TAMPEREEN
AMMATTIKORKEAKOULU

LIIKETALOUS

OPINNÄYTETYÖRAPORTTI

Perintätoimeksiantojen hallintaohjelman vaatimusmäärittely ja toteutussuunnitelma

Miika Kännälä

Tietojenkäsittelyn koulutusohjelma
Kesäkuu 2007
Työn ohjaaja: Jyrki Vehmas

TAMPERE 2007

TAMPEREEN
AMMATTIKORKEAKOULU

Tekijä(t): Miika Sakari Kännälä

Koulutusohjelma(t): Tietojenkäsittely

Opinnäytetyön nimi: Perintätoimeksiantojen hallintaohjelman vaatimusmäärittely ja toteutussuunnitelma

Työn valmistumis-kuukausi ja – vuosi: Kesäkuu 2007

Työn ohjaaja: Jyrki Vehmas **Sivumäärä:** 35

TIIVISTELMÄ

Opinnäytetyöni toimeksiantajana toimi Valkeakoskella sijaitseva perintätoimisto nimeltään Alfa & Omega Oy (AOC), jolla on käytössään perintätoimistoille tarkoitettu ohjelma Collector. Ohjelman monimutkaisuuden takia he halusivat yksinkertaisemman ja heidän vaatimuksiinsa paremmin soveltuvan ohjelman.

Työn tavoitteena oli kirjoittaa AOC:n toiveiden mukainen toiminnallinen vaatimusmääritelmä heidän haluamansa ohjelman sisällöstä. Toinen tavoite oli toteuttaa ohjelmasta pienimuotoinen prototyyppi, jossa on graafinen käyttöliittymä Javalla toteutettuna sekä taustalla MySQL-tietokanta. Opinnäytetyöhöni kuului myös myöhemmin valmistettavan ohjelman toteutussuunnitelma. Näiden työvaiheiden valmistuttua AOC päättää ohjelman lopullisen version valmistajan.

Toiminnallisen vaatimusmäärittelyn pohjatietona käytin ohjaajaltani saamia neuvoja sekä Wiegersin ja Pohjolan kirjoittamia kirjoja aiheesta. Myös muutamilta internet-sivuilta löytyi hyödyllisiä vinkkejä. Prototyypin ja toteutussuunnitelman tekemisessä hyödynsin koulussamme saamaani tietotaitoa sekä aiheeseen liittyvää kirjallisuutta.

Toiminnallisen vaatimusmäärittelyn valmistuttua AOC päätyi tilaamaan tuotteen ohjelmistotoimistolta, toimittamisvarmuuden ja -nopeuden takaamiseksi. Ohjelmistotoimisto pystyi silti hyödyntämään kirjoittamaani toiminnallista vaatimusmäärittelyä määritellessään projektin laajuutta, kustannuksia sekä sisältöä.

Opinnäytetyöstäni voi olla hyötyä koululllemme, koska opettajat pystyvät käyttämään prototyyppiäni kursseillaan mm. esimerkkinä tai tuntitehtävänä opiskelijoiden integroidessa prototyyppiini tietyn uuden ominaisuuden tai parannellessa jo valmiita ominaisuuksia.

Avainsanat: Java SQL Ohjelmistotuotanto

TAMPERE
POLYTECHNIC

Author(s):	Miika Sakari Kännälä
Degree Programme(s):	Business Information Systems
Title:	Requirement specification and implementation scheme for management software of collection orders
Month and year:	June 2007
Supervisor:	Jyrki Vehmas **Pages:** 35

ABSTRACT

The client of my thesis was a collection agency Alfa & Omega Oy (AOC) in Valkeakoski. AOC uses Collector which is a program designed for collection agencies. Because of the complexity of the program, they needed a simpler program which is better suited for their requirements.

First aim of the thesis was to make a requirement specification for a program that AOC needs according to their desires. Second aim was to implement a small-scale prototype of the program which has a graphical interface programmed with Java and a MySQL database in the background. An implementation scheme for the program that will be later produced is also included in the thesis. After these phases are completed, AOC decides who will do the final version of the program.

The instructions from my advisor and the books by Wiegers and Pohjola were used as basics for the requirement specification. Some Internet sites gave also useful hints. For the development of the prototype and the requirement specification I used the knowledge that I have gained in our school and the literature concerning the subject of my thesis.

After the requirement specification was finished AOC decided to order the product from a software agency to assure that the program is delivered for certain and on time. Nevertheless, the software agency was able to make use of my requirement specification to define the scale, the costs and the content of the project.

My thesis can be useful for our school for example because the teachers can use my prototype on their courses as an example or a task for students to integrate a new function to the prototype or to improve the completed functions.

Keywords: Java SQL Software development

Sisällysluettelo:

1 Johdanto

Tämän opinnäytetyön aiheena on Alfa & Omega Oy:lle (AOC) valmistettavan perintätoimeksiantojen hallintaohjelmiston toiminnallisen vaatimusmäärittelyn, toteutussuunnitelman sekä ohjelman prototyypin valmistaminen. Projektini toimeksiantaja on Valkeakoskella toimiva perintätoimisto.

Opinnäytetyön ensimmäisen osion tavoitteena on AOC:lle myöhemmin tuotettavan ohjelman toiminnallisen vaatimusmäärittelyn kirjoittaminen. Ohjelman vaatimukset on kirjattu kyseisen yrityksen edustajan Timo Sihvosen tarpeiden perusteella ja ne kuvaavat tarpeita määrittelyn tekohetkellä.

Ohjelmaa ei pyritä tuotteistamaan laajempaan käyttöön tai tulevaisuuden tarpeisiin, vaan siitä ohjelmoidaan ainoastaan prototyyppi. Sovimme AOC:n edustajan kanssa, että toiminnallisen vaatimusmäärittelyn ja prototyypin valmistumisen jälkeen he päättävät, mistä he tilaavat lopullisen ohjelman.

Ohjelma on tarkoitus suunnitella ainoastaan AOC:n käyttöön. Ohjelmaan on pyritty ottamaan samankaltaisia ominaisuuksia kuin jo valmiiksi markkinoilla oleva Collector -ohjelma sisältää. AOC:lla on ollut käytössä aiemmin Collector–ohjelma, mutta ohjelman monimutkaisuuden takia tavoitteeni on suunnitella heille yksinkertaisempi ohjelma, joka sisältää yksinomaan heidän tarvitsemansa ominaisuudet.

Opinnäytetyöni toinen tärkeä tavoite on ohjelman toteutusmenetelmien tutkiminen ja suunnitteleminen. Näiden tarkoituksena on antaa pohjaa ohjelman lopullisen version tekijälle tai esimerkiksi koulumme ohjelmointiin suuntautuneille opiskelijoille, mikäli he haluavat jatkokehittää prototyyppiäni tai tutkia vaihtoehtoja tämäntyyppisten projektien suorittamiseen.

Prototyyppi sisältää ainoastaan osan ohjelman ominaisuuksista. Opettajat voivat hyödyntää sitä kursseillaan antaen opiskelijoille tehtäväksi esimerkiksi tietyn uuden ominaisuuden integroimisen prototyyppiin tai jo valmiin ominaisuuden parantamisen. Prototyyppi on toteutettu ohjelmoimalla Java -ohjelmointikielellä graafinen käyttöliittymä, jonka taustalle on ohjelmoitu MySQL-tietokanta.

2 Toiminnallinen vaatimusmäärittely

2.1 Teoria vaatimusmäärittelystä

Toiminnallinen vaatimusmäärittely on ohjelmistoprojektin onnistumisen kannalta tärkein vaihe. Ohjelman toiminnallinen vaatimusmäärittely on dokumentti, johon kirjataan ohjelmaan sisällytettävät ominaisuudet ja toiminnallisuudet. Tavoitteena on saada ohjelmiston toimeksiantaja ja valmistaja samalle aaltopituudelle ohjelman sisällöstä.

Toiminnallisen vaatimusmäärittelyn tulee olla täsmällinen ja kattava. Tällöin se helpottaa merkittävästi valmistettavan ohjelman hinta- ja laajuusmäärittelyä sekä ohjelman tuottamista. Mikäli toiminnallisesta vaatimusmäärittelystä puuttuu jokin ominaisuus kokonaan tai sitä ei ole määritelty riittävän tarkasti, sen myöhäisempi lisääminen tai muuttaminen ohjelmaan saattaa aiheuttaa suurenkin ajallisen menetyksen.

Karl E. Wiegers (2003) kertoo kirjassaan hyvin laaditun toiminnallisen vaatimusmäärittelyn tavoittavan seuraavat edut:

- vähentää vaatimusvirheitä
- vähentää uudelleenkehittelyä
- vähentää tarpeettomia ominaisuuksia
- vähentää parannuskustannuksia
- nopeuttaa toimitusta
- vähentää väärinymmärryksiä
- vähentää projektin laajuusongelmia
- vähentää projektin kaaosta
- lisää huolellisia järjestelmän testaamisarvioita
- lisää toimeksiantajan ja tiimin jäsenten tyytyväisyyttä

2.2 Teorian soveltaminen

Toiminnallisen vaatimusmäärittelyn kirjoittamisen lähtökohtana käytin Karl E. Wiegers:n kirjaa ”Software Requirements” sekä Risto Pohjosen kirjaa ”Tietojärjestelmien kehittäminen”. Sovelsin kyseisten kirjojen teorioita esimerkiksi vaatimusmäärittelyn sisältöä ja vaatimusten selvitystapaa määrittäessäni. Myös ohjaajaltani sain paljon vihjeitä toiminnallisen vaatimusmäärittelyn sisältöä suunnitellessani.

Ohjelmani toiminnallisia vaatimuksia kirjatessani dokumenttiin käytin hyväksi K. Wiegersin (2003, 168–169) esittelemää vaatimusten hierarkkista numerointia. Mielestäni tämä oli selkein tapa kirjata vaatimukset dokumenttiin.

Vaatimusten keräämisessä sovelsin K. Pohjosen (2002, 28) esittelemien eri sidosryhmien haastattelu-, aivoriihi- ja ideointipalaveri-tekniikoita. Wiegers:n (2003, 96) mukaan varmin tapa ohjelman toiminnallisten vaatimusten selvittämisessä on valmistettavan tuotteen käyttäjien haastatteleminen. Ensimmäisessä projektiryhmämme palaverissa haastattelin AOC:n edustaja Timo Sihvosta, joka kertoi ominaisuuksista, joita ohjelmaan tulee sisällyttää. Näiden ominaisuuksien perusteella ideoin toiminnallista vaatimusmääritelmää AOC:n toiveiden mukaiseksi.

