SAIMAAN AMMATTIKORKEAKOULU
Tekniikka Lappeenranta
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Jaakko Purhonen

INNOMOBIILI - MOBIILI TIEDONKERUU- JA RAPORTONTISOVELLUS

Opinnäytetyö 2010

TIIVISTELMÄ

Jaakko Purhonen
Innomobiili - mobiili tiedonkeruu- ja raportointisovellus, 86 sivua
Saimaan ammattikorkeakoulu, Lappeenranta
Tekniikka, Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto
Opinnäytetyö 2010
Ohjaaja: lehtori Martti Ylä-Jussila, Saimaan ammattikorkeakoulu
Toimitusjohtaja Markus Heikkinen, Innotek Oy

Opinnäytetyön aiheena on Innotek Oy:lle tehtävä mobiili tiedonkeruu- ja raportointisovellus, Innomobiili. Innotek Oy on lieksalainen LVI-tuoteinnovaatio yritys, jonka taustalla on toimintamalli nimeltä Energo-ohjelma, jonka tarkoitus on auttaa yrityksen asiakkaita turhan vedenkulutuksen minimoimiseen, asuntokohteisiin tehtävien kartoitusten ja LVI-asennusten avulla.

Toistaiseksi kartoitukseen ja työtehtäviin liittyvä raportointi on hoidettu paperilomakkein, joka on aikaa ja resursseja hukkaavaa. Tehostaakseen toimintaansa Innotek Oy on aloittanut sähköiseen raportointimallin protoilun vuonna 2004. Tästä on tuloksena prototyypit Innomobiili-tiedonkeruu- ja raportointisovelluksesta sekä Innonet-toiminnanohjausjärjestelmästä. Kyseiset järjestelmät sekä muu tuotettu materiaali toimivat pohjana tässä työssä kehitettävälle Innomobiili-sovellukselle.

Työn tavoitteena on määritellä, suunnitella ja toteuttaa ensimmäinen toimiva versio Innomobiili-sovelluksesta. Sen toimintoihin kuuluvat käyttäjäkohtaisten työnantojen hakeminen sekä Energo-kartoitusten, asennusten, reklamaatioiden ja Laajojen kuntoraporttien kirjaaminen.

Innomobiili-sovelluksen kehitykseen on käytetty Visual Studio 2008-kehitysympäristöä sekä C#-ohjelmointikieltä. Kehityksessä käytetään Windows Mobile SDK:n mukana tulevaa .NET Compact Frameworkia sekä Pocket PC -emulaattoria. Tietokantana toimii MySQL-tietokanta.

Työn tuloksena määriteltiin yksi versio Innomobiili-sovelluksesta. Sovellukseen ideoitujen uusien muutosten takia työtä ei saatu valmiiksi tämän opinnäytetyön puitteissa, mutta projekti jatkuu Evgeni Kovalevin opinnäytetyönä

Avainsanat: mobiiliohjelmointi, Windows Mobile, Visual Studio 2008, C#, Energo-kartoitus

ABSTRACT
Jaakko Purhonen
Innomobiili - mobile data collection and reporting application, 86 pages
Saimaa University of Applied Sciences, Lappeenranta
Technology, Degree Programme in Information Technology
Software Engineering
Bachelor's Thesis 2010
Instructors: lecturer Martti Ylä-Jussila, Saimaa University of Applied Sciences
Managing director Markus Heikkinen, Innotek Oy

The purpose of this thesis was to develop a mobile application, Innomobiili, for Innotek Oy. The HVAC-oriented Innotek Oy is a company which aims at minimising its customers' everyday water consumption. The activity behind Innotek Oy is named as the Energo-programme and its main phases are the mapping of the customers' apartments HVAC-fittings and the installation of new, less consuming ones.

So far all of the data collecting and reporting has been mainly done using paper forms. This kind of reporting takes a lot of time and resources so Innotek Oy has started to alter its ways towards electronic reporting. The goal is that the electronic reporting system would completely replace the usage of paper forms.

The transition to electronic reporting at Innotek Oy was started in 2004. As a result of this Innotek Oy has two prototype systems: Innonet, a system for enterprise resource planning and Innomobiili, a mobile data collection application. These systems serve as a base in development of a new Innomobiili application.

The main features of Innomobiili are fetching the user's daily work assignments and the data collection for apartment mappings and installation procedures. Innomobiili communicates with a database but all the work assignment related tasks should be carried out without Internet connection.

Innomobiili application will be developed with Visual Studio 2008 using C#-programming language utilizing the .NET Compact Framework that is deployed with Windows Mobile SDK. It also provides a Pocket PC emulator that is used to test the application. Innomobiili uses a MySQL database.

One version of Innomobiili was specified and defined as a result of the work. Since Innomobiili experienced numerous changes in functional requirements, all the work could not be done during this thesis. The project Innomobiili continues as Evgeni Kovalev's bachelor's thesis.

Keywords: mobile programming, Windows Mobile, Visual Studio 2008, C#, Energo-programme

SISÄLTÖ

KÄSITTEET, TERMIT JA LYHENTEET

.NET Compact Framework
.NET Frameworkin suppeampi versio, joka on tarkoitettu erityisesti mobiili- ja sulautettuihin ohjelmistoihin ja järjestelmiin

.NET Framework
Microsoftin kehittämä ohjelmistokomponenttikirjasto, jota Microsoftin Visual Studio -ympäristössä kehitettävät ohjelmistot käyttävät

Android
Linux-pohjainen käyttöjärjestelmä ja alusta mobiililaitteille

Apache-palvelin
Avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma

Avoin lähdekoodi
Termi, joka tarkoittaa ohjelmia, joita kuka tahansa voi korjata, kehittää, kopioida ja käyttää vapaasti

C# (C sharp)
Microsoftin kehittämä, kaupallisessa ohjelmistokehityksessä käytetty .NET-konseptin olio-ohjelmointikieli

C++
Kaupallisessa ohjelmistokehityksessä käytetty olio-ohjelmointikieli

FTP (File Transfer Protocol)
Protokolla, jolla siirretään tiedostoja tietokoneiden välillä

HTML (Hypertext Markup Language)
Merkkauskieli, jolla kuvataan WWW-sivujen ulkonäköä ja rakennetta

HTTP (Hypertext Transfer Protocol)
Protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon

GPS-paikannus (Global Positioning System -paikannus)
Yhdysvaltain puolustusministeriön kehittämä ja rahoittama satelliittipaikannus-järjestelmä. Siviilikäytössä mahdollistaa kohteen paikantamisen noin 5 - 10 metrin tarkkuudella

IEEE (Institute of Electrical and Electronics Engineers)
Maailman suurin tekniikan alan järjestö, jonka toimintaan kuuluu muun muassa standardien luominen ja hallinta

IEEE 830
IEEE:n standardi vaatimusmäärittelyn kirjoittamisesta

IEEE 1016
IEEE:n standardi tekniselle määrittelylle

Emulaattori
Tietokoneohjelma, joka pyrkii mahdollisimman tarkasti matkimaan kohteena olevan laitteen toimintaa

IP-osoite (Internet Protocol -osoite)
Numerosarja, joka yksilöi jokaisen Internet-verkkoon kytketyn laitteen

MySQL
Suosittu ja tehokas avoimen lähdekoodin SQL-tietokannan hallintajärjestelmä

PHP (PHP: Hypertext Processor)
Avoimen lähdekoodin tulkattava ohjelmointikieli, jota käytetään yleisesti WWW-ympäristössä

phpMyAdmin
Ilmainen, MySQL-tietokannan hallintaan tehty, Internet-selaimella käytettävä, sovellus

Pocket PC
Matkapuhelin, jossa on perinteisten mobiilipuhelintoimintojen lisäksi kämmentietokonetta muistuttavia ominaisuuksia. Eroaa älypuhelimista siinä, että Pocket PC:ssä on aina mukana elektroninen kynä ja kosketusnäyttö. Pocket PC:n täytyy myös sisältää Windows CE -käyttöjärjestelmä

Protokolla
Protokolla eli yhteyskäytäntö on käytäntö tai standardi, joka määrittelee ja mahdollistaa laitteiden tai ohjelmien väliset yhteydet

SDK (Software Development Kit)
Ohjelmistokehitykseen tarkoitettu sarja kehitystyökaluja, dokumentaatiota, lähdekoodia ynnä muuta mitä tarvitaan tietyn tekniikan tai sovelluksen kehitykseen

SQL
IBM:n kehittämä standardoitu kyselykieli tietokantojen käsittelyyn

UML (Unified Modeling Language)
Object Management Groupin standardoima graafinen mallinnuskieli olio-pohjaisten tietokonejärjestelmien ja -ohjelmien suunnittelua ja kehitystä varten

Visual Basic
Kaupallisessa ohjelmistokehityksessä käytetty .NET-konseptin olio-ohjelmointikieli

Visual Studio (VS 2008)
Microsoftin kehittämä sovelluskehitysympäristö. Sisältää täyden tuen mm. ohjelmointikielille kuten (Visual) C#, (Visual) C++ ja Visual Basic

Windows CE (Windows Compact Edition)
Muun muassa Pocket PC -laitteille tarkoitettu, suppeampi versio Microsoftin kehittämästä Windows-käyttöjärjestelmästä

XAMPP
Apache Friends:n kehittämä ja kasaama, ilmainen ja todella helppokäyttöinen palvelin paketti

XML (eXtensible Markup Language)
Rakenteellinen kuvauskieli tai standardi, joka on tarkoitettu puhtaasti tiedon kuvaamiseen ja siirtoon

Älypuhelin (Smartphone)
Mikä tahansa nykyaikainen puhelinlaite, mikä sisältää tietokonetta muistuttavia ominaisuuksia

1 JOHDANTO

Vaikka tekniikka kehittyy nykypäivänä kovaa vauhtia, ovat ihmiset silti riippuvaisia tietyistä perusluonnonvaroista. Yksi tällainen on vesi. On arvioitu, että yksi suomalainen käyttää keskimäärin noin 155 litraa vettä vuorokaudessa, tavoitetason ollessa noin 130 litraa. Energiankulutuksen kannalta lämpimän käyttöveden osuus on noin viidennes asuinrakennuksen kokonaiskulutuksesta. Veden käyttötottumukset ovat avainasemassa vedenkulutuksen vähentämiseen, mutta myös vesikalusteilla on vaikutus kulutukseen. (Motiva 2009.)

Innotek Oy on Lieksassa sijaitseva LVI-alan tuoteinnovaatioyritys, joka pyrkii toiminnallaan minimoimaan veden kulutusta asuntokohteisiin asennettavilla tuotteilla ja korjauksilla. Tehtävien kartoitus- ja asennustöiden raportointi on tähän mennessä tehty kokonaan paperilomakkeita käyttäen. Kyseinen raportointimalli on toimiva, mutta aikaa vievä ja resursseja hukkaava. Tehostaakseen toimintaansa Innotek Oy on siirtymässä perinteisestä raportointimallista täysin sähköiseen raportointiin. Tavoite on se, että paperilomakkeista päästäisiin kokonaan eroon.

Suurin ongelma Innotek Oy:n tämänhetkisessä tilanteessa on yhtenäisen ja toimivan, yrityksen sisäiseen käyttöön tarkoitetun sähköisen raportointi- ja viestintäjärjestelmän puute. Sähköiseen raportointimalliin siirtyminen on jo aloitettu Innotek Oy:llä vuonna 2004. Tuloksena tästä Innotek Oy:llä on tällä hetkellä Internetissä käytettävä Innonet-raportointi- ja toiminnanohjausjärjestelmä, yksi versio Innomobiili-tiedonkeruusovelluksesta sekä yhteinen tietokanta, joka kattaa molemmat järjestelmät. Edellä mainitut kokonaisuudet ovat luonteeltaan prototyyppisovelluksia eivätkä tarkkaan määriteltyjä ja suunniteltuja, yhtenäisiä ohjelmistokokonaisuuksia

Opinnäytetyöni tavoitteena on selvittää, suunnitella ja kuvata Energokartoituksen ja siihen liittyvän LVI-tuotteiden asennuksen toimintaprosessit sekä niihin liittyvä tietojenkäsittely ja laatia tarvittavien tietojärjestelmien toiminnallinen määrittely.

2 INNOTEK OY

Innotek Oy on lieksalainen, pääasiassa LVI-tuotteita valmistuttava sekä vesikalusteita kartoittava ja asennuksia tekevä yritys. Sen asiakkaat kostuvat pääasiassa huoltoyhtiöistä, isännöintiyrityksistä sekä yksityisistä asiakkaista, ympäri Suomen. Innotek Oy:n päätoimipiste sijaitsee Lieksassa. Muut toimipisteet ovat Helsingissä ja Kempeleellä. Innotek Oy:llä on muutamia tuotevarastoja, ja ne sijaitsevat Lieksassa, Keravalla ja Kempeleellä. Henkilöstö koostuu tällä hetkellä kymmenestä työntekijästä. Liikevaihto on noin 1,2 miljoonaa euroa.

Innotek Oy:llä on käytössä yrityksen sisäinen, sähköinen laskutusjärjestelmä Econet 2000. Koko yrityksen laskutus tehdään kyseisellä järjestelmällä. Päivittäiset työt, kuten raportointi, työtehtävien aikataulutus sekä asiakkaiden tiedotus hoidetaan tällä hetkellä pääasiassa Excel- ja Word-dokumentein. Kirjanpito on ulkoistettu.

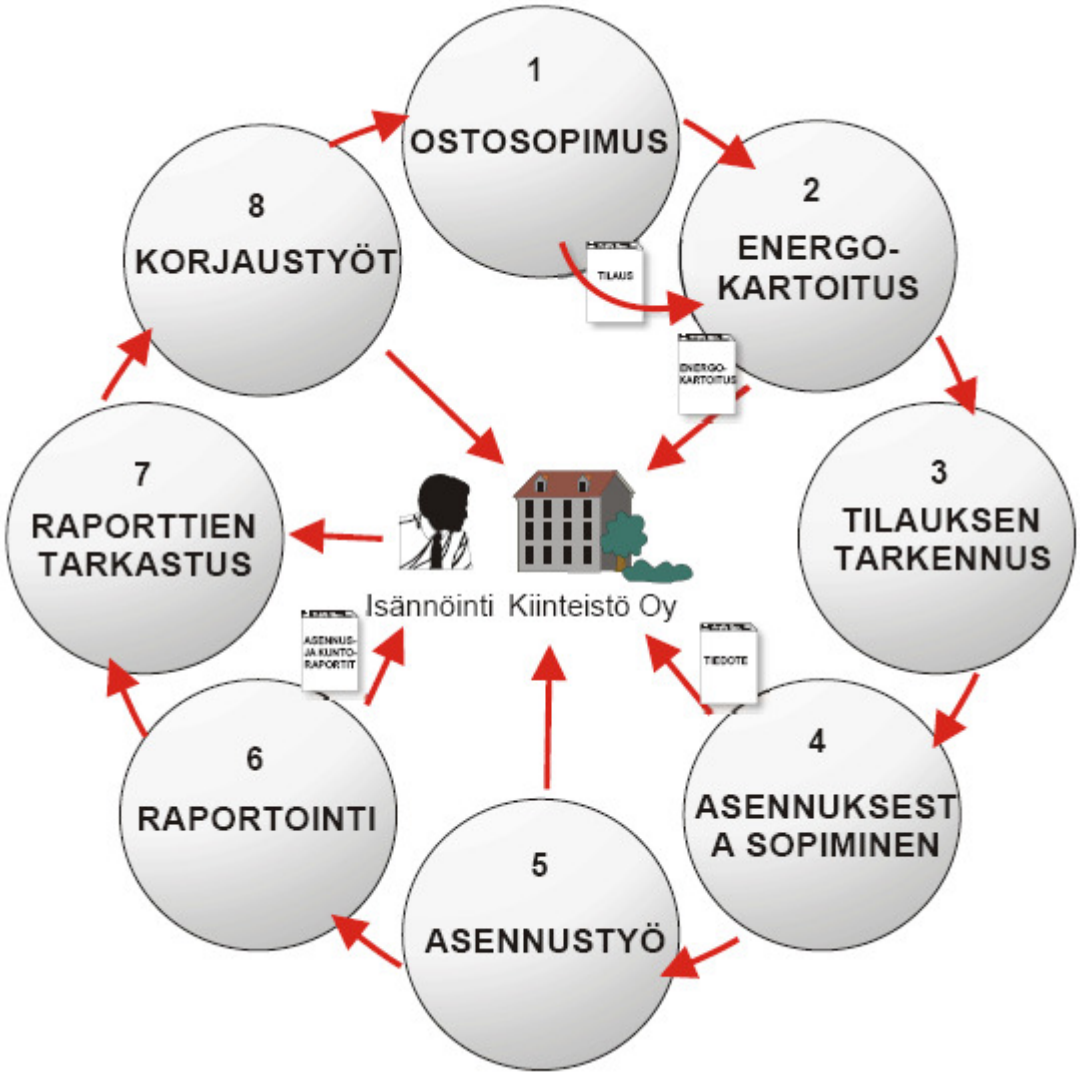
2.1 Energo-ohjelma

Innotek Oy:n toiminnan taustalla on toimintamalli nimeltä Energo-ohjelma, joka on Innotek Oy:n laatima ja kehittämä, veden ja energian kulutuksen minimointiin pyrkivä toimintamalli. Vedenkulutuksen pienentämiseen tarvitaan myös asukkaan omatoimisuutta, joten Innotek Oy tarjoaa asiakkailleen uusia teknisiä ratkaisuja ja tuotteita. Energo-ohjelmaan kuuluu palveluita sekä asennettavia tuotteita. Asuntokohteisiin tehtävistä Energo-kartoituksista tehdään isännöitsijälle Energo-kuntoraportti, josta selviää tiloissa ja kalusteissa mahdollisesti havaitut viat ja huomiot. Energo-kuntoraportin avulla isännöitsijä voi arvioida korjaustöiden tarpeellisuutta, kiireellisyyttä ja ajankohtaa. Energo-kuntoraportissa esille tulevat asiat:

- vesikalusteiden rikkonaisuus
- havaitut vuodot
- kosteiden tilojen pintojen rikkonaisuus
- asukkaiden esille tuomat viat
- selkeät haju- ja siisteyspoikkeamat

- mahdolliset muut viat, joilla on merkitystä isännöitsijälle tai kiinteistönomistajalle (Innotek 2000).

Energo-ohjelma koostuu kahdeksasta vaiheesta, jotka on esitetty kuvassa 2.1.

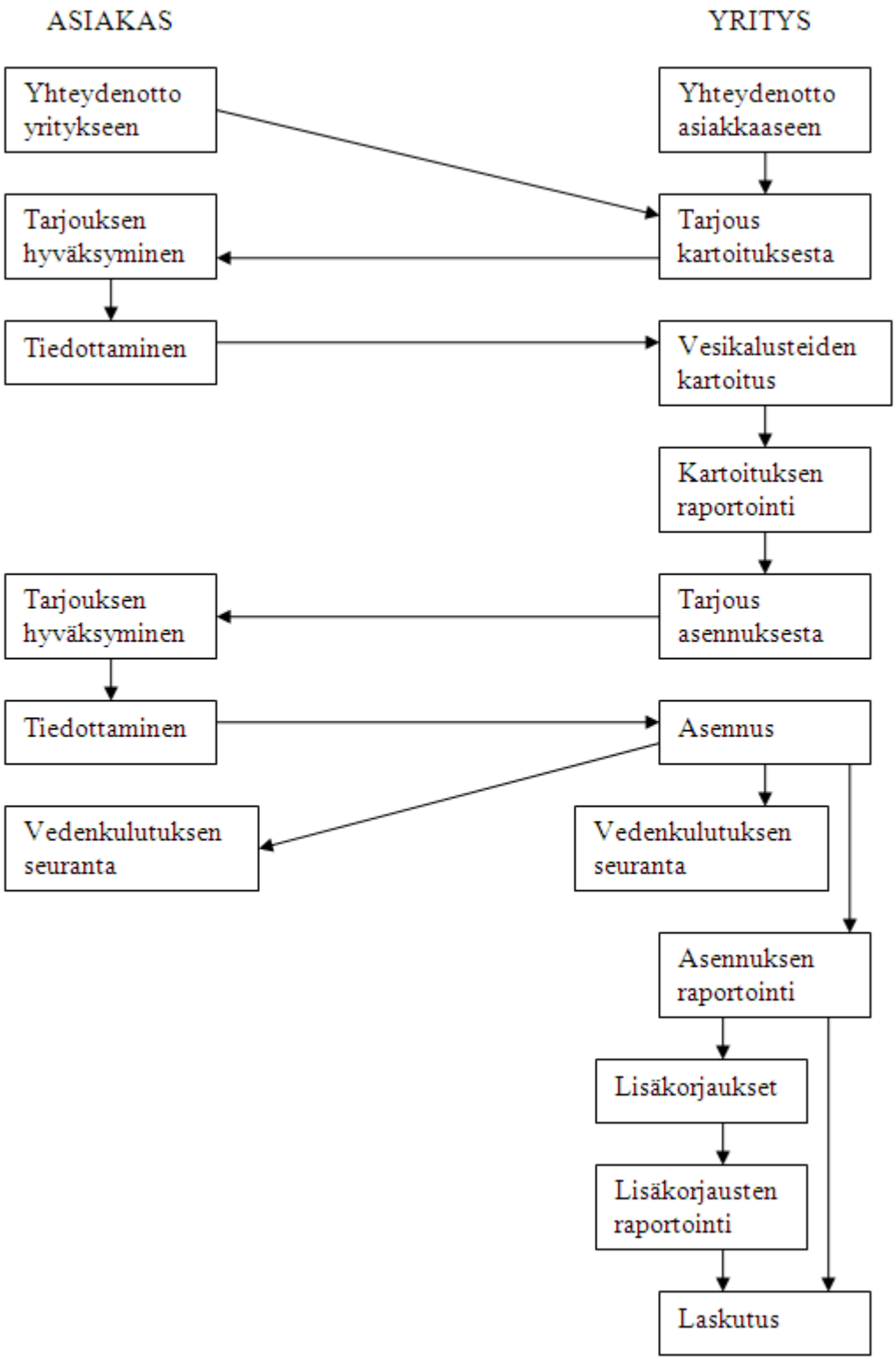


Kuva 2.1 Energo-ohjelman vaiheet (Innotek 2000)

1. Kiinteistön edustaja tekee Energo-tilauksen
2. Energo-asentajat suorittavat kiinteistössä
   - kalustokartoituksen
   - virtausmittaukset
   - huoltosuositukset
3. Energo-tilauksen laajuus tarkennetaan kartoituksen tulosten perusteella
4. Sovitaan asennusaika ja tiedotetaan kiinteistöyhtiössä

5. Energo-asentajat suorittavat asennustyön ja kuntoraportoinnin
6. Asennus- ja kuntoraportit toimitetaan laskun mukana isännöitsijälle
7. Isännöitsijä tarkastaa kuntoraportit ja tilaa tarvittaessa lisäkorjaustyöt
8. Suoritetaan mahdolliset lisäkorjaustyöt (Innotek 2000).

Energo-ohjelman ja samalla Innotek Oy:n toiminnan tarkempi kuvaus selviää seuraavasta prosessimallikuvasta (Kuva 2.2).



Kuva 2.2 Innotek OY:n prosessimalli

Tarjous

Yrityksen toiminta käynnistyy asiakkaan ottaessa yhteyttä yritykseen tai päinvastoin. Asiakkaalle tehdään tarjous vesikalusteiden kartoituksesta (Energokartoitus), jossa asentajat mittaavat kiinteistön vedenkulutuksen ja tutkivat, mitä yrityksen tuotteita kiinteistöön tulisi asentaa. Mikäli vesikalusteista löytyy vikoja, merkitään nekin raporttiin. Myyjä toimittaa asukastiedotepohjan kiinteistön isännöitsijälle, joka täydentää ne, hoitaa tiedottamisen kartoituksesta kiinteistön asukkaille ja toimittaa pohjan takaisin yritykselle.

Kartoitus

Kartoituksen valmistuttua asentaja toimittaa myyjälle laatimansa kartoitusraportin, minkä pohjalta tehdään kulutusanalyysi kiinteistöstä. Sen perusteella asiakkaalle tehdään tarjous suoritettavista asennustöistä. Kun asennettavat ja mahdolliset korjattavat tuotteet on määritelty ja aikataulu asennukselle sovittu, asiakas saa täytettäväkseen tiedotepohjat, joiden avulla hän huoltoliikkeen kautta hoitaa tiedottamisen kiinteistön asukkaille. Tiedotteet on toimitettava niin, että asukkaat saavat tiedon kaksi viikkoa ennen kartoitusta tai asennusta.

