SAIMAAN AMMATTIKORKEAKOULU
Tekniikka Lappeenranta
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Niko Rissanen

INNONET-TOIMINNANOHJAUSJÄRJESTELMÄ

Opinnäytetyö 2010

TIIVISTELMÄ
Niko Rissanen
Innonet-toiminnanohjausjärjestelmä, 64 sivua
Saimaan ammattikorkeakoulu, Lappeenranta
Tekniikka, tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö, 2010
Ohjaajat:Toimitusjohtaja Markus Heikkinen, Innotek Oy
Lehtori Martti Ylä-Jussila, Saimaan ammattikorkeakoulu Oy

Tämän opinnäytetyön tavoitteena on ollut Innonet-toiminnanohjausjärjestelmän jatkokehittäminen. Innonet on Innotek Oy:n Energo-ohjelman toiminnanohjauksen automatisointiin tarkoitettu järjestelmä. Järjestelmän tarkoitus on tuoda säästöjä työaika- ja materiaalikustannuksiin sekä tehostaa Energo-ohjelman toimintaa. Energo on Innotek Oy:n kehittämä ohjelma vesi- ja energiakulutuksen pienentämiseen kiinteistöissä.

Opinnäytetyön tavoitteet kuitenkin muuttuivat projektin edetessä ja uusiksi tavoitteiksi muodostui Innonet-toiminnanohjausjärjestelmän toteutuksen määrittäminen ja uuden tietokannan määrittely, suunnittelu ja toteutus.

Työssä on noudatettu ohjelmistotuotannossa yleisesti käytettyä vesiputousmallia. Työ on keskittynyt vesiputousmallin määrittelyvaiheeseen. Määrittelyvaihe tuottaa dokumenttina toiminnallisen määrittelyn.

Työn tuloksena on tuotettu toiminnallinen määrittelydokumentti Innonetin toteutuksesta ja toteutettu uusi tietokanta Innonetille.

Asiasanat: Innonet, toiminnanohjausjärjestelmä, CodeIgniter, Energo-kartoitus, toiminnallinen määrittely, Innotek Oy

ABSTRACT
Niko Rissanen
Innonet Enterprise resource planning, 64 pages
Saimaa University of Applied Sciences, Lappeenranta
Technology, Degree Programme in Information Technology
Software Engineering
Bachelor's Thesis, 2010
Instructors: CEO Markus Heikkinen, Innotek Oy
Lecturer Martti Ylä-Jussila, Saimaa University of Applied Sciences

The purpose of this thesis was to continue the development of Innonet enterprise resource planning system. The idea of Innonet system is to enhance the Energo program and reduce work and material costs of the program. The aim of the Energo program is to save water and energy consumption of buildings. The Energo program is developed by Innotek Oy.

The objectives of the thesis changed during the project and the new task was to design the functional specifications of Innonet ERP and create a new database for the system.

The project used a software development process known as waterfall model. The project focused on the requirement analysis phase of the waterfall model. The output of this phase is the functional specifications.

The functional specifications of the Innonet ERP have been made as the outcome of the project and the new database has been established as well.

Keywords: Innonet, Enterprise resource planning, CodeIgniter, Energo water flow analysis, functional specification, Innotek Oy

