Ylläpitodokumentti

NJC2

Helsinki 11.5.2004

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Eero Anttila
Olli Jokinen
Jesse Liukkonen
Jani Markkanen
Jere Salonen
Jouni Tuominen

Asiakas

Olli Lahti

Johtoryhmä

Juha Taina

Kotisivu

`http://www.cs.helsinki.fi/group/njc2/`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
0.1	10.5.2004	Ensimmäinen versio
1.0	11.5.2004	Ensimmäinen virallinen versio

Sisältö

1 Johdanto

Projektiryhmä NJC2 tuotti Helsingin yliopiston tietojenkäsittelytieteen laitoksen Ohjelmistotuotantoprojekti-kurssilla lehden toimituksen apuvälineen. Ryhmän tehtävänä oli tuottaa järjestelmä, joka helpottaa lehden julkaisuprosessin vaiheita sekä kommunikointia sidosryhmien välillä. Järjestelmän käyttäjänä tulee olemaan Nordic Journal of Computing -lehden toimituskunta tietojenkäsittelytieteen laitoksella. Yliopisto julkaisee ohjelmiston joko GNU General Public License- tai GNU Lesser General Public License-lisenssin alaisuudessa.

1.1 Dokumentin tarkoitus

Ylläpitodokumentin tarkoituksena on tarjota ohjelmiston ylläpitäjälle yleiskuvaus ohjelmistosta, sekä riittävät tiedot myöhempää kehittämistä varten. Tiukan aikataulun vuoksi projektiryhmä joutui jättämään muutamia ominaisuuksia kokonaan toteuttamatta, ja joihinkin toteutettuihin ominaisuuksiin jäi parantamisen varaa. Tämä dokumentti toimii ohjekirjana ohjelmiston kehitystyöhön projektin jälkeen.

1.2 Dokumentin rakenne

Luvussa 2 kuvataan järjestelmän arkkitehtuuri yleisellä tasolla lähdekoodin ymmärtämisen helpottamiseksi. Luvussa 3 on lueteltu ohjelmistoon suunniteltujen, mutta toteuttamatta jätettyjen ominaisuuksien lisäksi sellaisia ominaisuuksia, joiden projektiryhmä katsoo parantavan ohjelmiston laatua. Ominaisuuksien yhteydessä on kuvattu lyhyesti, millaisia muutoksia lähdekoodiin kunkin ominaisuuden korjaaminen tai lisääminen ylläpitäjältä vaatii.

2 Järjestelmän yleiskuvaus

2.1 Järjestelmän tarkoitus

Järjestelmän tarkoituksena on helpottaa Nordic Journal of Computing -lehden julkaisutoimintaa. Ohjelmisto hallinnoi julkaistavaksi tarjottujen artikkeleiden toimituksellista käsittelyä, edesauttaa toimituksen kanssa vuorovaikutuksessa olevien sidosryhmien toimintaa sekä yksinkertaistaa toimituksen ja sidosryhmien välistä kommunikointia.

Järjestelmän sidosryhmät ovat lehden toimittajakunta - johon kuuluu myös päätoimittaja - artikkeleiden kirjoittajat sekä joukko artikkeleita arvostelevia asiantuntijoita. Kuka tahansa voi tarjota artikkeliaan lehdelle; mahdollisen asiantuntijakierroksen jälkeen toimitus tekee lopullisen päätöksen artikkelin julkaisusta.

Artikkelin tyypillinen elinkaari kulkee toimituksen kautta päätoimittajalle, takaisin toimitukselle ja edelleen valituille asiantuntijoille. Asiantuntijoiden annettua lausunnot (asian-

tuntija voi myös olla hyväksymättä hänelle lähetettyä lausuntopyyntöä) toimitus tekee oman päätöksensä artikkelin kohtalosta saatujen lausuntojen perusteella. Puutteellinen artikkeli voi käydä läpi useita tämäntapaisia kierroksia, kunnes artikkeli lopulta joko päätyy julkaistavaan muotoon tai saa hylkäävän päätöksen.