Asennus

Asentaessaan tuotteet asentaja tutkii samalla kiinteistössä ilmeneviä muita korjausta vaativia vikoja ja kirjaa kaikki havaintonsa asennusten lisäksi huoneistoittain asennusraporttiin. Asentajat palaavat mahdollisesti myöhemmin kiinteistöihin korjaamaan havaitsemistaan lisävioista ne, jotka kuuluvat Innotek Oy:n toimialaan. Asennuksen jälkeen myyjä ja asiakas ovat yhteydessä toisiinsa ja seuraavat kiinteistön vedenkulutuksen kehitystä. Asennusraportit toimitetaan laskun mukana asiakkaalle. Innotek Oy:lle jää kopio asennusraportista myöhempää tarkastelua varten.

Muut palvelut

Vesikalusteiden kartoituksen lisäksi Innotek Oy tarjoaa asiakkailleen Laaja kuntokartoitus -nimistä palvelua. Kyseinen palvelu laajentaa Energo-kartoitusta niin, että kartoitettavat kalusteet eivät rajoitu vain vesikalusteisiin, vaan asiakkaan vaatimuksesta tai toiveesta kartoitetaan ja arvioidaan melkein mitkä huoneen kalusteet tahansa tai koko huoneen kunto. Laajan kuntokartoituksen tiedot lisätään asiakkaille meneviin raportteihin.

Mikäli asennetuissa ja/tai korjatuissa tuotteissa ilmenee jotain vikaa, asiakas voi tehdä reklamaation. Reklamaatioita tehdessä asiakas ottaa yleensä yhteyttä Innotek Oy:n työntekijöistä asennussopimuksen tehneeseen myyjään tai työn tehneeseen asentajaan. Molemmissa tapauksissa reklamaatiosta tehdään uusi työnanto, jonka asentaja käy tekemässä heti kun mahdollista.

3 OHJELMISTOTUOTANNON MENETELMÄT

Ohjelmistotuotannolle ja sen menetelmille on olemassa rajattomasti erilaisia vaihtoehtoja. Yleispäteviä ratkaisuja ei ole ja tutullakin sovellusalueella saatetaan menetelmiä joutua muuttamaan projektikohtaisesti. Nykypäivänä on kuitenkin saatavilla standardoituja sekä muuten hyväksi havaittuja menetelmiä jokaiseen ohjelmistotuotannon vaiheeseen. Tässä luvussa käsitellään tällä hetkellä käytössä olevia ohjelmistotuotannon menetelmiä.

3.1 Projektityö

Ohjelmistotuotanto tapahtuu yleisimmin projektityöskentelynä. Projekti jaetaan usein peräkkäin ja/tai rinnakkain eteneviin osaprojekteihin. Yleinen jakotapa on jako esitutkimusprojektiin, määrittelyprojektiin, toteutusprojektiin ja käyttöönottoprojektiin. Esitutkimusvaihe tai määrittelyvaihe saattaa myös joskus johtaa hankkeen keskeyttämiseen. (Haikala & Märijärvi 2004, 53, 225.)

Jokaisen projektin takana on aina organisaatio. Projekti koostuu projektin jäsenistä ja projektipäälliköstä. Projektipäällikkö on kokonaisvastuussa projektin läpiviemisestä ja projektin onnistuminen tai epäonnistuminen on suurelta osin kiinni projektipäälliköstä. Projektipäällikkö vastaa projektista ohjausryhmälle ja toimii yhteyshenkilönä myös projektin muiden sidosryhmien välillä, esimerkiksi ohjaajien ja muiden ryhmien välillä (Jyväskylän yliopisto). Projektipäällikön tehtävät ovat karkeasti seuraavat:

- Laatii projektisuunnitelman tai johtaa sen laatimista.
- Tekee projektin aikataulutuksen.
- Käynnistää projektiryhmän työskentelyn ja ohjaa projektiryhmää.
- Johtaa projektin toimeenpanoa ja tehtävien antoa sekä valvoo työn edistymistä.
- Suorittaa projektin riskien ja muutosten hallintaa.
- Raportoi ohjausryhmälle.
- Osallistuu ohjausryhmän työskentelyyn.
- Hoitaa suhteet sidosryhmiin.
- Varustaa projektiryhmän tarvittavilla tiedoilla ja koulutuksella.
- Huolehtii projektin raportoinnista, dokumentoinnista ja arkistoinnista.
- Laatii projektin loppuraportin ja suorittaa projektin päättämisen. (Hahtola 2009.)

Projektiin voi liittyä tukiryhmä, johon kootaan teknisiä asiantuntijoita, jotka antavat ohjeita mm. teknisten ratkaisujen suunnittelussa. Ohjausryhmä koostuu sekä toimittajan, että asiakkaan edustajista. Ohjausryhmä seuraa projektin edistymistä, auttaa isojen ongelmien ratkaisussa ja hyväksyy projektisuunnitelmaan tehtävät muutokset. (Haikala & Märijärvi 2004, 228, 229.)

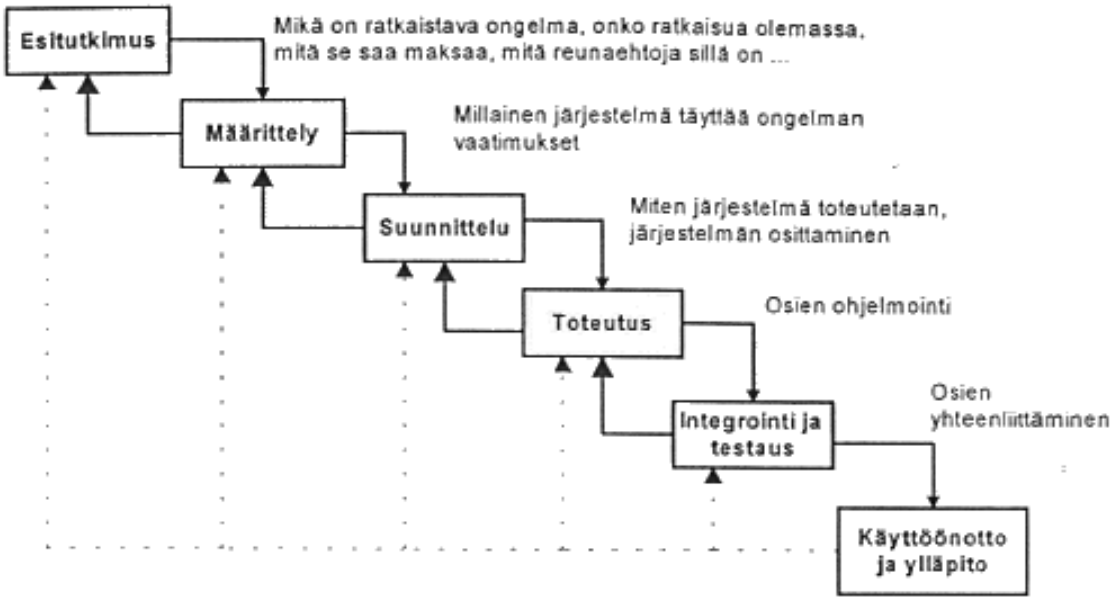
Projektisuunnittelussa projektin jaetaan päätyövaiheisiin. Kukin työvaihe paloitellaan tehtäviin ja tehtävien työmäärät arvioidaan. Tehtävät sijoitetaan kalenteriin ja jokaiselle tehtävälle valitaan sen suorittaja. Työmäärien pituuksille ei ole sääntöjä, mutta suositeltavaa on, että yhden tehtävän kesto on muutamasta viikosta muutamaan kuukauteen. (Haikala & Märijärvi 2004, 53 - 54.) Projektisuunnittelussa projektipäällikkö kirjoittaa projektisuunnitelman. Projektisuunnitelma on projektipäällikölle ohjenuora koko projektin läpiviemisestä, ja sitä voi-

daan käyttää selvittämään esimerkiksi projektin ohjausryhmälle projektin tilannetta ja etenemistä, aikatauluja ja työtehtäviä sekä riskejä. Koska projektin kulussa voi tapahtua yllättäviäkin muutoksia, on projektisuunnitelmaa päivitettävä, täsmennettävä ja pidettävä kokoajan ajan tasalla, projektipäällikön toimesta. Projektisuunnitelman pääotsikot ovat seuraavat:

- johdanto
- nykyinen järjestelmä
- hyödyt ja haitat
- projektin organisointi
- projektin tavoitteet ja päättyminen
- projektin ositus ja vaiheistus
- seuranta, ohjaus ja tiedottaminen
- standardit, direktiivit ja määräykset
- riskienhallintasuunnitelma
- koulutussuunnitelma
- asennussuunnitelma
- käyttöönottosuunnitelma
- kustannukset
- hylätyt ratkaisuvaihtoehdot ja jatkokehitysajatuksia
- projektin tehokkuus ja onnistuminen. (Ahtee 2009.)

3.2 Vaihejakomallit

Vaihejakomallilla tarkoitetaan tapaa, jolla ohjelmiston kehitystyö jaetaan vaiheisiin. Tavallisin vaihejakomalli on ns. vesiputousmalli (waterfall model, ks. kuva 3.2) (Haikala & Märijärvi 2004, 36 - 37). Vesiputousmalli on vaiheellinen ohjelmistotuotantoprosessi, jossa projektin työvaiheet etenevät aina yksi kerrallaan. Tarkoitus on se, että seuraavaan vaiheeseen ei edetä ennen kuin edellinen on kokonaan valmis. Vesiputousmalli-nimeä käytetään, koska vaiheiden kulku kuvataan yleensä alaspäin etenevänä, kuin vesiputous. Jokaisen vaiheen aikana tehdään laadunvarmistusta, kuten tarkastuksia ja testauksia ja niiden tarkoitus on kitkeä järjestelmän virheet mahdollisimman varhaisessa vaiheessa. Seuraavassa kerrotaan vesiputousmallin päävaiheista tarkemmin sekä dokumentit mitkä jokaisesta vaiheesta tuotetaan.



Kuva 3.1 Esimerkki vesiputousmallin vaiheista (Haikala & Märijärvi, 2004)

Esitutkimus

Esitutkimus on lyhyt alustava selvitys potentiaalisesta projektista. Esitutkimuksessa asetetaan projektissa syntyvälle järjestelmälle yleiset järjestelmätason vaatimukset. Esitutkimuksen pohjalta myös päätetään, lähdetäänkö projektia tekemään. Mikäli projekti toteutuu, niin esitutkimuksesta selviää, minkälaisesta ja minkä laajuisesta projektista suurin piirtein on kyse. Jälkeenpäin esitutkimuksen osia voidaan käyttää määrittelydokumenttien teossa. Esitutkimuksesta syntyy esitutkimus-dokumentti ja sen sisältö on yleensä seuraava:

- tuoteidea
- projektin organisointi
- havaitut ongelmat ja riskit
- tavoitteet ja vaatimukset
- rajaukset ja rajoitukset
- ympäristö ja liittymät
- projektin hyödyt
- projektin aikataulu
- toteutusvälineet
- kustannukset

- projektin kannattavuus
- lisätietoja. (Ahtee 2007.)

Määrittely

Määrittelyvaiheessa (vaatimusmäärittely) asiakasvaatimuksia analysoidaan ja niistä johdetaan toteutettavan järjestelmän ohjelmistovaatimukset (järjestelmävaatimukset, toiminnalliset vaatimukset). Määrittelyn tuloksena syntyy dokumentti nimeltä toiminnallinen määrittely.

Toiminnallinen määrittely on järjestelmän suunnittelun kannalta yksi tärkeimmistä dokumenteista. Siinä kuvataan kaikki järjestelmän toteuttavat toiminnot, tiedonkäsittely, käyttöliittymä sekä liitännät järjestelmän ulkopuolelle. Järjestelmän toiminta kuvataan aina käyttäjän näkökulmasta eli mitä sillä voidaan tehdä ja miten sekä toisaalta myös se miten järjestelmä ei saa toimia. Määrittelydokumentti ei ota kantaa järjestelmän toteutukseen. (Haikala 2009a.)

Määrittelydokumentti toimii myös apuvälineenä tilaajan ja toimittajan keskinäisessä kommunikaatiossa kehitettävästä sovelluksesta tai järjestelmästä. Sen perusteella todetaan, että projekti voidaan (tai ei voida) viedä läpi, toteuttaja tietää mitä tehdään, asiakas tietää mitä saa, epäselvät asiat ja riskit tulevat tunnetuksi sekä käsitteet ja termit täsmentyvät. (Haikala 2009a.)

Kaikkiin tilanteisiin soveltuvan dokumenttimallin tekeminen on mahdotonta ja siksi määrittelydokumentin runkoa on joskus sovellettava projektin mukaan. Esimerkki toiminnallisen määrittelyn sisältöluettelosta on kuvassa 3.2.

1. Johdanto
   1.1 Tarkoitus ja kattavuus
   1.2 Tuote
   1.3 Määritelmät, termit ja lyhenteet
   1.4 Viitteet
   1.5 Yleiskatsaus dokumenttiin
2. Yleiskuvaus
   2.1 Ympäristö
   2.2 Toiminta
   2.3 Käyttäjät
   2.4 Yleiset rajoitteet
   2.5 Oletukset ja riippuvuudet
3. Tiedot ja tietokannat
   3.1 Tietosisältö
      3.1.1 Käsite X (kukin omana alakohtana)
   3.2 Käyttöintensiteetti
   3.3 Kapasiteettivaatimukset
   3.4 Tiedostot ja asetustiedostot
4. Toiminnot
   4.1 Yleistä
   4.2 Järjestelmän toiminnot
5. Ulkoiset liittymät
   5.1 Laitteistoliittymät
   5.2 Ohjelmistoliittymät
   5.3 Tietoliikenneliittymät
6. Muut ominaisuudet
   6.1 Suorituskyky ja vasteajat
   6.2 Käytettävyys, toipuminen, turvallisuus ja suojaukset
   6.3 Ylläpidettävyys
   6.4 Siirrettävyys ja yhteensopivuus
   6.5 Käyttäjän ylläpitotoimet
7. Suunnittelurajoitteet
   7.1 Standardit ja suositukset
   7.2 Laitteistorajoitteet
   7.3 Ohjelmistorajoitteet
   7.4 Muut rajoitteet
8. Hylätyt ratkaisuvaihtoehdot
9. Jatkokehitysajatuksia
10. Vielä avoimet asiat

Kuva 3.2 Toiminnallisen määrittelyn sisältöluettelo (IEEE 830 Haikalan 2009a mukaan)

Suunnittelu

Suunnitteluvaiheessa suunnitellaan määritellyt toiminnot. Suunnitteluvaihe jaetaan usein kahteen tasoon, arkkitehtuurisuunnitteluun ja moduulisuunnitteluun. Arkkitehtuurisuunnittelussa järjestelmä jaetaan mahdollisimman itsenäisiin toisistaan riippumattomiin osiin, moduuleihin. Moduulisuunnitteluvaiheessa suunnitellaan moduulien sisäinen rakenne. (Haikala & Märijärvi 2004, 40.)

Arkkitehtuurisuunnittelussa syntyy dokumentti, jota sanotaan tekniseksi määrittelyksi. Toiminnallisen määrittelyn tapaan tekniselle määrittelylle ei ole yhtä ainoata mallia vaan tarpeen mukaan siihen tehdään lisälukuja. Esimerkki teknisen määrittelyn sisältöluettelosta on kuvassa 3.3.

1. Johdanto
   1.1 Tarkoitus ja kattavuus
   1.2 Tuote ja ympäristö
   1.3 Määritelmät, merkintätavat ja lyhenteet
   1.4 Liitteet
   1.5 Yleiskatsaus dokumenttiin
2. Järjestelmän kuvaus
   2.1 Sovellusalueen kuvaus
   2.2 Järjestelmän liittyminen ympäristöönsä
   2.3 Laitteistoympäristö
   2.4 Ohjelmistoympäristö
   2.5 Toteutuksen keskeiset reunaehdot
   2.6 Sopimukset ja standardit
3. Arkkitehtuurin kuvaus
   3.1 Suunnitteluperiaatteet
   3.2 Ohjelmistoarkkitehtuuri, moduulit ja prosessit
   3.3 Tietokanta-arkkitehtuuri
   3.4 Virhe- ja poikkeusmenettelyt
4. Moduuli-/luokka-/prosessi-kuvaukset
   4.1 Moduuli X (Kustakin moduulista oma alakohtansa (4.n)
      4.1.1 Yleiskuvaus
      4.1.2 Rajapinta yleisesti
      4.1.3 Rajapintafunktiot
      4.1.4 Moduulin toteutus
      4.1.5 Virhekäsittely
5. Valmisosat ja erityiset tekniset ratkaisut
6. Hylätyt ratkaisuvaihtoehdot
7. Jatkokehitysajatuksia
8. Vielä avoimet asiat

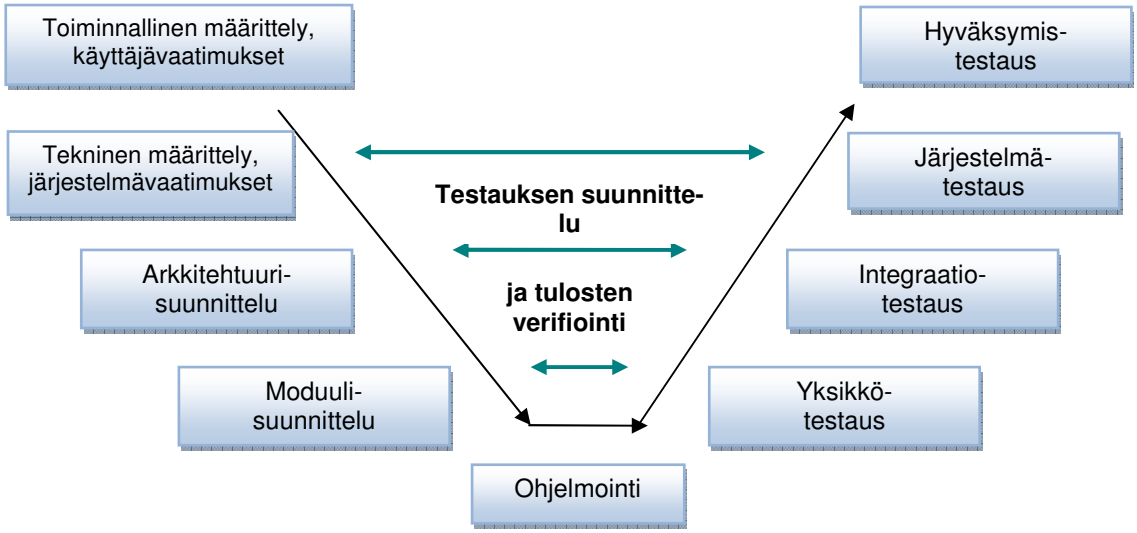
Kuva 3.3 Teknisen määrittelyn sisältöluettelo (IEEE 1016 Haikalan 2009b mukaan)

Toteutus

Toteutusvaihe pitää sisällään ohjelman kirjoitusvaiheen eli ohjelmoinnin. Ohjelma kirjoitetaan ensimmäiseen virheettömään käännökseen asti niin kuin se on määrittelydokumentteihin määritelty. Yleensä tarvitaan myös ohjelman valmistumisen jälkeistä muokkausta ja päivitystä, mutta siitä vastaa ylläpitovaihe.

Testaus (ja integrointi)

Testausvaiheen tarkoitus on löytää ohjelmasta virheitä. Yleensä testaus tapahtuu monella tasolla ns. V-mallin mukaisesti (ks. kuva 3.4). Mikäli testaus tehdään V-mallin mukaan, voidaan järjestelmän testaus ainakin osittain suunnitella jo määrittelyvaiheessa (Haikala & Märijärvi 2004, 40). Hieman vesiputousmallista poiketen olisikin hyvä, jos testausta suoritettaisiin koko projektin ajan. V-mallissa testaus jaetaan alla olevan kuvan 3.4 mukaisesti yksikkötestaukseen (moduulitestaus), integrointitestaukseen, järjestelmätestaukseen ja hyväksymistestaukseen.



Kuva 3.4 Esimerkki V-mallista (Sainio 2009, 31)

Yksikkötestauksessa etsitään vikoja yksittäisistä moduuleista, integraatiotestauksessa testataan moduulien yhteistoiminta ja järjestelmätestauksessa tutkitaan koko järjestelmän toiminta ja suorituskyky (Haikala & Märijärvi 2004, 40). Hyväksymistestauksen suorittaa asiakas ja siinä testataan, että järjestelmä vastaa sitä mitä on tilattu ja toimii niin kuin on sovittu.

Testauksesta voi syntyä dokumentteja varsin paljon. Testaussuunnitelmat tehdään yleensä järjestelmätestauksesta, integraatiotestauksesta ja jokaisesta integrointitestistä sekä yksikkötestauksesta ja jokaisesta yksikkötestistä. Pienehkössä projektissa järjestelmätestaussuunnitelma ja integraatiotestaussuunnitelma voidaan yhdistää. Vastaavasti jokaisesta suoritetusta testistä tehdään raportti. (Haikala & Märijärvi 2004, 299.)

Käyttöönotto ja ylläpito

Ohjelmiston valmistuessa suoritetaan sen käyttöönotto. Käyttöönoton ja ylläpidon organisointi ovat asiakasprojekteissa erittäin tärkeitä ja keskeisiä vaiheita. Käyttöönoton yhteydessä asiakkaalle voidaan toimittaa järjestelmän dokumentaatio. Laajemmissa järjestelmissä käyttöönotostakin tehdään suunnitelma. Kun ohjelmisto on saatu käyttöönotettua, siirrytään sen ylläpitoon.

Ylläpito voidaan karkeasti jakaa korjaavaan, adaptiiviseen ja täydentävään ylläpitoon. Korjaava ylläpito tarkoittaa virheiden korjaamista, adaptiivisessa ylläpidossa ohjelmaan tehdään muutoksia, ympäristön vaatimusten muuttuessa ja täydentävässä ylläpidossa parannellaan ohjelmaa ja lisätään mahdollisesti sen toiminnallisuutta. Ohjelmistotuotteiden tapauksessa yläpidolla ei yleensä ole varsinaista vaihetta, vaan ylläpito suoritetaan erillisessä projektissa. (Haikala & Märijärvi 2004, 41.)

Muita vaihejakomalleja

Vesiputousmallin lisäksi on olemassa muita käytössä olevia vaihejakomalleja. Yleisimpiä näistä ovat protoilumallit, inkrementaaliset mallit ja niin kutsutut ketterät menetelmät. Prototyyppilähestymistapa tarkoittaa tiettyä tuotteen ominaisuuden tai piirteen kokeilua ennen tuotteen varsinaista rakentamista. Valmistuneen prototyypin perusteella määritellään toteutettava järjestelmä ja toteutetaan alusta alkaen uudelleen tai vaihtoehtoisesti prototyyppi kehitetään valmiiksi järjestelmäksi. Myös välimuodot ovat mahdollisia. Protoilu on hyödyllistä esimerkiksi käyttöliittymien suunnittelussa, uusien teknisten ratkaisujen kokeilussa tai epäselvien asiakasvaatimusten etsimisessä. (Haikala & Märijärvi 2004, 42 - 43.)

Inkrementaalisilla kehitysmalleilla tarkoitetaan yleensä ohjelmistokehitystä, jossa lopputuotetta kehitetään pienehköinä inkrementteinä projektin sisällä. Jokainen inkrementtikierros tuottaa toimivan järjestelmän, mutta projektissa määritelty lopputulos saadaan aikaan useamman inkrementtikierroksen tuloksena. Inkrementaalisten mallien tapaisia kehitysmalleja ovat mm. Evo-malli ja spiraalimalli. Evo-mallin ideana on rakentaa ensimmäisessä projektissa ydinjärjestelmä, jota laajennetaan ja kehitetään edelleen seuraavissa projekteissa. Spiraalimalli puolestaan koostuu toistuvista kierroksista, joissa samat vaiheet toistuvat joka kierroksella. (Haikala & Märijärvi 2004, 45.)

Erityisesti pienehköihin ohjelmistoprojekteihin soveltuvat ketterät menetelmät ovat vastavoima byrokraattisille ja kankeille malleille ja menetelmille. Ketterille menetelmille on tunnusomaista hyvin lyhyet iteraatiot. XP-menetelmässä (Extreme Programming) työ aloitetaan ohjelmoimalla tehtävälle ohjelman osalle tes-

titapaukset. Ajettaessa testitapaukset todetaan, että ne raportoivat testien epäonnistuneen. Tämän jälkeen alkaa itse ohjelmakoodin kirjoittaminen, mikä jatkuu kunnes, kaikki testitapaukset saadaan virheettömästi ajettua läpi. (Haikala & Märijärvi 2004, 47.)