Seuraavaan palaveriin kirjoitin valmiiksi ensimmäisessä palaverissamme käydyn keskustelun perusteella ensimmäisen version toiminnallisesta vaatimusmäärittelystä. Palaverissa kävimme läpi kirjoittamani vaatimusmäärittelyn sisällön ja laadimme siihen tarvittavat parannukset. Tämän jälkeen työstin vaatimusmääritelmää palaverissa ilmenneiden korjausten ja täsmennysten perusteella. Tätä menetelmää hyödynsin, kunnes pääsimme toimeksiantajan kanssa yksimielisyyteen ohjelman toiminnallisista vaatimuksista.

Samaa menetelmää hyödynsin myös prototyypin ulkoasua suunnitellessamme. Prototyypin ulkoasun suunnittelin aluksi PowerPoint-ohjelmalla. Päästyämme yhteisymmärrykseen ulkoasusta, ohjelmoin Javalla prototyypin graafisen käyttöliittymän.

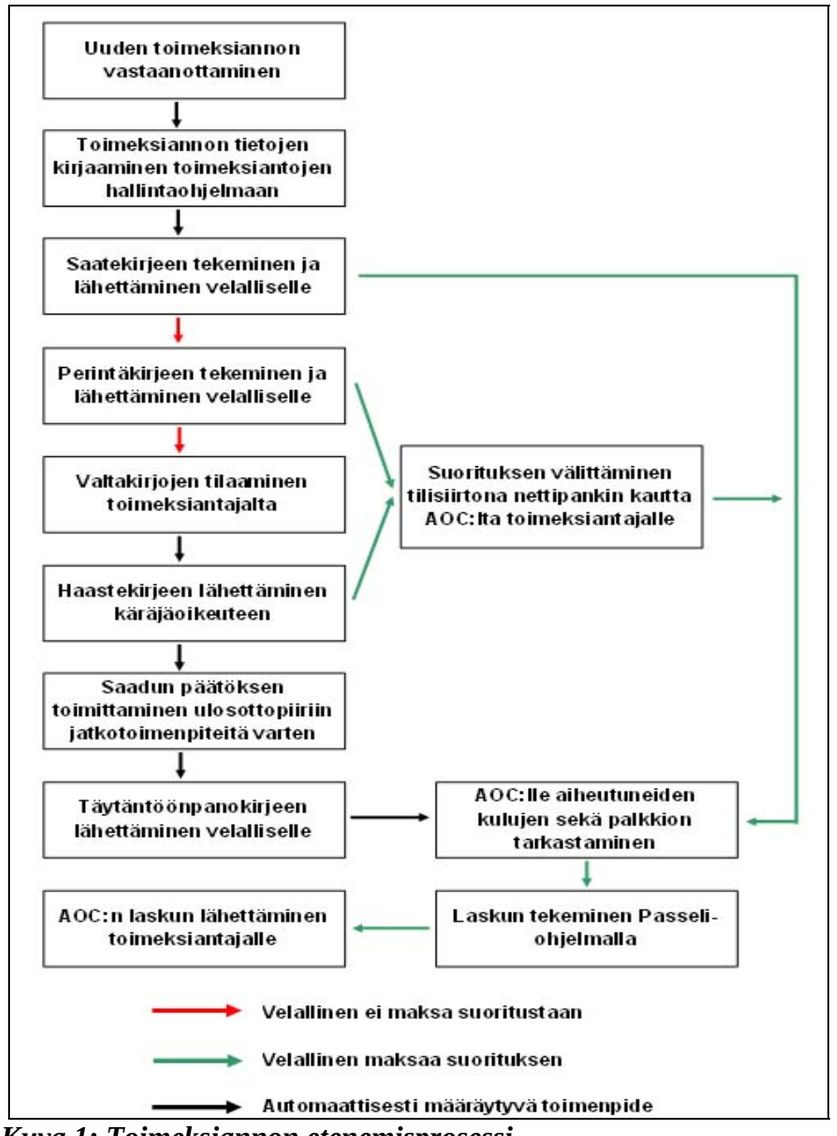
Wiegers:n (2003, 97) mukaan ohjelman käyttäjät kannattaa ryhmitellä käyttäjäryhmiksi ja jokaiselle ryhmälle tulee määritellä oikeudet ohjelman käyttämisen suhteen. AOC työllistää kuitenkin ainoastaan muutaman työntekijän ja jokainen työntekijä suorittaa ohjelmalla täysin samoja toimenpiteitä. Täten heidän tapauksessa käyttäjäryhmien muodostaminen oli turhaa.

3 Ohjelman toiminnallinen vaatimusmäärittely

3.1 Liiketoiminnalliset tavoitteet

3.1.1 Tuettava työprosessi

AOC:n koko toiminta liittyy velkomistoimeksiantojen perusteella tehtäviin toimenpiteisiin, asioiden etenemisen seurantaan ja eteenpäinvientiin. Toimeksiannon etenemistä kuvaava työprosessi on esitetty kuvassa 1.



Kuva 1: Toimeksiannon etenemisprosessi

Toimeksiannon käsitteleminen tapahtuu aina samassa järjestyksessä (kuva 1). Toimeksiantojen vastaanoton yhteydessä toimeksiantojen hallintaohjelmaan kirjataan toimeksiannon tiedot. Mikäli kyseessä on uusi asiakas, kirjataan ohjelmaan myös toimeksiantajan tiedot.

Toimeksiannon vastaanottamisessa ei ole mitään tiettyä tapaa. Yleisimmin AOC vastaanottaa toimeksiannot puhelimen välityksellä, tai toimeksiantajan ja AOC:n edustajan tapaamisen perusteella. Toimeksiantojen vastaanottaminen ja kirjaaminen on mahdollista myös esimerkiksi kirjeen, sähköpostin tai faksin välityksellä.

Toimenpiteiden etenemisjärjestys on seuraava:

- Saate- eli huomautuskirje
- Perintäkirje
- Haastekirje
- Täytäntöönpanokirje

Saatekirje on ensimmäinen AOC:n velalliselle lähettämä kirje. Siinä huomautetaan velallista maksamatta jätetystä suorituksesta ja pyydetään maksamaan se tiettyyn päivään mennessä suoraan toimeksiantajan tilille.

Mikäli velallinen ei maksa suoritustaan saatekirjeeseen merkittyyn eräpäivään mennessä, laaditaan ja lähetetään hänelle **perintäkirje**. Perintäkirjeestä aiheutuvat kustannukset ovat velalliselle suuremmat ja kirjeen teksti on uhkaavampaa. Tekstissä uhataan oikeudellisilla toimenpiteillä ja kerrotaan aiheutuvasta merkinnästä luottotietoihin, mikäli velallinen ei maksa suoritusta sekä perinnästä aiheutuneita kuluja uudelleenmääritettyyn eräpäivään mennessä. Perintäkirjeessä suoritus kehotetaan lähettämään AOC:n tilille. Velallisen tulee kirjoittaa laskun huomautuskenttään laskun numero, jonka perusteella ohjelma osaa kohdistaa maksusuorituksen oikeaan toimeksiantoon järjestelmään kirjaamista varten.

Maksusuoritusten välitys tapahtuu suoraan toimeksiantajalle ilman kulujen huomioimista. Ohjelman avulla voi seurata, onko AOC:n pankkitilille kertynyt suorituksia velallisilta. Välitys hoidetaan Osuuspankin eräsiirtopalvelun avulla, ja samalla päivitetään tiedot toimeksiannosta ohjelman tietokantaan.

Mikäli velallinen ei maksa saate- eikä perintäkirjeessä vaadittavia suorituksia, AOC tilaa toimeksiantajalta valtakirjat, joiden perusteella laaditaan **haastekirje** lähetettäväksi käräjäoikeuteen. Käräjäoikeudelta saatu päätös välitetään ulosottopiiriin toimenpiteitä varten. Seuraavaksi velalliselle lähetetään **täytäntöönpanokirje**, joka sisältää tiedot ulosottotoimenpiteistä.

Täytäntöönpanokirjeen lähettämisen jälkeen AOC tarkastaa toimeksiannosta aiheutuneet kulut sekä perittävän palkkion, joiden perusteella AOC:n edustaja laatii laskun heillä jo valmiiksi käytössä olevalla PASSELI–ohjelmalla. Tämän jälkeen lasku lähetetään toimeksiantajalle.

Toimeksiantajan maksusuoritusten rekisteröinti ja seuranta hoidetaan PASSELI-ohjelmalla. Vain suurista poikkeamista tulee merkintä toimeksiantojen hallintajärjestelmään, toimeksiantaja-ikkunan muistioon ko. toimeksiantajan tietoihin.

3.1.2 Tavoiteltavat hyödyt

Tällä hetkellä AOC:n muistiinpanot, kirjeet, yms. arkistoidaan kansioihin. Dokumentit ovat Word-asiakirjoja tai käsin kirjoitettuja muistioita. Tämän vuoksi ohjelma helpottaa AOC:n liiketoimintaa huomattavasti.

