Yhteenveto

Oppimistavoitteiden hallintajärjestelmä harri

Helsinki 14.12.2007

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (9 op)

Projektiryhmä

Petri Kinnunen
Lasse Leino
Anne Pääkkö
Minna Ulmala

Asiakas

Harri Laine

Johtoryhmä

Kimmo Simola, vastuuhenkilö
Aleksi Yrttiaho, ohjaaja

Kotisivu

`http://www.cs.helsinki.fi/group/harri`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
1.0	14.12.2007	Dokumentti valmis
0.3	11.12.2007	Dokumenttia tiivistetty
0.2	10.12.2007	Päättöanalyysi ja yhteenveto
0.1	10.12.2007	Dokumenttitiivistelmät

Sisältö

1 Johdanto

Tämä dokumentti antaa hyvän yleiskuvan Helsingin yliopiston Tietojenkäsittelytieteen laitokselle tehdystä Oppimistavoitteiden hallintajärjestelmästä eli harrista sekä ohjelmistotuotantoprojektista, jossa järjestelmä tehtiin syksyllä 2007. Dokumentti sisältää sanaston, lyhyen kuvauksen järjestelmästä, tiivistelmät kaikista projektin aikana tuotetuista dokumenteista, sekä päättöanalyysin, että yhteenvedon.

2 Sanasto

2.1 Vaatimusmäärittelyyn liittyvät termit

Oppimistavoite on sanallinen kuvaus siitä mitä opiskelijan tulee oppia kurssilla.

Taso luokittelee oppimistavoitteen kolmeen: lähestyy (läpäisee kurssin), saavuttaa (oppii kurssin asiat), syventää (syventää osaamistaan kurssin oppien yli).

Esitietovaatimus on kurssin esitietoina vaadittava osaaminen, jotta kurssin pystyy läpäisemään.

Teema on kurssilla opittava käsite tai asiakokonaisuus. Teemalla voi olla oppimistavoitteita ja se voi jakautua useampiin teemoihin.

Tilaa käytetään kurssin näkyvyyden säätelyyn. Niitä ovat muokattava, suojattu, tarkastettava, hylätty ja julkinen.

Syvyys on oppimistavoitteen ja esitietovaatimuksen ominaisuus, jota kuvataan kuusiportaisella Bloomin taksonomialla: ulkoa muistaminen, alkeellinen ymmärrys, soveltaminen, analysointi, syntetisointi ja evaluointi.

Kuvaus on esitietovaatimusta, teemaa tai oppimistavoitetta kuvaava teksti.

Opiskelija on järjestelmän loppukäyttäjä, joka tutkii julkaistuja kursseja, niiden esitietovaatimuksia, teemoja ja oppimistavoitteita. Opiskelija ei voi muuttaa kurssin tilaa.

Muokkaaja on vastuuhenkilön nimeämä henkilö, joka voi lisätä ja muokata kurssin sisältöä, kun kurssi on *muokattavassa* tilassa. Muokkaaja ei voi muuttaa kurssin tilaa.

Tarkastaja voi julkaista kurssin kaikille nähtäväksi tai hylätä sen, jolloin se palautetaan takaisin suojattuun tilaan vastuuhenkilön muokattavaksi.

Vastuuhenkilö vastaa ja syöttää pääosan kurssin sisällöstä sekä voi halutessaan nimetä kurssille muokkaajia. Vastuuhenkilö voi siirtää kurssin muokattavaan, suojattuun tai tarkastettavaan tilaan. Jokaisella kurssilla on oma vastuuhenkilönsä.

Järjestelmävastaava myöntää oikeudet tarkastajille ja vastuuhenkilöille järjestelmässä. Järjestelmävastaavalla on oikeudet kaikkiin järjestelmän tietoihin ja toimintoihin.

2.2 Suunnitteluun liittyvät termit

JSP eli Java Server Pages on Java-teknologia, jonka avulla voidaan dynaamisesti luoda HTML, XML tai muun tyyppisiä dokumentteja vastauksena Web-asiakkaan pyyntöön. Teknologia erottaa käyttöliittymän sisällöntuotannosta sallien muutokset käyttöliittymän ulkoasuun ilman että dynaamista sisältöä pitää muuttaa.

Java Servletit ovat alustasta riippumattomia palvelimella olevia moduuleita, joiden avulla saadaan lisättyä dynaamista sisältöä Web-palvelimelle. Servletit voivat hallita tiloja useiden palvelintransaktioiden läpi käyttäen evästeitä, istuntomuuttujia tai URL-uudelleenkirjoitusta.

MVC eli Model-View-Controller on arkkitehtuurityyli, jossa ohjelma jaetaan kolmeen osaan: malliin, näkymään ja ohjaimeen. Näiden tarkoitus on erottaa käyttöliittymä varsinaisesta toimintalogiikasta ja tietosisällöstä.

OSJ eli Opetuksensuunnittelujärjestelmä on Helsingin yliopiston Tietojenkäsittelytieteen laitoksen tietojärjestelmä, jossa on laitoksella työskentelevien henkilöiden ja luennoitavien kurssien tiedot.