3.3 Toimintoanalyysi

Toimintoanalyysia käytetään yrityksen prosessien kartoitukseen. Sen tavoitteena on parantaa yrityksen kannattavuutta ja suorituskykyä. Toimintoanalyysi alkaa kohdealueen, toimintoyksiköiden ja toimintojen määrittämisellä. Näiden jälkeen toiminnot järkeistetään sekä jaotellaan ensisijaisiin ja toissijaisiin toimintoihin. Lopuksi kerätään toimintotiedot sekä viimeistellään ja dokumentoidaan toimintomääritykset. Toimintoanalyysin tuloksena saadaan yleensä hierarkkinen lista kehitettävän sovelluksen kannalta olennaisista toiminnoista. Toimintojen kulut voidaan kuvata jollakin kuvaustekniikalla, yleensä aluksi vuokaavioin. Määrittelyvaiheessa kyseiset toimintojen kulut tarkentuvat käyttötapauksiksi ja käyttöliittymäsuunnitelmiksi. (Gröhn 2003, 14, 103.)

3.4 Käsiteanalyysi

Käsiteanalyysi on jonkin rajatun kokonaisuuden jäsentämistä yhtenäisiin, tietojen ja toimintojen kannalta tärkeisiin käsitteisiin sekä niiden välisiin suhteisiin ja ominaisuuksiin. Toisin sanoen käsiteanalyysi on eräänlaista todellisuuden mallintamista. Ohjelmistotuotannossa käsiteanalyysiä tarvitaan yleisimmin kehitettävän järjestelmän tietosisällön mallintamiseen. Käsiteanalyysissä nimetään, yksilöidään, luokitellaan ja ryhmitellään kaikki kiinnostuksen kohteet ja niiden ominaisuudet sekä kuvataan ne yhtenäiseksi käsitteistöksi, käsitemalliksi.

Käsitemalli kuvaa kokonaisuuden käsitteet karkealla tasolla. Tavoite on, ettei mallissa ole turhaa toistoa. Käsitemallia voidaan täsmentää myöhemmissä vaiheissa eikä se ota kantaa tekniikoihin tai toteutukseen. Tästä syystä käsitemallista voidaan edetä moniin eri tietokantatuotteisiin tai tekniikoihin ja tavallista on, että käsitemallia käytetään järjestelmän tietokantasuunnittelun ensimmäisenä vaiheena. Käsitemallin avulla järjestelmän kehittävät pääsevät myös yhteisym-

märrykseen siitä, miten he näkevät mallinnettavan kohdealueen. Käsitemalli kuvataan yleensä käsitekaaviona (ER-kaavio), jollakin kuvaustekniikalla. (Satakunnan ammattikorkeakoulu.)

3.5 Kuvaustekniikat

Kuvaustekniikoita eli notaatioita käytetään, kun kehitettävän järjestelmän vaatimuksista laaditaan spesifikaatiot, jotka dokumentoidaan. Kuvaustekniikoita on useita aina luonnollisesta kielestä formaaleihin ja tarkkaan määriteltyihin notaatioihin. Kuvaustekniikoita voidaan käyttää dokumentoinnin lisäksi suunnittelutyön apuvälineenä. Kuvaustekniikoilla ei yleensä voida kuvata kuin osa tietystä spesifikaatiosta. Käytännössä spesifikaatiot kuvataan suurelta osin luonnollisella kielellä, ja eri notaatioilla tehdyt kuvaukset täsmentävät niitä. (Haikala & Märijärvi 2004, 67.)

3.5.1 UML-mallinnuskieli

UML, Unified Modeling Language, on Object Management Groupin standardoima ja hallinnoima graafinen mallinnuskieli olio-pohjaisten tietojärjestelmien ja -ohjelmien suunnittelua ja kehitystä varten, jolla voidaan visualisoida, rakentaa ja dokumentoida tietojärjestelmän artefaktit (osa-alueet ja toiminta). UML tarjoaa standardin tavan kuvata järjestelmän konseptitason asiat, kuten liiketoimintaprosessit ja järjestelmän yleistoiminnot. Se kuvaa myös konkreettiset asiat esimerkiksi ohjelmointikielikäskyt, tietokannan skeemat ja uudelleenkäytettävät ohjelmakomponentit. (Sparx Systems 2010)

Uusin virallinen UML versio on UML 2.0 ja se koostuu 13 kaaviotyypistä jaettuna kolmeen pääkokoelmaan: rakennekaaviot, käyttäytymiskaaviot ja interaktiokaaviot.

Rakennekaaviot kuvaavat järjestelmän staattinen arkkitehtuuri eli luokat, oliot, rajapinnat ja fyysiset komponentit sekä niiden väliset yhteydet (Taulukko 3.1).

Taulukko 3.1 UML-mallinnuskielen rakennekaaviot (Turun yliopisto 2007)

Kaavion nimi	**Tarkoitus**
Ryhmittelykaavio	Jakaa järjestelmän mallin paketteihin ja kuvaa niiden välisen vuorovaikutuksen korkealla tasolla.
Luokkakaavio	Määrittää järjestelmän mallin perusrakenteen eli tyypit, luokat ja päämateriaalit, joista malli rakentuu.
Oliokaavio	Kuvaa luokkien instanssien käyttöä ja käyttäytymistä ajon aikana.
Koostumuskaavio	Esittävät elementtien ajonaikaisen luomisen sekä niiden rakenteet ja yhteydet.
Komponenttikaavio	Kuvaa koko järjestelmän rakennetta korkeammalla tasolla, esittäen osat joista järjestelmä rakentuu.
Sijoittelukaavio	Kuvaa järjestelmän fyysisiä piirteitä, kuten laitteisto, ohjelmisto sekä yhteydet eri laitteiden välillä.

Käyttäytymiskaaviot ovat tarkoitettu kuvaamaan mallien sisäiset interaktiot sekä kuinka järjestelmä toimii käytännössä mukaan lukien toimintojen vaikutukset ja seuraukset (Taulukko 3.2).

Taulukko 3.2 UML-mallinnuskielen käyttäytymiskaaviot (Turun yliopisto 2007)

Kaavion nimi	**Tarkoitus**
Käyttötapauskaavio	Kuvaa järjestelmän käyttötapauksien väliset suhteet sekä käyttötapauksiin osallistuvat järjestelmän ulkoiset toimijat eli aktorit (actor).
Aktiviteettikaavio	Kuvaa tietyn toiminnon etenemistä eli miten luokat ja oliot toimivat yhdessä tietyn toiminnon tuottamiseksi.
Tilakaavio	Kuvaa tietyn toiminnon tiloina ja niiden siirtyminä.

Interaktiokaaviot ovat johdettu käyttäytymiskaavioista, ja ne kuvaavat järjestelmän ulkoisen käyttäytymisen eli toiminnan järjestelmän käyttäjän näkökulmasta (Taulukko 3.3).

Taulukko 3.3 UML-mallinnuskielen interaktiokaaviot (Turun yliopisto 2007)

Kaavion nimi	**Tarkoitus**
Interaktiokaavio	Kuvaa järjestelmän tai sovelluksen rakenteen ja vuorovaikutuksen yksityiskohtaisesti.
Sekvenssikaavio	Kuvaa olioiden välisen vuorovaikutuksen jonkin toiminnon aikaansaamiseksi.
Ajoituskaavio	Käytetään silloin, kun tarkoituksena on kuvata vuorovaikutuksia aikasidonnaisesti.
Kommunikaatiokaavio (yhteistyökaavio UML 1:ssä)	Soveltuu sellaisen dynaamisen tilanteen kuvaamiseen, jossa olioiden välinen suhde on tärkeä.

Kaikki UML-kaaviot koostuvat elementeistä (element) ja niiden välisistä suhteista (relation). Monilla elementeillä voi myös olla oma sisäinen rakenne, esimerkiksi luokka koostuu attribuuteista (attribute) ja operaatioista (operation). Elementtien välisiin suhteisiin voidaan liittää lisätietoa suhteesta esimerkiksi suhteen kertautuminen ja rooli. Syntaktiset säännöt määräävät, mitkä suhteet ovat sallittuja elementtien välillä ja elementtejä voidaan tarkentaa kielen laajennusmekanismeilla. Myös omien elementtien määritys on mahdollista. (Turun yliopisto 2007)

Yleisiä elementtejä ovat mm.

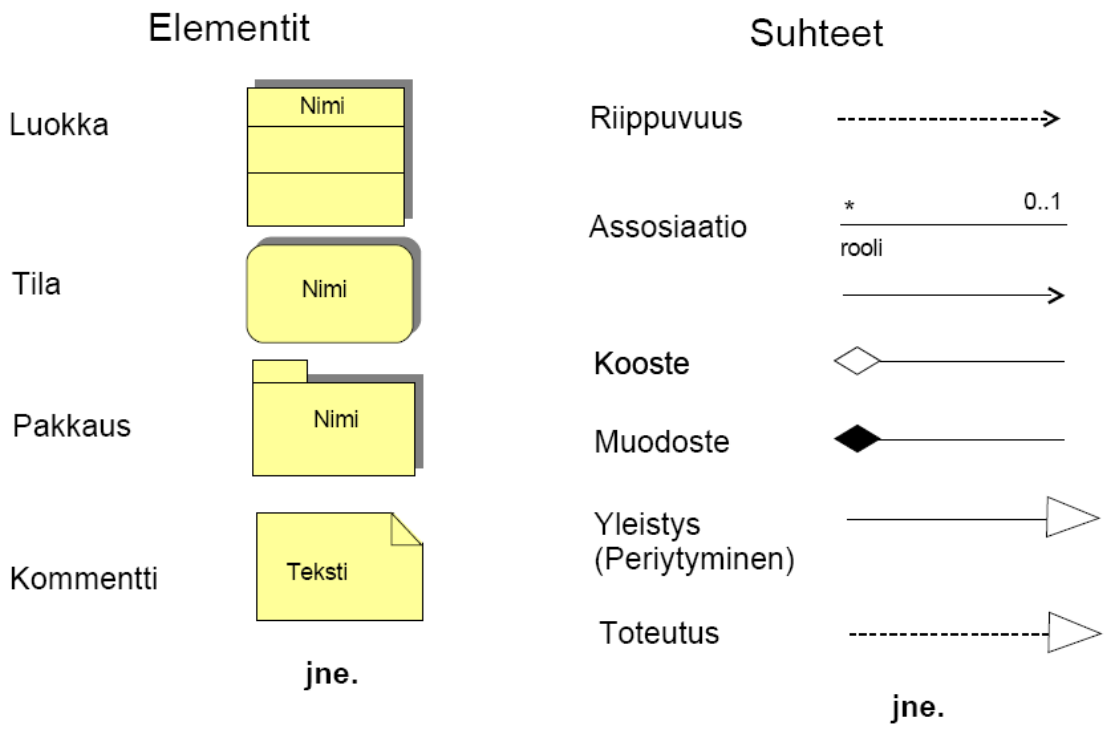
- luokka (class) ja olio (object)
- tila (state)
- pakkaus (package) ja
- kommentti (comment).

Yleisiä suhteita ovat mm.

- riippuvuus (dependency)

- assosiaatio (association)
- kooste (aggregation)
- muodoste (composition)
- yleistys (generalization) ja
- toteutus (realization).

Elementit ja suhteet kuvataan yksikäsitteisin graafisina symboleina, mutta symbolien graafinen ulkoasu on vain löyhästi määritelty. Eri työkaluvalmistajilla onkin omia tyylejä symbolien esittämiseen. Kuvassa 3.5 on esimerkki UML-kaavioiden elementtien ja suhteiden symboleista. (Turun yliopisto 2007)



Kuva 3.5 Esimerkkejä UML-kaavioiden elementtien symboleista (Turun yliopisto 2007)

3.6 Laadunhallinta

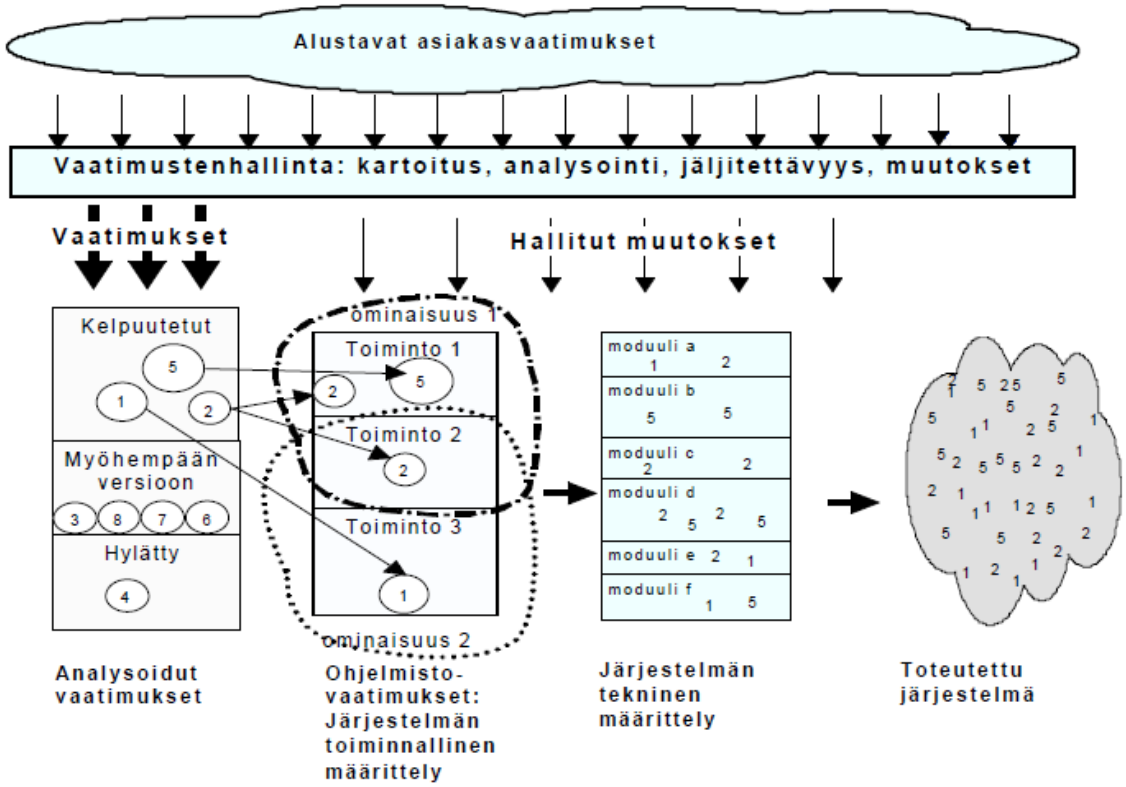
Laatu on subjektiivinen, käyttäjästä ja käyttöympäristöstä riippuva käsite. Termillä ”laatu” ei ohjelmistotuotannossa välttämättä tarkoiteta erityisesti hyvää tai huippulaatua vaan erilaisia tuotteen ja toiminnan mittaavia ominaisuuksia. Mo-

dernin ohjelmistotuotantoajattelun mukaan tuotteen laatuun vaikutetaan parhaiten toiminnan laadun kautta. (Haikala & Märijärvi 2004, 48.)

Yrityksen tuotteen tekemisessä käytettävää toimintatapaa kutsutaan laatujärjestelmäksi (quality management system, suomennetaan usein myös laadunhallintajärjestelmä). Laatujärjestelmää kuvaa yleensä yrityksen laatukäsikirja sekä siihen liittyvät muut ohjeistukset ja sen tavoite on taata, että tuotantoprosessi tuottaa suunniteltua laatutasoa olevia tuotteita aikataulun ja budjetin mukaisesti. Ulkopuolisen tahon on tarvittaessa pystyttävä varmistamaan, että yritys todellakin toimii laatujärjestelmän mukaisesti. Yritys voi myös hakea laatujärjestelmälleen sertifiointia eli todistetta siitä, että laatujärjestelmä on standardin mukainen ja yritys toimii laatujärjestelmän mukaisesti. Ohjelmistotuotannossakin tärkein ISO 9001 -standardi määrittelee tietyt perusasiat, jotka yrityksen laatujärjestelmän tulee sisältää yrityksen toimialasta riippumatta. (Haikala & Märijärvi 2004, 50.)

Laadunhallinta koostuu laadunohjauksesta, laadunvarmistuksesta, laadunsuunnittelusta ja laadun parantamisesta. Standardin määrittelyn mukaan laadunohjaus on niiden tekniikkojen ja toimintojen joukkoa, joita käyttämällä pyritään täyttämään laatuvaatimukset. Laadunvarmistus pyrkii saavuttamaan sekä organisaation sisäisten että ulkopuolisen asiakkaiden ja viranomaisten luottamuksen tavoitteiden toteutumisen suhteen (Haikala & Märijärvi 2004, 197). Laadunsuunnittelu ja parantaminen tähtäävät kehittämään ja muuttamaan yrityksen toimintaa tavalla, joka vaikuttaa positiivisesti yrityksen tuotteiden ominaisuuksiin ja projekteihin. (Haikala & Märijärvi 2004, 209)

Ohjelmistotuotannossa laadunhallinnalla pyritään asiakastyytyväisyyteen sekä kehitettävän tuotteen virheettömyyteen. Asiakastyytyväisyys saavutetaan täyttämällä asiakkaan vaatimukset ja tuotteen virheettömyys saavutetaan testaamalla tuotetta. Kuvassa 3.6 on esimerkki vaatimustenhallinnasta ohjelmistoprojektissa.



Kuva 3.6 Vaatimustenhallinta ohjelmistoprojektissa (Haikala & Märijärvi Ilkon mukaan)

Yleensä ohjelmistoprojekteissa vaatimustenhallinta on koko projektin ajan suoritettava tukitoiminto, jonka keskeisimmät osa-alueet ovat vaatimusten kartoittaminen ja analysointi, vaatimusten jäljitettävyys sekä vaatimusmuutosten hallinta. Vaatimusten kartoittamisen ja analysoinnin perusteella tehdään vaatimusdokumentti josta selviää vaatimusten perustelut, prioriteetit, liittymät muihin vaatimuksiin sekä mistä vaatimukset ovat peräisin. Koska vaatimusten toteutuminen on oltava todennettavissa, täytyy vaatimukset voida jäljittää asiakasvaatimuksista toteutukseen ja vastaavasti koodimoduleista takaisin asiakasvaatimuksiin. Vaatimusmuutosten hallinnan tarkoitus on vastata siihen miten vaatimusmuutokset hyväksytään ja miten niiden kerrannaisvaikutukset (jäljitettävyys) hoidetaan (Haikala & Märijärvi Ilkon mukaan). Tuotteen testaus jaetaan staattiseen ja dynaamiseen testaukseen. Staattinen testaus koostuu pääasiassa projektin dokumentaation sekä tuotteen ohjelmakoodin ja muun informaation tarkastamisesta ja katselmoinnista suorittamatta itse ohjelmaa. Dynaaminen testaus on testausta, joka suoritetaan ajamalla ohjelmakoodia ja tarkastelemalla sen suorituksenaikaista käyttäytymistä.

4 MOBIILILAITTEET JA -SOVELLUKSET

Mikroprosessorit kehittyvät koko ajan kovaa vauhtia. Mooren lain mukaan transistorien määrä mikropiirissä kaksinkertaistuu noin joka toinen vuosi (Intel Corporation). Laskentateho mikroprosessoreissa siis kasvaa entisestään, kun taas fyysinen koko pienenee. Tämä taas mahdollistaa yhä tehokkaamman tietotekniikan sijoittamista yhä pienempiin laitteisiin. Eräs suosittu tietotekniikan kohde nykypäivänä ovat mobiililaitteet.

Langattomien tietoverkkojen kehittyminen ja sen tarjoamat mahdollisuudet ovat luoneet monipuolisen pohjan mobiililaitteille ja mobiilisuudelle yleensä. Monet mieltävät mobiililaitteen taskuun mahtuvaksi, jonkin tasoista tietotekniikkaa hyödyntäväksi laitteeksi, esimerkiksi älypuhelimeksi, mutta teknisestä näkökulmasta asia ei ole näin yksinkertainen. Sellaista yksiselitteistä ominaisuus- tai piirrejoukkoa ei ole olemassa, joka oikeuttaisi kutsumaan tiettyä laitetta tai järjstelmää mobiiliksi tai vastaavasti ei-mobiiliksi. Syynä tähän voidaan pitää mobiilien järjestelmien nuorta ikää sekä sitä, että järjestelmien ns. esi-isät ovat käsitteellisesti kaukana toisistaan, kuten rannekello ja kannettava tietokone. Tässä työssä mobiililaitteilla tarkoitetaan kuitenkin mitä tahansa tietojärjestelmän sisältäviä laitteita, joita niiden käyttäjä voi kuljettaa mukanaan. (Mikkonen 2004, 1.)

4.1 Mobiililaitteiden perusominaisuuksia

Mobiililaitteen ehkä tärkein ominaisuus sen käyttäjälle on usein laitteen tarpeeksi pieni koko, jotta sitä on helppo kuljettaa mukanaan. Mobiililaitteiden pieni koko asettaa tekniikoille omat rajoituksensa, joten muita ominaispiirteitä ovat yleensä rajoitettu määrä muistia sekä suhteellisen heikko prosessointiteho. Laitteiden käytön kannalta mahdollisimman vähäinen energiankulutus luetaan myös ominaispiirteeksi. Nykyaikaisille mobiilijärjestelmille on ominaista laajennettavuus ja käyttäjän tarpeisiin mukautuminen. On hyvin tavallista, että mobiilijärjestelmille on saatavilla laaja kirjo erilaisia lisäosia aina fyysisistä lisälaitteista ohjelmistomoduuleihin. Näin järjestelmät palvelevat jokaista käyttäjää aiempaa yksilöllisemmin. Laajennukset tosin riippuvat mobiililaitteen perusominaisuuksis-

ta (Mikkonen 2004, 2-5). Seuraavassa muutama esimerkkikuva mobiililaitteissa (Kuvat 4.1 - 4.3).

Kuva 4.1 Kannettava tietokone sekä älypuhelin (Lehtiniitty 2008a)

Kuva 4.2 Digitaalikamera (Söderholm)

Kuva 4.3 Rannekellopuhelin (Lehtiniitty 2008b)

Kuvissa 4.1 ja 4.2 olevat esimerkkilaitteet ovat hyvin suosittuja nykyajan mobiililaitteita. Kuvan 4.3 rannekellopuhelin on puolestaan hieman harvinaisempi. Kyseinen digitaalisen kellon ja puhelimen ominaisuuksia yhdistävä laite on hyvä esimerkki tekniikan kehityksestä ja mobiililaitteiden monimuotoisuudesta.

4.2 Mobiilisovellukset ja -ohjelmistot

Ohjelmistot ovat mobiililaitteissa nykyään tärkeässä roolissa ja niiden merkitys on korostunut viimeisen 10 - 15 vuoden aikana. Voidaan jopa ajatella, että mobiililaitteet menettäisivät suuren osan houkuttelevuudestaan ilman nykyaikaisia ohjelmistoja (Mikkonen 2004, 1). Laitteet mahdollistavat myös valmiudet monipuolisten toimintojen suorittamiseksi. Johtavat käyttöjärjestelmävalmistajat ovat tehneet mobiililaitteille omat versiot käyttöjärjestelmistään, ja näin ollen esimerkiksi älypuhelimien käyttö muistuttaa nykyään tietokoneen käyttöä, pienemmässä koossa. Myös nykyaikaiset mobiilisovellukset ovat lähellä PC-sovelluksia ja saatavilla on ohjelmia tekstinkäsittelystä peleihin.

Varsinkin mobiilien puhelinlaitteiden käyttömahdollisuudet ovat laajentuneet huimasti niiden suosion myötä. Useimmissa nykyaikaisissa älypuhelimissa ja

Pocket PC -laitteissa on digikamerakuvien ja videoiden ottamiseen ja tallentamiseen. Digitaalinen musiikki kulkee mukana MP3-muodossa ja radio on melkeinpä itsestäänselvyys puhelinlaiteissa. Sisäänrakennettu GPS-paikannin on oletusominaisuus ja jos sellaista ei valmiiksi laitteesta löydy, on sellainen useissa tapauksissa saatavana erillisenä lisäosana. Etenkin langattomien tietoverkkoyhteyksien kehitys mahdollistaa muun muassa Internet-yhteyden muodostamisen mobiililaitteilla. Nykyaikaisissa mobiililaitteissa ovat täydet Internet-mahdollisuudet oletuksena ja erilaisiin sosiaalisiin verkostoihin liittyminen niin puhelin- kuin muillakin mobiililaitteilla on tehty todella helpoksi. Kannettavien tietokoneiden tavoin älypuhelimilla ja Pocket PC:illä työt ja dokumentit kulkevat aina tarvittaessa mukana ja työn teko onnistuu oikeastaan missä tahansa.

