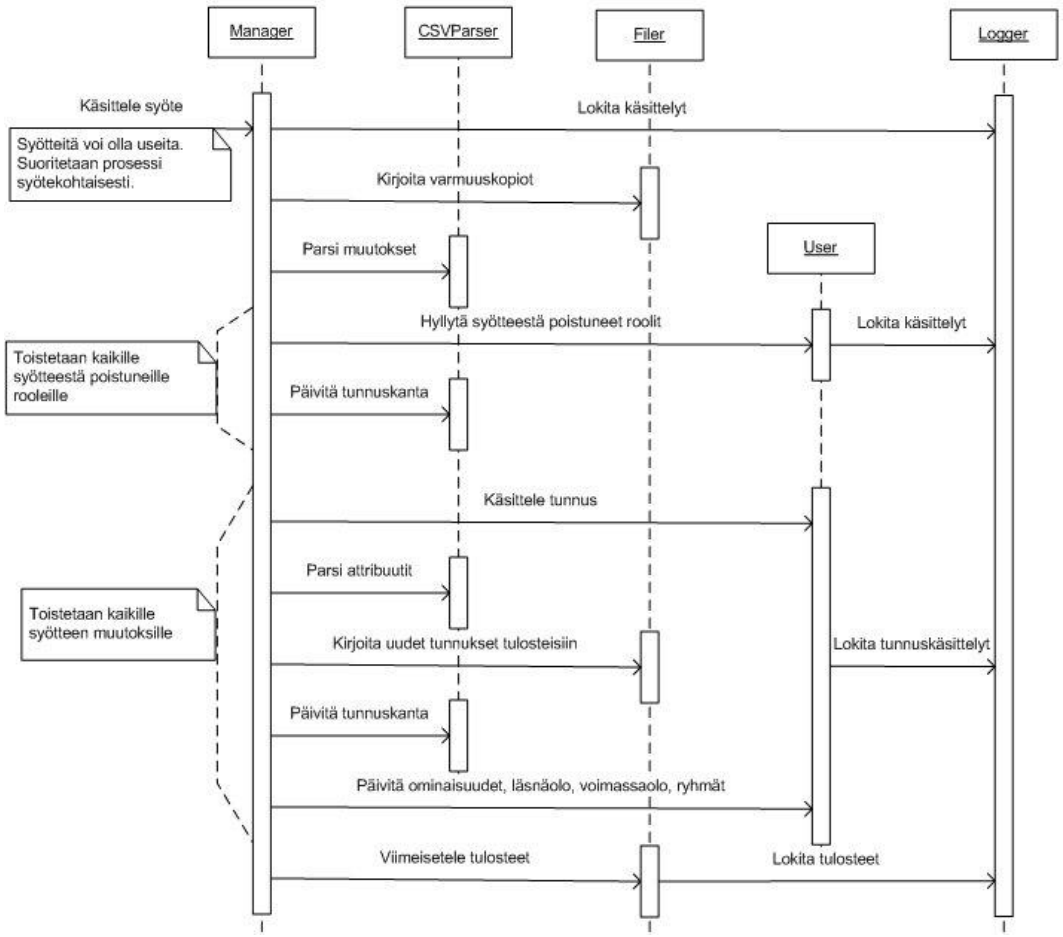
Mustalistaa ylläpidetään yksinkertaisena tekstitiedostona, jossa passiivisina pidettävät opiskelijanumerot listataan riveittäin. Mustalistaa ylläpidetään manuaalisesti.

4.3 Automatiikan toiminta ja rakenne

Automatiikka on rakennettu olio-periaatteiden mukaisesti. Kukin toiminto on jaettu oman luokan vastuulle. Koko automatiikka muodostuu viidestä luokasta, jotka ovat prosessin ohjaaja (*Manager*), syötteen käsittelijä (*CSVParser*), tiedostojen kirjoittaja (*Filer*), tapahtumien kirjaaja (*Logger*) sekä prosessin kannalta tärkeimmästä eli käyttäjäluokasta (*User*), joka huolehtii yksittäisten tilien luomisista ja ominaisuuksien päivittämisestä. Kuvion 4 luokkakaaviosta käyvät hyvin ilmi kunkin luokan ominaisuudet ja toiminnot, ja kuvion 5 sekvenssikaavio selventää luokkien keskinäistä vuorovaikutusta.



KUVIO 4. Automatiikan luokkakaavio.



KUVIO 5. Automatiikan sekvenssikaavio.

4.4 Ohjelmaluokat

4.4.1 Prosessin ohjaaja

Käyttäjätiliautomatiikkaa ohjaa yksittäinen hallintaolio, joka toimii säiliönä muille olioilla ja jakaa ohjelman vastuun näiden kesken. Ohjaajan vastuulla on ympäristö asetukset, mustalista, syötteen muutosten päivittäminen, syötteestä poistettujen käyttäjätilien passivoiminen sekä vanhojen, passiivisena yli puolitoistavuotta olleiden, tilien poistaminen.