Versio on kurssiversion eri tilojen jaottelu kahteen ryhmään, julkiseen ja työversioon. Ryhmään julkinen kuuluu vain kurssiversion tila julkinen ja ryhmään työversio kuuluvat muut kurssiversion tilat: muokattava, suojattu, tarkastettava ja hylätty

2.3 Testaukseen liittyvät termit

EUCT eli Extended Use Case Testing on järjestelmätestauksen suunnittelumalli, jossa vaatimusdokumentissa kuvatuista käyttötapauksista tehdään ns. laajennettuja käyttötapauksia.

JCoverage on lausekattavuuden laskemiseen tarkoitettu työkalu.

JUnit on Java-ohjelmointikielelle tarkoitettu testaustyökalu.

Mustalaatikkotestaus eli Black box testing on testausta, jossa komponenttia testataan tietämättä sen toteutuksen yksityiskohtia. Testaus perustuu syöte- ja tulostetietojen tarkasteluun.

Top-down -menetelmässä komponentit ja osajärjestelmät integroidaan ylhäältä alas eli saadaan nopeasti näkyviin järjestelmän runko ja käyttöliittymä, vaikka mitään todellista toiminnallisuutta näiden alla ei ole.

Tynkä on testauksessa käytettävä komponentti, joka kuvaa pelkistetysti alemman komponentin toimintaa.

3 Oppimistavoitteiden hallintajärjestelmä

Oppimistavoitteiden hallintajärjestelmän avulla voidaan määritellä ja hallita kurssien kolmitasoisia oppimistavoitteita. Järjestelmän käyttäjät, opiskelijat ja opettajat, voivat selata järjestelmän kursseja hakuehtoja tai kurssilistauksia käyttämällä. Oppimistavoitteiden

muokkaus ja kehitys on myös keskitetty samaan järjestelmään.

Järjestelmä näyttää kurssien oppimistavoitteet ja esitietovaatimukset hierarkisina listoina. Oppimistavoitteiden kohdalla voidaan tarkastella mihin teemoihin kurssi jakautuu. Oppimistavoitteita on kolmen tasoisia: lähestyy, saavuttaa ja syventää, jotka kuvaavat kuinka keskeinen ja välttämätön oppimistavoite on kurssin suorittamisen kannalta.

Esitietovaatimusten listauksessa näytetään kurssit, jotka tulee suorittaa ennen valitun kurssin suorittamista. Järjestelmä tarjoaa suorat linkit esitietoina vaadittaviin kursseihin järjestelmässä ja näyttää mihin kurssin osa-alueeseen (teemaan) esitieto liittyy.

Kurssien muokkaajat pääsevät muuttamaan niiden kurssien tietoja, joihin heillä on oikeudet. He voivat ylläpitää esitietovaatimuksia ja oppimistavoitteita. Lisäksi he voivat määrittää esitietovaatimuksille ja oppimistavoitteille syvyyden. Vertaamalla esitietovaatimuksen ja oppimistavoitteen syvyyttä saadan tarkistettua, ettei esitietovaatimuksessa vaadita asian syvempää osaamista kuin edeltävällä kurssilla on ollut tarkoitus oppia.

Jokaisella kurssilla on vastuuhenkilö, joka voi ylläpitää kurssia ja antaa kurssin muokkausoikeudet haluamilleen henkilöille sekä muuttaa kurssin suojattuun tilaan, kun hän ei enää halua muiden muuttavan kurssia. Kun kurssi ja sen esitietovaatimukset, teemat ja oppimistavoitteet ovat valmiit, vastuuhenkilö siirtää kurssin tarkastettavaan tilaan.

Opetushallinnon henkilö, jolla on valtuudet päättää kurssin oppimistavoitteiden julkaisemisesta, näkee tarkastettavassa tilassa olevat kurssit. Hän siirtää kurssin julkaistuksi tai hylätyksi, jolloin kurssi palaa suojattuun tilaan kurssin vastuuhenkilön muutettavaksi. Kun kurssi on julkaistu, se näkyy opiskelijoille ja opettajille.

Järjestelmävastaava antaa vastuuhenkilöille ja tarkastajille oikeudet ja hänellä on oikeudet kaikkiin järjestelmän toimintoihin. Järjestelmävastaavan, vastuuhenkilön ja tarkastajan on kirjauduttava järjestelmään voidakseen käyttää käyttäjäryhmänsä mukaisia toimintoja.

4 Dokumenttitiivistelmät

4.1 Projektisuunnitelma

4.1.1 Projektiorganisaatio

Projektiorganisaatioon kuuluu neljä projektitiimin jäsentä, asiakas Harri Laine sekä projektin ohjaaja Aleksi Yrttiaho ja Ohjelmistotuotantoprojektin vastuuhenkilö Kimmo Simola.