4.3 Mobiiliohjelmointi

Mobiililaitteiden nykyaikaisten käyttömahdollisuuksien johdosta mobiiliohjelmointi on noussut tärkeäksi ohjelmistotekniikan osa-alueeksi (Mikkonen 2004, 1). Vaikka mobiiliohjelmointi, varsinkin nykytekniikoilla ei ulkoisesti näytä kovinkaan paljon eroavan perinteisestä työasemaohjelmiston sovelluskehityksestä, on siinä kuitenkin omat erot ja rajoitteet. Vähäinen muistimäärä ja heikko prosessointiteho aiheuttavat ristiriidan perinteisten suunnitteluperiaatteiden kanssa, joita ovat lähinnä uudelleenkäytettävyys ja joustavuus. Nämä rajoitteet aiheuttavat myös sen, että ohjelmoijalla on käytettävissään vain hyvin rajallinen määrä palveluita ja resursseja. Mobiiliohjelmoinnissa on usein käytettävissä ainoastaan laite- tai alustatoimittajan omat rajapinnat ja kirjastokomponentit, kun taas työasemaympäristössä on usein mahdollista käyttää kenen tahansa kehittämiä ohjelmakoodikirjastoja. Koska useissa tapauksissa mobiililaitteelle on saatavissa paljon erilaisia lisälaitteita, on ohjelmoijan tehtävä tietyissä tapauksissa myös paljon ylimääräistä ohjelmakoodin optimointia tiettyä lisälaitetta varten. (Mikkonen 2004, 5 - 10.)

Ohjelmistopuolella mobiiliohjelmoinnin erot työasemaohjelmiston kehitykseen liittyvät ohjelmistojen luotettavuuteen ja käytettävyyteen, muistinvapautukseen, käynnistämiseen sekä energiankulutukseen. Siinä missä työasemien käyttäjät ovat tottuneet hyväksymään tietyn hitauden ja epäluotettavuuden järjestelmää

käytettäessä, odotetaan mobiililaitteen, esimerkiksi puhelimen toimivan nopeasti ja virheettömästi. Koska mobiililaite sammutetaan harvoin, ei resurssien automaattinen vapauttaminen tapahdu läheskään niin usein kuin esimerkiksi PC-laitteilla. Vähäisen muistin määrän takia muistin käytön suhteen täytyy olla erittäin tarkkana. Mobiililaite täytyy myös saada (lähes) virrattomasta tilasta käyttökuntoon todella nopeasti. Tähän voi olla jopa lainsäädännöllisiä rajoituksia, esimerkiksi miten pian käynnistämisen jälkeen täytyy voida soittaa hätäpuhelu. Mobiililaitteiden tarkoitus saattaa olla joidenkin asioiden mahdollistaminen joka paikassa ja tilanteessa ja tällöin lyhytkin odotusaika voi tuntua liian pitkältä. Energiankulutus on myös yksi ohjelmistosuunnittelijan huomioitavista asioista. Tämä tarkoittaa ohjelmoijan kannalta monimutkaisia optimointitehtäviä, joissa energiankulutuksen ja suorituskyvyn välistä suhdetta hienosäädetään. Joissain tapauksissa energiansäästöongelmat rajoittavat kehitettävää mobiilisovellusta niin, että jotain tiettyä ominaisuutta ei ole järkevää toteuttaa, sillä se estäisi energiansäästöominaisuuksien toiminnan totaalisesti. (Mikkonen 2004, 5 - 10.)

Mobiiliohjelmat ovat nykyisten Windows-sovellusten tapaan reaktiivisia. Toisin sanoen ohjelmien toiminta perustuu ympäristöstä saataviin ärsykkeisiin ja niihin reagointiin. Ärsykkeitä voivat olla esimerkiksi käyttäjän aikaansaama näppäimen painallus tai matkapuhelinverkon lähettämä viesti. Tällaisessa ympäristössä käytettävät ohjelmat vaativat tapahtumapohjaista ohjelmointia (event based programming). Kyseisen ohjelmointimallin perusidea on se, että järjestelmä pääasiassa odottaa syötteitä ympäristöltä. Herätteen saatuaan se suorittaa kyseiseen syötteeseen liitetyn toiminnon. Syötteitä kutsutaan tapahtumiksi (event) ja toimintoja kutsutaan tapahtumakäsittelijöiksi (event handler). Jokainen tapahtumakäsittelijä voi saada aikaan lisää tapahtumia, luoden monimutkaisempia toimintoja. (Mikkonen 2004, 9 - 10.)

Ohjelmoinnin näkökulmasta tapahtumakäsittelijät ovat yleensä aliohjelmia ja tapahtumat aliohjelmakutsuja. Joskus on kuitenkin tarpeen käyttää monimutkaisempia rakenteita, useiden rinnakkaisten herätteiden palvelemiseksi. Tapahtumakäsittelijöiden rakenteiden yksinkertaistamiseksi mobiiliohjelmointiympäristöt sisältävät usein perusversion sovelluksen pääohjelmasta sekä oletustapahtumakäsittelijän, perusympäristön tapahtumille. Ohjelmoijan tehtäväksi jää

mahdollisten sovelluskohtaisten tapahtumakäsittelijöiden lisääminen. Toteutus voi perustua joko käyttöjärjestelmän tarjoamiin tapahtumanvastaanottopalveluihin tai tiettyyn sovelluskehykseen, johon ohjelmoija esimerkiksi ylimäärittelemällä lisää omat rutiinit tarvittavien toimintojen aikaansaamiseksi. (Mikkonen 2004, 9 - 10)

Mobiiliohjelmistot kehitetään usein mobiiliohjelmistoalustojen avulla. Mobiilialustojen kehittäjiä on useita ja alustat poikkeavat toisistaan tietyin teknisin ominaisuuksin. Yleisimpiin mobiilialustoihin kuuluvat tällä hetkellä Java, Symbian OS, Palm OS, Microsoft smartphone (Mobile), Brew, Embedded Linux sekä Android. Mobiiliohjelmointiin käytetyt ohjelmointikielet ovat alustasta riippuen C, C++, C# ja Java.

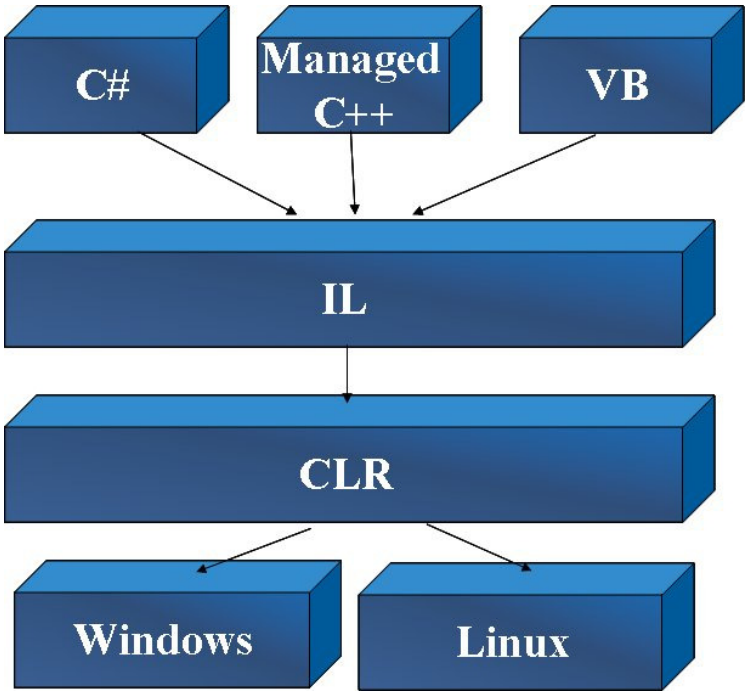
5 TYÖSSÄ KÄYTETYT TEKNIIKAT

Mobiilisovelluskehitykseen on saatavilla nykyään monenlaisia tekniikoita ja kehitysalustoja moneen tarpeeseen. Windows-alustan mobiililaitteet ovat täysin tuettuja uusimmissa Visual Studioissa ja mm. Microsoftin WWW-sivuilta on kaikki kehitykseen tarvittavat lisäosat saatavilla ilmaiseksi. Tilanteesta riippuen mobiilisovelluksia voidaan kehittää niin moderneilla ohjelmointikielillä kuin hieman vanhemmillakin.

5.1 Visual Studio 2008 Professional ja .NET-konsepti

Visual Studio 2008 on Microsoftin kehittämä ja ylläpitämä ohjelmistojen kehitysympäristö. Pääasiassa Microsoftin luoman, .NET-konseptin sovelluskehitykseen tarkoitettu Visual Studio 2008 pitää sisällään kaikki työkalut ja toiminnot, mitä tarvitaan ohjelmakoodin kirjoittamisesta valmiin ohjelman testaamiseen ja käyttöönottoon. Tärkeimmät Visual Studion tukemat ohjelmointikielet ovat Visual C#, Visual C++ ja Visual Basic, mutta myös esimerkiksi natiivien C- ja C++-sovellusten rakentaminen on tuettu.

.NET-konseptin yksi tärkeimmistä ominaisuuksista on kaikkien .NET-perheen ohjelmointikielien yhtenäisyys. Kuvassa 5.1 on esitetty .NET-konseptin pääpiirteinen toiminta.



Kuva 5.1 .NET-konseptin toiminta (Hariharan 2007)

Sovelluksia kirjoitettaessa kunkin kielinen sovellus, Visual C#, Visual C++ (Managed C++) tai Visual Basic, kirjoitetaan omalla hieman toistaan eroavalla syntaksilla. Kun kukin ohjelma käännetään ajettavaksi ohjelmaksi, niistä kaikista tehdään MSIL-välikieltä (Microsoft Intermediate Language, kutsutaan nykyisin myös nimellä CIL, Common Intermediate Language tai IL, Intermediate Language). Välikoodi käännetään ensimmäisellä ajokerralla suoritettavaksi ohjelmaksi, joka ajetaan CLR-virtuaalikoneessa (Common Language Runtime) (Mikkonen 2004, 14). Toisin sanoen on täysin mahdollista tehdä dynaaminen linkkikirjasto esimerkiksi Visual Basic -kielellä ja ottaa käyttöön Visual C++ -projektissa.

Mobiilisovellusten tekeminen Windows Mobile -alustalle on myös täysin tuettu Visual Studio 2008 Professionalissa. Oletuksena Visual Studio 2008 Professionalin mukana tulee mobiiliohjelmointiin tarkoitettu .NET Compact Framework sekä versiot mobiilisovellusten ajoon tarkoitetuista emulaattoreista. Microsoftin

verkkosivuilla on myös saatavilla Windows Mobile SDK:n uusimmat versiot ilmaiseksi.

5.2 C#

C# on moderni, yksinkertainen ja vahvasti tyypitetty olio-ohjelmointikieli. Alun perin Anders Hejlsbergin kehittämän C#:n juuret ovat C-kielessä, ja se on syntaksiltaan hyvin samanlainen, kuin edelleen suositut C-, C++- ja Java-ohjelmointikielet. .NET-kielenä C# tarjoaa monia ohjelmointityötä helpottavia ominaisuuksia, kuten muun muassa automaattisen muistin hallinnan, rakenteellisen poikkeusten käsittelyn ja tyyppiturvallisen muuttujien käsittelyn. Tarkkaan suunniteltu versiointi mahdollistaa sen, että vanhat C#-sovellukset toimivat myös uusilla, päivittyvillä luokkakirjastoilla. (Microsoft 2007)

Rakenteeltaan C# muistuttaa C++- ja Java-ohjelmointikieliä. Vahva tyypitys tarkoittaa sitä, että jokaisella muuttujalla on tyyppi ja se voi saada vain sille tyypille ominaisia arvoja. Muuttujien käsittely on myös tyyppiturvattu, joten alustamattomien muuttujien tietoa on mahdotonta lukea, taulukkomuuttujat on suojattu niin, ettei niitä voida indeksoida rajojen yli ja tarkistamattomia muuttujien tyyppimuunnoksia ei voida suorittaa (Microsoft 2007). Olio-ohjelmointikielenä C#-ohjelmat koostuvat olioista ja olioiden vuorovaikutuksesta keskenään, saaden aikaan haluttuja toimintoja ja ohjelman käyttäytymistä. Ohjelmistosuunnittelussa olioilla voidaan kuvata mitä tahansa abstraktia tai reaalimaailman käsitettä. Olio on luokan ilmentymä, ja luokka määrittelee, mitä tietoa ja toimintoja sen olioilla on. .NET Framework:in ansiosta C#-kielellä on oletuksena laaja luokkakirjasto.

Kaikilla C#:n tarjoamilla hyödyillä on kuitenkin myös haittapuolensa. C#-kieltä on kritisoitu muun muassa siitä, ettei sen nopeus ja tehokkuus yllä aivan samalle tasolle kuin esimerkiksi C++-kielen. Myös sovelluksista tehtävien asennuspakettien koko on kasvanut uusien versioiden myötä melko paljon. Näihin ongelmiin on kuitenkin lupailtu Microsoftin mukaan ratkaisuja uusimman C#-version, versio 4.0:n myötä.

Kuvassa 5.2 on yksinkertainen C#-kielinen ohjelma, joka kirjoittaa näytölle ”Hello world!”.

```
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello world!");
        }
    }
}
```

Kuva 5.2 Yksinkertainen C#-kielinen ohjelma

5.3 Windows Mobile SDK

Yleistä ohjelmistokehitystä varten tekniikoiden valmistajat tekevät ohjelmoijille käytettäväksi SDK:ita (Software Development Kit) eli tietyn tekniikan ohjelmistokehitykseen tarkoitettuja kehityspakkeja, jotka pitävät sisällään kaikki tarvittavat työkalut, apuohjelmat, dokumentaation, lähdekoodin ja luokkakirjastot, joita tarvitaan kyseisellä sovelluksella tai tekniikalla ohjelmoimiseen. Windows Mobile SDK on Windows Mobile -kehitykseen tarkoitettu ohjelmistokehityssarja, joka sisältää muun muassa .NET Compact Frameworkin sekä Pocket PC- ja älypuhelinemulaattorit.

5.3.1 .NET Compact Framework

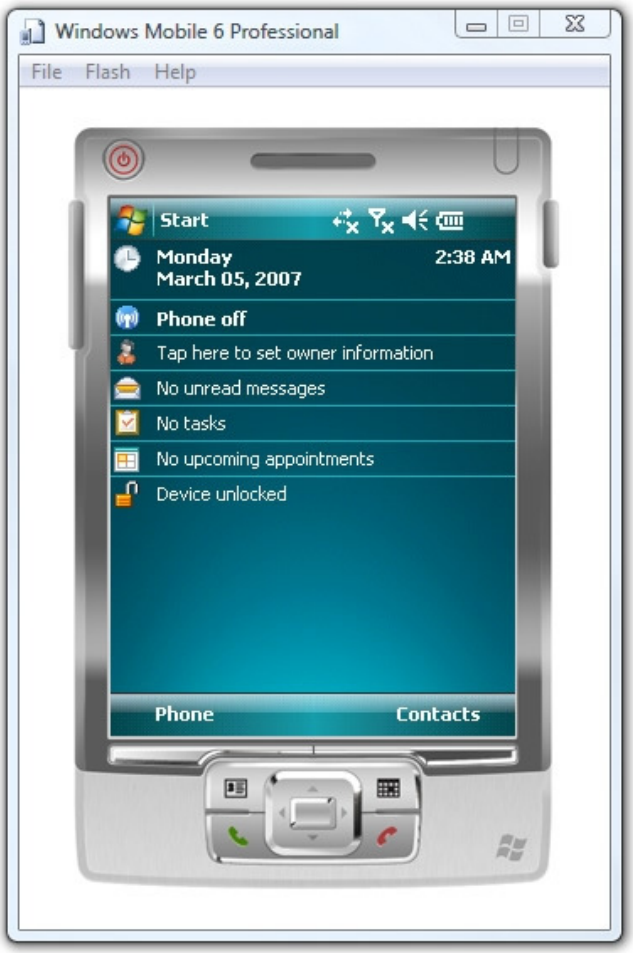
Koska mobiililaitteissa ja sulautetuissa laitteissa ja järjestelmissä on rajallinen määrä prosessointitehoa ja muistia, täytyy niiden käyttöympäristöjen ja sovellusten olla mahdollisimman karsittuja. Microsoft on kehittänyt pöytäkoneille suunnatun .NET Frameworkin rinnalle .NET Compact Frameworkin, juuri tähän tarkoitukseen. .NET Compact Framework sisältää noin 30 prosenttia täyden .NET Frameworkin toiminnoista ja luokkakirjastoista sekä lisäksi joitain mobiililaitteille ja sulautetuille laitteille ja järjestelmille tarvittavia toimintoja ja luokkia. (Microsoft 2010c) Kaikki tärkeimmät ominaisuudet kattavien ja toimintarikkaiden

sovellusten kehittämiseen ovat mukana, mutta joitain hyödyllisiä ominaisuuksia on jouduttu karsimaan kokonaan pois. Yksi mainitsemisen arvoisista karsituista ominaisuuksista on MySQL-tietokantatuki.

5.3.2 Pocket PC -emulaattori

Mobiililaitteille suunnattuja .NET-sovelluksia ja -ohjelmistoja kehitetään Visual Studiolla PC-ympäristössä, mutta niitä voidaan kuitenkin ajaa vain mobiililaitteilla Windows CE -ympäristössä. Tästä syystä Microsoft tarjoaa mobiilisovellusten ajamista varten PC:lle Windows Mobile -emulaattoreita.

Pocket PC -emulaattori on PC:llä toimiva, virtuaalinen Pocket PC -laite, joka suorittaa täysin samaa ohjelmistoa ja ohjelmakoodia kuin fyysinen Pocket PC-laite. Tämä mahdollistaa sen, että Visual Studiolla tehtävä Pocket PC -sovellus voidaan ajaa ja testata luotettavasti ilman fyysistä Pocket PC -laitetta. Myös ohjelmakoodin debuggaus eli ajonaikainen kooditason tarkastelu ohjelmointivirheiden havaitsemiseksi on mahdollista Pocket PC -emulaattoreilla. Emulaattorin käyttö nopeuttaa sovelluskehitystä, koska uutta ohjelmakäännöstä ei tarvitse ladata fyysiseen Pocket PC -laitteeseen jokaisen koodimuutoksen jälkeen, vaan se voidaan ajaa samalla tietokoneella, millä sitä kehitetäänkin. Kuvassa 5.3 on esimerkki Windows Mobile -emulaattorista.



Kuva 5.3 Windows Mobile 6 Professional -emulaattori (Chinnathambi 2007)

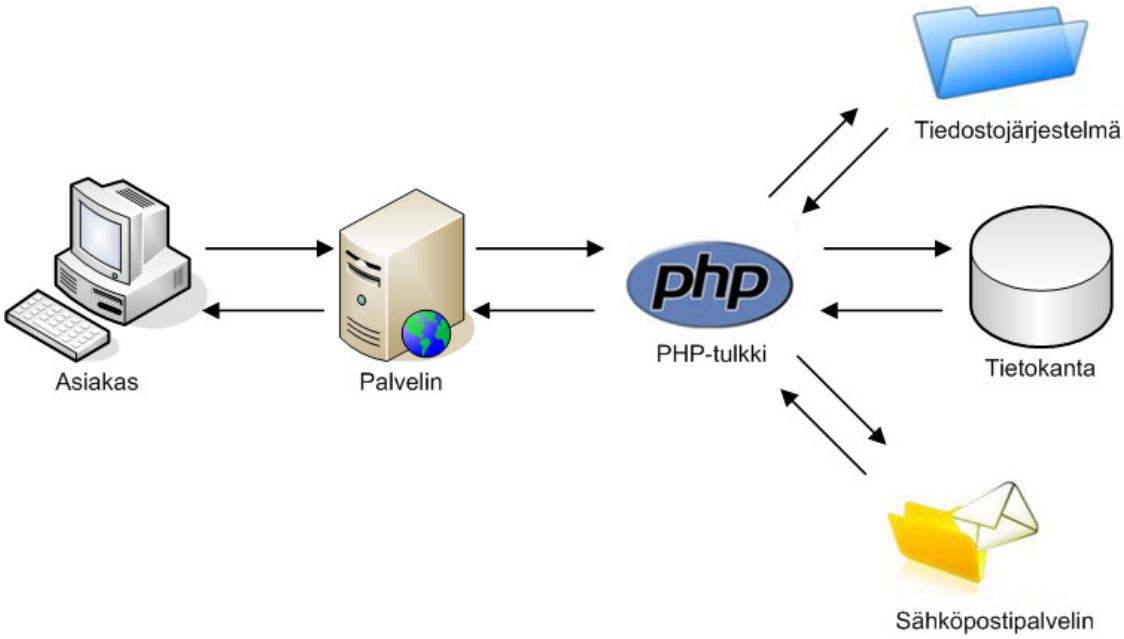
Mikäli Pocket PC -emulaattorilla halutaan muodostaa Internet-yhteys, täytyy emulaattori konfiguroida käyttämään verkkokorttia. Koska emulaattori on virtuaalinen jäljennös Pocket PC -laitteesta, sisällä muuta kuin Pocket PC -ajoympäristön ja sovellukset. Näin ollen emulaattorille täytyy asettaa ulkopuolinen verkkokortti, joka yleensä on tietokoneessa, jolla emulaattoria ajetaan. Emulaattorin ja tietokoneen verkkokortin väliin tarvitaan kuitenkin sovellusta liittämään fyysinen verkkokortti emulaattorin käytettäväksi. Microsoft tarjoaa tähän tarkoitukseen ilmaista Windows Virtual PC -sovellusta. Yksinkertaisimmillaan Pocket PC -emulaattori voidaan asettaa käyttämään tietokoneen verkkokorttia asentamalla tietokoneelle Windows Virtual PC -sovellus, jonka jälkeen emulaattorin asetuksista voidaan verkkokortiksi valita tietokoneen verkkokortti.

5.3.3 Mobiilisovelluksen käyttöönotto Pocket PC -laitteella

Mobiilisovelluksen käyttöönottoon Pocket PC -laitteella Microsoft tarjoaa ohjelmallista ratkaisua. Riippuen kehitysympäristön käyttöjärjestelmästä tiedostojen siirtoon ja synkronointiin käytetään joko sovellusta ActiveSync (Windows XP tai aikaisempi käyttöjärjestelmä) tai Windows Mobile Device Center (Windows Vista tai Windows 7 -käyttöjärjestelmä) (Microsoft 2010d). Kyseiset sovellukset toimivat pääpiirteissään samalla tavalla. Ensin käynnistetään käyttöjärjestelmän mukainen synkronointisovellus. Tämän jälkeen Pocket PC -laite kytketään tietokoneeseen laitteen mukana tulevan, tiedostojen siirtoon tarkoitetun, USB-johdon avulla. Synkronointisovellus tunnistaa automaattisesti kytketyn laitteen ja automaattisen alkukonfiguraation jälkeen voidaan kehitettävä mobiilisovellus ladata Pocket PC -laitteen muistiin. Sovelluksesta voidaan tehdä joko itse asentuva asennuspaketti tai vaihtoehtoisesti sovellus voidaan ladata ja käynnistää sellaisenaan Pocket PC -laitteella Visual Studion avulla.

5.4 PHP

PHP (rekursiivinen akronyymi sanoista PHP: Hypertext Preprocessor) on laajalti käytössä oleva, avoimen lähdekoodin, yleiskäyttöinen skriptauskieli, joka sopii erityisesti Web-kehitykseen ja joka on mahdollista upottaa HTML-kieleen (The PHP Group 2010). Kuvassa 5.4 on esitetty PHP-selainsovelluksen toiminta.



Kuva 5.4 PHP-selainsovelluksen toiminta

PHP vastaa syntaksiltaan C-kieliä ja Javaa, mutta toisin kuin esimerkiksi .NET-ohjelmointikielet, PHP-kieltä ei käännetä vaan koodin suorituksesta vastaa palvelimen komentotulkki, joka generoi PHP-komennoista HTML-kieltä. Valmis HTML-sivu lähetetään takaisin asiakkaalle ja asiakaspään Internet-selaimessa näkyy haluttu (HTML- ja PHP-kielinen) Internet-sivu. PHP:llä voidaan myös tehdä puhtaita PHP-skriptejä eli komentojonoja, jotka toimivat ns. näkymättöminä käyttäjälle.