Ympäristö ja automatiikan muut asetukset luetaan suorituksen alkuun. Asetuksista luetaan prosessin ympäristö, syötteestä kerättävät laitokset ja näistä pois jätettävät ryhmät sekä laitoskohtaiset kotihakemistot.

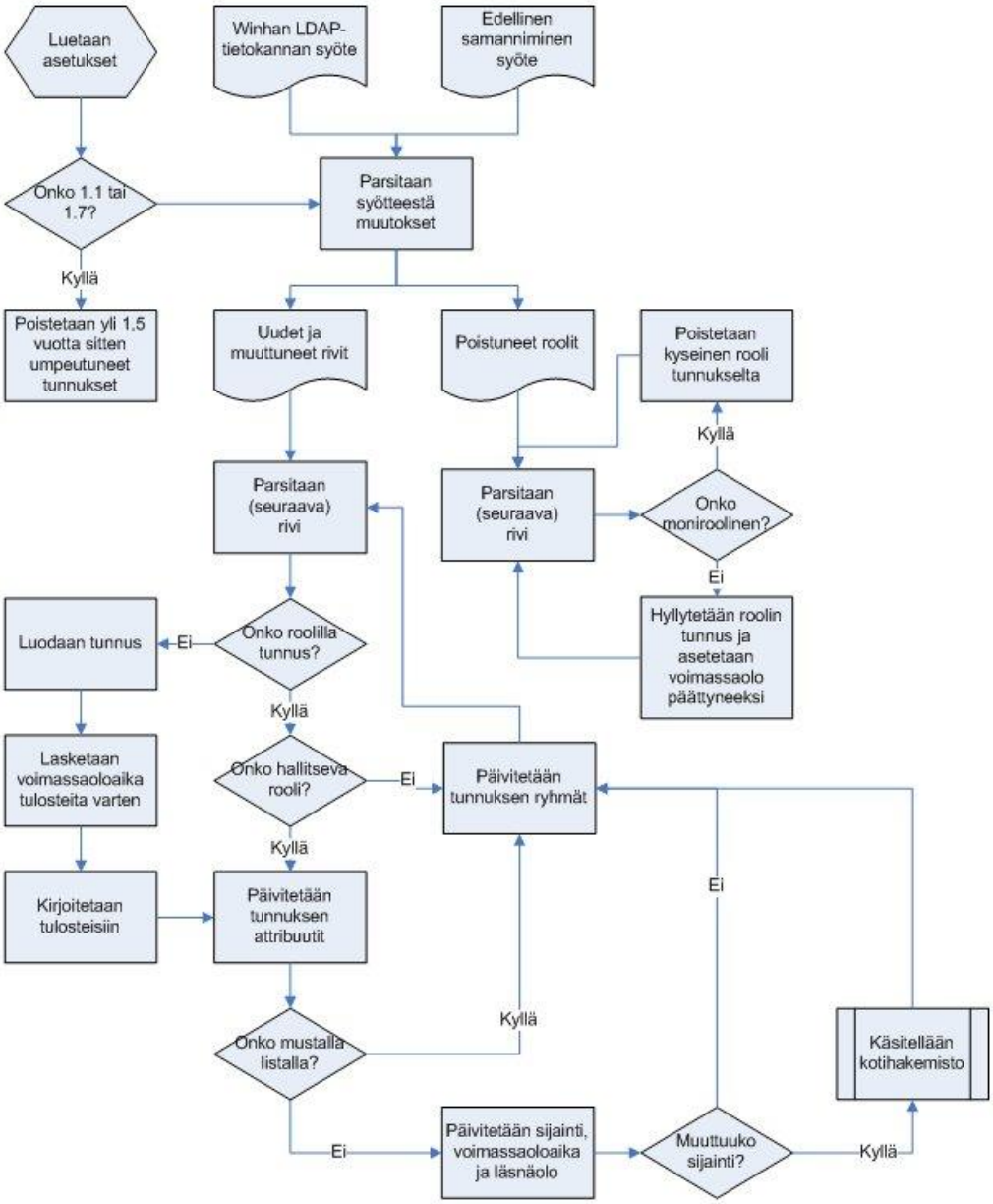
Asetusten luvun jälkeen luetaan ja käsitellään mustalista. Mustalistasta pidetään historiaa, jotta listalta poistetut tilit osataan aktivoida läsnäolokoodin mukaisesti. Mustalistalla olevat opiskelijat pidetään passiivisina. Käyttäjätilien käsittelyssä hyödynnetään käyttäjäluokkaa.

Vielä ennen päivityksiä tiedostojen kirjoittaja ottaa syötteestä ja muista käsittelytiedostoista varmuuskopiot.