Ohjelman avulla AOC tavoittaa seuraavat edut:

- tiedot säilyvät paremmassa järjestyksessä tietokannassa
- tiedot on helpommin saatavilla
- jokapäiväinen työ nopeutuu merkittävästi
- unohdusten määrä vähenee ohjelman huolehtiessa työtehtävien suoritusjärjestyksestä
- tietojen arkistointi helpottuu
- tietojen etsiminen arkistosta nopeutuu ja helpottuu
- tiedot sijaitsevat aina samassa paikassa
- tiedot ovat varmassa tallessa tietokannassa
- tietokannan varmuuskopioinnin ansiosta tietoja ei katoa
- kaikki tiedot kulkevat aina AOC:n kannettavan tietokoneen mukana, jolloin tietokoneen käyttäjä voi työskennellä minkä tahansa toimeksiannon parissa missä tahansa paikassa

Ohjelman avulla käyttäjä voi tehdä seuraavia toimenpiteitä:

- kirjata toimeksiantajiin, velallisiin ja toimeksiantoihin liittyvät tiedot hallitusti tietokantaan
- saada työlista kyseisenä päivänä tehtävistä toimenpiteistä
- tulostaa velalliselle lähetettävät kirjeet ja asiakirjat helposti
- muokata valmiita kirjepohjia ja asiakirjoja
- osittain automatisoida toimeksiantoihin liittyviä rahalaskelmia
- osittain automatisoida maksujen välittämistä pankkiyhteysliittymän avulla
- arkistoida toimeksiannot tehokkaasti
- koota erillään ylläpidetyt toimeksiantotiedot yhteen
- taulukoida ja tulostaa erilaisia raportteja toimeksiantajista sekä toimeksiannoista
- tarkastaa toimeksianto-listasta, mitä toimeksiantoa tulee tiettynä päivänä jatkaa eteenpäin

3.2 Vaatimukset järjestelmän toiminnallisuudelle

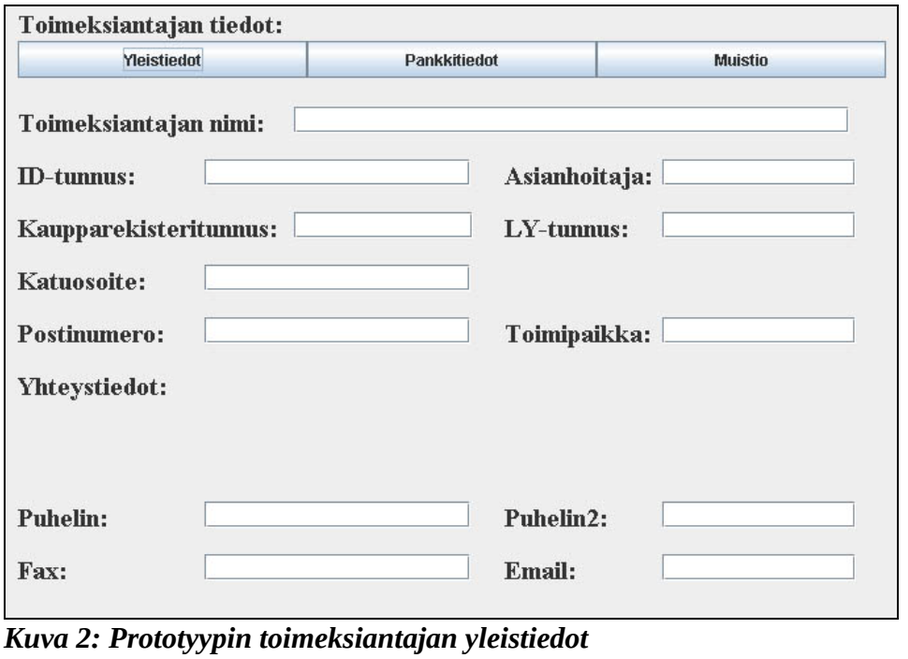
3.2.1 Salasanan kysely

Tietoturvan varmistamiseksi ohjelman käynnistyessä käyttäjältä kysytään salasana. Vasta käyttäjän kirjoitettua oikean salasanan, hän voi kirjautua sisään ohjelmaan ja aloittaa työskentelemisen. Mikäli käyttäjä ei ole tiettynä aikana tehnyt mitään tietokoneellaan ja käyttöjärjestelmä menee näytönsäästäjätilaan, käyttäjä pääsee tilasta pois ainoastaan oikean salasanan kirjoitettua. Myös tällä varmistetaan, etteivät ulkopuoliset voi käyttää ohjelmaa.

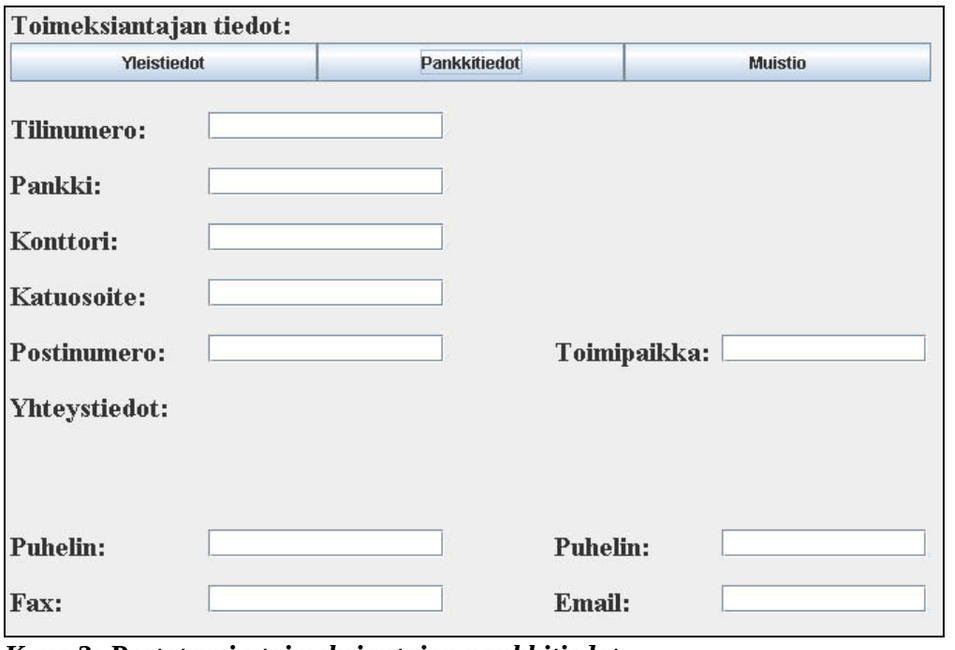
3.2.2 Toimeksiantajan tietojen kirjaaminen

Uuden toimeksiannon parissa työskenneltäessä ensimmäinen työvaihe on toimeksiantajan tietojen kirjaaminen tietokantaan. Mikäli toimeksiantaja on ennenkin ollut yrityksen asiakkaana, hänen tietonsa ovat jo valmiiksi tietokannassa. Tällöin käyttäjä voi siirtyä suoraan toimeksiannon tietojen kirjaamiseen.

Toimeksiantajien tiedot kirjataan ohjelman Toimeksiantajat -välilehdelle. Välilehdelle kirjataan toimeksiantajan yleistiedot sekä pankkitiedot.



Kuva 2: Prototyypin toimeksiantajan yleistiedot

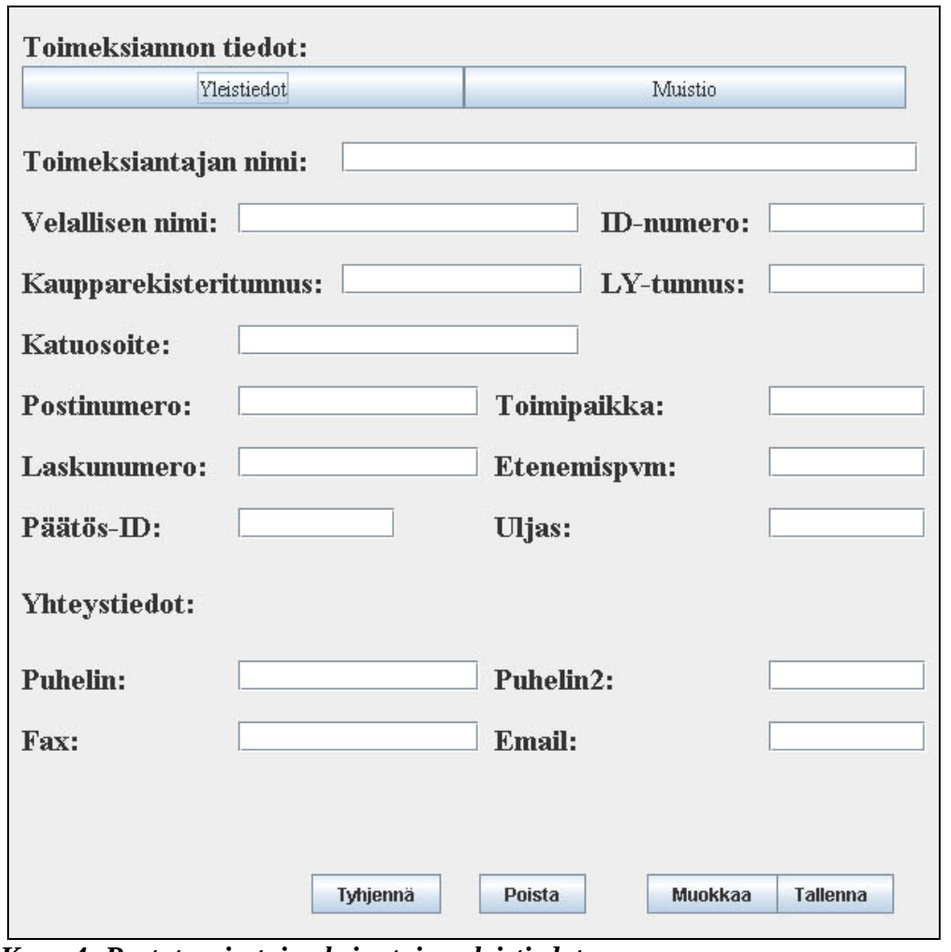


Kuva 3: Prototyypin toimeksiantajan pankkitiedot

Toimeksiantajat -välilehdellä on myös muistio, johon käyttäjä pystyy kirjoittamaan yleisiä muistiinpanoja kyseisestä toimeksiantajasta.

3.2.3 Toimeksiannon tietojen kirjaaminen

Toimeksiannon tiedot kirjataan toimeksiannot–välilehdelle. Ohjelman tulee antaa uudelle toimeksiannolle automaattisesti juokseva ID-numero, jonka avulla toimeksianto yksilöidään ja tunnistetaan. Toimeksiantojen kirjaaminen tapahtuu käyttämällä sille tarkoitettua ikkunaa. Ikkunassa on toimeksiannon tietoja varten kaksi välilehteä, yleistiedot sekä muistio.



Kuva 4: Prototyypin toimeksiantojen yleistiedot

Toimeksiannot-välilehdellä sijaitsee myös valintalista, josta käyttäjä voi valita haluamansa kirjeen tai rahalaskelma-linkin, tietojen muokkaamista tai tarkistamista varten. Valintalistan avulla käyttäjä voi:

- luoda uusia kirjeitä
- tutkia lähetettyjen kirjeiden sisältöä
- tulostaa valmiita kirjeitä lähetettäväksi velalliselle tai viranomaisille
- tarkistaa toimeksiannon etenemisvaihe
- tutkia ja muokata tietyn toimeksiannon rahaliikennettä

Lähetettyjen kirjeiden linkit näkyvät selkeyden vuoksi eri värein. Tällöin käyttäjän on helpompi seurata, missä vaiheessa kyseinen toimeksianto etenee.

Muistio-välilehdelle käyttäjä voi kirjoittaa lisätietoja kyseisestä toimeksiannosta.