5.5 Relaatiotietokannat ja SQL

Relaatiotietokannat ovat kokoelma loogisesti toisiinsa liittyvää tietoa. Relaatiotietokanta koostuu tauluista, jotka sisältävät rivejä ja tietokannan toiminnan perustana ovat taulujen ja rivien relaatiot eli yhteydet toisiinsa. Yhdellä tietokantataululla kuvataan yleensä, jotakin reaalimaailman käsitettä ja taulun sarakkeella käsitteen ominaisuutta. Esimerkkinä kuvan 5.5 mukainen Asiakas-taulu, mikä pitää sisällään sarakkeet asiakasID, nimi, osoite ja puhelinnumero. Jokainen taulun rivi sisältää yksilöivän ominaisuuden, ID:n, joka määritellään taulun primääriavaimeksi. Vierasavainten avulla puolestaan luodaan taulujen väliset yhteydet.

asiakasID	nimi	osoite	puhelinnumero
1	Antti	Katu 1	040 1111 222
2	Jani	Katu 2	050 1234 567
3	Mari	Testikatu 9	050 5555 123

Kuva 5.5 Asiakas-taulu

SQL (Structured Query Language) on IBM:n kehittämä, standardoitu relaatiotietokantojen hallintaan tarkoitettu kyselykieli. Sen avulla on mahdollista luoda tietokanta ja sen rakenne, hakea, poistaa, muokata ja lisätä tietokantaan tietoa sekä asettaa tietokannalle taulukohtaiset käyttöoikeudet. SQL toimii pohjana monille tietokannanhallinta sovelluksille ja yksi tunnetuimmista on MySQL. (Koulutus- ja konsultointipalvelu KK Mediat 2010)

MySQL on ruotsalaisen MySQL Ab:n kehittämä, yksi maailman suosituimmista avoimen lähdekoodin tietokantasovelluksista. Suorituskyvyltään tehokas MySQL on saatavilla lukuisille alustoille ja se soveltuu niin pienten, kuin suurtenkin järjestelmien taustalle. Kaikki suosituimmat ohjelmointikielet tarjoavat rajapinnat MySQL-tietokantojen käyttöön.

5.6 phpMyAdmin

phpMyAdmin on ilmainen, PHP:llä kirjoitettu, MySQL-tietokannan hallintaan tehty sovellus. Internet-selaimella käytettävällä phpMyAdmin:lla käyttäjä voi kattavien tietokannanhallinta operaatioiden (taulujen ja taulujen rivien luonnin, muokkauksen ja poiston, käyttäjien hallinnan, taulujen relaatioiden ja indeksoinnin jne.) lisäksi tehdä suoria kyselyitä haluamaansa tietokantaan. Helppokäyttöisyyden ja hyvän dokumentaation ansioista phpMyAdmin on suosittu työkalu.

5.7 XML

XML on lyhenne sanoista eXtensible Markup Language. Se on looginen kuvauksieli eli merkkauskieli. Toisin kuin muun muassa Internet-sivujen rakennetta ja ulkoasua kuvaamaan tehty HTML-kieli, XML on suunniteltu pelkästään tiedon siirtoon ja jäsentämiseen. XML-kieli ei itsessään tee mitään tai saa aikaan

minkäänlaista toiminnallisuutta vaan XML-kuvausten käyttöön aina tarvitaan ohjelmallinen XML-jäsennin (parser). Ulkoasultaan XML-kieli on hyvin lähellä HTML:ää. Kuten HTML-kielessä, XML:ssä on aloitus- ja lopetustagi, joiden avulla jäsennin tunnistaa, minkälaista tietoa kuvataan ja millä tavalla. Erona kuitenkin HTML:n tageihin on, että XML-tagit eivät ole millään tapaa ennalta määriteltyjä vaan ohjelmoija määrittää tagit itse. XML-merkkaus tapahtuu elementeillä (elements) ja ne koostuvat alkutagista (start tag), sisällöstä sekä lopputagista (end tag). (Refsnes Data 2010.)

```
<viesti>
     <vastaanottaja>Matti</vastaanottaja>
     <lahettaja>Jani</lahettaja>
     <otsikko>Muistutus</otsikko>
     <teksti>Nähdään viikonloppuna!</teksti>
</viesti>
```

Tässä esimerkkitapauksessa on XML-kielellä kuvattu viesti. Aloitustagi (<viesti>) aloittaa viestin ja kaikki mitä on aloitustagin ja lopetustagin (</viesti>) välissä sisältyy viestiin. <viesti> on tässä tapauksessa vanhempi-tagi ja <vastaanottaja>, <lahettaja>, <otsikko> ja <teksti> ovat lapsi-tageja. On hyvin tavallista, että sisäkkäiset, lapsi-tagit sisennetään tämän esimerkin tavoin lukemisen helpottamiseksi.

5.8 XAMPP

XAMPP on alustariippumaton, ilmainen, avoimen lähdekoodin Web-palvelin paketti. Sana XAMPP on akronyymi ja se tulee sanoista alustariippumaton (X), Apache (A), MySQL (M), PHP (P) ja Perl (P). XAMPP on valmis paketti, joka pitää sisällään vähintään edellä mainitut palvelimet ja ohjelmointikielituet Internet- ja tietokantapalvelimen pystyttämiseen. Uusimmat versiot pitävät sisällään paljon muitakin ominaisuuksia, kuten esimerkiksi phpMyAdmin-tietokantahallinta työkalun sekä FTP-palvelimen. XAMPP:n takana on ollut ajatus rakentaa helposti asennettava Apache-palvelinympäristö ohjelmakehittäjille (Seidler 2009). XAMPP-palvelimen asentaminen onkin tehty helpoksi eikä

yleensä vaadi kuin asennuspaketin purkamisen ja haluttujen palvelimien ja palvelujen käynnistämisen toimiakseen.

6 INNOMOBIILI-SOVELLUKSEN KEHITYSPROJEKTIN VAIHEET

Aloitin opinnäytetyöni Innomobiili-järjestelmästä marraskuussa 2009 opinnäytetyönä. Samaan aikaan aloitti Niko Rissasen opinnäytetyönsä Innonet-järjestelmästä. Pidimme ensimmäisen asiakaspalaverin 9.11.2009, jonka jälkeen tutustuimme Innotek Oy:n toimintaan sekä aiemmissa projekteissa Innomobiili- ja Innonet-järestelmistä tuotettuun materiaaliin.

6.1 Projektin organisointi

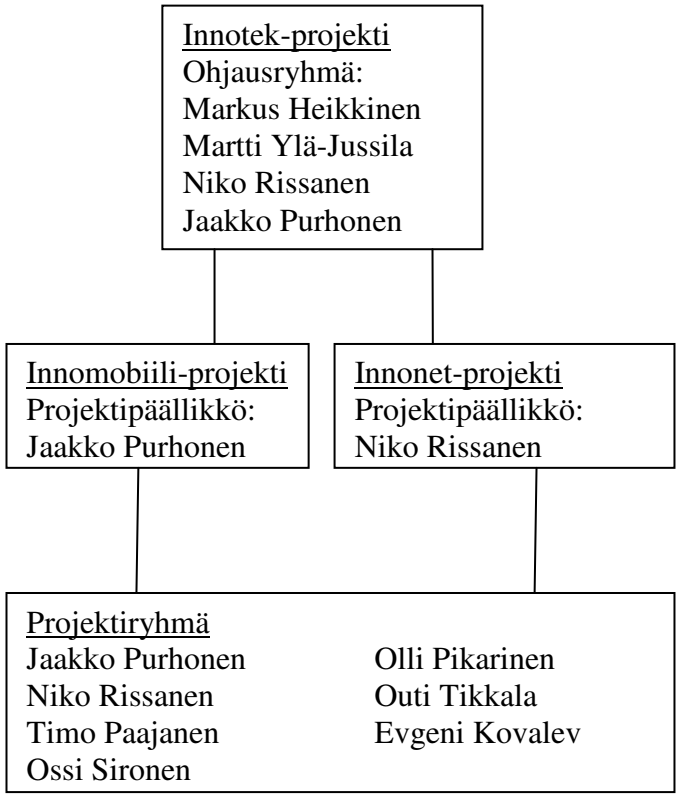
Alun perin opinnäytetyöni rajattiin kattamaan Innomobiili-sovelluksen toiminnallinen määrittely, suunnittelu ja toteutus. Niko Rissasen opinnäytetyö rajattiin vastaavalla tavalla. Koska kyseessä olivat projektiluontoiset työt, nimettiin meidät omien projektiemme projektipäälliköiksi. Työtehtävämme luonnollisesti poikkesivat tietyiltä osin perinteisistä projektipäällikön tehtävistä, mutta pääasiassa vastuu projektien johtamisesta ja toteutuksesta oli meillä itsellämme. Projektiorganisaatiolla oli myös ohjausryhmä, johon kuului asiakkaan edustajasta, Innotek Oy:n toimitusjohtaja Markus Heikkinen sekä molempien opinnäytetöiden ohjaaja, lehtori Martti Ylä-Jussila. Opinnäytetyötä hakiessani ohjaajana toimi lehtori Mikko Huhtanen, mutta pian aloituksen jälkeen ohjaajaksi vaihtui lehtori Martti Ylä-Jussila. Raportoimme projektin kulkua viikkopalavereissa ohjausryhmälle. Asiakkaalta saimme järjestelmiin kohdistuvat vaatimukset ja asiakkaan sekä opinnäytetöiden ohjaajan kanssa katselmoimme sopivin väliajoin projektin tuotoksia.

Alusta alkaen oli selvää, että tulisimme Niko Rissasen kanssa tekemään yhteistyötä projekteissamme, koska Innomobiili ja Innonet käyttävät samaa tietokantaa ja yhteisiä rajapintoja. Innomobiilin ja Innonetin suhde toisiinsa on sellainen, että Innonet toimii pohjana Innomobiilille. Innonetillä hallitaan muun muassa molempien järjestelmien käyttäjiä, asiakkaita ja työnantoja sekä sillä voidaan

tulostaa raportteja. Innomobiililla pääasiassa kirjataan Energo-kartoitus- ja asennus-työnantoja, jotka lähetetään Innonetille käsiteltäväksi ja tulostettavaksi raporttien muodossa.

Innomobiili-projektin aikataulutus oli alun perin seuraava. Esitutkimus ja projektisuunnitelma ovat valmiina 31.12.2009. Innomobiilin toiminnallisen määrittelyyn varattiin aikaa noin kolme kuukautta, joten se valmistuisi 1.3.2010 mennessä. Toteutukselle varattiin aikaa kuukausi, sovelluksen ensimmäisen version valmistuessa 1.4.2010. Innomobiili olisi testattu 15.4.2010, jonka jälkeen seuraisi noin viikon käyttöönotto. Projektin edetessä kohtasimme kuitenkin paljon muutoksia koskien sekä Innomobiilia että Innonetiä, joten aikataulutus vaati uudelleen arviointia useaan kertaan.

Toukokuussa 2010 alkoi Innomobiili- ja Innonet -projektien todelliset laajuudet olla selvillä. Huomasimme, että opinnäytetyön puitteissa emme ehtisi saamaan projektejamme valmiiksi, joten aloimme opinnäytetöidemme ohjaajan kanssa etsiä lisää työvoimaa projektiryhmiimme. Aiemmin mukaan tulleen Timo Paajasen lisäksi Innonet-projektiin liittyivät Outi Tikkala sekä Olli Pikarinen. Evgeni Kovalev liittyi Innomobiili-projektiin opinnäytetyötään tekemään. Ossi Sironen oli jo aiemmin liittynyt mukaan tekemään opinnäytetyötään Innomobiili- sekä Innonet-projektien testauksesta. Kaaviossa 6.1 on esitetty koko Innotek-projektin viimeisin organisaatio.



Kaavio 6.1 Innotek-projektin organisaatio

6.2 Vanha Innomobiili-sovellus

Innomobiilin kehitys ei alkanut täysin tyhjästä, vaan Innomobiili-sovelluksesta oli tehty prototyyppiversio Pohjois-Karjalan ammattikorkeakoulussa. Ensimmäisen asiakaspalaverin jälkeen asiakas toimitti minulle materiaalia koskien Innomobiili- ja Innonet-järjestelmiä. Materiaali piti sisällään prototyypin Innomobiili-sovelluksesta ja Innonet-sivustosta, tietokannan, valmiita näyttökuvia sekä Innotek Oy:n työnantojen prosessikuvauksia ja lomakkeita mitkä ovat vielä käytössä. Innomobiilin, Innonetin ja Innotek Oy:n tietokannan dokumentaatio puuttui kuitenkin täysin ja pian selvisi, ettei kunnollisia dokumentteja ollut. Tein esitutkimuksen Innomobiili-sovelluksesta edellä mainitun materiaalin pohjalta. Lähetin myös Innomobiilin prototyypin kehittäjille Pohjois-Karjalan ammattikorkeakouluun sähköpostia, että he lähettäisivät minulle kaiken mitä he olivat tehneet kyseisen prototyypin kehittämiseen liittyen. Paluusähköpostina sain uusimman version Innomobiilin prototyypistä sekä myöhemmin lyhyen dokumentin, joka kuvasi Innomobiilin prototyypin toiminnan pääpiirteittäin.

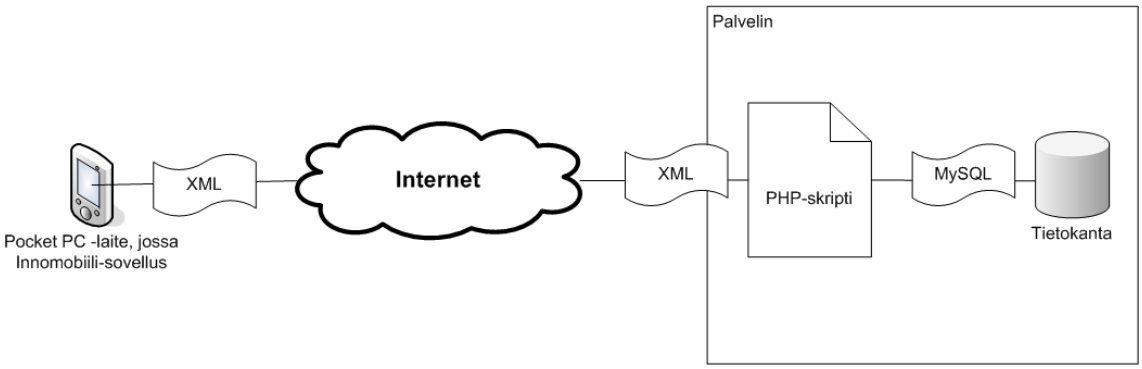
6.3 Vaatimusten selvittäminen, esitutkimus ja projektisuunnitelma

Esitutkimuksen kirjoittaminen sujui ongelmitta. Ensimmäisen asiakaspalaverin selvitysten ja minulle toimitetun materiaalin pohjalta minulle kehittyi selkeä yleiskuva kehitettävästä Innomobiili-sovelluksesta.

Sovelluksen toiminnalliset vaatimukset olivat tässä vaiheessa seuraavat:

- kirjautumisen kautta työnantojen hakeminen
- asuntokohteiden Energo-kartoitusten ja asennusten tekeminen, vikaosioineen
- tuntitöiden automaattinen ajanottaminen
- kohteista otettujen kuvien ja videoiden lisääminen kartoitusten ja asennusten vikaosioihin
- työmatkojen laskeminen mahdollisesti GPS-paikannusta hyödyntäen
- töiden väliaikainen tallennus Pocket PC -laitteen muistiin sekä Internet yhteyden ollessa saatavilla pysyvästi Innotek Oy:n tietokantapalvelimelle.

Aiempien prototyyppisovellusten pohjalta kaikki sovelluksen kehittämisessä käytettävät tekniikat olivat jo lyöty lukkoon. Sovellus tehdään pääasiassa C#-ohjelmointikielellä. Ainut poikkeus on tietokannan käsittely, joka täytyy .NET Compact Frameworkista johtuen hoitaa erillisellä PHP-skriptillä. Pocket PC-laitteesta lähetetään parametrisoitu pyyntö palvelimella sijaitsevaan .php-tiedostoon, joka kutsuu parametrien mukaisen tietokantakyselyn. Kaikki tietokantaan välitettävä tieto siirretään XML-muodossa. Innomobiili on osittain offline-sovellus, sillä työnantojen hakeminen ja tehdyn työn tallennus tietokantaan vaatii Internet-yhteyttä toimiakseen, mutta esimerkiksi työnantojen suorittamisen ja merkitsemisen sekä väliaikaisen tallennuksen on toimittava ilman Internet-yhteyttä. Kuvassa 6.1 on esitetty Innomobiili-järjestelmän yleisarkkitehtuuri.



Kuva 6.1 Innomobiili-järjestelmän yleisarkkitehtuuri

Koska Innomobiili-sovellusta käytetään Pocket PC -laitteilla, aiheuttaa se omat haasteensa käyttöliittymän suunnittelussa. Sovelluksen käyttö pitää tapahtua pääasiassa kosketusnäyttöä käyttäen. Erilaiset Pocket PC -laitteiden näyttökoot aiheuttavat sen, että kaikki sovelluksen näkymät täytyy skaalata sovelluksen ajon aikana näytölle sopivaksi. Koko Innomobiili-sovelluksen täytyy olla helppokäyttöinen, rakennuskohteessa kartoitusten ja asennusten aikana ja/tai päätteeksi käytettävä sovellus. Vaikka käyttöliittymä suunnittelu ei sinällään kuulu opinnäytetyöni rajaukseen, on asiakkaan vaatimuksia ja toiveita noudatettava myös tällä saralla.

Kun esitutkimus oli hyväksytty silloisen opinnäytetyöni ohjaajan, Mikko Huhtasen toimesta, laadin Innomobiili-projektin ensimmäisen projektisuunnitelman. Tässä vaiheessa projektin riskien ja työmäärien arviointi oli melko hankalaa, koska minulla ei vielä ollut perinpohjaista kuvaa Innomobiilin tulevasta laajuudesta tai vaiheista, joita kehitysprojekti tulisi vaatimaan. Näytti siltä, että paljon oli jo valmiina. Tein kuitenkin arviot parhaani mukaan tietäen, että projektisuunnitelma on dokumentti, jota täsmennetään ja päivitetään koko projektin ajan. Kun ensimmäinen projektisuunnitelmaversio oli valmis, sovimme asiakkaan kanssa opinnäytetyön virallisen aloituspalaverin ajankohdan. Tässä vaiheessa myös opinnäytetyöni ohjaaja vaihtui lehtori Mikko Huhtasesta lehtori Martti Ylä-Jussilaan.

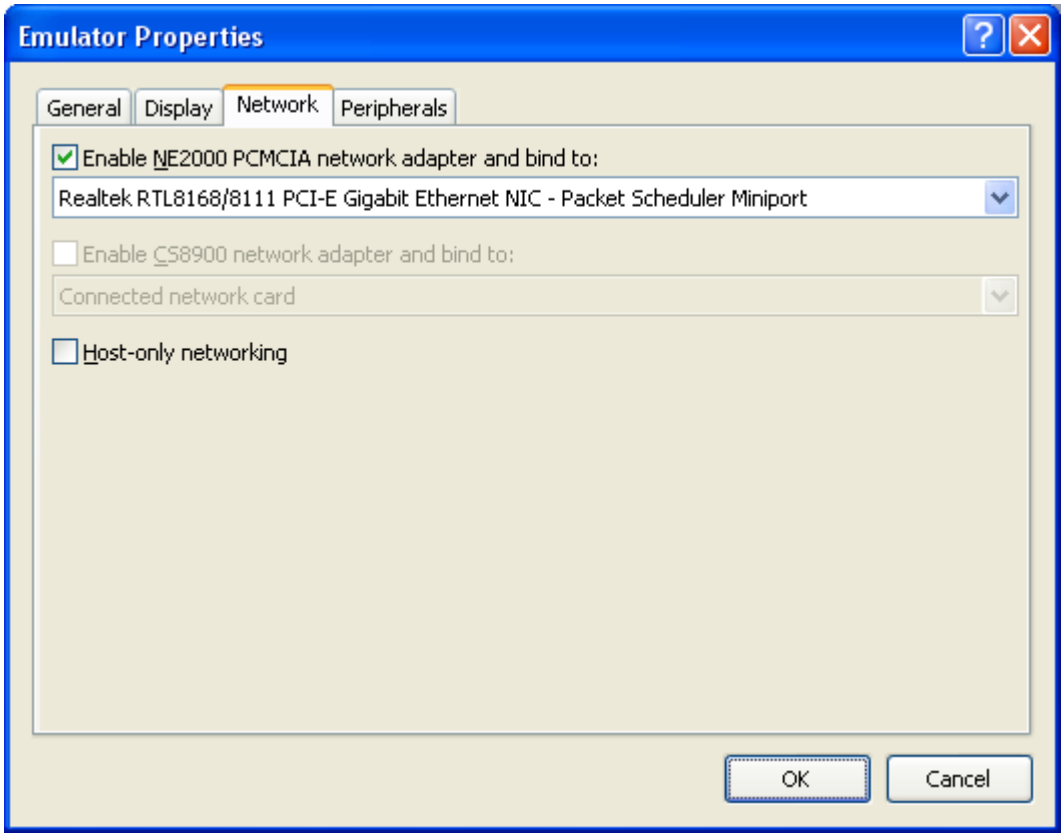
6.4 Projektin alkutoimenpiteet, kehitysympäristön pystytys ja tekniikoiden opiskelu

Koska Innomobiili käyttää tiedon varastoimiseen MySQL-tietokantaa, sovimme asiakkaan kanssa, että projektin ajaksi asennan kyseisen palvelimen omalle tietokoneelleni. Näin ollen tietokannan käyttö ja kehittäminen on helpompaa ja turvallisempaa kuin Innotek Oy:n käytössä olevan tietokannan kanssa. Palvelimen valinnassa päädyin Apache Friendsin kokoamaan XAMPP-palvelin pakettiin, koska se on ilmainen ja helppokäyttöinen. Asennukseen riitti asennuksenpaketin lataaminen ja purkaminen, jonka jälkeen palvelimien hallintaan tarkoitetun graafisen käyttöliittymän sai käyntiin ja sitä kautta pääsi hallitsemaan koko XAMPP-palvelin pakettia.

XAMPP sisältää kattavat ohjeet kuinka palvelimia ja muun muassa PHP-tukea voidaan konfiguroida ja hallita. Kyseisten ohjeiden avulla tein tunnukset phpMyAdmin-tietokannan hallinta työkaluun, jonka jälkeen latasin asiakkaalta saadun Innomobiili- ja Innonet-materiaalin mukana tulleen tietokannan siihen. Näin alkuperäinen tietokanta oli käytössä ja nopean testin jälkeen totesin palvelimen, PHP-tuen ja MySQL-tietokannan toimivan tietokoneellani. XAMPP:n asennuksen jälkeen asensin vielä uusimman version Windows Mobile SDK:sta sekä Service Pack 2:n Visual Studio 2008:lle. Itse Visual Studio 2008:n oli koneessa ennestään.

Kun kehitysympäristö oli kunnossa, aloitin tekniikoiden, asiakkaan toimittaman Pocket PC -laitteen ja ennen kaikkea Innomobiilin prototyyppiversion opiskelu. C#, PHP, SQL sekä XML-merkkauskieli olivat minulle jo ennestään tuttuja, kuten myös Visual Studio 2008 -kehitysympäristö. Täysin uutena asiana tuli Pocket PC -emulaattori sekä sen toiminta ja käyttö. En myöskään ollut ennen käyttänyt Pocket PC -laitetta. Internetistä löysin kuitenkin paljon tietoa ja esimerkkejä molemmista aiheista ja koska Pocket PC -emulaattorin ja -laitteen opiskelu tukivat toinen toistaan, pääsin nopeasti alkuun. Sovelluksen käynnistäminen Pocket PC -emulaattorilla ei tuottanut ongelmia, koska emulaattori käynnistyy automaattisesti Visual Studiossa mobiilisovelluksen käynnistyksen yhteydessä. Internet-yhteyden käyttöönotto emulaattorilla oli alussa hieman on-

gelmallista, koska emulaattorissa itsessään ei ole automaattisesti Internet-yhteyteen tarvittavia verkkolaitteita, joten emulaattorin asetuksista täytyy käytettävä verkkolaite asettaa käyttöön. Tässä tapauksessa emulaattori täytyi asettaa käyttämään tietokoneeni verkkokorttia Internet-yhteyden muodostamiseen. Aiemmin Internetistä löytämieni ohjeiden avulla tämä ei loppujenlopuksi ollut kovinkaan vaikea toimenpide. Ensin täytyi asentaa ilmainen Microsoft Virtual PC -sovellus, jonka jälkeen emulaattorin ollessa käynnissä sen tietoverkko-ominaisuuksista (network properties) sallitaan emulaattorin käyttää verkkokorttia (network adapter) ja valitaan listalta löytyvä, tietokoneessa fyysisesti oleva verkkokortin malli (Kuva 6.2).