TERMIT JA LYHENTEET

AC	AC (Auto-clean) on itsestään puhdistuva poresuutin.
ACID	ACID (Atomicity, Consistency, Isolation, Durability) on joukko sääntöjä, joiden avulla turvataan tietokannan tietojen eheys kaikissa tilanteissa.
ANSI	ANSI (American National Standards Institute) on yhdysvaltalainen yksityinen organisaatio, joka valvoo standardeja Yhdysvalloissa.
ERP	ERP (Enterprise Resource Planning) eli toiminnanohjausjärjestelmä on yrityksen liiketoimintaa ohjaava järjestelmä.
HTML	HTML (Hypertext Markup Language) on avoimesti standardoitu kuvauskieli WWW-sivujen luontia varten.
InnoDB	InnoDB on Innobase Oy:n kehittämä tietokantamoottori MySQL-tietokannan hallintajärjestelmälle.
ISO	ISO (International Organization for Standardization) on kansainvälinen standardisoimisjärjestö.
MDA	MDA (Model-driven architecture) on OMG:n kehittämä mallipohjaiseen suunnitteluun perustuva menetelmä.
MVC	MVC (Model–View–Controller) on Trygve Reenskaugin kehittämä ohjelmistokehitysmalli.
MyISAM	MyISAM on MySQL-tietotietokannan hallintajärjestelmän oletus tietokantamoottori.
MySQL	MySQL on avoimeen lähdekoodiin perustuva SQL-tietokannan hallintajärjestelmä.
OMG	OMG (Object Management Group) on kansainvälinen organisaatio, joka kehittää ja määrittelee oliohajautukseen liittyviä standardeja.
PCR	PRC (Pressure compensating aerator) eli painevakioitu poresuutin, pyrkii säilyttämään vakion veden virtauksen painevaihteluista riippumatta.
PHP	PHP (PHP: Hypertext Preprocessor) on palvelinpohjainen ohjelmointikieli, jota käytetään erityisesti WWW-palvelinympäristöissä dynaamisten WWW-sivujen luonnissa.

SQL	SQL (Structured Query Language) on standardoitu kieli relaatiotietokantojen käsittelyyn ja määrittelyyn.
UML	UML (Unified Modelling Language) on ohjelmistojen analyysiin ja suunnitteluun tarkoitettu standardoitu kuvauskieli.
XAMPP	XAMPP on ilmainen ja avoimeen lähdekoodiin perustuva alustariippumaton WWW-palvelinpaketti, joka sisältää Apache HTTP-palvelimen, MySQL-palvelimen, tuen PHP:lle ja Perlille.

SISÄLTÖ

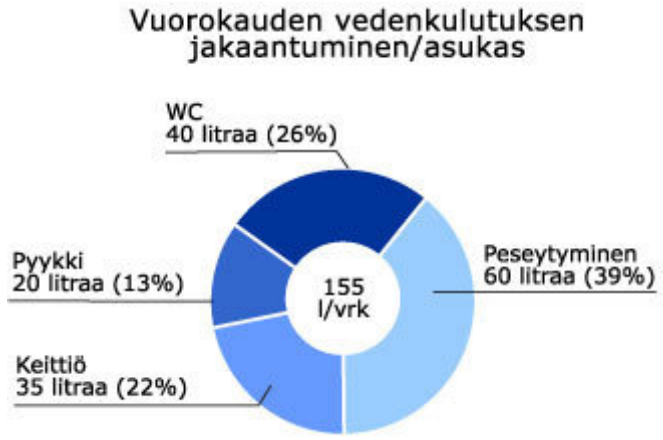
TIIVISTELMÄ
ABSTRACT
TERMIT JA LYHENTEET
1 JOHDANTO 8
2 ASIAKAS 10
2.1 Innotek Oy 10
2.2 Energo-ohjelma 10
2.3 Energo-tuotteet 11
2.3.1 VuotoPois 11
2.3.2 Vetovahti 12
2.3.3 Säästävä käsisuihku 12
2.3.4 PCR-säädin 12
2.3.5 Säästöporesuuttimet 13
2.3.6 Laaja kuntoraportointi 13
3 TOIMINNANOHJAUSJÄRJESTELMÄ 14
4 OHJELMISTOTUOTANTO 15
4.1 Ohjelmiston elinkaari 15
4.2 Vesiputousmalli 16
4.3 Inkrementaaliset mallit 19
4.4 Ketterät mallit 21
5 TYÖSSÄ KÄYTETYT MÄÄRITTELY- JA SUUNNITTELUMENETELMÄT 23
5.1 Käsiteanalyysi 23
5.2 Käyttötapausanalyysi 25
5.3 UML 27
5.3.1 Käyttötapauskaavio 28
5.3.2 Luokkakaavio 30
5.3.3 Oliokaavio 32
5.3.4 Sekvenssikaavio 33
5.3.5 Kommunikointikaavio 35
5.3.6 Pakkauskaavio 35
5.3.7 Tilakaavio 37
5.3.8 Aktiviteettikaavio 37
5.3.9 Komponenttikaavio 39
5.3.10 Sijoittelukaavio 39
5.3.11 Ajoituskaavio 40