Vastuualue	Vastuuhenkilö
Projektipäällikkö	Lasse Leino
Varaprojektipäällikkö	Anne Pääkkö
Vaatimusmäärittelyvastaava	Minna Ulmala
Suunnittelu- ja testausvastaava	Anne Pääkkö
Dokumentti- ja koodivastaava	Petri Kinnunen
Kotisivu- ja SVN-vastaava	Petri Kinnunen

4.1.2 Riskianalyysi

Projektikohtaisia riskejä on 16 kappaletta ja tuotekohtaisia kaksi. Niistä on selvitetty nimi, todennäköisyys, vakavuusaste ja vastatoimet. Sekä todennäköisyyden että vakavuusasteen hahmottamisessa on käytetty viisiportaista asteikkoa.

4.1.3 Laitteisto- ja ohjelmistoympäristön vaatimukset

Järjestelmästä kehitettävä protyyppi ohjelmoidaan Javalla ja se käyttää hyväkseen tietokantayhteyttä. Käyttöliittymä toteutetaan selainympäristöön. Järjestelmän lähdekoodin kehitykseen Eclipseä ja SVN-versionhallintaa. Kaikki dokumentit ladotaan lopulliseen muotoonsa LaTeX:illa. Muistiot ja pöytäkirjat saavat olla ASCII-tekstinä.

4.1.4 Koko- ja kustannusarviot

Projektin funktiopisteitä on yhteensä 49 kappaletta ja niiden kertoimien summaksi saadaan 226. Järjestelmän yleispiirteiden kertoimien summaksi tulee 37. Funktiopistearvoksi saadaan täten noin 230.

Ohjelmiston toteutuskielinä ovat: Java + HTML + CSS + SQL. Lopullinen SLOC (Source Lines of Code) arvo on funktiopistearvon avulla laskettuna 7146,12.

Tarkempi analyysi löytyy projektisuunnitelmasta.

4.1.5 Työn ositus

Projektissa noudatetaan vesiputousmallia ja se jakautuu mallin mukaisiin vaiheisiin. Vaatimusten esiinkaivelu tehdään pääosin asiakas- ja asiantuntija tapaamisten avulla. Lopulta vaatimukset priorisoidaan, dokumentoidaan ja hyväksytetään asiakkaalla.

Suunnittelussa edetään arkkitehtuurisuunnittelusta komponenttisuunniteluun. Komponenttisuunnittelu jakautuu tietokantasuunnitteluun, käyttöliittymäsuunnitteluun ja integrointisuunnitteluun. Lopuksi siirrytään luokkasuunnitteluun. Testaussuunnitelmaa laaditaan suunnittelun ohella.

Toteus aloitetaan suunnitteludokumentin mukaisesti. Testausta tehdään tätä vastoin testaussuunnitelman mukaisesti. Puutteet kirjataan ylläpitodokumenttiin.

Projekti viimeistellään ja luovutetaan asiakkaalle.

4.1.6 Aikataulu

Projekti alkoi 3.9.2007 ja se kestää 14 viikkoa. Projekti valmistuu 14.12.2007. Tarkempi aikataulu ja vaiheiden eteneminen löytyy projektisuunnitelmasta.

4.1.7 Seuranta- ja raportointimenetelmät

Projektisuunnitelmaa päivitetään koko projektin ajan. Joka viikon ensimmäisessä palaverissa pidetään aluksi seurantakatsaus, jossa jokainen kertoo, mitä on viime viikolla saanut aikaan ja mitkä asiat vaativat vielä työtä.

4.2 Vaatimusdokumentti

Vaatimusdokumentti kuvaa Oppimistavoitteiden hallintajärjestelmän vaatimukset ja toimii sopimuksena järjestelmän tilaajan ja projektiryhmän välillä siitä, millainen järjestelmä tilaajalle tehdään ja toimitetaan. Oppimistavoitteiden hallintajärjestelmän avulla ylläpidetään Helsingin yliopiston Tietojenkäsittelytieteen laitoksen kurssien oppimistavoitteita. Järjestelmä tarjoaa kurssien suunnittelijoille, eli opettajille, mahdollisuuden kurssin teemojen hierarkiseen esittämiseen ja kolmitasoisten oppimistavoitteiden liittämisen niihin. Kurssitietoihin voidaan liittää myös esimerkkejä ja tenttikysymyksiä. Järjestelmä mahdollistaa esitietovaatimusten ja oppimistavoitteiden vastaavuuden valvomisen kurssien välillä. Muille opettajille ja opiskelijoille järjestelmä näyttää selkeän esityksen kurssien oppimistavoitteista ja esitietovaatimuksista.

Järjestelmä integroidaan Tietojenkäsittelyteiteen laitoksen Opetuksensuunnittelujärjestelmän kanssa, josta saadaan laitoksen kurssien koodit ja nimet sekä laitoksella käytössä olevat käyttäjätunnukset, nimilyhennetunnukset ja käyttäjien nimet.

Vaatimusdokumentissa esitetyt vaatimukset on kerätty asiakkaan ja projektiryhmän yhteisissä tapaamisissa sekä projektiryhmän ja asiantuntija Heikki Lokin tapaamisessa. Asiakas on hyväksynyt vaatimusdokumentin.