Kuva 6.2 Pocket PC -emulaattorin verkkokortin asettaminen käyttöön

Verkkokortin asetuksen jälkeen täytyi emulaattori käynnistää uudestaan. Kun emulaattori oli taas käynnissä, totesin sen käyttöjärjestelmästä, että liitettävyys (connectivity) valikko näytti verkkokortin olevan kytkettynä (Network Card – Connected). Viimeisenä toimenpiteenä asetin vielä emulaattorin Käyttöjärjestelmän yhteydet-asetusten (connection settings) verkkokortit-valinnan (Network

Cards / Network Adapters) alta verkkortille luvan muodostaa yhteys Internetiin. Nyt Pocket PC -emulaattorilla oli mahdollista muodostaa yhteys Internetiin. Lopuksi tallensin emulaattorin tilan sen Save State -toiminnolla.

6.5 Innomobiili-prototyypin analysointi

Seuraavaksi aloin tutkia asiakkaan minulle aiemmin toimittamaa Innomobiilin-prototyyppiversiota. Kävin sovellusta läpi niin ajonaikaisesti kuin ohjelmakoodia tutkimalla, jonka tuloksena piirsin sovelluksesta alustavat ja melko karkeat näyttö-, tila- ja aktiviteettikaaviot. Sovelluksen prototyypin tilaa kuvasti hyvin se, että jouduin itse tutkimaan ja korjaamaan muutamia ajon aikana ilmenneitä, sovelluksen toiminnan kaatavia ohjelmointivirheitä, jotta sain ajettua sovelluksen jokaisen näytön ja osa-alueen. Luonnollisesti sovelluksen ohjelmakoodia täytyi muuttaa myös niin, että sen käyttämän tietokantapalvelimen IP-osoite vastasi oman tietokoneeni IP-osoitetta, jotta sovellus yhdistyi oman tietokoneeni XAMPP-palvelimeen ja sitä kautta tietokantaan.

Vaikka Innomobiili-sovelluksen prototyyppi oli selvästi tietyiltä osa-alueilta vajavainen, oli siihen tehty jonkinlaiset toteutukset osaan toiminnoista. Alustavan analyysi- ja opiskeluvaiheen jälkeen järjestelmän tilanne oli seuraava:

- Kirjautuminen ja töiden hakeminen Pocket PC:n muistista sekä tietokannasta eri päivämäärien mukaan näytti toimivan moitteetta.
- Työnannon valinta ja uuden Energo-kartoituksen teko ja tallennus väliaikaisena Pocket PC:n muistiin toimi.
- Asennuksen suoritus ei toiminut kunnolla ja vikojen lisääminen puuttui kokonaan.
- Tehdyn työn (Energo-kartoitus tai asennus) tallennus tietokantaan ei toiminut ollenkaan.
- Tuntitöiden ja työmatkojen laskeminen puuttuivat kokonaan.
- Palvelimella sijaitseva, tietokantakyselyihin käytettävä PHP-skripti näytti toimivan, mutta sekin tietyissä tapauksissa vajavaisesti.

Seuraavassa asiakaspalaverissa kävimme asiakkaan kanssa läpi sen hetkiset Innomobiilin näyttökuvat siltä varalta, että asiakkaalla olisi muutoksia ja uusia

vaatimuksia niihin. Näytöissä ilmeni jonkin verran muutettavaa, lähinnä käytettävyyden ja esitettävän tiedon suhteen. Kokonaan puuttuvien ominaisuuksien kuten työaikojen ja -matkojen laskentaa varten minun täytyi suunnitella tarvittavat näytöt ja toimintaperiaatteet. Määrittelyvaihetta varten tein kaikista asioista muistiinpanot.

Oma työni jatkui tästä Pocket PC -laitteen GPS:n ominaisuuksien käytön ja ohjelmoinnin opiskelulla. Internetistä saatujen ohjeiden ja Windows Mobile SDK:n mukana tulleiden esimerkkien avulla pääsin nopeasti alkuun. Tein yksinkertaisen prototyyppisovelluksen, jonka tarkoitus oli näyttää sen hetkiset GPS-koordinaatit Pocket PC:n näytöllä. Ongelmaksi kuitenkin muodostui se, ettei Pocket PC -emulaattorissa ollut GPS-paikanninta, koska se oli vain virtuaalinen Pocket PC -laite. Windows Mobile SDK:n dokumentaatiosta löysin kuitenkin ratkaisun ongelmaan. SDK:n mukana tulleella FakeGPS-ohjelmalla oli mahdollisuus syöttää emulaattorille tekaistut, staattiset GPS-koordinaatit ja pian kyseisen sovelluksen käyttöönoton jälkeen sain Pocket PC -emulaattorin näytölle näkymään koordinaatit. Tekaistut GPS-koordinaatit eivät kuitenkaan ratkaise Innomobiili-sovellukseen vaadittua paikkatietojen laskentaa, vaan seuraavaksi minun täytyi selvittää kuinka saisin vastaanotettua oikeat sijaintikoordinaatit. Ajattelin, että helpoiten tämä onnistuu käyttämällä Pocket PC -laitetta toimivan GPS-sovelluksen kanssa. Latasin erään Windows Mobile SDK:n mukana tulleen GPS-sovelluksen Pocket PC -laitteeseen, sen varalta, että oma prototyyppisovellukseni sisältäisi ohjelmointivirheitä. En kuitenkaan saanut minkäänlaisia paikkatietoja sovelluksen avulla vastaanotettua, joten tutkin vielä hieman lisää dokumentaatiota, Internet-lähteitä ja Pocket PC -laitteen asetuksia. Mitään lopullista ratkaisua en onnistunut GPS:n toimintaan löytämään, koska en voinut käyttää loputtomasti aikaa projektin tässä vaiheessa yhden tietyn asian ratkomiseen. GPS-paikannuksen tekninen toteutus jäi siis toistaiseksi auki.

6.6 Analysointivaihe ja toiminnallinen määrittely

Koska kunnollinen dokumentaatio puuttui täysin silloisesta Innomobiili-sovelluksesta, päätimme heti alusta alkaen opinnäytetyön ohjaajan, Martti Ylä-Jussilan kanssa, että ensimmäinen Innomobiili-projektin tärkeä tavoite on tehdä

kattava määrittely koko sovelluksesta ja sen liittymistä. Opinnäytetyöni varsinainen työvaihe alkoi siis prototyyppi-järjestelmän analysoinnilla ja analyysin perusteella tehtävän toiminnallisen määrittelyn kirjoittamisella. Aloitus ja johdantolukujen jälkeen hahmottelin Innomobiilin ja Innonet:n käyttämää tietokantaa. Hyvin pian huomasin, että ilman minkäänlaista dokumentaatiota tietokannan tutkiminen, tulkitseminen ja ymmärtäminen olivat todella vaikeaa ja työlästä. Tietokannassa oli myös käytetty englantia ja suomea sekaisin sekä lyhenteitä, jotka eivät olleet yksiselitteisiä. Myöskään tietokantataulujen välisiä yhteyksiä ei ollut määritetty suoraan tietokantaan vaan ne täytyi päätellä primääri- ja vierasavainten perusteella. Kun havaittiin, että kaikissa tapauksissa monta moneen -yhteyksiä ei ollut toteutettu oikeaoppisesti välitaulun avulla, aloin epäillä koko tietokannan toimintaa ja laajennettavuutta. Koska Innonet-projektin projektipäällikkö Niko Rissanen päätyi tutkimuksissaan samaan tulokseen kuin minä, päätimme yhteistuumin, että tietokanta tullaan suunnittelemaan kokonaan uudestaan. Pian tämän jälkeen Innonet-projektiin nimettiin toiseksi opinnäytetyön tekijäksi Timo Paajanen. Niko Rissanen ja Timo Paajanen aloittivat tietokannan uudelleensuunnittelun Innonetin näkökulmasta ja itse siirryin tekemään Innomobiilin käyttötapauskuvauksia.

Aloitin käyttötapausten dokumentoinnin tekemällä ensin karkean käyttötapauskaavion koko Innomobiilin toiminnasta. Tämän jälkeen jaoin pääkäyttötapaukset pienempiin osiin muodostaen käyttötapauskokoelmia, jotka kuuluivat tietylle pääkäyttötapaukselle. Kun laajemmat käyttötapaukset oli jaettu tarpeeksi pieniin osiin, niin että yksi osa selkeästi vastasi yhtä toimintoa tai toimenpidettä, aloin tehdä osista yksityiskohtaisia käyttötapauskuvauksia sekvenssikaavioineen. Tällä menetelmällä minun oli tarkoitus kartoittaa koko Innomobiili sovellus sekä kaikki vielä puuttuvat toiminnot.

Käyttötapausten kirjoittamisen ohessa suunnittelin Niko Rissasen ja Timo Paajasen kanssa tietokantaa niiltä osin mitkä liittyivät Innomobiilin toimintaan. Käyttötapausten analysoinnin ja suunnittelun pohjalta pystyin esittämään tietokannan toimintaan ja tietosisältöön liittyviä vaatimuksia, jotka mallinsimme yhdessä. Tarkan miettimisen ja suunnittelun avulla tietokanta muodostui eheäksi kokonaisuudeksi. Ongelmia ilmeni tietokannan alati kasvavan koon kanssa. Koska

kyseessä on melko monimutkainen järjestelmä, halusimme alusta alkaen, että tietokanta on todella hyvin suunniteltu, joustava ja mahdollisimman hyvin laajennettavissa, tulevaisuuden varalle. Mitä monimutkaisempi järjestelmä, sitä laajempi ja monimutkaisempi tietokanta yleensä vaaditaan, koska tieto täytyy pilkkoa pieniksi osiksi, jotta sen käsittely olisi mahdollisimman järjestelmällistä ja loogista. Tähän kun vielä lisätään tietokannan mahdollisimman hyvä laajennettavuus, alkaa tietokantatauluja väkisinkin kertyä. Vaikka tietokanta olisi kuinka hyvin ja selkeästi suunniteltu, on se vaikeasti hahmotettava, kun tietokantatauluja on tarpeeksi paljon. Oman tietokantamme kanssa kävi juuri näin. Kun huomasimme, että tauluja oli jo yli 50 kappaletta eikä kaikkia tietokannan osia ollut vielä edes suunniteltu, mietimme, olisiko tähän ongelmaan jotain ratkaisua. Tulimme siihen tulokseen, että jos haluamme tehdä tämän laajuiselle järjestelmälle hyvän tietokannan, on se toteutettava juuri niin kuin tähänkin asti, vaikka tauluja olisi lopullisessa tietokannassa kaksi kertaa enemmän.

Kun tietokantasuunnittelu ja Innomobiilin toiminnallinen määrittely alkoivat olla hyvässä vauhdissa, asiakas ilmoitti tulevansa pitämään palaveria. Palaveri tuli hyvään aikaan, koska katselmoitavaa riitti molemmissa projekteissa. Myös epäselviä asioita oli kertynyt, jotka täytyi käydä asiakkaan kanssa läpi. Kyseinen palaveri osoittautui kuitenkin melkoiseksi käännekohdaksi Innomobiilin ja Innonetin kannalta, koska molempiin järjestelmiin ilmeni paljon uusia vaatimuksia. Innomobiilin jo määriteltyihin ominaisuuksiin ja toiminnallisuuksiin myös kohdistui muutoksia. Vasta tässä kohtaa aloimme tosissaan hahmottaa, kuinka monimutkaisia ja laajoja molemmista järjestelmistä olisi tulossa. Kahden päivän aikana palaveerasimme yhteensä noin kymmenen tuntia ja lopullinen vaatimuslista Innomobiilin pääominaisuuksille näytti palaverien jälkeen tältä:

- Sisäänkirjautuminen, jonka yhteydessä haetaan Pocket PC:n muistista ja tietokannasta (Internet-yhteyden ollessa saatavilla) kaikki päivän työnannot sovelluksen näkymään.
- Työnantojen hakeminen valitulta päivämäärältä.
- Energo-kartoituksen tekeminen ja tallentaminen Pocket PC:n muistiin ja tietokantaan.
- Asennuksen tekeminen ja tallentaminen Pocket PC:n muistiin ja tietokantaan.

- Laajan kuntoraportin tekeminen ja tallentaminen Pocket PC:n muistiin ja tietokantaan.
- Reklamaatiota vastaavan asennuksen tekeminen ja tallentaminen Pocket PC:n muistiin ja tietokantaan.
- Tuntitöiden tekemiseen käytettävä automaattinen ajastintoiminto.
- Liittymä Innonetin varastonhallintaan asennustöissä tarvittavien tuotteiden ja varaosien osalta.
- Muistiinpanomahdollisuus kesken työpäivän tulevien asiakasyhteydenottojen varalle.

Vaikka juuri tämäntyyppiset vaatimusmuutokset ovat tietotekniikkaprojekteissa melko yleisiä, on niihin silti usein vaikeaa varautua. Muutokset saattavat olla toiminnallisesti hyvinkin pieniä, mutta pahimmassa tapauksessa ne vaikuttavat suureen osaan järjestelmän tai sovelluksen toimintaa. Tämä taas väistämättä lisää työmääriä ja kuluja. Mitä aikaisemmassa vaiheessa muutokset tulevat, sitä helpompi ja nopeampi ne on korjata. Innomobiilin tapauksessa muutokset olivat rakenteellisesti kuitenkin sitä luokkaa, että sen aikainen toiminnallinen määrittely jouduttiin käyttötapausten osalta tekemään uudestaan. Uusia toiminnallisia vaatimuksia ja ominaisuuksia tuli myös niin paljon, arvioin käyttötapausten määrän kaksinkertaistuvan. Myös Innonet-järjestelmään tuli merkittäviä muutoksia ja lisävaatimuksia. Muutosten jälkeen Innonet-järjestelmä muistuttaa yhä enemmän toiminnanohjausjärjestelmää kuin pelkästään raportointiin tarkoitettua sovellusta. Lisäksi Innonet ja Innomobiili ovat paljon tiiviimmin sidoksissa toisiinsa kuin mitä alun perin ymmärsimme. Sen lisäksi, että Innomobiililla kerätään asuntokohteissa tehtyjen työiden tiedot, täytyy Innomobiilissa olla liittymä Innonetin uuteen varastonhallintatoimintoon. Myös Innonetin töidenhallintaan täytyy osittain päästä käsiksi Innomobiilista. Innonetillä puolestaan täytyy omien toiminnanohjaustoimintojen lisäksi voida suorittaa kaikki samat toimenpiteet kuin Innomobiilillä.

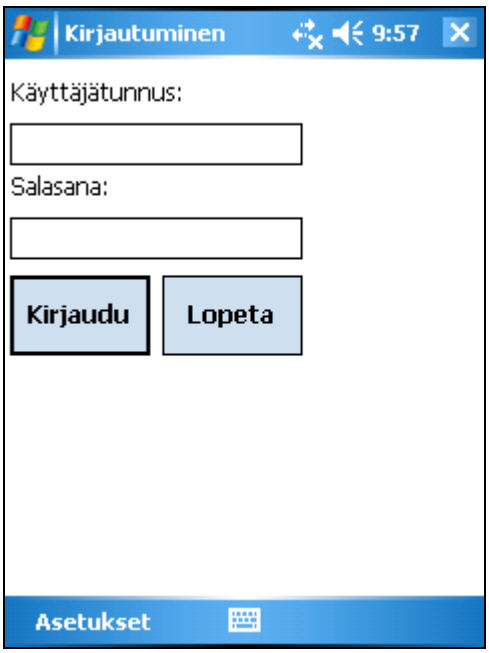
Tästä alkoi Innomobiilin toiminnallisen määrittelyn muuttaminen ja korjaaminen vaatimuksien mukaiseksi. Ongelmana tässä oli kuitenkin se, että opinnäytetyöhön varatut työtunnit alkoivat omalla kohdallani täyttyä. Tämä tarkoitti sitä, että opinnäytetyöni puitteissa ehtisi tehdä uutta toiminnallista määrittelyä valmiiksi.

7 INNOMOBIILI-SOVELLUS

Kuten edellisestä luvusta käy ilmi, Innomobiili-sovellus on vasta määrittelyvaiheessa. Tässä kappaleessa esitetyt toiminnot ja ominaisuudet voivat vielä muuttua määrittelyn edetessä. Varsinkin näyttökuvat tulevat todennäköisesti vielä muuttumaan. Tämä luku kuvaa siis Innomobiili-sovelluksen tilannetta tämän raportin kirjoitushetkenä.

7.1 Sisäänkirjautuminen

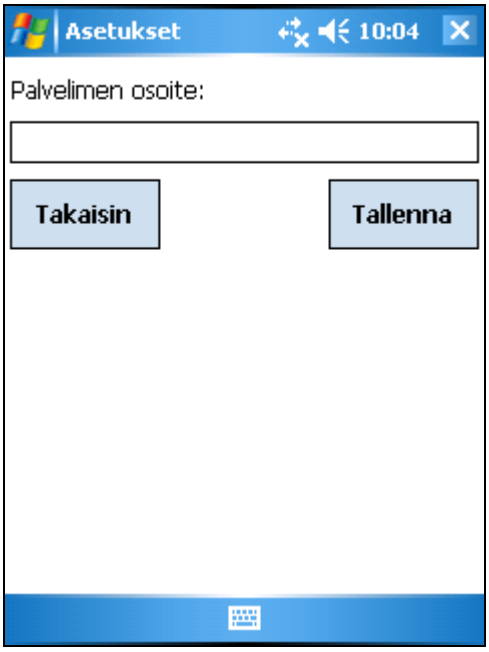
Kun Innomobiili-sovellus käynnistetään, avautuu ensimmäisenä Kirjautuminen-näkymä (Kuva 7.1). Sovellukseen voidaan kirjautua kirjoittamalla käyttäjätunnus ja salasana niille varatuille tekstilaatikoille ja painamalla Kirjaudu-painiketta. Asetukset painikkeesta avautuu Asetukset-näkymä (Kuva 7.2), jolla käyttäjä voi vaihtaa Innomobiili-sovelluksen käyttämän tietokannan osoitetta. Lopeta-painike sulkee sovelluksen.



Kuva 7.1 Kirjautuminen-näkymä

7.2 Asetukset

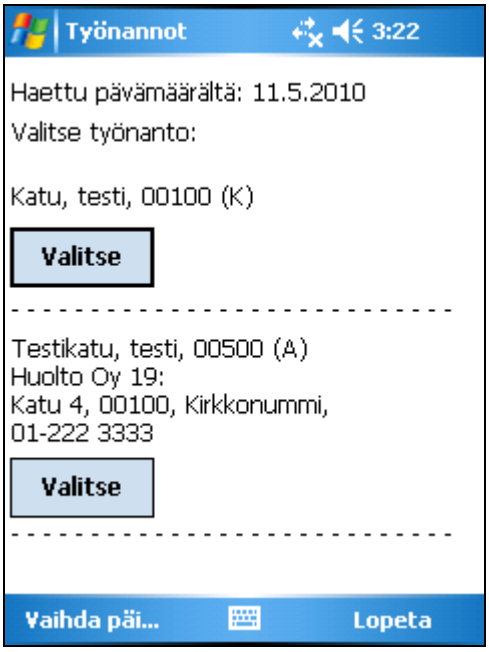
Asetukset-näkymässä (Kuva 7.2) käyttäjä voi vaihtaa Innomobiili-sovelluksen käyttämän tietokannan osoitteen haluamakseen kirjoittamalla sen sille varattuun tekstilaatikkoon. Takaisin-painike palaa takaisin Sisäänkirjautuminen-näkymään tallentamatta muutoksia, Tallenna-painike tallentaa muutokset Pocket PC:n keskusmuistiin sekä asetukset.xml-nimiseen tiedostoon, josta se sovelluksen käynnistyksen yhteydessä voidaan ladata Pocket PC:n keskusmuistiin.



Kuva 7.2 Asetukset-näkymä

7.3.1 Työnantojen hakeminen (oletus päivämäärän mukaan)

Onnistuneen sisäänkirjautumisen yhteydessä Innomobiili-sovellukseen avautuu Työnannot-näkymä (Kuva 7.3), johon ladataan automaattisesti Pocket PC:n muistiin mahdollisesti tallennetut työnannot XML-tiedostosta sekä Internet-yhteyden ollessa saatavilla kyseiselle käyttäjälle ja päivämäärälle asetetut työnannot tietokannasta. Työnannoista tulostetaan työn kohteen perustiedot sekä työnannon tyyppi (K tarkoittaa Energo-kartoitusta, A tarkoittaa asennusta). Jokaista työnantoa kohden on näkymässä Valitse-painike, jolla tehdään halutun työnannon esivalinta. Vaihda päivä -painikkeesta avautuu päivämäärän valinta näkymä. Lopeta-painike sulkee sovelluksen.



Kuva 7.3 Työnannot-näkymä, johon tulostettu työnantoja

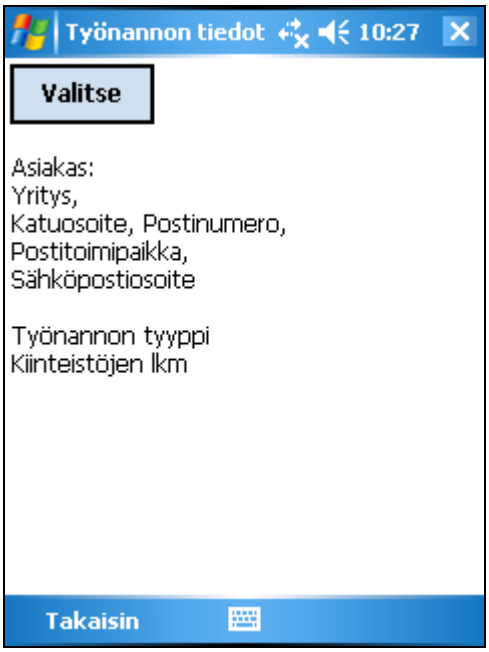
7.3.2 Työnantojen hakeminen (valitun päivämäärän mukaan)

Työnannot-näkymän (Kuva 7.3) Vaihda päivä -painikkeesta avautuu päivämäärän valinta näkymä (Kuva 7.4), jolla käyttäjä voi valita päivämäärän minkä mukaan työnantoja haetaan tietokannasta. Valitse päivä -näkymän kalenterilla valitaan haluttu päivämäärä ja painetaan Hae-painiketta. Sovellus palaa takaisin työnantojen valitsemiseen ja hakemiseen tarkoitettuun näkymään ja lataa automaattisesti Pocket PC:n muistiin mahdollisesti tallennetut työnannot XML-tiedostosta sekä, Internet-yhteyden ollessa saatavilla, kyseiselle käyttäjälle ja juuri valitulle päivämäärälle asetetut työnannot tietokannasta. Työnannoista tulostetaan työn kohteen perustiedot sekä suluissa työnannon tyyppi (K tarkoittaa Energo-kartoitusta, A tarkoittaa asennusta).

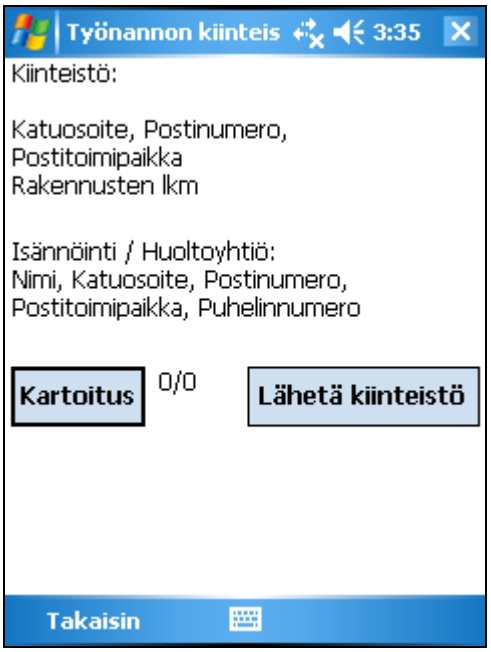
Kuva 7.4 Valitse päivä -näkymä

7.4 Työnannon valinta

Kun Työnannot-näkymään (Kuva 7.3) on haettu työnantoja, voidaan haluttu työnanto esivalita työnannon perustietojen yläpuolella sijaitsevalla Valitse-painikkeella. Ensimmäisen Valitse-painikkeen painalluksen jälkeen avautuu Työnannon tiedot -näkymä, johon on tulostettu työnannon yksityiskohtaisempia tietoja (Kuva 7.5). Takaisin-painikkeella päästään edelliseen, työnantojen valitsemiseen ja hakemiseen tarkoitettuun näkymään. Valitse-painikkeesta valitaan kyseinen työnanto tehtäväksi ja sovellus avaa Työnannon kiinteistö-näkymän (Kuva 7.6).