5.3.12 Koostekaavio 41
5.3.13 Kokoava vuorovaikutuskaavio 41
5.3.14 Profiilit 43
6 TYÖSSÄ KÄYTETTÄVÄT TEKNIIKAT 43
6.1 PHP 44
6.2 Relaatiotietokannat ja SQL 45
6.2.1 MySQL 45
6.2.2 MyISAM 46
6.2.3 InnoDB 46
6.3 PHP-ohjelmistokehykset 46
6.3.1 CodeIgniter 47
6.4 StarUML 47
7 PROJEKTIN KULKU 49
7.1 Projektin organisaatio 49
7.2 Esitutkimus ja alustus 50
7.3 Vanhan Innonet-järjestelmän opiskelu ja dokumentointi 50
7.4 Innonet-järjestelmän uudelleen määrittely 51
8 UUDEN INNONET-JÄRJESTELMÄN MÄÄRITTELYN ESITTELY 52
8.1 Kirjautuminen 53
8.2 Asiakkaiden hallinta 54
8.3 Kiinteistöjen hallinta 55
8.4 Tilauksien hallinta 56
8.5 Töiden hallinta 56
8.6 Raporttien hallinta 57
8.7 Laskujen hallinta 57
8.8 Tuotteiden hallinta 57
8.9 Varaston hallinta 57
8.10 Ylläpito 58
8.11 Tietokanta 58
9 YHTEENVETO 59
KUVAT 60
TAULUKOT 61
LÄHTEET 62

1 JOHDANTO

Puhdas makea vesi on muuttumassa maailmassa ylellisyydeksi. Sitä voi kuitenkin olla hieman vaikeaa huomata maassa, jonka vesivaroista käytössä on kaksi prosenttia. (Maailma.net.)

Jokainen suomalainen kuluttaa keskimäärin 155 litraa vettä vuorokaudessa. Tästä vedestä lämpimän veden osuus on noin 45 litraa. Kuvasta 1.1 selviää tarkempi vedenkulutuksen jakaantuminen. Veden lämmittämiseen kuluu noin 20 % koko asuinrakennuksen energiakulutuksesta. (Motiva, 2009.)



Kuva 1.1 Vuorokauden vedenkulutuksen jakaantuminen (Innotek Oy, 2000a)

Innotek Oy:n Energo-ohjelman tarkoitus on säästää vettä, energiaa ja niihin liittyviä kustannuksia. Ohjelmalla saavutetaan vedenkulutuksessa 25–30 prosentin säästö. (Innotek Oy, 2000b.)

Tämän opinnäytetyön tavoitteena on jatkokehittää Innotek Oy:n Energo-ohjelman toiminnanohjausjärjestelmää, Innonetiä. Innonet on toteutettu PHP:llä ja sitä käytetään Internet-selaimella. Innonetin kehittäminen on aloitettu vuonna 2004. Tässä opinnäytetyössä järjestelmän kehitystä jatketaan siitä, mihin se on jäänyt.

Opinnäytetyön tavoitteet kuitenkin muuttuvat radikaalisti projektin edetessä ja lopulta uudeksi tavoitteeksi muodostuu Innonetin määrittelydokumentaation

tuottaminen ja uuden tietokannan määrittely, suunnittelu ja toteutus.

Tarve Innonet-toiminnanohjausjärjestelmälle syntyi, kun haluttiin automatisoida Energo-ohjelman toiminnanohjaus. Innotek Oy:llä ei ole tällä hetkellä vastaavaa järjestelmää käytössä ja toiminnanohjaus hoidetaan Excel-taulukoilla ja paperilomakkeilla. Innonetin tarkoitus on helpottaa vedenkulutuksen kartoitusta, asennusten raportointia sekä laajaa kuntoraportointia. Järjestelmän avulla paperilomakkeiden määrää saadaan vähennettyä ja tiedon useaan kertaan kirjoittaminen vähenee. Tämä tuo säästöjä työaika- ja materiaalikustannuksiin ja tehostaa Energo-ohjelman toimintaa.

Tulevaisuudessa asentajat käyttävät Innonetiä kämmenmikroilla. Kämmenmikroille toteutetaan oma järjestelmänsä, Innomobiili. Innomobiili toteutetaan omana opinnäytetyönään ja sitä ei käsitellä tässä opinnäytetyössä.