Tämän jälkeen syötteenkäsittelijä kerää syötteestä muutokset. Ensin syötteestä pois jääneet tilit passivoidaan, minkä jälkeen muuttuneet rivit käsitellään yksitellen. Yksittäisistä käyttäjätilien päivityksistä huolehtii käyttäjäluokka, kaikista päivityksista lokia kirjoittaa tapahtumien kirjaaja ja tulosteista huolehtii tulosteiden kirjoittaja. Kunkin luokan tekemiset tarkennetaan omissa kohdissaan.

Poikkeuskäsittelynä vanhojen tunnusten poistaminen suoritetaan kahdesti vuodessa tammi- ja heinäkuun ensimmäisinä päivinä. Tällöin oppilastunnuksista poistetaan ne tunnukset, jotka ovat olleet passiivisina vähintään puolitoista vuotta.

Kuviossa 6 havainnollistetaan prosessikaaviolla automatiikan ja erityisesti prosessin ohjaajan toimintaa. Suoritus alkaa kuvion vasemmasta yläkulmasta asetusten lukemisella ja päättyy, kun kaikki muuttuneet syöterivit on käsitelty.



KUVIO 6. Automatiikan prosessikaavio.

4.4.2 Syötteen käsittelijä

Syötetiedosto on suodattamaton tekstimuotoinen tuloste *Winhan* koko oppilastietokannasta, joten ennen kuin syötettä voidaan käsitellä, tulee se suodattaa automatiikan parametrien mukaisesti.

Syötteen käsittelijä saa prosessin ohjaajalta asetuksista luetut laitos- ja ryhmäkoodit, joiden mukaan tämä parsii syötteen. Syötteen käsittelijä muodostaa

vertailun pohjalta suodatetun todellisen käyttäjäkantatiedoston, jossa on mukana vain haluttujen laitosten rivit poislukien automatiikan ulkopuolelle jätettävät ryhmät. Tämä tiedosto on varsinainen ajossa käytettävä tiedosto. Tämän suodatetun käyttäjäkantatiedoston viimeisin version tulee aina olla yhtenevä *aktiivihakemiston* käyttäjätilien kanssa.

Ensimmäiseksi edellä saatua todellista syötetiedostoa verrrataan edellisen ajon syötteeseen, josta poimitaan pois jääneet rivit. Tällaiset rivit, jotka eivät poistu syötteestä läsnäolokoodin vaihtumisella, ovat ns. vielailevia opiskelijarooleja ja ne asetetaan passiivisiksi välittömästi, kun niitä ei enää ole syötteessä mukana.

Tämän jälkeen syötteestä otetaan mukaan parametrien mukaiset laitokset, joista vielä suodatetaan pois poikkeusryhmät. Näin saadusta todellisesta syötteestä tutkitaan muutokset, jotka päivitetään *aktiivihakemistoon*.

Syötetiedoston suodattamisen ja muutosten etsinnän lisäksi syötteen käsittelijän vastuulla on tietää, mihin järjestykseen roolien ominaisuudet on kirjoitettu CSV-syötteeseen.

4.4.3 Tulosteiden kirjoittaja

Tiedostojen käsittelijä *Filer* huolehtii opintotoimiston tunnus- ja sitoumuslomakkeiden muodostuksesta. Ajoaikana kirjoittaja kerää kaikista luoduista käyttäjätileistä väliaikatiedostoja, joiden perusteella se lopulta viimeistelee tulosteet automatiikan ajokansioon opintotoimiston ja Unix-sähköpostin omiin alihakemistoihin.

Tarkempi kuvaus tiedostojen käsittelijän tulosteista löytyy kohdasta 4.2.3 Tuloste sähköpostipalvelimelle ja 4.2.4 Tunnus- ja sitoumuslomakkeet.

4.4.4 Tapahtumien kirjaaja

Tapahtumalokista huolehtii *Logger*-luokka. Tapahtumalokeja pidetään automatiikan toimintaympäristössä omassa alihakemistossa.

Lokiin kirjattavat tapahtumat jaetaan viiteen lokitiedostoon: AJOT, TUNNUKSET, KOTIHAKEMISTOT, TIEDOSTOT ja VIRHEET.

AJOT-tiedostoon kirjoitetaan automatiikan suorittamat toimenpiteet kuten suorituskerrat yleisellä tasolla. TUNNUS-tiedostoon kirjataan kaikki käyttäjätileihin kohdistuneet toimenpiteet kuten tilin luonti, aktivointi, passivointi ja siirto. KOTIHAKEMISTOT-tiedostoon kirjoitetaan kotihakemistojen käsittelyt, kuten luonnit, siirrot ja poistot. Automatiikan muodostamista tulostetiedostoista kirjoitetaan omaa lokitiedostoa TULOSTEET. Virheille on varattu VIRHEET-tiedosto, johon kirjoitetaan kaikki automatiikan havaitsemat virheelliset toimenpiteet.