Kuva 7.5 Työnannon tiedot -näkymä



Kuva 7.6 Työnannon kiinteistöt -näkymä, johon valittu Energo-kartoitus-työnanto

7.5 Energo-kartoituksen kirjaaminen

Energo-kartoituksen suorittamiseksi täytyy käyttäjän valita K-tyyppinen työnanto. Nyt Työnannon kiinteistö-näkymään tultaessa (Kuva 7.6) tulostuu näytölle kiinteistön perustietoja sekä Kartoitus- ja Lähetä kiinteistö -painikkeet. Kartoitus-

painikkeen viereen tulostetaan numeroin valmiiksi kuitatut ja kuittaamattomat, kyseiselle kiinteistölle tehdyt Energo-kartoitukset. Lisäksi näkymän alalaidassa on Takaisin-painike. Kartoitus painiketta painamalla aloitetaan Energo-kartoitusten kirjaaminen, Takaisin painikkeella palataan Työnannon tiedot-näkymään (Kuva 7.5).

Energo-kartoituksen kirjaaminen tapahtuu painamalla Kiinteistö-näkymän Kartoitus-painiketta. Tällöin sovellus avaa Energo-kartoitukset-näkymän (Kuva 7.7).

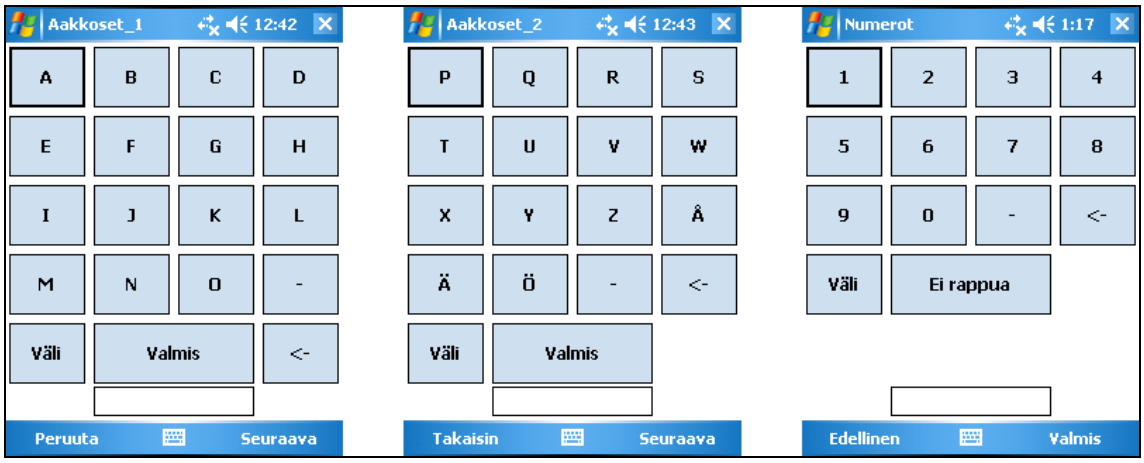
Kuva 7.7 Energo-kartoitukset-näkymä

Kyseiseen näkymään tulostetaan kaikki kyseiseen kiinteistöön tehdyt Energo-kartoitukset painikkeen muodossa. Jokaisessa painikkeessa on tekstinä rappu- ja huoneistonumero sen mukaan mihin kyseinen Energo-kartoitus on tehty. Jokaisen painikkeen vieressä on kyseisen tilan rakennus- ja kerrosnumero sekä tieto, onko kyseinen kartoitus kuitattu valmiiksi vai ei. Mikäli näkymässä on tehtyjä Energo-kartoituksia, päästään niitä muokkaamaan tilan painiketta painamalla, mutta vain, jos Energo-kartoitusta ei ole kuitattu valmiiksi. Uusi Energo-kartoitus voidaan aloittaa painamalla Uusi-painiketta ja Takaisin-painike palaa takaisin Työnannon kiinteistöt -näkymään (Kuva 7.6).

Kun käyttäjä aloittaa kokonaan uuden tai muokkaa vanhaa Energo-kartoitusta, ensimmäiseksi Energo-kartoituksen kirjaamisessa tullaan aina Energo-kartoitus-näkymän Tila-välilehdelle (Kuva 7.8).

Kuva 7.8 Energo-kartoitus-näkymän Tila-välilehti

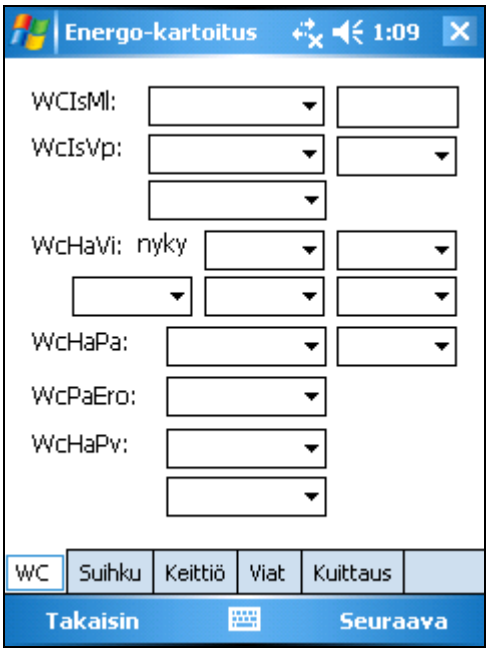
Tila-välilehdellä määritetään tila, mihin Energo-kartoitus tullaan tekemään. Tilan tiedot koostuvat rakennusnumerosta, rappu-tiedosta, huoneistonumerosta, kerroksesta, huoneiston tyypistä sekä lisätiedot-kentästä. Käyttäjä voi valita rapun ja huoneistonumeron joko nuolinäppäimillä selaamalla tai painamalla rapun tai huoneistonumeron painiketta, jonka johdosta sovellus avaa Aakkoset_1-näkymän (Kuva 7.9). Kyseisen näkymän tarkoitus on helpottaa käyttäjää kirjoittamaan haluttu rappu tai huoneistonumero, jos kyseessä on erikoistilanne, esimerkiksi rapuksi täytyisi kirjoittaa ”A – C”. Tilan kerros ja tyyppi valitaan valmiista listoista selaamalla ja tilaan liittyvien kuvien ja videoiden lisäys onnistuu erillisestä näkymästä (Kuva 7.10), Kuvat ja videot -painiketta painamalla.



Kuva 7.9 Aakkosten ja numeroiden syöttö -näkymät

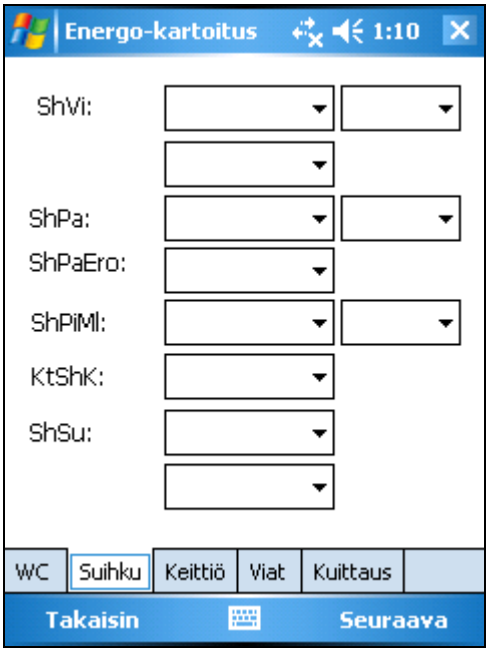
Kuva 7.10 Kuvat ja videot -näkymä

Tila-välilehdeltä (Kuva 7.8) mennään kirjaamaan itse kartoituksessa saadut tiedot Huoneet-painiketta painamalla. WC-välilehdellä (Kuva 7.11) kirjataan wc-istuimen sekä -hanan tiedot sekä suositukset uusille tuotteille.



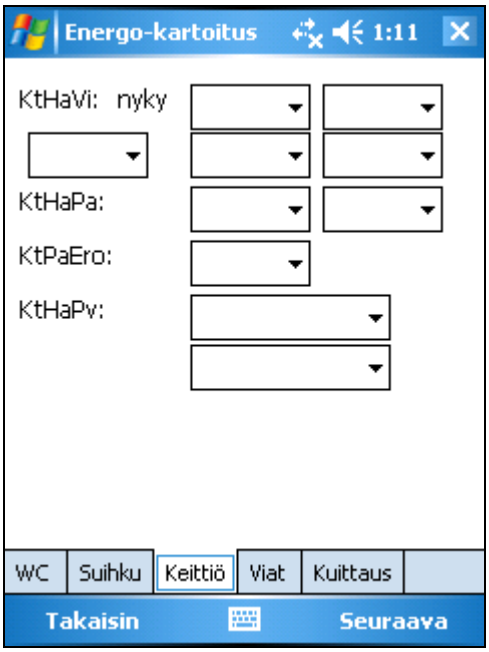
Kuva 7.11 Energo-kartoitus-näkymän WC-välilehti

Suihku-välilehdellä (Kuva 8.12) merkitään suihku-tilaan liittyviä tietoja ja itse suihkun tiedot sekä myös suositukset suihkun suuttimille.



Kuva 7.12 Energo-kartoitus-näkymän Suihku-välilehti

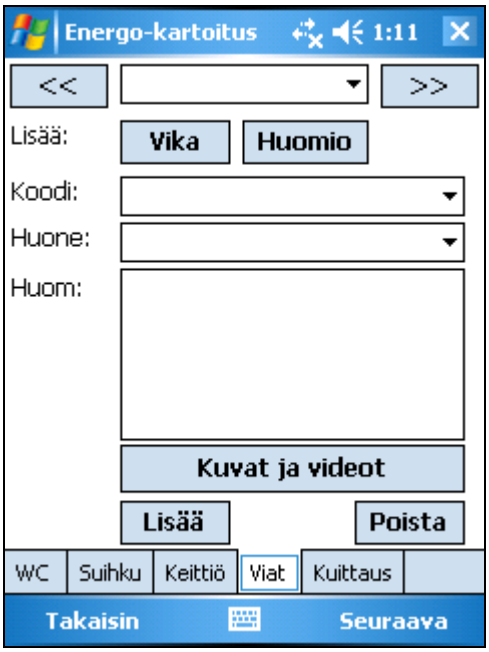
Keittiö-välilehdellä (Kuva 7.13) merkataan keittiön hanan tiedot sekä hanaan liittyvät suositukset.



Kuva 7.13 Energo-kartoitus-näkymän Keittiö-välilehti

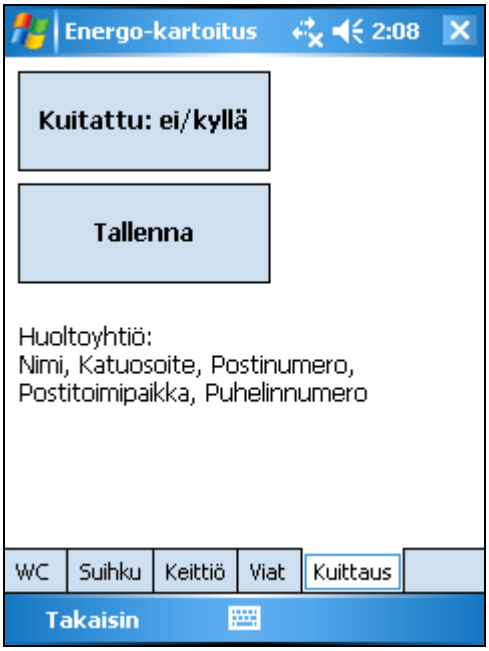
WC-, Suihku- ja Keittiö-välilehtien tietojen kirjaamiseen tarkoitetut alasvetovalikot toimivat niin, että jos käyttäjä ei ole tehnyt mitään valintaa on alasvetovalikon taustaväri keltainen. Kun valinta on tehty, muuttuu kyseisen alasvetovalikon taustaväri valkoiseksi. Värien avulla siis havainnollistetaan käyttäjälle, mitkä tiedot on syötetty ja mitkä ei.

Energo-kartoitus-näkymän Viat-välilehdellä (Kuva 7.14) voidaan lisätä tilassa tai vesikalusteissa huomattuja vikoja ja huomioita. Ensin valitaan painikkeella lisätäänkö vika vai huomio (viat korjataan, huomioita ei). Mikäli lisätään vika, voidaan Huom-kenttä jättää halutessa tyhjäksi, huomiota lisätessä Huom-kenttä on pakollinen tieto. Vikaa lisätessä myös koodi täytyy valita vialle. Huomiota lisätessä koodi on automaattisesti ”Huom”. Vialle tai huomiolle voidaan lisätä asiaa havainnollistavia kuvia ja/tai videoita Kuvat ja videot -painikkeella. Kun kaikki tarvittavat tiedot on syötetty, lisätään vika painamalla Lisää-painiketta.



Kuva 7.14 Energo-kartoitus-näkymän Viat-välilehti

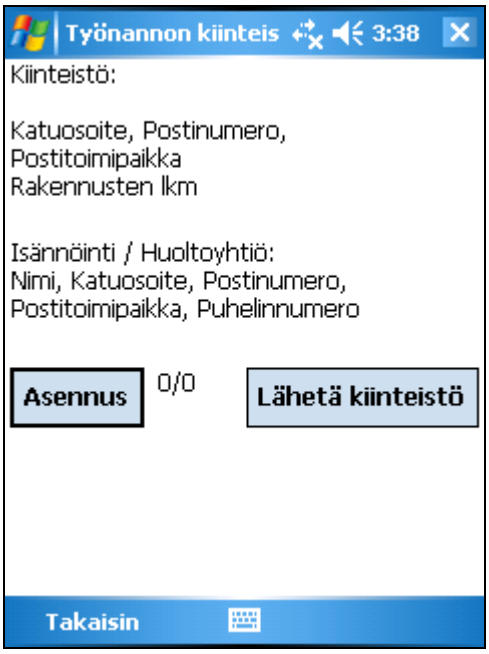
Energo-kartoitus-näkymän Vahvistus-välilehdellä (Kuva 7.15) käyttäjä voi kuitata ja/tai tallentaa tehtävän Energo-kartoituksen Pocket PC:n muistiin, XML-tiedostoon. Käyttäjä voi kuitata Energo-kartoituksen valmiiksi painamalla Kuittaus-painketta, jolloin sen tekstiksi vaihtuu ”Kuitattu: kyllä”. Tallenna-painiketta painamalla sovellus tallentaa Energo-kartoituksen tiedot ja palaa takaisin Energo-kartoitukset-näkymään.



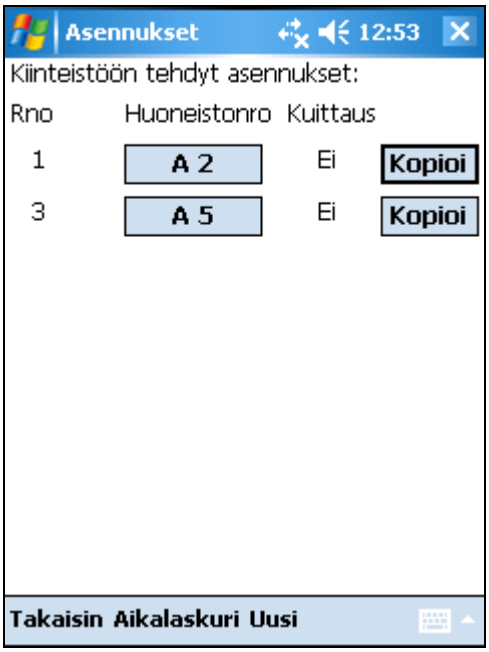
Kuva 7.15 Energo-kartoitus-näkymän Kuittaus-välilehti

7.6 Asennuksen kirjaaminen

Asennuksen suorittamiseksi täytyy käyttäjän valita työnanto, jonka tyyppi on työnantojen valitsemisen yhteydessä A eli Asennus. Näin ollen Työnannon kiinteistö -näkymään tultaessa tulostuu näytölle kiinteistön perustietoja sekä Asennus- ja Lähetä kiinteistö -painikkeet (Kuva 7.16). Asennus-painikkeen viereen tulostetaan numeroin valmiiksi kuitatut ja kuittaamattomat, kyseiselle kiinteistölle tehdyt Asennukset. Lisäksi näkymän alalaidassa on Takaisin-painike. Asennus painiketta painamalla päästään Asennukset-näkymään (Kuva 7.17), Takaisin painikkeella palataan Työnannon tiedot -näkymään (Kuva 7.5).



Kuva 7.16 Työnannon kiinteistö -näkymä, johon valittu asennus-työnanto



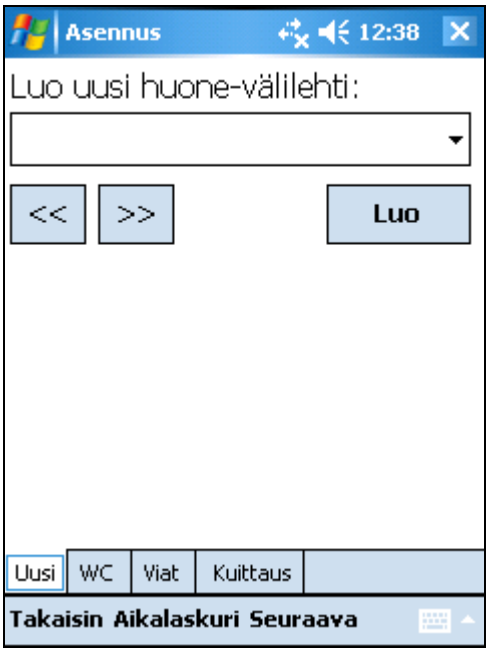
Kuva 7.17 Asennukset-näkymä

Asennukset-näkymään tulostetaan kaikki kyseiseen kiinteistöön tehdyt Asennukset painikkeen muodossa. Jokaisessa painikkeessa on tekstinä rappu- ja huoneistonumero sen mukaan, mihin kyseinen asennus on tehty. Jokaisen painikkeen vieressä on rakennusnumero sekä tieto siitä, onko kyseinen asennus kuitattu valmiiksi vai ei. Mikäli näkymässä on tehtyjä asennuksia, päästään niitä muokkaamaan tilan painiketta painamalla, mutta vain jos Asennusta ei ole kuitattu valmiiksi. Uusi asennus voidaan aloittaa painamalla Uusi-painiketta, Takaisin-painikeella voi palata takaisin Kiinteistö-näkymään ja Aikalaskuri-painike avaa työtehtävien ajanottoon tarkoitetun Aikalaskuri-näkymän (vielä määrittelemätön toiminto). Tehtyjen asennusten painikkeiden vieressä on myös Kopioi-painike, jolla voidaan myös aloittaa uuden asennuksen kirjaaminen, mutta se kopioi valitun tilan huoneet asennukseen valmiiksi kuitenkin niin, etteivät tuotteet viat tai huomiot kopioidu asennukseen. Mikäli käyttäjä aloittaa uuden asennuksen tai muokkaa hän vanhaa, ensimmäisenä asennuksen kirjaamisessa tullaan Asennus-näkymän Tila-välilehdelle (Kuva 7.18).

Kuva 7.18 Asennus-näkymän Tila-välilehti

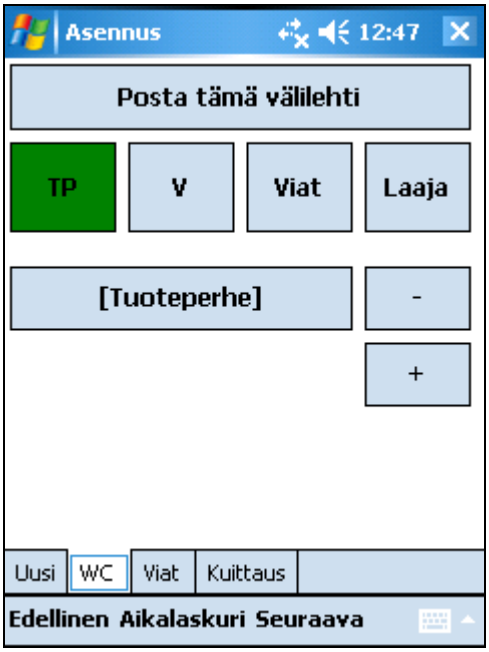
Tila-välilehdellä määritetään mihin tilaan asennus tullaan tekemään. Tilan tiedot koostuvat rappu-tiedosta, huoneistonumerosta huoneiston tyypistä sekä Lisätiedot-kentästä. Käyttäjä voi valita rapun ja huoneistonumeron joko nuolinäppäimillä selaamalla tai painamalla rapun tai huoneistonumeron painiketta, jonka johdosta sovellus avaa aakkosten ja numeroiden kirjoittamiseen tarkoitetun näkymän (Kuva 7.9). Kyseisen näkymän tarkoitus on helpottaa käyttäjää kirjoittamaan haluttu rappu tai huoneistonumero, jos kyseessä on erikoistilanne, esimerkiksi rapuksi täytyisi kirjoittaa ”A – C”. Tilan tyyppi valitaan valmiista listasta selaamalla.

Kun asennettava tila on määritetty, voidaan siirtyä asennuksen tietojen kirjaamiseen. Kuvan 7.19 mukaisella Uusi-välilehdellä voidaan asennukseen luoda uusia huoneita.



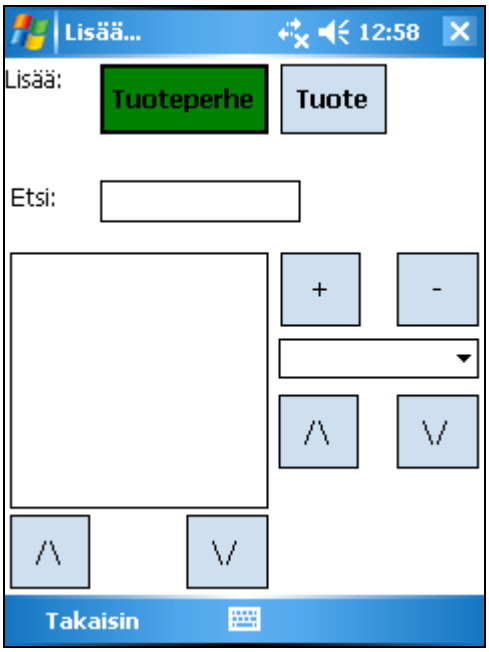
Kuva 7.19 Asennus-näkymän Uusi-välilehti

Huoneen tyyppi valitaan valmiista listasta ja painetaan Luo-painiketta, jolloin valitun huoneen välilehti luodaan asennukseen. Itse Huone-välilehdeltä, esimerkiksi WC, määritetään kaikki mitä kyseiseen huoneeseen tehdään (Kuva 7.20).



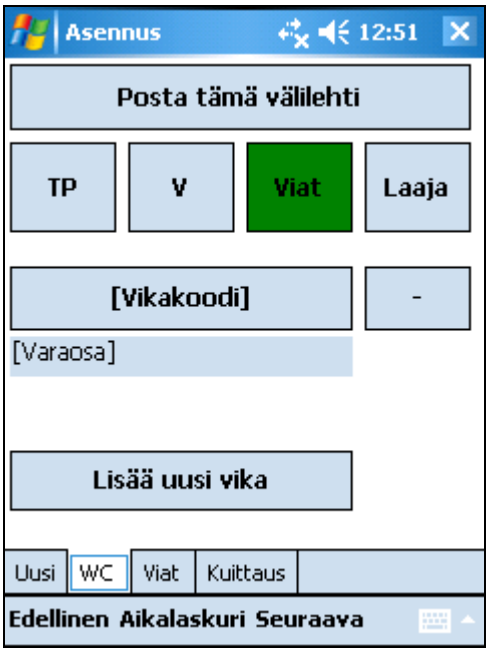
Kuva 7.20 Asennus-näkymän, WC-välilehden tuoteperheet-alinäkymä

Jokainen Huone-välilehti on rakenteeltaan samanlainen. Huone koostuu alinäkymistä tuoteperhe (TP), tuote (T), varaosa (V) sekä viat ja huomiot (Viat). Tuoteperhe-, tuote-, sekä varaosa-alinäkymä ovat toiminnaltaan samanlaisia. Näkymien plus(+)-painikkeista päästään hakutoimintoon (Kuva 7.21), jolla voidaan hakea listalta halutut tuoteperheet, tuotteet tai varaosat ja lisätä huoneeseen. Tuoteperheen, tuotteen tai varaosan kohdalla oleva miinus(-)-painike poistaa kyseisen asian huoneesta.