2 ASIAKAS

2.1 Innotek Oy

Opinnäytetyön asiakkaana on Innotek Oy. Innotek Oy on lieksalainen yritys, joka erikoistunut LVI-alan palveluihin. Yrityksen edustajana ja opinnäytetyön ohjaajana toimii Markus Heikkinen.

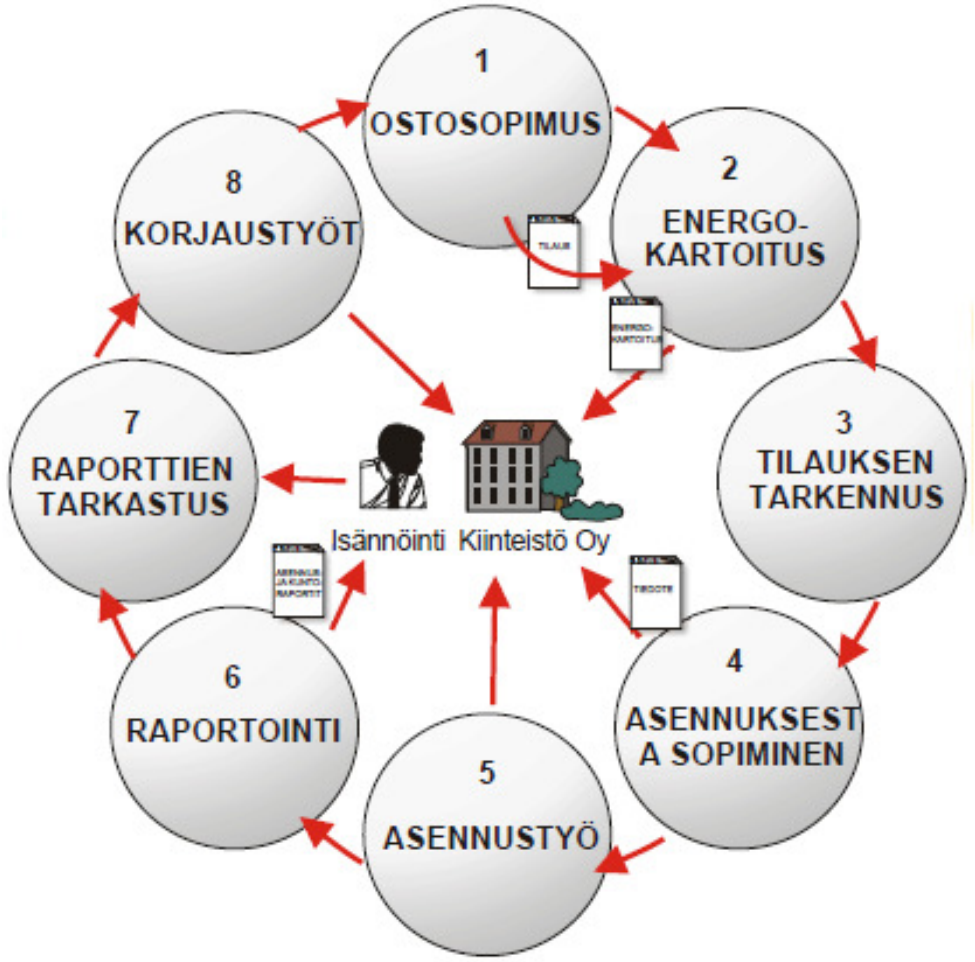
Innotek Oy:n pääkonttori sijaitsee Lieksassa, muut yrityksen toimipisteet sijaitsevat Helsingissä ja Kempeleellä. Yrityksen varastot sijaitsevat Lieksassa, Keravalla ja Kempeleellä. Innotek Oy työllistää 10 henkilöä ja sen vuosittainen liikevaihto on noin 1,2 miljoonaa euroa. Innotek Oy:n laskutus on yrityksen sisäinen ja kirjanpito on ulkoistettu Tilitoimisto Optimus Oy:lle.

2.2 Energo-ohjelma

Innotek Oy toi vuonna 2000 markkinoille Energo-ohjelman, jonka tarkoitus on vähentää vedenkulutusta kiinteistöissä. Energo-ohjelma jakautuu eri vaiheisiin oheisen kuvan (kuva 2.1) mukaisesti.