4.2.1 Järjestelmävaatimukset

Toiminnalliset vaatimukset

Seuraavaksi esiteltävät toiminnot on ryhmitelty käyttäjäryhmittäin ja ne ovat ensimmäisen prioriteetin toimintoja (poikkeuksena haku), jotka ovat järjestelmän toiminnan kannalta kriittisiä ja jotka toteutettiin lopullisessa prototyypissä. Pienemmän prioriteetin toiminnalliset vaatimukset löytyvät vaatimusdokumentista.

Toiminnot kaikille

- Kurssin löytäminen listasta
- Esitietovaatimusten valinta
- Haku

Muokkaaja, vastuuhenkilö, tarkastaja ja järjestelmävastaava

- Kirjautuminen

- Uloskirjautuminen

Muokkaaja, vastuuhenkilö ja järjestelmävastaava

- Pääteeman lisääminen ja poistaminen
- Alateeman lisääminen ja poistaminen
- Oppimistavoitekstin lisääminen ja poistaminen
- Oppimistavoitteen tason ja syvyyden valinta
- Esitietovaatimuksen lisääminen ja poistaminen
- Esitietovaatimuksen syvyyden valinta
- Esitietovaatimuksen kytkeminen kurssiin tai teemaan

Vastuuhenkilö ja järjestelmävastaava

- Kurssin tilan muutos muokattavaksi, suojatuksi tai tarkastettavaksi
- Muokkaajan nimeäminen ja poistaminen

Tarkastaja ja järjestelmävastaava

- Kurssitietojen julkaisu
- Kurssin tilan muutos suojatuksi

Järjestelmävastaava

- Kurssin poisto julkisesta tilasta
- Vastuuhenkilön luominen ja poistaminen
- Tarkastajan luominen ja poistaminen

Ei-toiminnalliset vaatimukset

Ei toiminnallisissa vaatimuksissa on ensimmäisellä prioriteetilla mm. oppimistavoitteiden kuvausten kaksi kielisyys (suomi - englanti) sekä kiinnitetty huomiota järjestelmän turvallisuuteen. Samasta kurssista on järjestelmässä sekä julkinen, että työversio. Selainympäristössä toimiva käyttöliittymä on suomeksi.

Järjestelmäarkkitehtuuri ja järjestelmämalli

Järjestelmän arkkitehtuurina käytetään MVC-mallia. Tietovarastoina toimivat harri-järjestelmän oma tietokanta, sekä OSJ.

4.3 Suunnitteludokumentti

Suunnitteludokumentissa kuvataan, miten harri-järjestelmä tullaan toteuttamaan. Suunnittelun tuloksena saadaan kuvaukset käytetystä arkkitehtuurityylistä, osajärjestelmistä, kaikista luokista, tietokannasta ja integroinnista OSJ:ään.

Projektiryhmä suunnittelee ja toteuttaa Vaatimusdokumentissa 1. prioriteetilla sekä osan 2. prioriteetilla kuvatuista toiminnoista.

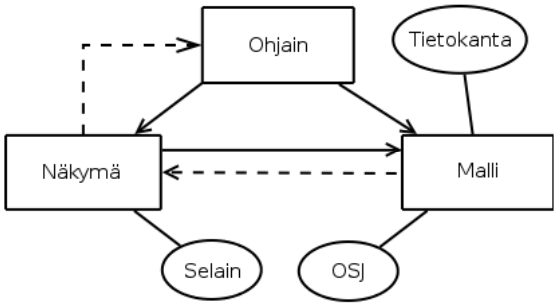
4.3.1 Toteutuskieli ja -ympäristö

Ohjelmisto toteutetaan Java-ohjelmointikielen versiolla 1.5 . Koska järjestelmä on tietokantapohjainen websovellus, käyttöliittymän toteutuksessa käytetään JSP-tekniikkaa. Käyttöliittymän ulkonäköön käytetään CSS-tyylimäärittelyjä. Tietokanta tehdään Oracle:n versiolla 10g Enterprise Edition Release 10.2.0.3.0. Tietokantaa käsittelevät luokat toteutetaan Java Servletteinä, joissa ovat SQL-lauseet.

4.3.2 Arkkitehtuurisuunnittelu

Osajärjestelmät

MVC-arkkitehtuurityylin mukaisesti järjestelmä koostuu kolmesta osajärjestelmästä: *näkymästä, ohjaimesta* ja *mallista*. Näiden lisäksi järjestelmässä on kaksi muuta osajärjestelmää (kuva 1): *tietokanta* ja ulkoinen järjestelmä *OSJ*.