4.4.5 Käyttäjä

Käyttäjätilikohtainen *aktiivihakemiston* käsittely on käyttäjäluokan (*User*) vastuulla. Kuitenkin vanhojen tilien poisto ja syötteestä arkistoituneiden roolien käsittelyt *aktiivihakemistoon* hoitaa prosessin ohjaaja (*Manager*), koska näissä käsittelyissä ei ole tarvetta luoda käyttäjäkohtaisia olioita.

Käyttäjäluokka huolehtii uusien käyttäjätilien luonneista, siirroista, ominaisuuksista, voimassaolosta sekä ryhmistä. Ryhmistä huolehtiessa tarkistetaan roolin syötetiedoston ryhmien olemassa olo *aktiivihakemistossa*. Mikäli syötteessä olevaa ryhmää ei *aktiivihakemistosta* löydy, luo käyttäjäluokka sen. Lisäksi käyttäjäluokan vastuulla on käyttäjien kotihakemistojen luonnit ja poistot sekä siirrot tarvittaessa.

Syötteessä esiintyy sekä oppilasrooleja että henkilökuntarooleja. Kunkin rivin kohdalla tämä tieto on ilmoitettu kentässä *eduPersonAffiliation*. Oppilas ja

henkilökuntaroolit eroavat käyttäjätunnusten muodustuksen ja voimossaolon osalta.

Uuden käyttäjätilin luonnin yhteydessä automatiikka muodostaa tilille käyttäjätunnuksen ja salasanan. Käyttäjätunnus muodostetaan oppilaille sukunimen neljästä ja henkilökunnalla viidestä ensimmäisestä kirjaimesta, mihin lisätään etunimen kirjaimia niin, että käyttäjätunnuksen enimmäispituus on kahdeksan merkkiä. Lisäksi käyttäjätunnuksesta korvataan skandinaaviset erikoismerkit vastaavilla perusaakkosilla. Esim. ä korvataan a:lla. Lisäksi käyttäjätunnusta muodostettaessa tarkistetaan, ettei *aktiivihakemistosta* jo löydy kyseistä käyttäjätunnusta. Jos se löytyy, korvataan tunnuksen viimeinen merkki ensimmäisellä vapaalla juoksevalla numerolla niin, ettei kahta samannimistä käyttäjätunnusta pääse syntymään.

Uusille käyttäjätileille muodostetaan satunnainen salasana, jossa on oltava isoja ja pieniä kirjaimia sekä vähintään yksi numero ja yksi erikoismerkki. Kuitenkaan puolipistettä (;) ja kenoviivaa (\) ei hyväksytä osaksi salasanaan. Lisäksi uusien käyttäjätilien käyttäjät pakotetaan ensimmäisellä kirjautumisella muuttamaan tämä salasana omaksi haluamakseen.

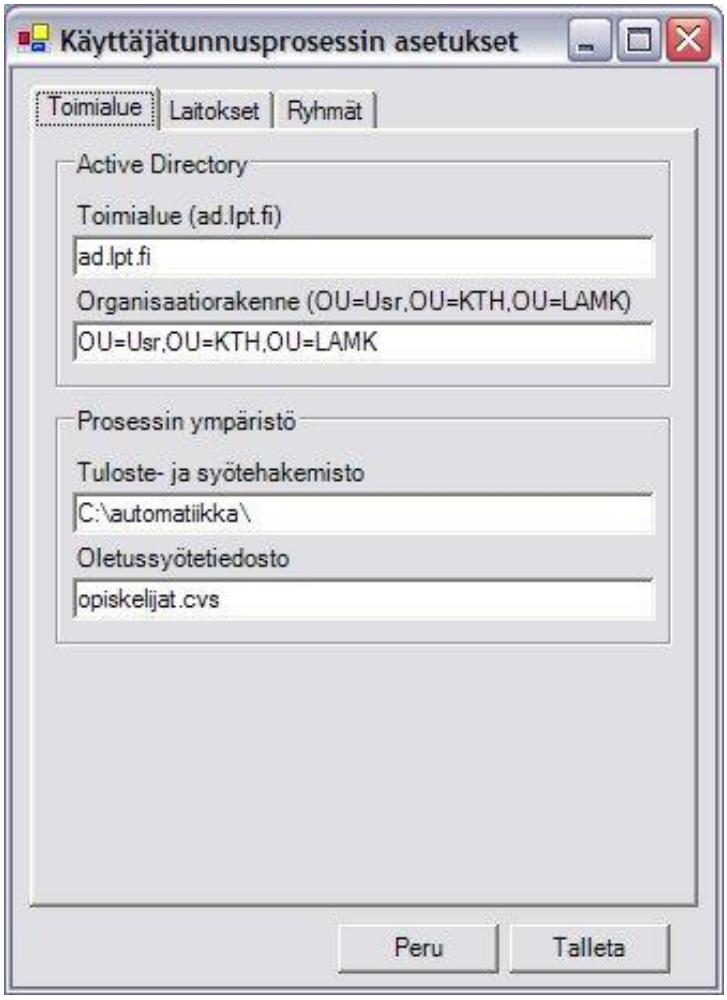
Winhassa yhdellä oppilaalla voi olla useita opiskelijanumeroita, jotka kukin esiintyvät syötteessä omana rivinään. *Aktiivihakemistoon* kyseiselle oppilaalle ei kuitenkaan haluta muodostaa käyttäjätiliä jokaista roolia kohden, joten käyttäjäluokka huolehtii moniroolisten opiskelijoiden kunkin roolin yhdistämisestä yhteen ja samaan käyttäjätiliin. *funetEduPersonId* on moniarvoinen ominaisuus *aktiivihakemistojen* käyttäjätileillä, mutta kyseiseen kenttään tallennetaan vain kyseisen oppilaan hallitseva opiskelijanumero. Moniroolisilla käyttäjillä muiden roolien opiskelijanumerot talletetaan W*inhan* skeeman ominaisuuteen nimeltä *winhaPersonStudentOtherIdList*. Ensisijainen hallitseva rooli käyttäjällä on viimeisin henkilökuntarooli. Jos käyttäjällä ei ole henkilökuntaroolia, asetetaan hallitsevaksi rooliksi viimeisin opiskelijarooli.