Ensimmäisessä vaiheessa kiinteistön edustaja tekee tilauksen Energo-kartoituksesta. Tilauksen pohjalta Energo-asentajat suorittavat kiinteistössä Energo-kartoituksen (vaihe 2). Kartoituksella tarkoitetaan käytännössä huoneistossa olevien hanojen ja suihkujen virtausmittausten tekemistä ja wc-istuimen huoltotarpeen selvittämistä. Tarvittaessa mitataan myös vuorokauden kokonaiskulutus huoneistossa. Kartoituksen yhteydessä tarkastetaan myös kiinteistön päävesimittari. Asennustilaus tarkentuu Energo-kartoituksesta saatujen tulosten perusteella (vaihe 3). Kartoituksen perusteella valitaan toimenpiteet, joilla saavutetaan haluttu säästötaso. Neljännessä vaiheessa sovitaan asennusaika asiakkaan kanssa ja tiedotetaan asennuksista kohdekiinteistöyhtiössä. Viides vaihe on itse asennustyö. Energo-asentajat tekevät sovitut asennus- ja huoltotyöt kohteissa. Asennuksen yhteydessä tarkastetaan myös asunnon kosteat tilat ja havaituista vioista tehdään kuntoraportti asiakkaalle. Asennus- ja kuntoraportit

toimitetaan laskun mukana kiinteistön isännöitsijälle (vaihe 6). Isännöitsijä tarkastaa kuntoraportit ja tilaa tarvittaessa lisäkorjaustyöt (vaihe 7). Ja viimeisenä vaiheena suoritetaan mahdolliset lisäkorjaustyöt (vaihe 8). (Innotek Oy, 2000a.)



Kuva 2.1 Energo-ohjelman vaiheet (Innotek Oy, 2000a)

2.3 Energo-tuotteet

Energo-ohjelma sisältää monia erilaisia veden kulutusta vähentäviä tuotteita ja palveluita. Alla on käyty nämä lyhyesti läpi.

2.3.1 VuotoPois

VuotoPois on huoltopalvelu, joka voidaan suorittaa WC-istuimiin. VuotoPois-

huoltopalvelun tarkoituksena on eliminoida WC-istuimessa olevat huomaamattomat vesivuodot. VuotoPois-huoltopalveluita on kahta eri tyyppiä, VuotoPois-kokohuolto ja VuotoPois-puolihuolto. Näistä ensimmäinen on tarkoitettu yli 10 vuotta vanhoihin WC-istuimiin, ja sitä suositellaan silloin, kun halutaan minimoida vuotoriskit ja varmistaa WC-istuimen ongelmaton käyttö vuosiksi eteenpäin. VuotoPois-kokohuollossa uusitaan kaikki kuluvat tiivisteet tulo- ja poistoventtiileistä ja puhdistetaan tiivistepinnat. Jälkimmäinen huoltopalvelu on VuotoPois-puolihuolto, joka on tarkoitettu alle 10 vuotta vanhoihin WC-istuimiin. VuotoPois-puolihuollossa huolletaan WC-istuimen poistoventtiili ja istukkapinta. (Innotek Oy, 2000a.)

2.3.2 Vetovahti

Vetovahti on WC-istuimeen jälkiasennettava säästölaite, jossa on kaksi eri huuhtelua erikokoisia tarpeita varten. Vetovahdin teho perustuu korkeampaan huuhtelupaineeseen. Huuhtelupainetta lisäämällä WC-istuimen kulho huuhtoutuu hygieenisesti jo puolta pienemmällä vesimäärällä. (Innotek Oy, 2000a.)

2.3.3 Säästävä käsisuihku

Säästävän käsisuihkun rungossa oleva vakiovirtausventtiili tasaa veden virtaaman halutulle tasolle. Tällöin käsisuihku kuluttaa normaalia vähemmän vettä. Virtausvaihtoehdot ovat 6, 8, 9, 10, 10.5 ja 12 l/min. (Innotek Oy, 2000a.)