Kuva 1: Järjestelmän arkkitehtuuri

4.3.3 Komponenttisuunnittelu

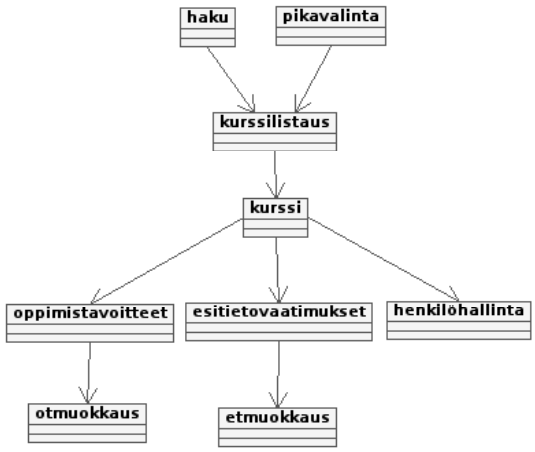
Malli koostuu Java Servleteistä ja Java-luokista. Komponentin servletteihin on toteutettu tietokantaa muokkaavat tai tietoja kyselevät SQL-lauseet.

Näkymä koostuu JSP-tiedostoista.

Ohjaimessa on vain yksi komponentti, joka vastaanottaa näkymältä tulevat muokkauspyynnöt ja lähettää nämä eteenpäin mallille.

Käyttöliittymäsuunnittelu

Käyttöliittymä koostuu yhdeksästä käyttöliittymäkomponentista, joita ovat: haku, pikavalinta, kurssilistaus, kurssi, oppimistavoitteet, oppimistavoitteiden muokkaus, esitietovaatimukset, esitietovaatimusten muokkaus ja henkilöhallinta (kuva 2). Komponentit ovat käyttöliittymän osia, joita yhdistelemällä muodostetaan käyttäjälle näkyviä sivuja.



Kuva 2: Käyttöliittymän sivukartta

Käyttöliittymäsuunnitelman tarkempi kuvaus löytyy suunnitteludokumentista.

Tietokantasuunnittelu

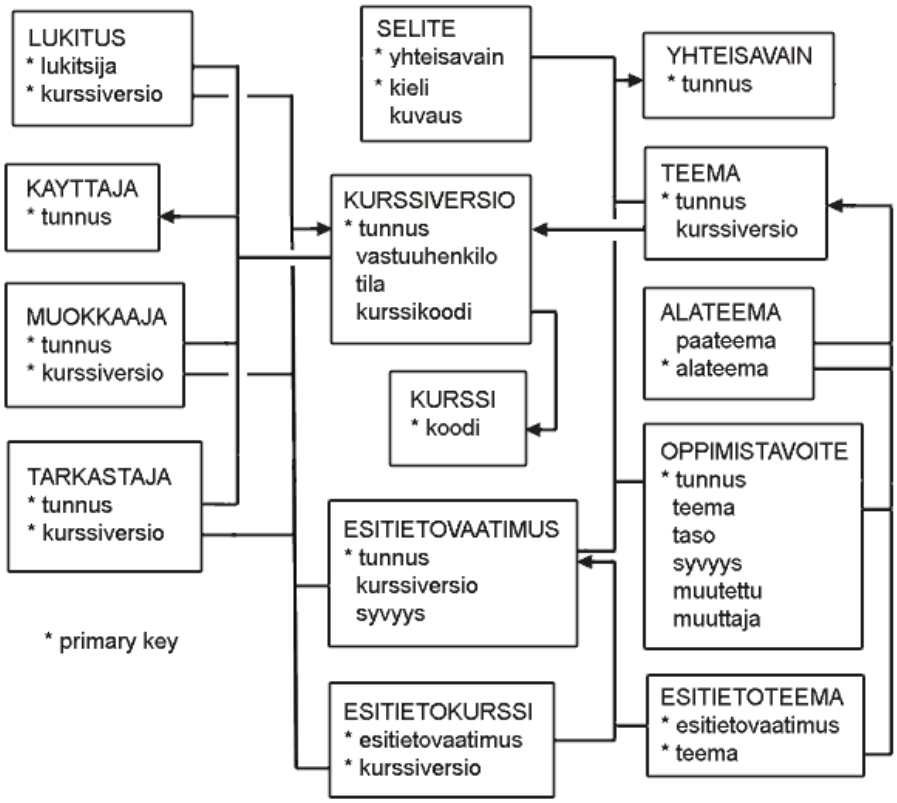
Tietokanta on suunniteltu vaatimusdokumentissa esitettyjen käsitteiden pohjalta. Tietokantakaaviosta (kuva 3) näkyvät taulujen kentät ja niiden väliset yhteydet. Tietokannan taulut ovat Boyce-Codd -normaalimuodossa.

Oppimistavoite-taulussa olevien kenttien taso ja syvyys selväkieliset tekstit ovat asetustiedostossa. Yhteisavain-taulussa luodaan yksilölliset avaimet esitietovaatimuksille, teemoille ja oppimistavoitteille. Toisin sanoen, näiden taulujen avaimet muodostavat yhtenäisen sarjan: 1, 2, 3 jne. Kurssin versiohallinta hoidetaan kurssiversio-taulun avulla.