Ohjelmisto tarjoaa mahdollisuuden artikkelin elinkaarten hallinnoimiseen, automatisoiden samalla toimituksen ja asiantuntijoiden sekä toimituksen ja kirjoittajien välillä tapahtuvaa kommunikointia.

2.2 Järjestelmän arkkitehtuuri

Järjestelmän selkeästä kolmijakoisuudesta johtuen suunnittelumalliksi valittiin *Model-ViewController (MVC)*, jossa järjestelmä jaetaan kolmeen eri kerrokseen: Model, View ja Controller. Ensimmäinen huolehtii tietokantakyselyistä; toinen muokkaa saadut tulokset käyttäjälle esitettävään muotoon; kolmas määrittelee järjestelmän toiminnallisuuden ja ottaa vastaan pyynnöt käyttäjältä. Tällainen suunnittelumalli selkeyttää koodin rakennetta ja sulautuu hyvin arkkitehtuuriin, joka toteutetun järjestelmän tapauksessa muodostuu tietokannasta (Model), Java-moduulista (Controller) ja käyttöliittymästä (View).

2.2.1 Tietokanta

Tietokantaa ja Java-moduulia suunniteltaessa lähteenä on käytetty *DAO (Data Access Object)* -suunnittelumallia. Ideana on, että yhteys tietokantaan toimii rajapinnan kautta. Tämä mahdollistaa tietoresurssin vaihtamisen esimerkiksi XML-muotoon tai johonkin toiseen tietokantatyyppiin aiheuttamatta muutoksia muualle kuin luokkaan DAO.

Kaikki tietokantaoperaatiot suoritetaan keskitetysti luokassa DAO. Uusien tietokantakyselyiden lisääminen ja nykyisten kyselyiden muokkaaminen on siten varsin suoraviivaista: mikäli kyselyyn liittyvät attribuutit ja taulut löytyvät tietokannasta, riittää muokata vain kyseistä luokan DAO metodia.

Uusien attribuuttien lisääminen tietokantatauluihin ei aiheuta muutoksia muualle kuin itse tietokantaan. Olemassaolevien attribuuttien muokkaaminen tai poistaminen vaatii kuitenkin myös luokassa DAO olevien, kyseisiä attribuutteja kyselyissään käyttävien metodeiden muokkaamista uutta tietokantaa vastaaviksi.

Tietokanta on suunniteltu siten, että ohjelmisto on tiettyyn pisteeseen asti laajennettavissa ja paranneltavissa nykyisiä tietokantatauluja ja attribuutteja hyväksikäyttäen; suuritöisimmät muutokset liittyvät yleensä luokkaan DAO. Laajamittaisempi kehittäminen vaatii todennäköisesti kuitenkin myös uusien taulujen ja attribuuttien lisäämistä.

2.2.2 Käyttöliittymä

Käyttäjälle näkyvä osuus järjestelmästä on JSP-tekniikalla toteutettu käyttöliittymä. Käyttöliittymäsivut on jaettu sidosryhmittäin siten, että jokainen sidosryhmä pääsee käsiksi

oman ryhmänsä ja sitä "alempien"ryhmien sivuihin: kirjoittajalla on pääsy vain kirjoittajien sivuille, asiantuntijoilla on pääsy sekä asiantuntijoiden että kirjoittajien sivuille ja toimittajilla on pääsy kaikille sivuille. Erikoistapauksena päätoimittajalla on muiden sivujen lisäksi oma erillinen päätoimittajanäkymä. Käyttäjien navigointi sivuilla tapahtuu pääasiassa vasemmassa kehyksessä olevan valikon avulla.

JSP-sivujen sisältämät tiedot luodaan dynaamisesti kutsumalla sivuilla sopivaa DAO-luokan metodia. Esimerkiksi kaikille sidosryhmille näkyvät artikkeliluettelot ja niihin liittyvät lausuntojen tilat luodaan HTML-koodin seassa sisäkkäisten silmukoiden avulla, joissa kutsutaan halutun tiedon palauttavaa metodia. Saatujen arvojen perusteella tulostetaan sivuille halutun näköinen tuloste.