4.5 Asetukset

Automatiikka kehitettiin lopulta täysin riippumattomaksi organisaatiorakenteista, ja kaikki Lahden ammattikorkeakoulun omat tiedot haluttiin erottaa ohjelmakoodista. Tämä oli mahdollista vain joustavilla asetuksilla, joiden ansiosta automatiikka voidaan sellaisenaan ottaa käyttöön myös muissa laitoksissa ja organisaatioissa, joissa verkkoympäristön vaatimukset ovat yhteneviä Lahden ammattikorkeakoulun kanssa.

Asetuksilla voidaan muuttaa automatiikan syötetiedostoa sekä tuloste- ja syötehakemistoa. Samalla asetuksissa kerrotaan automatiikalle *aktiivihakemistossa* käytettävän käyttäjäkannan juuri eli säiliö, jonka alihakemistoissa käyttäjätilejä ylläpidetään. Lisäksi automatiikka lukee asetuksista laitokset, jotka otetaan mukaan syötteestä, sekä ryhmät, jotka jätetään sen ulkopuolelle.

Asetukset kirjoitetaan yksinkertaiseen tekstitiedostoon, jota ei kuitenkaan tarvitse käsitellä erillisillä editoreilla, vaan sen muokkaamiseen on oma graafinen sovellus ASETUKSET.EXE. Kuvio 7 havainnollistaa asetusohjelman käyttöliittymää.



KUVIO 7. Asetusten toimialue-välilehti.

TAULUKKO 1. Asetukset ja niiden käyttötarkoitus.

Välilehti	**Asetus**	**Käyttötarkoitus**
Toimialue	Toimialue	Osoite, jolla automatiikka ottaa yhteyden *aktiivihakemistoon*. Esim. ad.lpt.fi
	Organisaatiorakenne	Automatiikan äitisäiliö *aktiivihakemiston* käyttäjätileille. Esim. OU=Usr,OU=KTH,OU=PHKK
	Tuloste- ja syötehakemisto	Automatiikan ajohakemisto palvelimella. Tämän kansion alta automatiikka lukee syötteen ja kirjoittaa tulosteet. Esim. C:\automatiikka\
	Oletussyötetiedosto	*Winhasta* tulevan syötetiedoston nimi. Esim. syöte.csv
Laitokset	Laitos	Automatiikan huolehtiman laitoksen tunniste syötetiedostossa. Esim. 07TL
	Oppilashakemisto	Verkkopolku kyseisen laitoksen oppilaiden kotihakemistoille. Esim. \\aitta\07TL
	Luodaan opiskelijoille jako	Mikäli kyseisen laitoksen oppilaskotihakemistopalvelimella ei ole automaattista levyjaon luontia käytössä, tulee tämä vaihtoehto ottaa käyttöön.
	Henkilöhakemisto	Verkkopolku kyseisen laitoksen henkilökunnan kotihakemistoille. Esim. \\aitta\henkkunt
	Luodaan henkilökunnalle jako	Mikäli kyseisen laitoksen henkilökuntakotihakemistopalvelimella ei ole automaattista levyjaon luontia käytössä, tulee tämä vaihtoehto ottaa käyttöön.
Ryhmät	Saapumisryhmän koodi	Automatiikan ulkopuolelle jätettävä ryhmä. Osa saapumisryhmän tunnuksesta riittää. Esim. OPETU jättää käsittelemättä ryhmät 06OPETU, 07OPETU, jne.