3.2.4 Tulevien toimenpiteiden seuranta ja suorittaminen

Ohjelman haku-välilehdelle tulee lista, josta käyttäjä näkee listan tänään suoritettavista toimeksiannoista. Kustakin toimeksiannosta listaan tulee ID-numero, toimeksiantajan sekä velallisen nimi. Listaan voi myös lisätä tulevia toimeksiantoja päivä kerrallaan. Valitsemalla toimeksiannon käyttäjä pääsee tarkastamaan kyseisen toimeksiannon etenemisvaiheen, sekä työskentelemään toimeksiannon parissa.

Käyttäjän suoritettua tietyn toimeksiannon vaatimat toimenpiteet, ohjelma poistaa automaattisesti toimeksiannon tänään suoritettavien toimeksiantojen listasta. Toimeksiannon tiedot säilyvät kuitenkin ohjelman tietokannassa.

Käyttäjä voi työskennellä myös muina päivinä ajankohtaiseksi tulevien toimeksiantojen parissa selaamalla listalta eri päiviä, mutta ensisijaisesti tarkoituksena on suorittaa toimenpiteet päivä kerrallaan. Mikäli käyttäjä ei ole jonain päivänä ehtinyt suorittaa kaikkia tarvittavia toimenpiteitä, ohjelma huomauttaa aiheesta automaattisesti ja listaa selaamalla käyttäjä pääsee tekemään tarvittavat toimenpiteet toimeksiantoihin.

Haku-ikkunan avulla voi hakea järjestelmään jo valmiiksi tallennettuja toimeksiantojen tai toimeksiantajien tietoja. Haku onnistuu ID-numeron, nimen tai nimen osan perusteella. Mikäli nimen osalla löytyy ainoastaan yksi toimeksiantaja tai toimeksianto, ohjelma siirtyy automaattisesti ko. tietoihin. Muutoin ohjelma antaa valintalistan, josta käyttäjä voi valita etsimänsä tiedot. Mikäli toimeksiantajaa ei ole kirjattuna järjestelmään, valintalistasta pääsee automaattisesti toimeksiantajat–välilehdelle kirjaamaan toimeksiantajan tiedot tietokantaan.

3.2.5 Toimenpiteiden tekeminen toimeksiannolle

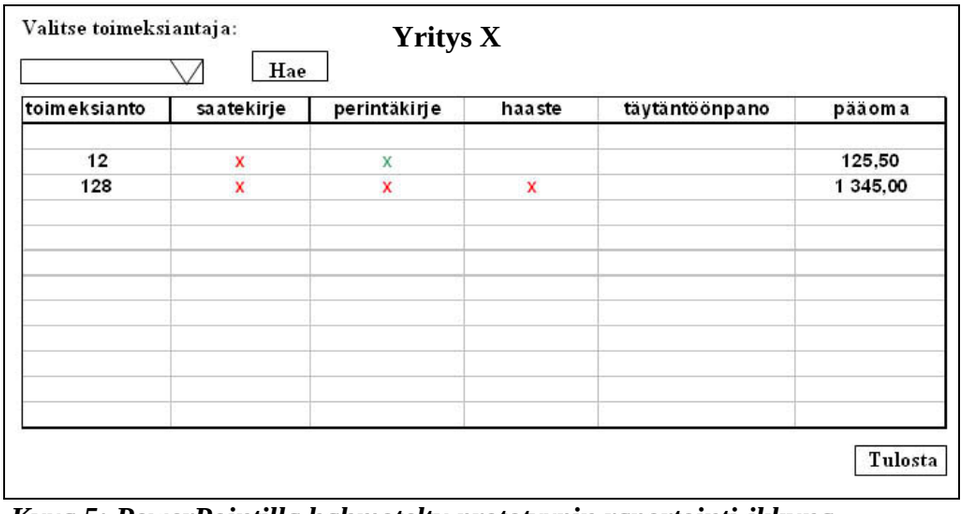
Toimeksiantojen toimenpiteet suoritetaan toimeksiannot-välilehdellä. Käyttäjä voi valita listalta toimeksiannossa seuraavaksi hoidettavan kirjeen, jolloin ohjelma hakee tarvittavat tiedot tietokannasta ja käyttäjä voi manuaalisesti täyttää puuttuvat kohdat. Kyseisten tietojen perusteella ohjelma luo valmiin kirjeen käyttäjälle, tulostamista ja velalliselle lähettämistä varten. Kirjeet arkistoidaan tietokantaan myöhempää tutkimista varten.

3.2.6 Rahaliikenne

Ohjelman rahaliikenne-välilehdellä käyttäjä voi hallinnoida tietyn toimeksiannon rahaliikennettä. Välilehdellä käyttäjä voi esimerkiksi suunnitella ja kirjata tietyn toimeksiannon maksuerien määrän, suuruuden, sekä seurata toimeksiannon rahaliikennettä.

3.2.7 Raportointi tietyn toimeksiantajan toimeksiannoista

Ohjelmaan tulee myös raportointi-välilehti, jossa käyttäjä pystyy seuraamaan tietyn toimeksiantajan toimeksiantoja. Tiedot sijaitsevat taulukossa, josta käyttäjä näkee toimeksiannon etenemisvaiheen. Toimeksiannon edetessä ohjelma kirjaa automaattisesti tietokantaan etenemiskohdan raportteja varten. Ohjelma seuraa myös toimeksiannon suorituksia. Mikäli raportissa nähtävän kirjeen vaadittu suoritus ei ole saapunut AOC:n tilille, kirjeestä on raportissa punainen merkintä. Saapuneen suorituksen merkintä on vihreä.



Kuva 5: PowerPointilla hahmoteltu prototyypin raportointi-ikkuna

Kuvasta 5 on nähtävillä Yritys X:ltä perittävät toimeksiannot ID-numeroiltaan 12 ja 128. Toimeksiannosta ID-numerolla 12 on lähetetty Yritys X:lle saatekirje sekä perintäkirje. Raportista selviää, että Yritys X on maksanut suorituksensa eräpäivään mennessä AOC:n tilille. Toimeksiannosta ID-numerolla 128 on lähetetty Yritys X:lle saate- ja perintäkirje. Lisäksi käräjäoikeudelle on lähetetty haaste, mutta yritys ei ole maksanut velkojaan. Taulukon pääoma-sarakkeessa sijaitsee toimeksiannon puhdas pääoma, eli toimeksiantajan perittävänä oleva kokonaissumma ilman korkoja.

3.2.8 Pankkiyhteys

Saapuvien maksusuoritusten rekisteröinti

”Osuuspankin eräsiirtopalvelun välityksellä asiakas voi käyttää omalla pankkiyhteysohjelmistollaan osuuspankkien tarjoamia pankkipalveluita” (Osuuspankin Eräsiirtopalvelun liittymäkuvaus, 1).

Ohjelman rahaliikenne-välilehti sisältää myös pankkiyhteysosion. Ohjelman pankkiyhteysosion ja Osuuspankin eräsiirtopalvelun välille tulee rajapinta, jotta ohjelma ja Osuuspankin eräsiirtopalvelu voivat kommunikoida keskenään. Ohjelmaan tulevan toiminnon avulla käyttäjä voi tarkistaa Osuuspankin eräsiirtopalvelusta, onko AOC:n tilille tullut suorituksia toimeksiannoista. Toimeksiantojen tunnistamiseksi velallisen tulee kirjoittaa aina viitenumeroksi kyseisen toimeksiannon ID-numero. Näin toimeksianto saadaan yksilöityä. Ohjelma kirjaa suorituksen suoraan järjestelmään ja päivittää automaattisesti toimeksiannon tiedot ajan tasalle.

Ennen automaattista päivittämistä tulee käyttäjälle näkyviin ikkuna, josta näkee toimeksiannon päivitettävät tiedot ja voi tarkastaa niiden paikkansapitävyyden. Mikäli suoritus on myöhässä, ohjelma laskee automaattisesti suorituksen viivästyskoron ja kirjaa sen kyseisen toimeksiannon tietoihin. Kun velallinen on maksanut velkasumman kokonaisuudessaan, ohjelma tarkastaa mahdolliset viivästyksestä aiheutuneet korot, ja näiden tietojen perusteella käyttäjä voi lähettää velalliselle vielä ylimääräisen laskun.

Saatujen maksujen välittäminen toimeksiantajalle

Käyttäjän tarkastettua toimeksiannon suorituksen paikkansapitävyyden ja hyväksyttyään sen, ohjelma kirjaa automaattisesti suorituksen kyseisen toimeksiannon rahaliikenteeseen. Tämän jälkeen ohjelma ottaa yhteyden Osuuspankin eräsiirtopalveluun, lähettää rahat automaattisesti perintätoimiston pankkitililtä toimeksiantajan tilille ja ilmoittaa käyttäjälle toimenpiteestä. Virheiden välttämiseksi ohjelma kysyy ennen rahojen lähettämistä käyttäjältä varmistuksen toimenpiteeseen.

3.2.9 Eri koneilla pidettyjen toimeksiantojen synkronointi

Ohjelman tulee toimia useammalla eri tietokoneella yhtä aikaa. AOC:n käytössä on kaksi tietokonetta; toimistolla pöytätietokone, sekä asiakastapaamisia varten kannettava tietokone. Sekaannusten välttämiseksi tiedot tulee synkronoida tietyin aikavälein, jotta ne pysyvät ajan tasalla kummassakin koneessa. Tietojen synkronointi tulee toteuttaa kerran päivässä. Esimerkiksi sähköpostin välityksellä tai USB-muistitikulla tietojen päivittäminen koneiden välillä on yksinkertaista ja nopeaa.

AOC:n edustajan mukaan tietojen päivittäminen kerran päivässä on tarpeeksi usein, koska saman päivän aikana ei koskaan työskentele kahta eri henkilöä saman toimeksiannon parissa. Tällöin tiedoissa ei pääse tapahtumaan päällekkäisyyksiä tai ristiriitoja.

3.2.10 Kirjepohjien muokkaaminen

Ohjelmaan tulee sisällyttää AOC:n toiveiden mukaiset velallisille sekä viranomaisille lähetettävät kirjepohjat, joihin ohjelma osaa käyttäjän toimintojen perusteella hakea tietokannastaan tarvittavat tiedot. Puuttuvat tiedot käyttäjä lisää kirjeeseen manuaalisesti.