Käyttäjien lomakkeisiin syöttämät tiedot lähetetään tapahtumien (event) avulla Java-moduulin ControllerServlet-luokan käsiteltäväksi. Jokaista käyttäjän sivuilla tekemää toimintoa vastaa oma yksikäsitteinen tapahtumansa, joka annetaan parametriksi lomakkeen lähettämisen yhteydessä. ControllerServlet tulkitsee tapahtuman ja suorittaa tarvittavat operaatiot, minkä jälkeen käyttäjälle näytetään päivittynyt JSP-sivu.

Esimerkiksi käyttäjän painaessa artikkeliluettelosta tiettyä artikkelia, lähettää käyttöliittymäsivu parametrina ControllerServletille tiedon tapahtumasta - käytännössä siis jonkin Event-luokan vakioista - jolloin ControllerServlet huomaa, että haluttu tapahtuma oli artikkelin avaaminen. Muiden parametrien avulla ControllerServlet päivittää kehykset ja avaa alempaan kehykseen valittuun artikkeliin liittyvät tiedot.

Uusien JSP-sivujen lisääminen vaatii siis sivuilla aiheutettuja tapahtumia vastaavat metodit ControllerServletiin sekä kyseisiä tapahtumia vastaavat luokkavakiot luokkaan Event. Todennäköisesti uudet sivut vaativat myös uusien kyselyiden lisäämistä luokkaan DAO. Nykyisten sivujen päivittäminen ja muokkaaminen saattaa niin ikään vaatia muutoksia edellämainittuihin luokkiin.

Väliaikaisia ja istuntokohtaisia tietoja säilytetään SessionData-luokan ilmentymässä, josta ne saadaan kutsumalla JSP-sivuilla vastaavia metodeita. Esimerkiksi käyttäjän rooli on sijoitettu SessionDataan, ja se tarkistetaan jokaisen sivun alussa.

2.2.3 Java-moduuli

Järjestelmän Java-moduuli sisältää tarvittavat tietotyypit, servletit, tietokantayhteyteen tarvittavat luokat sekä tietokantakyselyistä huolehtivan luokan DAO.

Tietotyyppejä ovat esimerkiksi Artikkeli ja Lausunto. Uusien tietotyyppien lisääminen ei itsessään vaadi muutoksia muihin luokkiin, mutta nykyisten tietotyyppien muokkaaminen vaatii kyseisiä tyyppejä käyttävien DAO:n metodeiden muokkaamista.

Ylläpitäjälle mielenkiintoisimmat luokat ovat ControllerServlet ja DAO. Käytännössä kaikki toiminnallisuuden lisääminen vaatii muutoksia näihin kahteen luokkaan. Jokainen uusi tietokantakysely täytyy lisätä omaksi metodikseen luokkaan DAO, ja mikäli kysely liittyy johonkin käyttöliittymäsivuilla aiheutettuun tapahtumaan, täytyy sitä varten lisätä oma metodinsa myös luokkaan ControllerServlet.

Käytännössä jokainen painikkeen (ja osan linkeistä) painaminen aiheuttaa tapahtuman,

joka täytyy tulkita ControllerServletissä. Ilman käyttäjän eksplisiittistä toimintaa suoritettavat metodikutsut sen sijaan eivät aiheuta tapahtumaa. Esimerkkinä jälkimmäisestä mainittakoon kaikkien sidosryhmien sivujen yläkehykseen tulostettava artikkeliluettelo, joka tuotetaan kutsumalla suoraan JSP-sivulla halutun luettelon tuottavaa DAO:n metodia sivun jokaisen latauskerran yhteydessä.