4.6 Käyttöönotto

4.6.1 Skeemalaajennukset

Automatiikan lisäksi työn määrityksiin kuului *Funetin* ja *Winhan* skeeman lisääminen *aktiivihakemistoon*.

Funetin skeemasta on saatavilla LDIF-muotoinen tiedosto. *Funetin* skeema sisältää laajennetun funetEduPerson -luokan, jonka ominaisuuksia ovat mm. Suomen henkilötunnus, suuntautumisvaihtoehto, koulutusohjelma sekä opiskelijanumero. Ensisijaisesti *Funetin* skeema on tarkoitettu korkeakoulurajojen ylittäviin käyttäjätietokantojen yhteensopivuuden parantamiseen.

Winhan skeema puolestaan sisältää WINHAPRO -ohjelman käyttämän oman laajennnuksen, joka löytyi myös LDIF-muotoisena tiedostona. Winhan skeema on oikeastaan laajennos Funetin skeemaan tarjoten lähinnä läsnäolokoodin ja verkon käyttöoikeudet uusina automatiikassa käytettyinä ominaisuuksina.

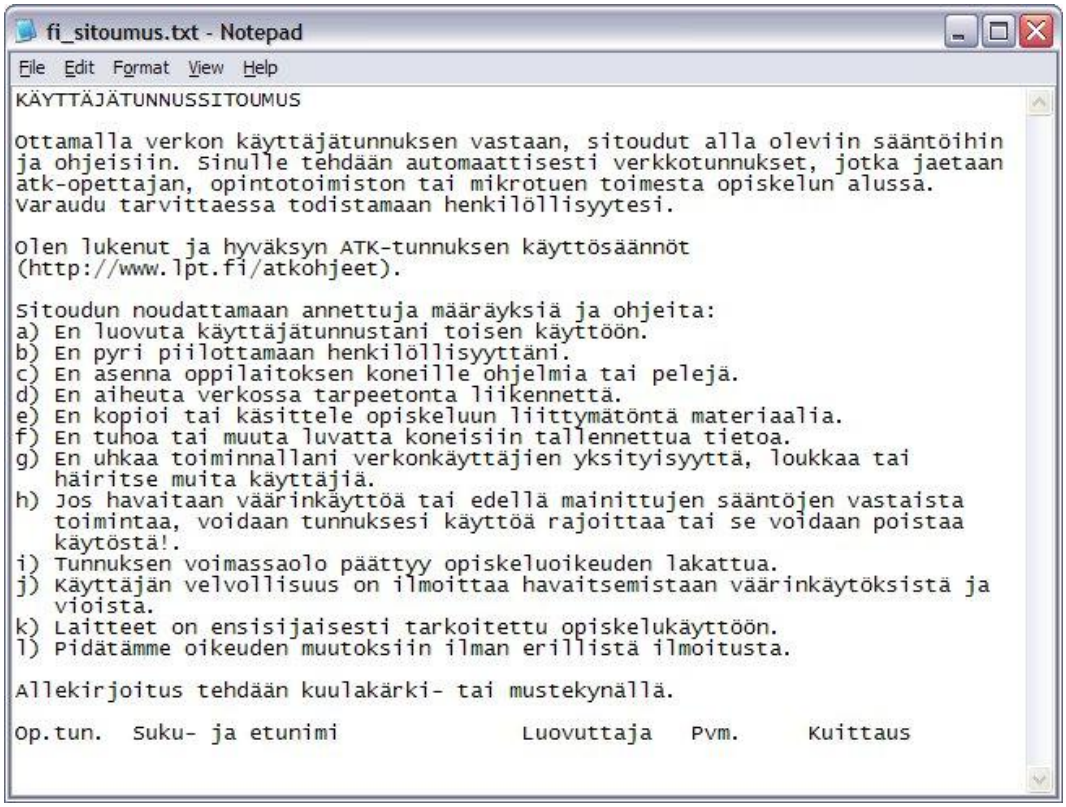
LDIF-muotoiset skeeama voidaan ottaa hakemistoissa käyttöön komennolla LDIFDE, joka on tarkoitettu tiedon tuomiseen ja viemiseen hakemistoista. LDIFDE komennolla tuodaan tietoa *aktiivihakemistoon* ”-i” -argumentilla. Skeeman muokkauksen yhteydessä on hyvä ottaa *aktiivihakemistosta* ensin varmuuskopio. Skeeman tuominen onnistuu yksinkertaisesti komennolla ”ldifde -i skeeamatiedosto.ldif”. Komennon ajamiseen luonnollisesti tarvitaan oikeudet muokata *aktiivihakemiston* skeemaa.