Kirjestandardien, yms. muuttuessa aika-ajoin, yksi ohjelman tärkeistä ominaisuuksista on käyttäjän mahdollisuus muokata valmiiksi luotuja kirjepohjia. Tähän tulee suunnitella mahdollisimman yksinkertainen päivitysmenetelmä. Kirjepohjien muokkaamisen tulee onnistua myös käyttäjältä, joka ei hallitse lainkaan alkuperäisten kirjepohjien muodostamisessa käytettyä ohjelmointikieltä.

3.2.11 Tietokannan varmuuskopiointi

Tietokannassa olevien tietojen katoamisen välttämiseksi tiedoista tulee ottaa varmuuskopio erilliselle tallennusmedialle säännöllisin väliajoin. Esimerkiksi jokaisen työpäivän päätteeksi eri tietokoneiden tietokannat tulee synkronoida ajan tasalle ja ottaa tietokannan uusimmasta versiosta varmuuskopio. Näin ollen käyttäjän tietokoneen hajotessa, tai jonkin suuren ongelman sattuessa, ainoastaan yhden vuorokauden aikana kirjatut tiedot katoavat.

Tutkittuani tallennusvaihtoehtoja totesin USB-muistitikun käyttämisen varmuuskopioinnissa parhaimmaksi vaihtoehdoksi. Sen käyttäminen on nopeaa ja helppoa. Olisi myös järkevää tallentaa varmuuskopiot muistitikulta esimerkiksi kerran kuukaudessa CD-levylle arkistointia varten. Mikäli uusimpien tietojen päivittäminen USB-muistitikulle ei ole mahdollista, esimerkiksi eri työpisteiden takia, paras vaihtoehto on uusimman tietokannan lähettäminen tietokoneelta toiselle sähköpostin välityksellä.

3.3 Projektin vaiheistus

Projektin vaiheistamisessa käytän inkrementaali- eli evoluutiomallia. Evoluutiomallilla tarkoitetaan projektin suorittamista vaihe kerrallaan. Ensimmäisessä vaiheessa toteutetaan ohjelman ydin ja seuraavissa vaiheissa lisätään ominaisuuksia yksi kerrallaan (Kettunen 2002, 58).

Prototyyppiini ohjelmoin yksinkertaistetun version graafisesta käyttöliittymästä, toimeksiannot-välilehden toiminnot, sekä niiden taustalle tietokannan tietojen säilyttämispaikaksi.

Myöhemmin ohjelmaan toteutetaan seuraavat ominaisuudet:

- toimeksiantajat-välilehden toimintojen ohjelmointi sekä osion lisääminen tietokantaan
- haku-välilehden toimintojen ohjelmointi
- rahaliikenne-välilehden toimintojen ohjelmointi
- kirjepohjien ohjelmointi
- lomakkeiden muokkausominaisuuden ohjelmointi sekä dokumentointi
- Osuuspankin Eräsiirtopalvelun rajapinnan ohjelmointi

4 Toteutussuunnitelma

AOC:n toiveiden mukaisen ohjelman valmistamisessa on useita toteutusvaihtoehtoja. Seuraavissa kappaleissa tutkin osaamistani parhaiten vastaavien vaihtoehtojen suunnittelemista.

4.1 Toteutusmenetelmät

Ohjelman toteutustyökaluja valitessani tärkeimmät huomioon otettavat asiat olivat ohjelman toteuttamisen helppous sekä AOC:lle aiheutuvat kustannukset. Pohdinnan ja tutkimusteni jälkeen totesin koulussamme opetettavien ohjelmointikielten perusteella parhaan tavan ohjelman toteuttamiseen olevan Javan sekä MySQL:n käyttäminen. Kyseisten kielten yhteiskäyttäminen on koulumme ohjelmistotuotannon suuntautumisvaihtoehdokseen valinneille opiskelijoille tuttua Tietokanta- ja Käyttöliittymäohjelmointi-kurssilta. Kurssilla opetetaan ohjelmoimaan Javalla graafinen käyttöliittymä, jonka taustalle luodaan MySQL-tietokanta tietojen säilytyspaikaksi.

Toinen vaihtoehto koulussamme saatavien oppien perusteella ohjelman toteuttamiseen on HTML- sekä PHP -ohjelmointikielten käyttäminen. Koulussamme opetetaan molempia ohjelmointikieliä ja nämä kielet hallitsevat opiskelijat pystyvät varmasti, aiheeseen perehdyttyään, toteuttamaan ohjelman kyseisillä työkaluilla.

4.1.1 Graafinen käyttöliittymä

Käyttöliittymällä tarkoitetaan niitä keinoja, joilla muodostetaan yhteys tietokoneen ja käyttäjän välille. Prototyyppini graafinen käyttöliittymä on ohjelmoitu Java-ohjelmointikielellä. **”**Java on Sun Microsystemsin kehittämä tietoverkkoihin suunnattu olio-ohjelmointikieli. Se on C++:n pohjalta rakennettu lausekieli, jolle tyypillisiä piirteitä ovat muun muassa puhdas oliopohjaisuus, vahva tyypitys, C++:aa muistuttava syntaksi, automaattinen roskien keruu, viitesemantiikka, tulkittavuus ja WWW-ohjelmointituki.**”** (Westerholm & Kyppö 2001, 17)

Javalla graafisen käyttöliittymän ohjelmoiminen tapahtuu käyttämällä AWT -luokkajoukkoa (Abstract Windowing Toolkit). ”AWT on joukko Javan perusluokkia, joiden avulla luodaan graafinen käyttöliittymä ja hallitaan sitä.” (Cadenhead 2002, 400). Prototyyppini käyttöliittymän ohjelmoinnissa käytin AWT:n laajennusta nimeltään Swing. Siinä on joukko luokkia käyttöliittymän ulkoasun, toimintojen ja hallintamahdollisuuksien parantamiseksi (Cadenhead 2002, 296).

Prototyyppini graafinen käyttöliittymä on jaettu asiakokonaisuuksien mukaan välilehdille. Välilehtien käyttäminen ohjelmassa selkeyttää ohjelman käyttämistä huomattavasti verrattuna esimerkiksi ohjelmaan, jossa navigoidaan sivulta toiselle liukuvalikoiden avulla. Välilehtien otsikot ovat koko ajan käyttäjän nähtävillä, mikä helpottaa ohjelman kokonaishahmottamista ja nopeuttaa ohjelman käyttämistä.

Kuva 6: Prototyypin välilehdet

Jotta käyttäjä ei pysty vahingossa poistamaan käyttöliittymään kirjaamiaan tietoja, ohjelmaan on lisätty ominaisuus varmistamaan, ettei käyttäjä pysty navigoimaan välilehdeltä toiselle ilman välilehden tietojen tallentamista tai tyhjentämistä. Mikäli käyttäjä on tehnyt muutoksia avoimena olevan välilehden tietoihin ilman tallentamista, siirryttäessä välilehdeltä toiseen ohjelma varmistaa, haluaako käyttäjä tallentaa tekemänsä muutokset tietokantaan. Tällä varmistetaan kaikkien haluttujen tietojen tallentuminen tietokantaan.

4.1.2 Tietokanta

Tietokanta on kokoelma tietoja, jotka on organisoitu siten, että sen sisältöön pääsee helposti käsiksi, sitä voi järjestellä ja päivittää. Yleisin tietokantatyyppi on taulukkona esitetty relaatiotietokantamalli, jonka toisiinsa yhteydessä olevat tiedot on määritelty siten, että niitä voi järjestellä useiden kriteerien mukaan.

Koulussamme opittujen ohjelmointitekniikoiden perusteella valitsin tietokannan toteuttamistavaksi upotetun MySQL:n käyttämisen. Tällä tarkoitetaan MySQL-komentojen kirjoittamista toisen ohjelmointikielen, tässä tapauksessa Javan sisälle. Ohjelmassani kaikki käyttäjän lisäämät tiedot toimeksiannoista, toimeksiantajista, velallisista, yms. kirjataan graafisen käyttöliittymän taustalla pyörivään relaatiotietokantaan.

Relaatiotietokannalla tarkoitetaan tietokantaa, jossa kaikki tiedot tallennetaan tauluihin. Yhteen tauluun tallennetaan vain yhteen asiakokonaisuuteen liittyviä tietoja. Prototyyppiin tallensin ainoastaan toimeksiannon tiedot. Kaikki toimeksiannon tiedot tallensin samaan tauluun. Tällöin kullekin riville tulee yhden toimeksiannon tiedot ja kuhunkin sarakkeeseen tulee yksi tietue, esimerkiksi toimeksiannon ID-numero. Valitsin toimeksiannot-taulun perusavaimek-

si automaattisesti tietokoneen lisäämän ID-numeron, joka erottaa tietueet toisistaan.

4.1.3 Tekijänoikeus

Ohjelman markkinoinnin ja myynnin takia on kiinnitettävä huomiota tekijänoikeuksiin. Jokaisella ihmisellä on oikeus tehdä sovelluksia Javalla sekä MySQL:llä. Näin ollen tuottamani ohjelma ei ole tekijänoikeus-lain alainen.

Java ja MySQL ovat nykyään molemmat vapaan lähdekoodin (open source) alaisia. Open source eli avoin lähdekoodi on lisenssi, joka oikeuttaa esimerkiksi Javan käyttäjiä ohjelmoimaan omia sovelluksia ilmaiseksi edellyttäen, että käyttäjät noudattavat GNU General Public License –sopimusta. GNU-sopimuksen noudattamisella tarkoitetaan sitä, että käyttäjällä on oikeus kopioida ja muuttaa, sekä jakaa edelleen ohjelmia ja niiden lähdekoodia. Tämä kuitenkin koskee ainoastaan tilannetta, jossa Javan tai MySQL:n moottoreihin tehdään muutoksia. Ohjelmani valmistuksessa tehdään ainoastaan komentoja kyseisillä ohjelmointikielillä, joten lähdekoodeja ei tarvitse näyttää kenellekään ulkopuoliselle.