Melkein kaikkien tapahtumien tutkiminen noudattaa samaa kaavaa. Käyttäjän painaessa käyttöliittymäsivulla jotain sivun painikkeista, lähetetään lomakkeen tiedot ControllerServletille, joka päättelee piilotettuna kenttänä lähetetyn tapahtuman (Event-luokan vakio) avulla oikean case-haaran ja kutsuu sitä vastaavaa yksityistä metodiaan. Metodin sisällä kutsutaan halutun tietokantaoperaation suorittavaa DAO-luokan metodia ja ohjataan käyttäjä halutulle sivulle. Tietokantakyselyissä tarvittavat parametrit saadaan yleensä SessionDatasta. Tarkempia esimerkkejä yleisimmistä käyttötapauksista sekvenssikaavioineen löytyy sekä suunnitteludokumentista että toteutusdokumentista.

Järjestelmän kehittäminen vaatii ylläpitäjältä perehtymistä ainakin luokkiin DAO ja ControllerServlet. Todennäköisesti myös nykyisten JSP-sivujen toiminnallisuuden tarkempi selvittäminen on tarpeen, sillä keskenään samantapaisten sivujen toteutuksessa on yhdenmukaisuuden vuoksi käytetty samaa perusideaa. Erityisesti artikkeliluettelot tulostetaan eri sivuilla pitkälti samaa kaavaa noudattaen.

Muuttujien ja metodien lisääminen SessionDataan on ylläpitäjän harkinnan varassa. Ohjenuorana vain sellaiset tiedot, jotka halutaan säilyttää siirryttäessä sivulta toiselle — esimerkiksi käyttäjän rooli tai osittain täytetty lomake — on syytä lisätä SessionDataan.

3 Parannusehdotuksia

Tässä luvussa esitellään toteuttamatta jääneitä alemman prioriteetin ideoita sekä parannusehdotuksia. Jokaisen parannusehdotuksen yhteydessä kuvataan lyhyesti, mitä muutoksia järjestelmään on tehtävä.

3.1 Raportit

Yleistä Ohjelmisto tarjoaa toimitukselle erilaisia raportteja artikkeleiden ja käyttäjien tiedoista. Raporttien tarkoituksena on antaa toimittajille jonkinlainen yleiskuva järjestelmän tilasta.

Parannusehdotus Toimitus saattaa tulevaisuudessa haluta nykyistä laajempia ja tarkempia raportteja. Uusien raporttien lisäämiseksi vaaditaan päivityksiä kolmeen komponenttiin: raportit tulostavalle JSP-sivulle, Raportti-luokkaan sekä raportteja tietokantakyselyiden avulla muodostavaan DAO:n metodiin getRaportti().

Uusi tietokantakysely lisätään samaan metodiin, mistä raportit tulostava JSP-sivu kutsuu sitä oikealla parametrilla. Mahdolliset parametrit ovat luokassa Raportti määriteltyjä luokkavakioita; jokaista lisättävää raporttia kohti täytyy lisätä uusi luokkavakio. Staattiselle raporttisivulle uusi raportti lisätään nykyisten raporttien tapaan

kutsumalla HTML-koodin keskellä metodia getRaportti() ja antamalla parametriksi lisätty luokkavakio.

3.2 Asiantuntijoiden värikoodaus

Yleistä Toimitus valitsee jokaiselle artikkelille sopivat asiantuntijat arvostelijoiksi. Valintaan vaikuttaa asiantuntijan soveltuvuus aiheeseen sekä se, kuinka paljon kyseistä asiantuntijaa on jo kuormitettu (ts. kuinka paljon odottavia lausuntoja asiantuntijalla on). Tällä hetkellä soveltuvuuden saa selville asiantuntijalle määritellyistä osaamisaloista, mutta asiantuntijan kuormitusta ei näe.