2.3.4 PCR-säädin

PCR (painevakioitu)-säädin asennetaan hanan yhteyteen niissä käsisuihkumalleissa, joissa itse suihkukahvaa ei voida vaihtaa. PCR-säätimessä oleva vakiovirtausventtiili tasaa veden virtaaman. PCR-säätimiä on saatavissa samoilla virtausvaihtoehdoilla kuin käsisuihkujakin. (Innotek Oy, 2000a.)

2.3.5 Säästöporesuuttimet

Energo-ohjelmassa käytettäviä säästöporesuuttimia on useita eri malleja. Säästömalli E on perusmalli, joka tasaa vedenvirtaaman 7,5 l/min. (Innotek Oy, 2000a.)

Säästömalli AC (itsestään puhdistuva) on tarkoitettu kohteisiin, joissa kalkin muodostus on voimakasta. PCR AC-suuttimia on saatavilla 6 l ja 8 l versioina. (Innotek Oy, 2000a.)

PRC Laminar -suutin poistaa ilman vedestä ja on tarkoitettu terveydenhuollon kohteisiin, joissa halutaan ehkäistä ilmakulkuisten bakteerien päätymistä virtaukseen (Innotek Oy, 2000a).

PCR Spray -suutin on tarkoitettu kohteisiin, joissa käytetään matalaa virtausta. PCR Spray-suutin soveltuu erityisesti julkisiin tiloihin, joissa halutaan säästää tehokkaasti turhaa veden kulutusta. (Innotek Oy, 2000a.)

Säästösuuttimia on saatavana myös ilkivaltasuojattuina. Ilkivaltasuojatut (Vandal resistant) mallit on varustettu pyörivällä holkilla, jonka avaaminen onnistuu vain erikoisavaimella. (Innotek Oy, 2000a.)

2.3.6 Laaja kuntoraportointi

Laaja kuntoraportointi on erillinen palvelu, jonka kiinteistön edustaja voi tilata suoritettavaksi. Siinä Energo-asentajat kartoittavat kohteen pinnat ja kalusteet kosteiden tilojen lisäksi. Laajasta kuntoraportoinnista asiakkaalle toimitetaan laaja kuntoraportti. (Innotek Oy, 2000a.)

3 TOIMINNANOHJAUSJÄRJESTELMÄ

Nykyisessä yhteiskunnassa tiedon hallinnan, jalostuksen ja hyödyntämisen tehokkuus määrää pitkälle yrityksen menestymisen. Tietojärjestelmien kehittyessä ja yrityksen tarpeiden muuttuessa yritykset ovat vähitellen ottaneet käyttöön yhä laajempia tietojärjestelmäkokonaisuuksia. Kehitys on keskittynyt järjestelmien, tietokantojen ja sovellusten kasvavaan integraatioon. Toiminnanohjausjärjestelmät edustavat omalta osaltaan tällaisia kokonaisratkaisuja. (Kettunen & Simons, 2001.)

Toiminnanohjausjärjestelmällä tarkoitetaan yrityksen tietojärjestelmää, jolla ohjataan ja valvotaan yrityksen toimintaa, liiketoimintaprosesseja ja sen käytössä olevia resursseja. Toiminnanohjausjärjestelmistä käytetään myös nimitystä ERP-järjestelmä (Enterprise Resource Planning). ERP-järjestelmä muodostuu erilaisten modulaaristen sovellusten yhteisestä kokonaisuudesta. Järjestelmällä integroidaan yrityksen eri toimintoja, ja siihen voi sisältyä monia eri sovelluksia kuten esimerkiksi palkanlaskenta, myynti, tilaukset, kirjanpito, ostoreskontra, myyntireskontra, varastonhallinta, tuotannonohjaus, projektien hallinta sekä erilaisia materiaalin hallintaan liittyviä sovelluksia. (TIEKE, 2008.)

Alun pitäen toiminnanohjausjärjestelmiä on kehitetty lähinnä suurten yritysten ja organisaatioiden tarpeisiin. Viimeaikainen kehitys ja kilpailun koveneminen on kuitenkin johtanut toiminnanohjausjärjestelmien käyttöönottoon myös pk-yrityksissä. Pk-yritykset ovat kuitenkin vaikeassa asemassa toiminnanohjausjärjestelmien käyttöönoton osalta. Suurille yrityksille suunnatut raskaat ja kalliit järjestelmät eivät sovellu parhaalla mahdollisella tavalla pk-yritysten tarpeisiin. Tämän lisäksi pk-yritysympäristö asettaa järjestelmille sekä teknisesti että toiminnallisesti erilaisia vaatimuksia kuin mihin suuremmissa ja jo perusvalmiuksiltaan kehittyneemmissä organisaatioissa törmätään. (Kettunen & Simons, 2001.)