4.6.2 Asennus

Automatiikalla ei ole erillistä asennusohjelmaa, vaan asennus onnistuu purkamalla binääripaketin haluttuun paikkaan ja suorittamalla ajotiedosto samasta kansiosta.

Ennen ensimmäistä ajoa tarvitsee kuitenkin vielä muodostaa asetustiedosto, joka kannattaa tehdä omalla asetusohjelmalla ASETUKSET.EXE.

Lisäksi ajokansiossa on oltava sitoumuslomakkeiden tekstimuotoiset mallitiedostot EN_COMMITMENTS.TXT ja FI_SITOUMUS.TXT.



KUVIO 8. Esimerkki sitoumuslomakkeesta.

4.6.3 Ajastus

Asennuskansiossa oleva TUNNUSAUTOMATIIKKA.EXE on varsinainen prosessin ajotiedosto, joka ajastetaan käyttöjärjestelmän ajastustoiminnolla. Ajastuksen yhteydessä tulee määrittää myös käyttäjätili, jolla prosessia suoritetaan.

Tällä tilillä on oltava riittävän laajat oikeudet käyttäjäkannan tilien luomiseen, siirtämiseen, muokkaamiseen ja poistamiseen. Lisäksi tilillä on oltava oikeudet myös verkkojakojen kotihakemistojen ja -jakojen luomiseen.

Lahden ammattikorkeakoulussa automatiikka ajastettiin ajamaan kerran vuorokaudessa, aamuisin sen jälkeen, kun *Winhasta* on ensin ajettu ajastetusti päivän syöte automatiikan ympäristöön.

Mikään ei estä prosessia ajastetusti ajettavaksi useampaankin otteeseen vuorokaudessa, mutta prosessia on turha ajaa useammin, kuin lähdetietokannasta ajetaan syöte prosessille.

4.6.4 Käyttöönoton vaiheistus

Automatiikan asennus ja ajastaminen eivät vielä päivitä käyttäjäkantaa uuteen rakenteeseen, vaan myös vanhat käyttäjätilit on saatava mukaan automatiikkaan, jotta nämä vastaisuudessa päivittyvät ja poistuvat a*ktiivihakemitosta,* kuten uudet käyttäjätilit. Tämän johdosta automatiikan käyttöönotto vaiheistettiin niin, että ensimmäisessä vaiheessa automatiikka käsitteli vain uusia käyttäjätilejä.

Koko syötteen käsittely edellyttää, että myös vanhoille käyttäjätileille lisätään *Funetin* skeeman mukainen opiskelijanumero. Osalla vanhoista käyttäjätileistä opiskelijanumero löytyi valmiiksi hakemistosta erinäisistä lisäominaisuuksista, joista kyseinen arvo asetettiin *funetEduPeronStudentId*-ominaisuuteen. Osalla vanhoista käyttäjistä jouduttiin keräämään opiskelijanumerot manuaalisesti, jolloin nämä käyttäjätilit kerättiin ensin *aktiivihakemistosta* erilliseen listaan, jonka ylläpito joutui täyttämään keräämällä kyseisten tilien opiskelijanumerot ryhmien ohjaajilta tai muulta henkilökunnalta.

Viimeinen versio automatiikasta otettiin käyttöön tammikuun viimeinen päivä 2005. Ohjelmallisia ongelmia automatiikassa ei ole tekijän tietojen mukaan ilmennyt tämän jälkeen.

5 JOHTOPÄÄTÖKSET

5.1 Automatiikka pähkinänkuoressa

Käyttäjätiliautomatiikka on koulutuskonsernin omasta tarpeesta esille noussut koulun tietoverkon ylläpitoa helpottava prosessi, jonka tavoitteena on yhtenäistää opintotoimiston oppilastietokanta ja koulun tietoverkon käyttäjätilit. Perusperiaatteeltaan automatiikka toimii rajapintana kahden tietokannan välillä, jolloin sen tehtävä on ylläpitää *aktiivihakemistoa Winhan* päivittäisten tulosteiden mukaisesti.

Automatiikka on tiedostopohjainen ajastettu prosessi, jonka omia asetuksia muuttamalla voidaan vaikuttaa syötetiedoston suodattamiseen itse syötteeseen koskematta. Automatiikka on toteutettu ADSI-ohjelmointirajapinnalla, joka mahdollistaa kieliriippumattoman lähestymistavan prosessin jatkokehittämiseen. Automatiikka kirjoitettiin Visual Basicin .NET -versiolla.