Parannusehdotus Lista, josta asiantuntijoita valitaan artikkelille voisi olla värikoodattu siten, että asiantuntijan tietojen taustavärin tummuus riippuu asiantuntijan kuormituksesta: mitä enemmän odottavia lausuntoja, sitä tummempi taustaväri. KayttajaListaan-metodiin lisätään tällöin uusi kenttä kuormitukselle, int kuormitus. DAO:n metodissa getAsiantuntijat(hakuehto) liitetään palautettaviin KayttajaListaan-olioihin arvostelijan kuormitusaste. Kuormitus saadaan selville relaatiosta hakemalla asiantuntijalla arvosteltavana olevien artikkelien määrä tarvittavilla ehdoilla esim. vuoden sisällä arvosteltujen tai lausuntoa odottavien artikkelien määrä. JSP-sivulla värikoodaus toteutetaan switch-case -rakenteella.

3.3 Kielituki eri kielille

Yleistä Ohjelmiston käyttöliittymäkieleksi on valittu englanti, jotta mahdollisimman monella käyttäjällä olisi mahdollisuus käyttää sitä.

Parannusehdotus Käyttöliittymästä voisi tehdä esimerkiksi suomenkielisen version, jotta ohjelmiston käyttö olisi helpompaa suomalaisille käyttäjille. Kielitiedostojen avulla ohjelman käyttöliittymän voisi muuttaa helposti esimerkiksi saksan- tai ranskankieliseksi.

Ohjelmisto ei tue tällä hetkellä kielitiedostojen käyttöä, vaan kaikki käyttöliittymätekstit on kovakoodattu Java-tiedostoihin sekä JSP-sivuille. Jotta käyttöliittymäkielen muuttaminen olisi mahdollisimman helppoa, kaikki käyttöliittymätekstit tulisi lukea yhdestä tekstitiedostosta. Tämän tekstitiedoston lukeminen hoituu esimerkiksi Asetukset-luokan tapaisella luokalla. Luokassa tulee olla yksi metodi "String getTeksti(int)"sekä joukko luokkavakioita, joilla kuvataan käyttöliittymätekstien nimet.

Käyttöliittymäluokan voi toteuttaa myös niin, että se tukee eri kielitiedostoja käyttäjän valinnan mukaan. Tällöin käyttäjä voi valita rekisteröitymisen yhteydessä halutun käyttöliittymäkielen, jolloin ohjelmisto toimii käyttäjän valitsemalla kielellä. Tämä ominaisuus vaatii sarakkeen lisäämistä tietokantaan Käyttäjä-tauluun sekä luokkaan DAO uuden metodin getKayttajanKieli(int kayt_id).

3.4 Asiantuntijan ehdotus artikkelille erikoisalan perusteella

Yleistä Toimitus on kerännyt asiantuntijoille erikoisalalistan, joka kuvaa asiantuntijoiden osaamista mahdollisimman hyvin. Asiantuntijat pääsevät myös itse muokkaamaan tätä listaa.

Jokaisen artikkelin lähettämisen yhteydessä käyttäjä määrittelee avainsanoja, jotka kertovat mihin aihepiiriin artikkeli kuuluu.

Parannusehdotus Toimituksen valitessa asiantuntijoita artikkelille, järjestelmä voisi ottaa huomioon artikkelin aihepiirin sekä asiantuntijoiden erikoisalat ja antaa ehdotuksia artikkelille sopivista asiantuntijoista. Ominaisuuden lisääminen vaatii artikkelienhallinta.jsp-sivun muokkausta. Sivulle tulee lisätä algoritmi, joka poimii asiantuntijalistasta ne, jotka saattaisivat olla sopivia arvostelijoita artikkelille.

3.5 Saman asiantuntijan käyttö oletuksena artikkelin eri versioille

Yleistä Toimitus valitsee jokaiseen artikkeliin haluamansa asiantuntijat. Artikkelin uudessa versiossa ei oteta huomioon edelliseen versioon valittuja asiantuntijoita.

Parannusehdotus Saman artikkelin vanhempien versioiden asiantuntijat voisivat olla oletuksena uuden version asiantuntijoiksi. Ominaisuus vaatii muutoksen artikkelienhallinta.jsp-sivulle, jossa uuden version asiantuntijaksi lisätään tällöin vanhan version asiantuntijat.