4.2 Tulevat työvaiheet ja niiden ongelmakohdat

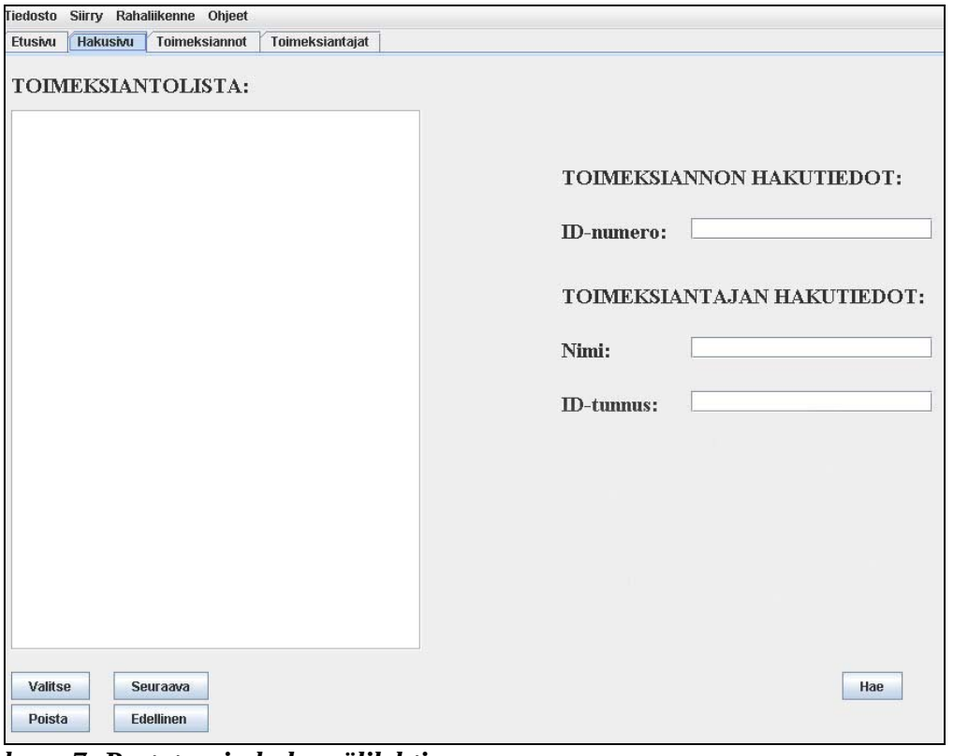
Lopullisen ohjelman valmistamisen täydelliseen onnistumiseen liittyy useita ongelmakohtia. Suurimmat näistä liittyvät muutamien työvaiheiden monimutkaisuuteen. Ohjelman tuottaminen noudattaen täysin AOC:n vaatimuksia vaatisi minulta useaan täysin uuteen asiaan perehtymistä. Näiden asioiden sisäistämiseen vaadittavat resurssit olisivat varsin suuria.

Seuraavissa kappaleissa olen pohtinut tulevia työvaiheita, niiden ongelmakohtia ja ratkaisuehdotuksia. Huomioitavaa on, että jokaisen työvaiheen viimeisenä osiona on tietokannan päivittäminen ajan tasalle. Tätä en siis mainitse työvaiheiden toteutusmenetelmissä enää erikseen.

4.2.1 Toimeksiantajat- ja haku-välilehtien ohjelmointi

Toimeksiantajat-välilehden toimintojen ohjelmoinnissa voi käyttää samaa menetelmää, kuin prototyyppiini ohjelmoimassani toimeksiannot-välilehdessä. Kyseisen toteutusvaihe on erittäin yksinkertainen, koska toimeksiannot-välilehden toteutuksesta saa selkeät ohjeet ohjelmoimiseen. Tietokannan ohjelmoinnissa tulee huomioida, että tietokannan laajentuessa normalisointi on erittäin suotavaa, jotta tietokannan käyttäminen pysyy nopeana. Ainoat toi-

meksiantajat-välilehdeltä puuttuvat asiat ovat toimintojen ohjelmoiminen, sekä tietokannan rakentaminen taustalle.



kuva 7: Prototyypin haku-välilehti

Haku-välilehdelle tulee ohjelmoida toiminto, joka hakee tietokannasta listan kyseisenä päivänä suoritettavista toimeksiannoista. Listasta käyttäjä voi valita haluamansa toimeksiannon ja siirtyä Valitse-painiketta painamalla työskentelemään toimeksiannon parissa. Poista-painikkeella tietokannasta tulee pystyä poistamaan listasta valittuna oleva toimeksianto. Seuraava- ja edellinen-painikkeilla pystyy päivittämään toimeksiantolistaan eri päivien toimeksiantoja. Suoritetut toimeksiannon toiminnot merkitään listaan esimerkiksi pienemmällä kirjasimella tai eri värillä, jolloin käyttäjä huomaa toimeksiannon hoidetuksi.

Haku-välilehdellä pitää myös pystyä siirtymään hakukriteerien perusteella haluamansa toimeksiannon tai toimeksiantajan tietoihin.

4.2.2 Rahaliikenne-välilehden ohjelmointi

Prototyyppini rahaliikenne-välilehdelle ohjelmoin ainoastaan AOC:n toiveiden mukaisen taulukon, josta ohjelman käyttäjä näkee kaikki tarvittavat tiedot toimeksiannon rahaliikenteestä. AOC:n vaatimusten mukaan prototyyppiini ei kuitenkaan voi ohjelmoida toimintoja ennen kuin eräsiirtopalvelun rajapinta on muodostettu.

Ohjelman valmiiseen versioon välilehdelle tulee sisällyttää useita toimintoja, jotka prototyypistäni puuttuvat. Esimerkiksi rahaliikenne-välilehdeltä täytyy pystyä muodostamaan yhteys Osuuspankin eräsiirtopalveluun. Välilehdellä tulee olla painike, jota painamalla ohjelma tarkistaa AOC:n pankkitililtä, onko sinne saapunut suorituksia velallisilta. Suorituksen löydyttäessä ohjelman tulee osata yhdistää suoritus oikeaan toimeksiantoon ja hakea tarvittavat tiedot rahaliikenne-välilehdelle. Mikäli pankkitilille on tullut useita suorituksia, ohjelma käsittelee suoritukset yksitellen saapumisjärjestyksessä.

Ohjelman haettua tarvittavat tiedot AOC:n pankkitililtä, sen tulee pystyä selvittämään, onko suoritus tullut ajoissa AOC:n tilille sekä kirjaamaan suoritus kyseenomaisen toimeksiannon tietoihin. Mikäli suoritus on myöhässä, ohjelman tulee laskea automaattisesti kertynyt viivästyskorko ja kirjata saatu suoritus ja viivästyskorko tietokantaan.

Yksinkertaistettu versio

Rahaliikenne-osiosta voi tehdä yksinkertaistetun version, jolloin siitä jätettäisiin pois rajapinta ohjelman ja Osuuspankin eräsiirtopalvelun väliltä. Tällöin rahaliikenne-osion sisällöstä tulee samankaltainen kuin prototyypissä, mutta ohjelma ei ota yhteyttä Osuuspankin eräsiirtopalveluun, vaan käyttäjä käy itse Osuuspankin internet-sivustolta katsomassa, onko suorituksia tullut velallisilta. Mikäli pankkitilille on tullut suoritus, käyttäjän tulee päivittää järjestelmään manuaalisesti oikean toimeksiannon tiedot suorituksen perusteella ja välittää suoritus toimeksiantajan pankkitilille.

4.2.3 Kirjepohjien luominen ja muokkaaminen

Yksi vaikeimmista ominaisuuksista on kirjepohjien sisällyttäminen ohjelmaan. AOC:n vaatimuksena oli, että ohjelmaan pitää laatia valmiit kirjepohjat jokaista kirjemallia varten. Ohjelman tulee osata hakea tietokannasta kyseisen toimeksiannon saatavilla olevat tiedot käyttäjän täyttäessä manuaalisesti puuttuvat kohdat. Tämä ominaisuus nopeuttaa huomattavasti käyttäjää kirjeiden lähettämisessä velallisille, muttei ole toiminnan kannalta välttämätön ominaisuus.

Ensimmäiseksi ongelmaksi muodostuu kirjepohjien luomiseen käytettävän työkalun valinta. Yksi varteenotettava ja toimiva vaihtoehto on PHP- ja HTML-ohjelmointikielien käyttäminen ongelman ratkaisemisessa. Itse en hallitse kumpaakaan kieltä, joten ominaisuuden ohjelmoiminen vaatisi kahden uuden kielen sisäistämistä, sekä niiden yhteiskäytön opettelemista.

Toisen ongelman muodostaa kirjepohjien muokkaaminen jälkikäteen. AOC:n vaatimusten mukaan heidän tulee pystyä muokkaamaan kirjeitä jälkikäteen, vaikka heidän yrityksessään kukaan ei hallitse mitään ohjelmointikieltä. Kirje-

pohjia pitää pystyä editoimaan yhtä yksinkertaisesti kuin esimerkiksi Word-tekstinkäsittelyohjelman dokumentteja.

Kirjepohjien alkuperäisen luomistavan selvittämisen ja toteuttamisen jälkeen kirjepohjien muokkaaminen onnistuu helposti käyttämällä samaa tekniikkaa kuin luomisessa. Ongelman muodostaa kuitenkin luomistavan opettaminen ohjelman käyttäjille. Kirjepohjien muokkaaminen vaatii PHP- ja HTML-ohjelmointikielien hallitsemista.

Ongelman ratkaisemiseksi luomistavasta voi toteuttaa ohjekirjan. Ohjekirjan kirjoittaminen kuitenkin lisää projektin laajuutta entisestään. Tarpeeksi ymmärrettävän sekä kattavan ohjekirjan luominen vaatii useita työtunteja ja lisää ohjelman yhteiskustannuksia.

AOC:n eräs vaihtoehto on hankkia tarvittaessa teknistä tukea esimerkiksi ohjelman laatijalta. Laatija voi toimia avustajana myös projektin päättymisen jälkeen ja tekee ohjelmaan muutoksia heidän vaatimustensa perusteella sekä muokkaa tarvittaessa kirjepohjia. Tällä menetelmällä voi myös päivittää ohjelmaa AOC:n toivomusten mukaan, joten ohjelman toimintatapoja voi parannella tarvittaessa entistä paremmin soveltumaan heidän vaatimuksiinsa. Tämä menetelmä kuitenkin muodostaa lisäkustannuksia AOC:lle.

4.2.4 Osuuspankin eräsiirtopalvelun rajapinnan ohjelmointi

Suurin este projektin onnistumiselle AOC:n vaatimusten mukaisesti on rajapinnan ohjelmoiminen ohjelman ja Osuuspankin eräsiirtopalvelun välille. Kyseinen ominaisuus on AOC:n mielestä erittäin olennainen, mutta toteuttamiseen vaadittavaa ohjelmointitaitoa ei koulumme kursseilla saa.

Osion sisällyttäminen ohjelmaan tulee ainoastaan nopeuttamaan käyttäjän työskentelyä. Täten osio ei ole välttämätön. Ilman tätä ominaisuutta käyttäjä tosin joutuu internet-selaimen avulla tarkistamaan AOC:n pankkitilille tulleet velallisten suoritukset, sekä välittämään manuaalisesti suoritukset toimeksiantajan tilille, päivitettyään ensin toimeksiannon tiedot ohjelman tietokantaan.