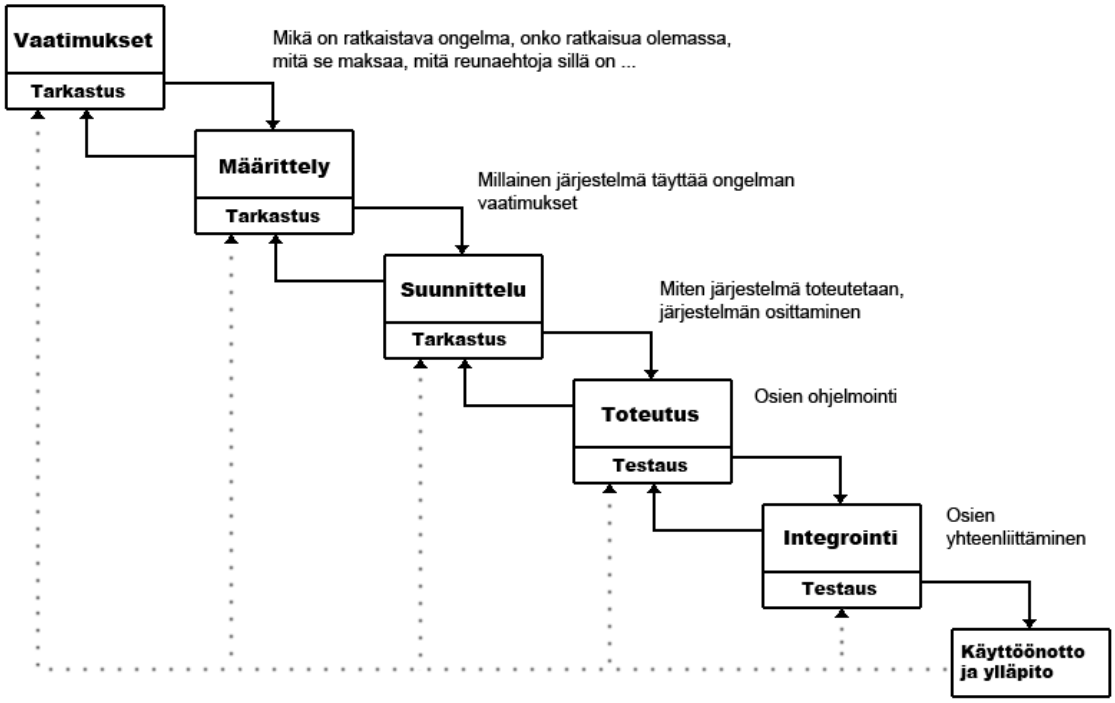
4 OHJELMISTOTUOTANTO

Ohjelmistotuotanto on ohjelmistotyötä, jonka tuloksena syntyvät järjestelmät täyttävät käyttäjiensä kohtuulliset toiveet ja odotukset ja tämän lisäksi valmistuvat laadittujen aikataulujen ja kustannusarvioiden puitteissa. Ohjelmistotuotanto käsittää kaikki ohjelmistotyöhön liittyvät osa-alueet, joita ovat määrittely, suunnittelu, toteutus, testaus, käyttöönotto ja ylläpito. Tämän lisäksi ohjelmistoprojekteihin liittyy koko projektin ja ohjelman elinkaaren ajan kestäviä tukitoimintoja. Tärkeimmät tukitoiminnot koostuvat laadunvarmistuksesta, tuotteenhallinnasta ja dokumentoinnista. (Immonen, 2003; Haikala & Märijärvi, 2004.)