Integroinnin suunnittelu

Rakennettavan järjestelmän tietokantaan luodaan näkymiä OSJ:ään, josta haetaan laitoksella työskentelevien henkilöiden ja pidettävien kurssien tietoja. *Opettaja-näkymä*n avulla saadaan rakennettavan järjestelmän käyttöön laitoksella työskentelevien henkilöiden nimet, nimilyhennetunnukset, jotka ovat yksilöiviä, sekä käyttäjätunnus, jota käytetään käyttäjän tunnistamiseen rakennettavan järjestelmän sisäänkirjautumisessa. *Opjaksonäkymä*n avulla saadaan rakennettavan järjestelmän käyttöön laitoksella järjestettävien opintojaksojen eli kurssien koodit sekä suomen- ja englanninkieliset nimet.

*Käyttäjätunnistus*ta varten järjestelmä tehdään laitoksen Webbi-palvelimelle, sysdb-palvelimelle. Kun käyttäjä menee järjestelmän URL-osoitteeseen, sysdb-palvelin on määritelty näyttämään laitoksen Intran sisäänkirjaus-sivu, jossa kysytään käyttäjän käyttäjätunnus ja salasana. Sysdb-palvelin tunnistaa käyttäjän vertaamalla käyttäjän antamia tietoja sysdb-



Kuva 3: Järjestelmän tietokantakaavio

palvelimella olevaan opettajalistaan. Jos tiedot löytyvät opettajalistasta, niin näytetään järjestelmän ensimmäinen sivu. Järjestelmä saa käyttäjätunnuksen ensimmäisellä sivulla luotavan HttpServletRequest-olion metodilla getRemoteUser. Sen jälkeen käyttäjän nimilyhennetunnus ja nimi haetaan tk_opha.opettaja-näkymästä, jollion tiedetään, kuka käyttää järjestelmää.

4.3.4 Luokkien suunnittelu

Malli

Mallissa on kolme luokkaa: Kurssihallinta, Kayttajahallinta ja Tilahallinta, jotka tarjoavat pääsyn tietokannan tietoihin. Mallin luokat Selite, Kurssi, Teema, Esitietovaatimus ja Oppimistavoite mahdollistavat tietokannan tietojen välittämisen. Useissa metodeissa pyydetään parametrina kutsuvan käyttäjän käyttäjätunnusta, jolla varmistetaan, että kutsuvalla taholla on oikeus kyseisiin operaatioihin. Rajapinta List löytyy paketista java.util ja muut projektin omasta paketista harri.

Selite on abstrakti luokka, jonka metodeilla hallitaan kuvauksia ja niiden eri kieliversioita. Perivät luokat ovat Kurssi, Teema, Oppimistavoite ja Esitietovaatimus.

Kurssi on luokka, jolla avulla välitettään kurssin (kurssiversion) tietoja.

Teema on luokka, jolla välitetään teeman tietoja. Teema-olioon kapseloidaan pääteema-alateema-oppimistavoite -rakenne.

Oppimistavoite on luokka, jolla välitetään oppimistavoiteen tietoja.

Esitietovaatimus on luokka, jolla välitetään esitietovaatimusten tietoja.

Näkymän JSP-sivut

Käyttöliittymäsuunnittelussa esiteltiin yhdeksän eri käyttöliittymä komponenttia, joita yhdistelemällä saadaan rakennettua käyttäjälle www-sivuja. Jokaisesta komponentista tehdään oma JSP-sivunsa. Tallennukseen liittyvät operaatiot käsitellään omilla JSP-sivuillaan.

Ohjain

Kaikki tiedon päivitys hoidetaan ohjaimen kautta. Kaikissa metodeissa parametria *String kayttaja* käytetään tarkistamaan, onko kutsujalla tarvittavat oikeudet kurssiin.

4.4 Testaussuunnitelma

Testaussuunnitelmassa määritellään toteutettavan järjestelmän testaussuunnitelma: mikä on projektin testausaikataulu, mitä testataan ja miten testaus tehdään. Lisäksi se määrittelee milloin järjestelmää on testattu tarpeeksi.

Testauksen tavoitteena on tarkistaa, että järjestelmästä tulee laadukas ja asiakkaan toivoma ohjelmisto. Testauksella pyritään estämään virheiden synty mahdollisimman aikaisessa vaiheessa sekä havaitsemaan mahdollisia puutteita. Testausuunnitelman tavoitteena on asettaa projektille testauksen suuntaviivat, joiden avulla testausta tehdään koko kehitystyön aikana.

Testauksen ensisijaiset kohteet ovat toiminnot ja vaatimukset, jotka on määritelty vaatimusdokumentissa prioriteetilla 1 ja on näin valittu toteutettavaksi. Nämä toiminnot täytyy siis olla toteutettu ja testattu, jotta tuote voidaan hyväksyä.

4.4.1 Testauksen organisointi

Testausprosessissa seurataan testauksen V-mallia, jossa on kolme vaihetta: yksikkötestaus, integrointitestaus ja järjestelmätestaus.