Koulustamme saatavien ohjelmointioppien avulla rajapinnan muodostaminen on erittäin vaativa ja työläs toteuttaa. Eräs vaihtoehto ongelman ratkaisemiseen on, että joku koulumme opiskelijoista muodostaa rajapinnan, sekä siihen liittyvän dokumentoinnin esimerkiksi omana opinnäytetyönään tai projektiopintoina.

Opiskelijalla olisi mahdollisuus käyttää oppaana Osuuspankilta tai heidän internet-sivuilta saatavaa 94 sivuista Osuuspankin Eräsiirtopalvelun liittymäkuvausta. Perehdyttyäni kyseiseen liittymäkuvaukseen, huomasin sen olevan erittäin vaikeaselkoinen. Perusteellisenkaan syventymisen jälkeen en löytänyt ratkaisua ongelmaan. Mikäli joku opiskelijoistamme ohjelmoi kyseisen rajapin-

nan, hänellä on mahdollisuus pyytää yksityisopetusta Osuuspankin ATK-tukihenkilöiltä.

Eräs vaihtoehto ongelman ratkaisemiseksi on antaa rajapinnan ohjelmoiminen ohjelmistotoimiston hoidettavaksi. Ohjelmistotoimistoilla pitäisi olla tarvittavat resurssit sekä tietotaito ongelman ratkaisemiseen. Kyseisessä tapauksessa ohjelman valmistuskustannukset kasvaisivat merkittävästi. Tällöin tulisi selvittää, onko ominaisuus sen arvoinen.

5 Ohjelmani vaatimat sovellukset sekä niiden kustannukset

Java-sovellukset sekä MySQL-tietokanta tarvitsevat toimiakseen tietyt ohjelmistot. Ohjelmat ovat ilmaisohjelmia, jotka ovat ladattavissa esimerkiksi internetistä.

5.1 Graafisen käyttöliittymän ohjelmat

Käyttäjän tulee asentaa tietokoneelleen Java-ohjelmien suorittamiseen tarvittavat työkalut. Javan kotisivuilta java.sun.com löytyy useita versioita, joista käyttäjä voi valita tarpeisiinsa parhaiten soveltuvan version. Suosittelen Javan uusimman ohjelmiston käyttämistä, mikä on *Java 2 Platform, Standard Edition, v 1.4.2 (J2SE).*

SDK (Software Development Kit)

SDK on Sun Microsystemsin valmistama Java-työkalu. SDK sisältää Java-sovellusten käyttämiseen tarvittavia sovelluksia, sekä ohjelmani käyttämiseen ja lähdekoodin muokkaamiseen tarvittavat työkalut. SDK:n uusin versio on Java(TM) 2 SDK, Standard Edition 1.4.2_13. Kyseisen paketin koko on 49.14 MB.

JRE (Java Runtime Environment)

Vaihtoehtoinen Java-työkalu ohjelmani toiminnan varmistamiseksi on JRE. SDK:sta poiketen JRE sisältää ainoastaan ohjelman käyttämiseen tarvittavan työkalun. Mikäli käyttäjä asentaa tietokoneeseensa JRE:n, hän ei pysty muokkaamaan Javalla toteutetun ohjelman lähdekoodia. JRE:stä uusin saatavilla oleva versio on *Java(TM) 2 Runtime Environment, Standard Edition 1.4.2_13*. Internetistä ladattavan paketin koko on 15.24 MB.

Nykyään tietokoneiden kovalevykapasiteetit ovat todella suuria ja edellä mainitut työkalut tarvitsevat kumpikin suhteellisen vähän tilaa, joten suosittelisin SDK:n asentamista käyttäjän tietokoneeseen. Kyseisestä työkalusta on mahdollisesti hyötyä myös muiden Java-sovellusten käyttämisessä kyseisellä tietokoneella.

5.2 Tietokannan ohjelmat

MySQL 5.0

Jotta ohjelman käyttämiseen tarvittavan tietokannan saa toimimaan, käyttäjän tietokoneelle tulee asentaa MySQL tietokantapalvelin. Suosittelen uusimman version, eli MySQL 5.0 käyttämistä. Tämän ohjelman avulla luotuun tietokantaan säilötään kaikki käyttäjän tallentamat tiedot toimeksiannoista, toimeksiantajista, kirjeistä, yms. Kyseisen ohjelman voi ladata ilmaiseksi internetistä osoitteesta dev.mysql.com/downloads/.

JDBC (Java Database Connectivity)

JDBC on universaali ja toimintariippumaton standardi, jonka avulla muodostin yhteyden Javan ja MySQL:n välille. JDBC on rajapinta eli väylä, jonka avulla Java-ohjelma sekä SQL-tietokanta kykenevät kommunikoimaan keskenään. Tämän rajapinnan avulla käyttäjä pystyy varastoimaan graafisessa käyttöliittymässä tallennetut tiedot tietokantaan, sekä hakemaan tietokannasta tietoja näytettäväksi graafisessa käyttöliittymässä. (Uusitalo 1997)

JDBC-rajapinnan voi ladata ilmaiseksi internetistä www.java.sun.com -sivustolta. Tällä hetkellä uusin saatavilla oleva versio on *JDBC 4.0 API***.**

5.3 Sovellusten lisäkustannukset

Javan sekä MySQL:n käyttäminen ohjelmani tekemiseen on ilmaista. Java ja MySQL ovat GNU General Public License -lisensointioption alaisia ohjelmointikieliä (www.gnu.org, www.mysql.com). Myös ohjelmani käyttämiseen tarvittavat ohjelmat JRE, SDK, JDBC sekä MySQL ovat ilmaisohjelmia.

Javan ja MySQL:n käyttäminen ei siis aiheuta lainkaan lisäkustannuksia AOC:lle. Näin ollen ohjelman valmistuksesta ei tule minulle mitään ohjelmointikielien aiheuttamia kustannuksia, joista joutuisin laskuttamaan AOC:ta.

6 Lopullisen ohjelmistohankinnan vaihtoehdot

Yrityksillä on useita mahdollisuuksia valita parhaiten soveltuva tapa hankkia jokin ohjelma. Seuraavaksi tutkin AOC:lle kolmea mielestäni varteenotettavinta vaihtoehtoa tuotteen hankkimiseen kirjoittamani toiminnallisen vaatimusmäärittelyn perusteella.

Tutkinnan pohjana käytin Sami Kettusen Tietojärjestelmän ostaminen -kirjassaan esittelemää taulukkoa, jossa vertaillaan valmiin tuotteen ja projektityön etuja sekä haittoja (Kettunen 2002, 38).

A) Markkinoilla oleva valmis tuote:

Markkinoilla on muutamia AOC:n käyttöön soveltuvia ohjelmia. Yrityksellä on jo ennestään Collector-niminen tuote, jonka he haluavat korvata uudella, yksinkertaisemmalla versiolla, sekä paremmin heidän käyttötarkoitukseensa soveltuvalla ohjelmalla.

Hyödyt:

- toimintavarma, valmiiksi testattu tuote
- ammattilaisten valmistama
- toimivat tukipalvelut
- useita valmiita rajapintoja muihin sovelluksiin
- kattava dokumentointi
- uudelleenkehitys
- AOC saisi tuotteen käyttöönsä heti

Haitat:

- AOC ei voi vaikuttaa tuotteen ominaisuuksiin, joten ohjelma ei välttämättä ole AOC:n tarpeiden mukainen
- kallis

Tuotteen tilaaminen ohjelmistotoimistolta

AOC:n toinen vaihtoehto tuotteen hankintaan on ohjelmiston tilaaminen ohjelmistotoimistolta heidän käyttötarkoituksiinsa soveltuvaksi räätälöitynä.

Hyödyt:

- AOC voi vaikuttaa tuotteen ominaisuuksiin
- toimivat tukipalvelut
- yksilöllinen ja vaikea kopioida
- ohjelmistotoimistolla on resurssit ja ammattitaito tehdä monimutkaisiakin sovellusratkaisuja
    - valmiiden lomakkeiden valmistaminen sekä muokkaaminen
    - tilisiirto OP:n nettipankin kautta
- ammattilaisten valmistama tuote

Haitat:

- vaihtoehdoista kallein
- integrointi muihin ohjelmiin voi tuottaa ongelmia
- jatkokehittäminen kohtuuttoman kallista

Tuote oppilastyönä TAMK:n kautta

Kolmas vaihtoehto olisi ohjelman hankkiminen TAMK:n opiskelijan tekemänä.

Hyödyt:

- huomattavasti edullisempi kuin kaksi aiempaa vaihtoehtoa
- AOC voi vaikuttaa tuotteen sisältämiin ominaisuuksiin

Haitat:

- ylitsepääsemättömien ongelmien mahdollisuus
- ohjelmoijan huomattavasti vähäisempi kokemus sekä ammattitaito verrattuna aiempiin vaihtoehtoihin
- vaikeuksia jatkokehittämisen kanssa
- dokumentoinnin epäpätevyys

Prototyyppini valmistumisen jälkeen AOC päätti hankkia tuotteen ohjelmistotoimistolta. Hankintatavan kalleudesta huolimatta he päätyivät kyseiseen vaihtoehtoon. Tärkeimmiksi syiksi he kokivat ohjelman toimintavarmuuden sekä valmistumisnopeuden.

7 Parannusehdotukset

Tallennusmuoto

Tietokoneille, joilla käytetään prototyyppini pohjalta tehtyä ohjelmaa, kannattaisi luoda kaksi komentoa:

- tietokannan pakkaaminen normaalimuodosta XML-muotoon
- tietokannan purkaminen XML-muodosta normaalimuotoon

Komennot voisi luoda esimerkiksi tietokoneen työpöydälle ja komentokuvaketta painamalla tietokone tekisi automaattisesti kyseisen komennon.

XML-muoto olisi järkevä ratkaisu silloin, kun tietokannan tietoja täytyisi siirtää tietokoneelta toiselle. Tietokannan pakkaaminen XML-muotoon pienentäisi tiedostokokoa huomattavasti, jolloin esimerkiksi sähköpostin välityksellä tietokannan lähettäminen tietokannalta toiselle olisi käytännöllisempää.