3.6 Lehden julkaisu

Yleistä Toimittaja voi valita artikkeleita julkaistavaksi lehden eri numeroihin, mutta lehteä ei voi julkaista. Tämän vuoksi jo julkaistut artikkelit jäävät valintalistoihin, vaikka ne voisi siirtää esimerkiksi erilliseen arkistoon.

Parannusehdotus Artikkeleille voi määritellä uuden tilan (julkaistu lehdessä). Artikkelilistoja tuottavia jsp-sivuja joudutaan muokkaamaan sen verran, että julkaistuja artikkeleita ei näytetä niissä.

Arkisto vaatii oman jsp-sivun ja linkin sille menu.jsp-sivulle. Arkisto-sivulla voidaan sitten ryhmitellä artikkeleita esimerkiksi lehden mukaan, joissa ne on julkaistu.

3.7 Tiedostojen MIME-tyypit

Yleistä Tiedostoille täytyy määritellä MIME-tyyppi, jotta käyttäjän selain osaa avata tiedostot oikealla ohjelmalla. Tällä hetkellä tiedoston MIME-tyyppi määritellään tiedoston päätteen mukaan ja määrittelyt on annettu TiedostoServlet-luokassa.

Parannusehdotus MIME-tyypit voitaisiin lukea erillisestä asetustiedosta. Tämä vaatisi TiedostoServlet-luokan muuttamista niin, että luokka hakee päätteitä vastaavat MIME-tyypit tiedostosta.

MIME-tyypin määrittelyn voi tehdä myös muulla perusteella kuin tiedostonimellä. Tämä vaatii TiedostoServlet-luokkaan esimerkiksi sellaisen metodin laatimista, joka osaa päätellä MIME-tyypin tiedoston sisällön perusteella. Tällöin selain osaisi avata esimerkiksi .txt-päätteisen PDF-tiedoston oikein.

3.8 Konfiguraatiotiedosto

Yleistä Toimituksen käytössä on erillinen asetustiedosto, johon on määritelty tarvittavia parametreja ohjelman käyttöön. Näihin parametreihin kuuluu mm. CSS-tyylitiedoston sijainti ja tietokantatunnukset.

Parannusehdotus Toimitus saattaa haluta ohjelmistoon enemmän konfiguroitavia parametrejä. Esimerkiksi voisi olla ihan kiva, jos toimitus pystyisi muuttamaan artikkelin tilaa kuvaavien palkkien väriä.

Asetukset-luokan arvoString ja arvoInt -metodit palauttavat kyseistä avainta vastaavan rivin asetustiedosta, joten muutos vaatii vain uuden rivin lisäämisen asetustiedostoon sekä JSP-sivujen, joissa kyseistä asetusta käytetään, muokkausta niin, että nämä arvot luetaan Asetukset-luokasta.

4 Virheet ja puutteet

4.1 Asiantuntijan tai toimittajan muuttaminen kirjoittajaksi

Virheen kuvaus Asiantuntijan tai toimittajan ”alentaminen” kirjoittajaksi ei poista kyseisen käyttäjän riviä Asiantuntija-taulusta. Tämän seurauksena käyttäjä näkyy edelleen asiantuntijalistoissa ja hänet voidaan valita asiantuntijaksi artikkelille.

Korjausehdotus DAO:in lisätään metodi removeAsiantuntija(int kayt_id), joka poistaa käyttäjän rivin Asiantuntija-taulusta. Tätä metodia kutsutaan ControllerServletin paivitaProfiili-metodin sisällä tarvittaessa.

4.2 Käyttäjien poistaminen järjestelmästä

Puutteen kuvaus Järjestelmästä ei voi poistaa käyttäjiä suoraan käyttöliittymän kautta vaan mahdolliset poistot täytyy tehdä SQL-operaatioilla suoraan tietokannasta.

Korjausehdotus DAO:in lisätään metodi poistaKayttaja(int kayt_id) ja esimerkiksi kayttajienhallinta.jsp-sivulle lisätään nappi, jota painamalla tunnus saadaan poistettua järjestelmästä.