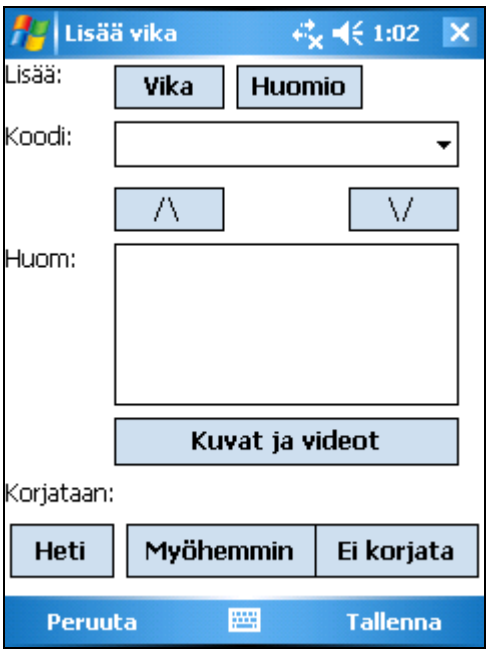
Kuva 7.21 Lisää tuoteperheitä ja tuotteita -näkymä

Käyttäjän painaessa lisätyn tuoteperheen, tuotteen tai varaosan kuvaavaa painiketta muuttuu kyseinen painike vihreäksi tarkoittaen, että kyseinen asia on asennettu huoneeseen. Tuhoa tämä välilehti -painike poistaa kyseisen huonevälilehden asennuksesta. Viat ja huomiot -alinäkymän (Kuva 7.22) toiminta poikkeaa hieman muista Huone-välilehtien alinäkymistä.



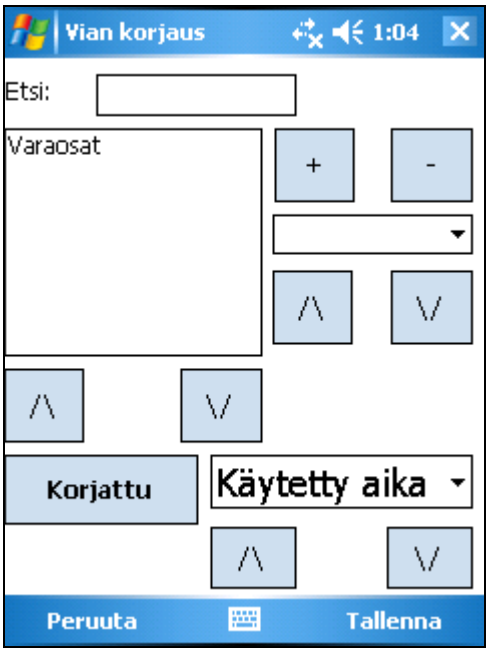
Kuva 7.22 Asennus-näkymän, WC-välilehden Viat ja huomiot -alinäkymä

Viat ja huomiot -alinäkymässä näytetään huoneen kalusteissa havaitut ja huoneelle lisätyt viat ja huomiot sekä mahdollisesti vioille lisätyt varaosat. Lisää uusi vika -painikkeesta avautuu kuvan 7.23 mukainen vian ja/tai huomion lisäämiseen tarkoitettu näkymä.



Kuva 7.23 Lisää vika -näkymä

Lisää vika (Kuva 7.23) voidaan lisätä huoneen vesikalusteissa huomattuja vikoja ja huomioita sekä lisätä vioille varaosia ja aloittaa korjaaminen. Ensin valitaan painikkeella, lisätäänkö vika vai huomio (viat korjataan, huomioita ei). Mikäli lisätään vika, voidaan Huom-kenttä jättää halutessa tyhjäksi. Huomiota lisättäessä Huom-kenttä on pakollinen tieto. Vikaa lisätessä vialle täytyy valita koodi. Huomiota lisätessä koodi on automaattisesti ”Huom”. Vialle tai huomiolle voidaan lisätä asiaa havainnollistavia kuvia ja/tai videoita Kuvat ja videot -painikkeella. Vian lisäyksessä lopuksi valitaan, milloin vika korjataan vai korjataanko Innotek Oy:n toimesta ollenkaan. Mikäli vikaa ei korjata heti, painetaan joko Myöhemmin- tai Ei korjata -painiketta. Tämän jälkeen vika lisätään Tallenna-painikkeella kyseessä olevaan huoneeseen. Jos vika kuitenkin korjataan heti, painetaan Heti-painiketta. Tästä alkaa varsinainen korjaustoimenpide, jonka johdosta aikalaskuri käynnistyy (ei näy käyttäjälle) ja Vian korjaus -näkymä (Kuva 7.24) avautuu.



Kuva 7.24 Vian korjaus -näkymä

Varaosien lisäys -näkymällä käyttäjä voi lisätä listasta tarvittavat varaosat kyseessä olevalle vialle. Varaosat-listasta käyttäjä voi hakea tai selata varaosat ja plus(+)- tai miinus(-) painikkeilla lisätä vialle. Kun varaosat on lisätty, suorittaa käyttäjä korjaustoimenpiteet, jonka aikana aikalaskuri koko ajan ottaa aikaa. Kun korjaukset on suoritettu, painaa käyttäjä Korjattu-painiketta, jonka johdosta

kyseinen korjaus kuitataan suoritetuksi, aikalaskuri pysähtyy. Korjaukseen käytetty aika tulee Käytetty aika -valikkoon. Aikalaskurin aika voidaan tarvittaessa ohittaa ja sen tilalle voidaan valita sopiva aika. Tallenna-painiketta painamalla sovellus palaa takaisin Asennus-näkymän kyseessä olevalle Huone-välilehdelle Viat ja huomiot -alinäkymään (Kuva 7.22). Kaikki kyseisen alinäkymän vikoja kuvaavat painikkeet, jotka on korjattu, kuvataan vihreällä värillä, korjaamattomat viat sinertävällä.

Asennus-näkymän Vahvistus-välilehdellä (Kuva 7.25) voidaan katsoa kyseisen tilan asennuksen yhteenveto, kuitata asennus valmiiksi sekä tallentaa tehtävän asennuksen Pocket PC:n muistiin, XML-tiedostoon. Yhteenvedossa näkyy tilan perustiedot sekä asennetut tuoteperheet, tuotteet, viat ja huomiot sekä vikojen varaosat. Kuittaus tapahtuu painamalla Kuittaus-painiketta, jolloin sen tekstiksi vaihtuu ”Kuitattu: kyllä”. Tallenna-painike tekee tallennustoimenpiteet, joiden jälkeen sovellus palaa Asennukset-näkymään (Kuva 7.17).

Kuva 7.25 Asennus-näkymän Kuittaus-välilehti

7.7 Kiinteistön tallentaminen tietokantaan

Kun käyttäjä on kirjannut ja tallentanut kiinteistöön Energo-kartoituksia tai asennuksia, voi hän tallentaa kyseisen kiinteistön tiedot tietokantaan kuvien 7.6 ja

7.16 mukaisten Työnannon kiinteistö-näkymien Lähetä kiinteistö -painikkeella. Toiminnon aikana Pocket PC:stä lähetetään tietokantapalvelimelle Energo-kartoituksissa tai asennuksissa XML-tiedostoon tallennetut tiedot sekä mahdolliset kuvat ja videot. Palvelimella sijaitsevan PHP-skriptin avulla XML-tiedoston sisältö tallennetaan Innotek Oy:n MySQL-tietokantaan. Kuva- ja videotiedostot tallentuvat palvelimen kiintolevylle ja tietokantaan tallentuvat vain niiden tiedostonimet ja sijainti palvelimella. Kun kaikki tiedot on onnistuneesti tallennettu tietokantapalvelimelle, voi käyttäjä halutessaan poistaa ne Pocket PC:n muistista. Kyseinen toiminto luonnollisesti vaatii, että Internet-yhteys on saatavilla.

7.8 Määrittelyä vailla olevat toiminnot

Koska Innomobiili on edelleen toiminnallisten vaatimusten suhteen muuttuva ja laajentuva projekti, on tätä raporttia kirjoittaessani paljon asioita ja toiminnallisuutta, joita ei ole vielä ehditty suunnitella tai määritellä. Tässä kappaleessa käydään läpi toiminnot, jotka ovat tulossa Innomobiiliin, mutta joista ei voida vielä tarkkoja toiminnankuvauksia kertoa niiden keskeneräisyyden takia.

7.8.1 Laaja kuntoraportti

Alun perin jatkokehitysideana ollut laajan kuntoraportin kirjaaminen tuli Innomobiiliin mukaan 31.3.2010 - 1.4.2010 pidetyssä asiakaspalaverissa. Laaja kuntoraportti on laajennus Energo-kartoitukselle siinä mielessä, että kun Energo-kartoituksessa kartoitetaan vain tietyt vesikalusteet, niin laajassa kuntoraportissa täytyy voida kartoittaa vapaasti valittava tietyn huoneen kaluste tai huone kartoituksen asiakkaan toiveiden tai vaatimusten mukaan. Innomobiilin suhteen vaatimukset ovat sellaiset, että laaja kuntoraportti täytyisi voida tehdä asennuksen yhteydessä ja että asennukseen pitää voida hakea laajan kuntoraportin tietoja tarvittaessa.

7.8.2 Reklamaatio

Mikäli asennettujen tuotteiden, varaosien ja/tai kalusteiden kanssa ilmenee asennuksen jälkeen vikoja tai muuta poikkeavaa, voi asiakas tehdä asennuk-

sesta reklamaation. Näin ollen, jos reklamaatio todetaan aiheelliseksi, täytyy vialliset tuotteet, varaosat ja/tai kalusteet korjata tai vaihtaa ehjiin. Näin ollen asentajien on pystyttävä hakemaan toistensa tehtyjä töitä tietokannasta, koska reklamaatiotyönanto ei välttämättä tule sille asentajalle, joka alkuperäisen asennustyön on tehnyt. Nyt työnantojen haku toimii niin, että asentajat voivat hakea vain itselle asetettuja työnantoja. Reklamaatiossa alkuperäisestä asennuksesta täytyy saada selville muun muassa työn tekijä sekä työn ajankohta. Kiinteistöön tehtyjen asennusten listassa (Kuva 7.17) täytyy olla mahdollisuus aloittaa reklamaatioasennuksen tekeminen tehdyn asennuksen pohjalta. Itse reklamaatioasennuksen kirjaaminen suoritetaan samalla tavalla kuin asennuksenkin.

7.8.3 Tuntitöiden aikalaskuri

Asennuksen kirjaamisen yhteydessä olevan vikojen korjaukseen liittyvän aikalaskurin lisäksi asennuksessa ja reklamaatioasennuksessa täytyy olla myös vapaasti käytettävä ja kontrolloitava aikalaskuri. Kyseistä toimintoa olen jo suunnitellut ja yhdet prototyyppinäytöt ja näyttöjen kuvaukset tehnyt, mutta ne tulevat vielä muuttumaan. Kyseisen aikalaskurin päätoiminta on kuitenkin seuraavanlainen. Kun asennustyönanto on valittu, täytyy aikalaskuriin päästä käsiksi mistä tahansa näytöstä, niin kauan kun asennusta ollaan tekemässä. Kyseistä aikalaskuria käytetään muun muassa työmatkojen raportointiin. Tällaisia työmatkoja ovat esimerkiksi tavaranhaku tukusta tai avainten haku isännöitsijältä. Ajanottoon täytyy siis aina voida liittää työtehtävä ja mahdolliset matkakilometrit, jotka on ajettu työtehtävän aikana. Käyttäjän täytyy myös voida merkitä, onko tehty työtehtävä tai -matka laskutettavaa työtä vai ei. Viimeisimmän vaatimuksen mukaan aikalaskurin työtehtäviä täytyy myös voida jälkeenpäin osittain muokata.

7.8.4 Liittymä varastonhallintaan

Koska asentajien on tiedettävä, onko heillä tarpeeksi tavaraa mukana asennustyönantojen tekemiseen, täytyy heidän pitää kirjaa autossaan olevista tuotteista ja varaosista. Tätä on siis ideoitu niin, että keskusvarastojen lisäksi asentajien

pakettiautot ovat myös varastoja. Näin ollen he voivat hakea keskusvarastolta tavaraa ja tehdä Tavaran siirto -toiminnon lähteenä valittu keskusvarasto ja kohteena oma auto. Näin ollen järjestelmän tietokanta pysyy ajan tasalla tavaramääristä. Tämä myös toimii siinä tilanteessa, jos asentajien täytyy vaihtaa keskenään tavaraa. Siinä vaiheessa, kun tietty tuote tai varaosa asennetaan, poistuu se varastosta. Koko Varastonhallinta-toiminto on Innomobiilin näkökulmasta vasta ideapohjalla, mutta tietokantaa on jo suunniteltu varastonhallinnan osalta.

7.8.5 Työnannon tekemisen aikana tulevat asiakkaiden yhteydenotot

31.3.2010 - 1.4.2010 käydyssä palaverissa lisättiin vaatimus, jonka mukaan Innomobiililla täytyy voida merkitä ylös asentajalle esimerkiksi asennustyönannon aikana tullut asiakkaan yhteydenotto. Tätä raporttia kirjoittaessani en ole vielä ehtinyt ideoita kyseistä toimintoa sen tarkemmin, mutta todennäköisesti se vaatii joko jonkinlaista liittymää Innonetin töidenhallintaan, tai sitten Innomobiiliin vain yksinkertaisesti kirjoitetaan ylös yhteydenottajan tiedot. Joka tapauksessa kyseinen toiminto tarkentuu tulevaisuudessa.

8 YHTEENVETO JA POHDINTA

Tavoitteeni opinnäytetyön suhteen oli ottaa sellainen aihe, joka tarjoaa tarpeeksi haasteita ja uusia tekniikoita opiskeltavaksi, joten tässä suhteessa annettu tehtävä on ollut hyvä. Innomobiili on ollut haastava projekti. Muiden tekemän ohjelman ymmärtäminen on aina vaikeaa vaikka asiallinen dokumentaatio olisi saatavilla. Tässä projektissa kunnollinen dokumentaatio kuitenkin puuttui melkein täysin, joten projektin käynnistyminen täyteen vauhtiin vei runsaasti aikaa. Myös mobiiliohjelmointi Windows-ympäristössä oli minulle täysin uusi asia, joten siihen tutustuminen ja sen opiskelu toi mukanaan omat haasteensa. Järjestelmän vaatimuksiin tehdyt muutokset ovat monimutkaistaneet asioita välillä paljonkin.

Tätä raporttia kirjoittaessani opinnäytetyöni varsinaisen työn osuuden tilanne on seuraava: Toiminnallinen määrittely on melko pahasti keskeneräinen käyttötapausten osalta. Vaatimukset ovat toistaiseksi saatu kirjattua ylös, mutta ne täytyy osittain suunnitella sekä dokumentoida tarpeeksi tarkasti ja standardeja noudattaen. Tietokannan tietosisältö ja toiminta täytyy suunnitella loppuun uusien toiminnallisten vaatimusten pohjalta ja dokumentoida. Kun tämä kaikki on huolellisesti tehty, voi kuka tahansa toteuttaa Innomobiili-sovelluksen yksinomaan sen toiminnallisen määrittelyn pohjalta. Arvioin karkeasti Innomobiili-projektiin vielä vaadittavat työtunnit ja summa on tällä hetkellä yli 1000 työtuntia.

Vaikka opinnäytetyöni puitteissa ei päästykään alkuperäiseen tavoitteeseen, sain silti käännettyä Innomobiili-projektin oikeaan suuntaan. Toisaalta voisi ajatella, että toiminnallisen määrittelyn ensimmäisen version kirjoittaminen oli melko lailla hukkaan mennyttä aikaa, mutta toisaalta taas ilman sitä emme olisi koskaan päässeet asiakkaan kanssa siitä konkreettisesti keskustelemaan. Virhearvion tein kuitenkin alussa Innomobiilin prototyypin suhteen, jonka perusteella ensimmäisen määrittelydokumentin tein. Luulin, että kyseisen prototyypin toiminnot olisivat olleet siihen asti toimivia ja tarkkaan mietittyjä ja että uudet ominaisuudet tulisivat vain vanhojen jatkoksi. Myös asiakkaan myönteinen suhtautuminen ensimmäisissä asiakaspalavereissa katselmoituihin Innomobiili-prototyypin näyttöihin vahvisti omia luulojani. Toisin kuitenkin kävi ja loppujen lopuksi uusien ja osittain vaihtuvien toiminnallisten vaatimusten takia melkein koko Innomobiili-sovellus joudutaan tekemään alusta alkaen uusiksi. Prototyyppisovelluksesta voidaan käyttää tiettyjä osia ja osatoimintoja uutta sovellusta ohjelmoitaessa, mutta muuten prototyyppisovellus joudutaan hylkäämään. Kuitenkin sitä ennen täytyy toiminnallinen määrittely saada kaikkien Innomobiili-sovelluksen toimintojen suhteen valmiiksi.

Yhteistyö Innonet-projektin tekijöiden, Niko Rissasen ja Timo Paajasen, kanssa on ollut todella helppoa. Varsinkin opinnäytetöidemme loppupuolella olemme tehneet todella tiivistä yhteistyötä projektiemme kesken. Osittain molemmat projektit sitä vaativatkin, koska ne liittyvät kuitenkin niin tiiviisti toisiinsa. Olemme myös päätelleet, että jos Innomobiililla ja Innonetillä ei olisi alusta alkaen ollut yhtäaikaisia tekijöitä, olisi järjestelmien kehittäminen ollut todella hankalaa. Var-

sinkin tietokannan suunnittelu ei luultavasti olisi onnistunut kovinkaan hyvin, koska molemmat järjestelmät asettavat omat vaatimukset sen tietosisältöön ja toimintaan. Näin ollen mikäli tietokanta olisi suunniteltu vain toisen järjestelmän näkökulmasta, olisi se melko varmasti täytynyt ainakin osittain suunnitella uudestaan siinä vaiheessa, kun taas toisen järjestelmän kehittäminen olisi jossain vaiheessa alkanut.

Paljon on vielä työtä edessä ennen kuin Innomobiili on täysin toimiva. Tällä hetkellä olisi kuitenkin tärkeää, että niin Innomobiili- kuin Innonet-projektissa tähän asti olleet henkilöt jatkaisivat projektien tekoa opinnäytetöiden jälkeenkin. Minulla, Niko Rissasella ja Timo Paajasella on toistaiseksi vielä paras käsitys tämän hetkisistä projektien tilanteista, joten uusien projektihenkilöiden mukaan tulon kannalta olisi hyvä, että olisimme kaikki heitä Innomobiiliin ja Innonetiin perehdyttämässä sekä tietenkin jatkamassa työtä niin pitkälle kuin mahdollista.

KUVAT

KUVIOT

TAULUKOT

LÄHTEET

Ahtee, T. 2007. Esitutkimus-dokumentti.
http://www.cs.tut.fi/~projekti/dokumentit/esit-sisalto.txt (Luettu 7.5.2010)

Ahtee, T. 2009. Projektisuunnitelma.
http://www.cs.tut.fi/~projekti/dokumentit/proj-sisalto.txt (Luettu 7.5.2010)

Chinnathambi, K. 2007. Run/Emulate Windows Mobile 6 on your Computer. http://blog.kirupa.com/?p=70 (Luettu 28.5.2010)

Gröhn, J. 2003. Kuljetusten toimintolaskennan sovellukset ja toteutus. Liikenne ja viestintäviraston julkaisuja. Helsinki.
http://www.lvm.fi/fileserver/17_2003.pdf (Luettu 17.5.2010)

Hahtola, I. 2009. Projektityöskentely. Projektityöskentely-luennot-alkuosa.pdf

Haikala, I. 2009a. Toiminnallinen määrittely.
http://www.cs.tut.fi/ohj/dokumenttipohjat/pohjat/maarittely/sovellusohje_hytt_drmaarittely.pdf (Luettu 8.5.2010)

Haikala, I. 2009b. Tekninen määrittely.
http://www.cs.tut.fi/ohj/dokumenttipohjat/pohjat/suunnittelu/sovellusohje_hytt_drsuunnittelu.pdf (Luettu 8.5.2010)

Haikala, I. & Märijärvi, K. 2004. Ohjelmistotuotanto. 10. uud. painos. Helsinki: Talentum

Hariaharan, S. 2007. What is .NET? http://cnx.org/content/m14638/latest/ (Luettu 28.5.2010)

Ilkko, L. Oulun seudun ammattikorkeakoulu, Teema 4 Vaatimustenhallinta
http://www.students.oamk.fi/~s6matu00/sekalaista/ot/teema%204%20vaatimustenhallinta.ppt (Luettu 28.5.2010)

Innotek. 2000. Energo-säästöohjelma-esityskalvot.
http://www.innotek.fi/datapankki/esityskalvot1428kt.pdf (Luettu 7.5.2010)

Intel Corporation. Moore's Law.
http://www.intel.com/technology/mooreslaw/index.htm (Luettu 29.3.2010)

Jyväskylän yliopisto. Tietojenkäsittelytieteiden laitoksen projektiopinnot - projektipäällikön tehtävät.
http://projekti.it.jyu.fi/suoritustavat/TJTS431_Projektin_johtaminen/opiskelijat/projektipaallikon_ja_sihteerin_tehtavat/projektipaallikon_tehtavat/ (Luettu 9.5.2010)

Koulutus- ja konsultointipalvelu KK Mediat. 2010. Johdatus SQL:n maailmaan. http://www.2kmediat.com/sql/alkeet.asp (Luettu 18.5.2010)

Lehtiniitty, M. 2008a. Nokia hyökkää Mail on Ovi -sähköpostillaan. http://www.puhelinvertailu.com/uutiset.cfm/2008/12/23/nokia_hyokkaa_mail_on_ovi_sahkopostillaan (Luettu 28.5.2008)

Lehtiniitty, M. 2008b. LG:ltä GD910-rannekellopuhelin. http://www.puhelinvertailu.com/uutiset.cfm/2008/12/29/lg_lta_gd910_rannekellopuhelin (Luettu 28.5.2010)
Microsoft. 2007. C# Language Specification Version 3.0.

Microsoft. 2010a. Windows Mobile. http://www.microsoft.com/mobile/buyersguide/educateme/default.asp (Luettu 29.3.2010)

Microsoft. 2010b. Selvitä, onko laite Smartphone- vai Pocket PC -puhelin. http://www.microsoft.com/finland/windowsmobile/help/OSversions.mspx (Luettu 29.3.2010)

Microsoft. 2010c. Differences Between the .NET Compact Framework and the .NET Framework. http://msdn.microsoft.com/fi-fi/library/2weec7k5(en-us).aspx (Luettu 30.3.2010)

Microsoft. 2010d. Synching your mobile phone and PC using ActiveSync and Windows Mobile Device Center. http://www.microsoft.com/windowsmobile/en-us/help/synchronize/device-synch.mspx (Luettu 26.4.2010)

Mikkonen, T. 2004. Mobiiliohjelmointi. 2. uud. painos. Helsinki: Talentum

Motiva. 2009. Vedenkulutus http://www.motiva.fi/koti_ja_asuminen/mihin_energiaa_kuluu/vedenkulutus (Luettu 13.5.2010)

Refsnes Data. 2010. Introduction to XML. http://www.w3schools.com/xml/xml_whatis.asp (Luettu 26.4.2010)

Sainio, L. 2009. Ohjelmistotestauksen menetelmät ja työvälineet. Saimaan ammattikorkeakoulu. Lukumateriaali.

Satakunnan ammattikorkeakoulu. Relaatiotietokannan suunnittelu. http://www.tp.spt.fi/~salabra/ha/Relaatiotietokannat/Kasiteanalyys.html (luettu 17.5.2010)

Seidler, K. 2009. XAMPP. http://www.apachefriends.org/en/xampp.html (Luettu 26.4.2010)

Sparx Systems. 2010. UML Tutorial. http://www.sparxsystems.com/uml-tutorial.html (Luettu 3.4.2010)

Söderholm, J. Digikamerat kalastuskäyttöön. http://www.heittokalastus.info/KAUPPA/26 (Luettu 28.5.2010)

The PHP Group. 2010.
http://fi.php.net/manual/en/intro-whatis.php (Luettu 30.3.2010)

Turun yliopisto. 2007. UML-Mallinnus: lyhyt oppimäärä.
http://staff.cs.utu.fi/kurssit/ohjelmistotuotanto/kevat07/UML-mallinnus.pdf (Luettu 3.4.2010)