Transaktio

Suurempien tietokantojen suunnittelemisessa transaktioiden huomioiminen on tärkeää tietokannan eheyden sekä tietojen katoamisen välttämiseksi. Transaktiolla tarkoitetaan tietokantaan tehtyjen toimintojen ketjuttamista yhtenäiseksi kokonaisuudeksi, jossa toiminnot suoritetaan loogisessa järjestyksessä. Transaktion onnistuminen edellyttää jokaisen transaktion sisältämän toiminnon onnistumista. Vasta tämän jälkeen toiminnot suoritetaan lopullisesti ja muutokset ovat muiden prosessien saatavilla ja tiedot tallennetaan tietokantaan. Mikäli yksikin transaktion toiminto epäonnistuu, kaikki transaktion toimenpiteet perutaan. (Gilmore, 2005, 683)

Tässä ohjelmassa transaktiosta olisi hyötyä esimerkiksi rahaliikenteen hoitamisessa. Mikäli ohjelmaan saadaan integroitua rajapinta OP:n eräsiirtopalveluun, saatavien välittäminen AOC:lta toimeksiantajalle tulisi rakentaa transaktioksi. Transaktion ensimmäinen toiminto olisi velalliselta saadun rahan ottaminen AOC:n tililtä, toinen toiminto olisi suorituksen tietojen kirjaaminen järjestelmään ja viimeinen toiminto olisi rahan lähettäminen toimeksiantajan tilille. Vasta kun kaikki toiminnot on onnistuneesti suoritettu, transaktio hyväksytään ja muutokset tulevat voimaan. Transaktion käyttäminen antaa käyttäjälle mahdollisuuden perua toimenpide mahdollisen näppäilyhäiriön, tms. tapahtuessa ennen kuin raha lähetetään eteenpäin.

Normalisointi

Tietokannan suunnitteleminen on olennainen ja erittäin tärkeä osa tietokannan ohjelmoimisessa. Ohjelman viimeistellyssä versiossa tietokanta tulisi ehdottomasti normalisoida. Normalisointi on menetelmä, jossa tietorakenteet järjes-

tellään ”parempaan” tallennusmuotoon. Normalisoitu tietokanta on tavallista nopeampi, tietojen turhat toistamiset on minimoitu ja tietojen eheys on varmistettu. Kun tietokanta on normalisoitu, tietokanta on ns. normaalimuodossa. (Hovi & Huotari & Lahdenmäki 2005)

Dokumentointi

Koska tietokannan rakenteen suunnittelussa ei ole yhtä ainoaa oikeaa ratkaisua, rakenteen myöhempää muokkaamista helpottaakseen tietokannan luojan tulisi kirjoittaa tietokannasta dokumentti. Mitä isommasta ja monimutkaisemmasta tietokannasta on kyse, sen tärkeämmäksi dokumentin kirjoittaminen muodostuu. Näin ollen pitkänkin tauon jälkeen tietokannan rakenteen muokkaaminen sujuu helpommin muokkaajalta. Dokumentin avulla uusikin ohjelmoija pystyy sisäistämään helposti tietokannan rakenteen.

Stephensin, Plewin, Morganin ja Perkinsin mukaan dokumentin tulisi sisältää seuraavat asiat:

- kuvaus tietokannan tarkoituksesta ja käyttäjistä
- kuvaus varsinaisen tietokannan tärkeistä taustatiedoista: millä se on luotu, tietokannan oletuskoko tai lokitiedoston koko
- tietokannan asennus- ja poistoscriptit sekä ohjeet tietokantaan liittyvien import/export-työkalujen käytöstä
- yksityiskohtaiset kuvaukset jokaisesta tietokannan taulusta ja niiden tarkoitukset sovelluksen kannalta
- kuvauksen jokaisen taulun rakenteesta mukaan lukien kaikki kentät ja niiden tietotyypit kommentteineen, indekseistä ja näkymistä
- lähdekoodin kaikista tietokannan tallennetuista proseduureista ja triggereistä
- kuvauksen tietokannan rajoituksista kuten yksilöllisistä tai NOT NULL -arvoista. Dokumentissa tulee myös mainita, huolehtiiko näistä rajoituksista tietokanta itse, vai onko ohjelmoijan tehtävä se koodissa.
  (Stephens & Plew & Morgan & Perkins 1999, 178)

8 Arviointi

8.1 Projektin toteutus

8.1.1 Toiminnallisen vaatimusmäärittelyn kirjoittaminen

Koin toiminnallisen vaatimusmäärittelyn kirjoittamisen erittäin tärkeäksi, koska sen perusteella saimme AOC:n edustajan kanssa yhtenäisen käsityksen valmistettavan ohjelman ominaisuuksista. Vaatimusmäärittelyn tekemisen jälkeen oli helpompi hahmottaa projektin vaativuus ja laajuus. Tosin en vielä vaatimusmäärittelyn kirjoittamisenkaan jälkeen täysin käsittänyt, miten laajasta ja vaikeasta projektista on kyse.

8.1.2 Prototyypin graafisen käyttöliittymän suunnittelu ja toteutus

Toiminnallisen vaatimusmäärittelyn jälkeinen työvaihe oli graafisen käyttöliittymän suunnitteleminen ja luominen. Mielestäni projektin alkuaikoina pidetyt ohjelman vaatimuksia ja ulkoasua koskevat palaverit onnistuivat niin hyvin, että niiden perusteella graafisen käyttöliittymän ulkoasun suunnitteleminen oli helppoa.

Graafisen käyttöliittymän ohjelmoiminen Java-kielellä oli minulle jo ennestään tuttu asia Käyttöliittymä- ja Tietokantaohjelmointi -kurssilta. Tosin prototyyppini graafisen käyttöliittymän ohjelmoimisen toteutin yli vuosi kyseisen kurssin päättymisen jälkeen ja asioiden mieleen palauttamiseen meni jonkun verran aikaa.

8.1.3 Prototyypin tietokannan suunnittelu ja toteutus

Prototyyppiini suunnittelin ja toteutin ainoastaan tietokannan toimeksiantajien tiedoista. Tietokannan rakentamisessa käytin Käyttöliittymä- ja Tietokantaohjelmointi -kurssilla oppimaani ns. yksinkertaistettua tapaa. Toimeksiannot -taulussa en käyttänyt normalisointia, enkä transaktioita, vaan loin ainoastaan yhden taulun, johon sijoitin kaikki tiedot toimeksiannosta.

Tietokannan taulun sekä tarvittavien SQL-komentojen ohjelmoiminen sujui mielestäni hyvin. Tietokannan suunnittelussa ainoa ongelma liittyi JDBC-rajapinnan muodostamiseen. Aluksi yhteyden muodostaminen tuotti ongelmia, koska prototyyppiä ohjelmoidessani saatavillani oli eri sovellus, kuin mihin olin tottunut *Tietokanta- ja käyttöliittymäohjelmointi -kurssilla*. Lopulta onnis-

tuin lisäämään Java-projektiini tarvittavan External JAR -paketin, jonka avulla rajapinnan muodostaminen onnistui.

8.1.4 Toteutussuunnitelman kirjoittaminen

Ohjelman toteutusmenetelmän valinta oli helppo. Jo ensimmäisessä palaverissa päädyimme ehdotukseni perusteella yksimieliseen ratkaisuun toteutusmenetelmästä. Mielestäni kaikkein järkevimmältä tuntui tuottaa prototyyppi Javalla ja MySQL:llä, koska ne olivat minulle jo ennestään tuttuja ohjelmointikieliä.

8.2 Projektin tulokset

Projektin toiminnallinen vaatimusmäärittely on toteutettu täysin AOC:n toiveiden mukaisesti. Tämä osa-alue on mielestäni onnistunut suunnitelmien mukaan. Pääsimme AOC:n edustajan kanssa hyvin yhteisymmärrykseen ohjelman vaatimusten suhteen.

Vaikka AOC päättikin hankkia ohjelman ohjelmistotoimistolta, uskon opinnäytetyöstäni olleen apua AOC:n ja ohjelmistotoimiston ensimmäisissä palavereissa, ohjelman sisältöä selvittäessä.

8.3 Kokemuksia projektista

Projekti käynnistyi erittäin hyvin. Projekti tuntui mielenkiintoiselta ja haastavalta. Motivaatiota lisäsi myös se, että pääsin tekemään opinnäytetyöni oikealle yritykselle ja valmistamastani ohjelmasta tulisi tärkeä työkalu heidän jokapäiväisessä työssään.

Projekti oli erittäin monipuolinen ja vaativa. Pääsin hyödyntämään koulumme kursseilla oppimiani ohjelmointitaitoja sekä Javassa että MySQL:ssä. Molemmista ohjelmointikielistä opin paljon uusia asioita, sekä tuttujen asioiden ohjelmoiminen rutinoitui entisestään. Vaikka AOC päättikin prototyyppini valmistumisen jälkeen hankkia ohjelmansa ohjelmistotoimistolta, koin prototyypin sekä kirjallisen osuuden valmistamisen erittäin hyödylliseksi.

Lähteet

Cadenhead, Rogers, 2002. *Java 2, Trainer Kit*. Helsinki: Edita Prima Oy.

Gilmore, Jason, W, 2005. *PHP & MySQL tehokas hallinta*. Jyväskylä: Gummerus Kirjapaino Oy.

GNU and the Java language [online] [viitattu 17.1.2007]. www.gnu.org/software/java/

Hovi, Ari & Huotari, Jouni & Lahdenmäki, Tapio, 2005. *Tietokantojen suunnittelu & indeksointi*. Jyväskylä: Docendo Finland Oy.

Kettunen, Sami, 2002. *Tietojärjestelmän ostaminen -käytännön opas yrityksille*. WS Bookwell Oy.

MySQL Open Source License. [online] [viitattu 18.1.2007]. www.mysql.com/company/legal/licensing/open source-license.html

Osuuspankin Eräsiirtopalvelun liittymäkuvaus-dokumentti

Pohjonen, Risto, 2002. *Tietojärjestelmien kehittäminen*. Jyväskylä: Docendo Finland Oy.

Stephens, Ryan, K. & Plew, Ronald, R. & Morgan Bryan & Perkins Jeff, 1999. *SQL Tietokantaohjelmointi, Pro Trainer*. Jyväskylä: Gummerus Kirjapaino Oy.

Uusitalo, M 1997. Intranet. [online] [viitattu 18.1.2007]. www.tml.tkk.fi/Studies/Tik-110.300/1997/Essays/jdbc.html

Westerholm, Mika & Kyppö, Jorma, 2001. *Java-ohjelmointi, Pro Training*. Helsinki: Satku.

Wiegers, Karl, E., 2003. *Software Requirements*. Washington: Microsoft Press.