Testausprosessi ei ole erillinen vaihe ohjelmiston elinkaaressa vaan sitä tehdään koko toteutusvaiheen ajan. Ryhmän jokainen jäsen on vastuussa järjestelmän testauksesta. Yksikkötestausta tehdään heti, kun luokka on toteutettu. Testaustapahtumista ja -tuloksista pidetään kirjaa erilliseen testausraporttiin.

Jokainen ryhmän jäsen pystyttää testausta varten oman testausympäristön Tietojenkäsittelytieteen laitoksen tietokantapalvelimelle, jossa yksikkö- ja integrointitestaus suoritetaan. Testaustyökaluina käytetään JUnitia ja JCoveragea. Tietokannan toimintaa simuloidaan sopivalla testimateriaalilla. Järjestelmätestauksessa käytetään vain yhtä yhteistä testausympäristöä. Koska järjestelmä on web-sovellus opiskelijoiden käyttöön, käyttöliittymää testataan ainakin kahdella selaimella, esimerkiksi Internet Explorerilla ja Firefoxilla.

Virhetilanteessa testaaja pyrkii katsomaan, mistä virhe on aiheutunut ja korjaamaan sen. Tämän jälkeen testitapaus toistetaan. Virhetilanteet ja uusittu testitapahtuma dokumentoidaan raporttiin. Raporttiin kirjatut virheet, joita ei saada korjatuksi, dokumentoidaan lopulta ylläpitodokumenttiin.

Järjestelmätestauksessa riittää kirjata onnistuiko testitapaus. Jos testi ei mennyt läpi, kirjataan vielä epäonnistumisen syy. Järjestelmätestaus on suoritettu hyväksytysti, kun laajennetut käyttötapaukset on testattu ja järjestelmä ei kaadu käytön aikana. Jotta testitapaus on hyväksytty, pitää kaikkien skenaarioiden mennä läpi.

Kun järjestelmätestaus on tehty, suoritetaan hyväksymistestaus asiakkaan kanssa.

4.5 Käyttöohje

Käyttöohjeen johdanto antaa yleiskuvan Oppimistavoitteiden hallintajärjestelmästä, harrista. Siksi jokaisen käyttäjän on hyvä lukea se ennenkuin alkaa käyttää järjestelmää. Sanastoon on listattu aakkosjärjestykseen käyttöohjeen keskeiset termit, jotta käyttäjän on helppo etsiä niiden selitykset. Ohjelman käynnistäminen on kerrottu käyttäjäryhmittäin.

Toimintojen käyttöohjeet on ryhmitelty niin, että jokaisen käyttäjäryhmän omat toiminnot on kerätty yhteen kappaleeseen. Kappaleen alkuun on kirjoitettu, minkä toisen käyttäjäryhmän toiminnot ovat myös ko. käyttäjäryhmän käytettävissä. Näin käyttäjä tietää, mitkä kaikki toiminnot ovat hänen käytettävissään. Jokaisesta toiminnosta on sanallinen kuvaus, miten toimintoa käytetään, mitä vaihtoehtoja sen käyttämisessä on ja mitä käyttäjän tulee toiminnosta tietää. Kaikkiin toimintoihin on liitetty vielä toimintoa havainnollistava kuva.

4.6 Ylläpitodokumentti

Ryhmä on toteuttanut vaatimusdokumentin ensimmäisen prioriteettitason toiminnoista 29 ja kolmannen tason toiminnoista yhden.

Suunnitteludokumentin toisessa luvussa (s. 2) esiteltyä MVC-arkkitehtuurimallia käytettiin toteutuksen pohjana, mutta järjestelmän Java-luokkia tai niiden metodeja ei toteutettu läheskään kaikilta osin luvussa neljä (s. 14) esitetyssä muodossa. Erot ovat paikoin suuria ja tarkat tiedot toteutetuista luokista ja metodeista tulisi katsoa Javadoc-kuvauksista ja lähdekoodista.

Tietokantaan on tullut jotain muutoksia verrattuna suunnitteludokumentissä esiteltyyn ja tässä luvussa esitellään toteutettu tietokanta. Järjestelmässä suoritettavat SQL-lauseet on tehty käytetyn Oraclen tietokantaversion mukaisesti, eikä niiden yhteensopivuutta tai toimintaa muissa ympäristöissä ole määritelty.

Ohjelmistoa ei asennettu palvelimelle muuta kuin toteutusta, testausta ja järjestelmän esittelyä varten, eikä sovellusalueen todellista tietosisältöä koskaan käytetty. Täten mitään toiminnassa olevaa ylläpidettävää järjestelmää projektin päättyessä ei ole.

5 Päättöanalyysi

Ohjelmisto saatiin ensimmäisen prioriteetin vaatimusten osalta valmiiksi 10.12.2007 pidettyyn hyväksymistestaukseen mennessä. Ohjelmistoon jäi muutamia bugeja, jotka on eritelty testausraportissa ja ylläpitodokumentissa. Ajan puutteen takia suurin osa toisen prioriteetin vaatimuksista jätettiin toteuttamatta. Kolmannen prioriteetin vaatimuksista toteutettiin haku.