5.2 Tavoitteiden täyttyminen

Automatiikka onnistui hyvin täyttämään alkuperäiset määritykset sekä vielä ensimmäisen käyttöönoton jälkeen tulleet jatkotoiveet, joita esim. mustalista ja sitoumuslomakkeet olivat. Lopputulos oli ohjelma-arkkitehtuuriltaan ehkä perinteisempi sovellus kuin alkuun oli ajateltu, koska alkuun puhuttiin vain skriptistä, ja lopputulos olikin käännetty binääriohjelma laajoine parametrointimahdollisuuksineen.

Jälkiviisaana olisi ollut syytä alun alkaen suunnitella hieman tarkemmin syötekysely *Winhan* käyttäjätietokannasta, koska syötteessä oli mukana suuri määrä rooleja, joita ei haluttu ottaa mukaan *aktiivihakemistoon*. Lopulta automatiikkaan jouduttiin lisäämään paljon älyä syötteen käsittelyyn, vaikka sama

äly olisi voitu korvata järkevästi *Winhan* tulosteen suodattamisella sitä ajettaessa *Winhasta*.

Ohjelmistorojektin läpivienti kehitti tekijän ammattitaitoa kiitettävästi. Erityisesti tiedonhaku- sekä ongelmanratkaisu taidot kehittyivät itsenäisessä työnteossa. Lisäksi koko harjoitteluaika laajensi näkemystä suurten organisaatioiden sovellusympäristöistä ja -tarpeista.

5.3 Automatiikan tulevaisuuden näkymiä

Automatiikka otettiin käyttöön Lahden ammattikorkeakoulun lisäksi Päijät-Hämeen koulutuskonsernin toisessa liikelaitoksessa, Koulutuskeskus Salpauksessa, jossa opiskelijoita on vuosittain lähes 8000. Salapauksen automatiikka toimii samassa *aktiivihakemistossa* kuin Lahden ammattikorkeakoulun automatiikka. Tämän johdosta automatiikkaa jouduttiin hieman muokkaamaan Salpauksen tarpeisiin, jotteivät päällekkäiset opiskelijanumerot aiheuta ongelmia yhteisessä *aktiivihakemissa* Lahden ammattikorkeakoulun kanssa. Lisäksi muut koulut ympäri Suomea ovat olleet kiinnostuneita automatiikan käyttöönotosta omissa laitoksissaan. (Päijät-Hämeen koulutuskonserni 2007.)

Automatiikka olisi sellaisenaan varsin käytettävä rajapintaprosessi minkä hyvänsä kahden käyttäjätietokannan välillä. Tässä opinnäytetyössä automatiikka toteutettiin Lahden ammattikorkeakoulun opiskelijoiden ja osittain myös henkilökunnan käyttäjätilien automatisointiin, mutta tulevaisuuden erilaisissa pseudoverkoissa vastaavat käyttäjätilien yhtenäistämiset lienevät oleellinen osa käyttäjien personointia ja erilaisten palveluiden saatavuutta.

Vastaavaa yhtenäistämistä voisi ajatella kirjastojen ja muiden opiskeluun läheisesti liittyvien julkishallinnollisten palveluiden ja koko maanlaajuisen opetushallinnon opiskelijarekistereiden yhtenäistämisellä organisaatiorajat

ylittäväksi käyttäjien tunnistautumiseksi. Suunta on ilmeinen, mutta prosessien toteuttaminen laajalti hajautettuun verkkoympäristöön on vielä työn alla.

LÄHTEET

ADSI Architecture. 2000. Microsoft Corporation. Saatavissa: http://www.microsoft.com/technet/scriptcenter/guide/sas_ads_ecfv.mspx?mfr=true

Creating a User. 2004. Microsoft Corporation. Saatavissa: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ad/ad/creating_a_user.asp

CSC Tieteen tietotekniikan keskus. 2004. LDAP-hakemistopalvelu [viitattu 7.2.2004]. saatavissa: http://www.csc.fi/suomi/funet/ldap.html

Mikä Känkänen. 2004. Aktiivihakemiston käyttöönotto Lahden ammattikorkeakoulussa

Päijät-Hämeen koulutuskonserni. 2004. Tunnuslukuja Päijät-Hämeen koulutuskonsernista. Saatavissa: http://www.phkk.fi/esittely/tunnusluvut/tunnusluvut.html

Sakari Kouti & Mika Seitsonen. 2002. Active Directory Schema. Saatavissa: http://www.awprofessional.com/articles/article.asp?p=26136&seqNum=3&rl=1

Simon Robinson. 1999. Professional ADSI Programming. Wrox Press, USA

Ted Pattison. 2001. Visual Basic .NET: New Programming Model and Language Enhancements Boost Development Power. Saatavissa: http://msdn.microsoft.com/msdnmag/issues/01/02/vbnet/default.aspx

Visual Studio .NET Dokumentaatio. 2004.

William Boswell. 2003. Understanding Active Directory Services [viitattu 7.2.2005]. Saatavissa: http://www.windowsitlibrary.com/Content/716/06/toc.html