5.0.1 Projektin vaiheet

Ryhmän kanssa käytyjen keskustelujen pohjalta on selvää, että vaikka ohjelmisto informaatiojärjestelmänä ei ollut teknisesti vaikea toteutettava, osoittautui vaatimusten esiinkaivelu erittäin haastavaksi. Oppimistavoitteiden laatimiseen, hyväksyntään ja julkaisuun liittyy paljon byrokratiaa, ja asiaan kunnolla paneutuminen vaatisi projektiryhmän osallistumisen esimerkiksi laitoksen palavereihin, jossa näitä asioita puidaan. Tämä ei taas enää kuulu laajuudessaan ohjelmistotuotanto projektin piiriin.

Koska asiaan paneutuminen vaati paljon aikaa, vaatimusmäärittelyyn käytettiin yli kuukausi, eli lähes koko ensimmäinen periodi. Suunnittelu aloitettiin limittäin vaatimusmäärittelyn kanssa. Ryhmä oli yksimielinen siitä, että vaatimusmäärittelyn vaikeudet näkyivät myös suunnittelussa. Osalti myös suunnittelua vaikeutti tarvittavien työkalujen teknisen osaamisen puuttuminen, joten opeteltavaa oli kaikilta osin paljon. Suunnitteluvaihe venyi myös viikolla, mutta tämä oli tarpellista, koska toteuttaminen lähti viikon lisätyön jälkeen sujuvasti käyntiin. Suunnittelun ohella laadittiin myös testaussuunnitelma.

Jälkiviisaana voi kuitenkin todeta, että suunnittelun olisi pitänyt pohjautua enemmän käyttöliittymäsuunnitteluun, sillä toteutuksen eri vaiheissa, muutoksia ohjelmiston rakenteeseen tehtiin lähinnä käyttöliittymän tarpeisiin. Koska tätä ei heti suunnittelun alkaessa ymmärretty, suunnitteluvaihe venyi, mutta ei ollut kuitenkaan tarpeeksi kattava.

Toteutusvaihe eteni sujuvasti. Tietokanta ja luokat valmistuivat hyvissä ajoin. Käyttöliittymän ohjelmointiin olisi kannattanut laittaa enemmän resursseja, jotta se olisi valmistunut aiemmin ja sen tarpeisiin olisi voitu vastata luokkia ohjelmoidessa aiemmin. Ehkä viimeistään tässä vaiheessa ryhmän resurssipula tuli parhaiten esille. Aikaisemmissa vaiheissa lisämiehityksellä ei olisi ollut merkitystä.

Vaikka vaiheet venyivät, yksikkötestaukselle ja integrointitestaukselle jäi hyvin aikaa. Näitä ei pystytty kuitenkaan toteuttamaan ryhmän toivomassa laajuudessa. Yksityiskohdat ilmenevät testausraportista.

5.0.2 Ryhmätyöskentely

Ohjelmiston haastavuudesta huolimatta ryhmä toimi hyvin yhteen. Heti ensimmäisellä viikolla ryhmän koko pieneni kuudesta hengestä neljään. Tämä tapahtui onneksi jo niin aikaisessa vaiheessa, joten tästä ei ollut varsinaisesti haittaa projektin edistymiselle. Kuten todettua, lisämiehistölle oli tarvetta ehkä vasta toteutusvaiheessa. Ehkä osalti tämän takia

tuntimäärät eivät jakautuneet aivan tasan ryhmän kesken. Osa ryhmän jäsenistä oli syksyn aikaan myös päivätöissä. Teknisen osaamisen opettelu osoittautui myös ryhmän jäsenille haastavaksi.

Huolimatta pienestä koostaan, ryhmä teki kuitenkin saman yhteistuntimäärän kuin muut ohjelmistotuotantoprojektiryhmät. Ryhmän henki säilyi loppuun asti positiivisena ja rakentavana. Keskustelua käytiin avoimesti niin palavereissa kuin niiden ulkopuolellakin. Silti, kaksi ryhmän jäsentä ylittivät ohjelmistotuotanto projektille suositellun tuntimäärän. Se jo yksin kertoo, että neljän hengen ryhmä on auttamattoman pieni 14 viikon projektille.

6 Yhteenveto

Jokatapauksessa, ohjelmisto saatiin vaatimusmäärittelyn pääprioriteettien osalta tehtyä. Tuontantokäyttöön kypsää järjestelmää ei saatu näillä resursseilla aikaan, mutta pieni katsaus siihen, millainen järjestelmä oppimistavoitteiden hallintaan voidaan laatia. Järjestelmän jatkokehitystä ajatellen, tässä projektissa laaditut dokumentit toimivat hyvänä pohjana jatkokehityksen aloittamiselle. Seuraava projektiryhmä pystyy niiden avulla välttämään samat sudenkuopat ja etenemään heti alkajaisiksi oikeaan suuntaan niin vaatimusmäärittelyssä, suunnittelussa kuin toteutuksessakin.