4.1 Ohjelmiston elinkaari

Ohjelmiston elinkaari on se aika, joka kuluu ohjelmiston kehittämisen aloittamisesta sen poistamiseen käytöstä. Elinkaarimallien avulla ohjelmistotyön ongelmia voidaan tarkastella systemaattisesti. Useista olemassa olevista elinkaarimalleista yleisin on myös tässä projektissa käytetty vesiputousmalli (kuva 4.1). (Immonen, 2003; Haikala & Märijärvi, 2004.)

Kaikkiin elinkaarimallin vaiheisiin liittyy laadunvarmistustoimenpiteitä, kuten tarkastuksia, katselmointia ja testausta. Tarkastuksilla ja testauksella on tarkoitus kitkeä virheet järjestelmästä mahdollisimman aikaisessa vaiheessa. Katselmointitilaisuuksia pidetään yleensä vaiheiden päätteeksi. Niissä todetaan projektin tilanne ja se, että kaikki vaiheeseen liittyvät tavoitteet on saavutettu ja kaikki sovitut dokumentit tuotettu. (Haikala & Märijärvi, 2004.)



Kuva 4.1 Tyypillinen vesiputousmalli (Haikala & Märijärvi, 2004)

4.2 Vesiputousmalli

Vesiputousmallia pidetään ensimmäisenä varsinaisena prosessimallina. Se jakaa prosessin lineaarisiin vaiheisiin. Edellisen vaiheen tulos on seuraavan vaiheen syöte. Yleensä vesiputousmalli sisältää ainakin kuvan 4.1 mukaiset vaiheet. Vaiheen lopussa sen tuotos tarkastetaan ja tarkastuksen läpäistyään jäädytetään. Jäädytettyä dokumenttia ei saa muuttaa ilman projektin johtoryhmän lupaa. Vesiputousmalli perustuu hyvään dokumentaatioon ja tuotosten huolelliseen tarkastamiseen. (Immonen, 2003; Luukkainen & Laine, 2009.)

Esitutkimuksessa kartoitetaan yleisiä järjestelmälle asetettuja vaatimuksia ja niiden toteuttamismahdollisuuksia. Esitutkimus perustuu asiakasvaatimuksiin, jotka kertovat, mitä järjestelmän tulisi tehdä tyydyttääkseen asiakkaan tarpeet. Esitutkimuksen perusteella syntyy päätös järjestelmän kehittämisen jatkamisesta tai koko projektin peruuttamisesta. Esitutkimus on yksi ohjelmiston elinkaaren tärkeimmistä vaiheista, koska väärinymmärrettyjen tai puutteellisten asiakasvaatimusten perusteella ei voida toteuttaa hyvää järjestelmää. Sen suurin on-

gelma on asiakkaan todellisten tarpeiden selvittäminen ja täydellinen ymmärtäminen. (Immonen, 2003; Haikala & Märijärvi, 2004.)

Määrittelyvaiheessa asiakasvaatimukset kootaan ja analysoidaan ja niistä johdetaan ohjelmistovaatimukset, jotka määrittelevät toteutettavan järjestelmän. Asiakasvaatimukset pyritään selvittämään mahdollisimman tarkasti ja yksiselitteisesti. Selvitystyö tapahtuu aina asiakkaan kanssa yhteistyönä käyttäen apuna muun muassa haastatteluja, palavereita ja aivoriihiä. Määrittelyprosessin tuloksena syntyvä dokumentti on toiminnallinen määrittely. Toiminnallinen määrittely sisältää ohjelmiston toimintojen kuvauksen, toteutukselle asetettavat ei-toiminnalliset vaatimukset sekä rajoitukset. Toimintojen yhteydessä määritellään ohjelmistolla toteutettavat ominaisuudet, käyttöliittymä ja kommunikointi muiden järjestelmien kanssa. Ei-toiminnallisia vaatimuksia ovat suoritusteho, vasteaika ja käytettävyys. Rajoituksia ovat muun muassa toteutus tietyllä ohjelmointikielellä sekä käytettävissä oleva levytila. Toiminnallisessa määrittelyssä ei oteta kantaa järjestelmän teknisiin ratkaisuihin. Kuvassa 4.2 on esimerkki toiminnallisen määrittelyn sisältörungosta. (Immonen, 2003; Haikala & Märijärvi, 2004.)

1. Johdanto
    - 1.1 Tarkoitus
    - 1.2 Tuote
    - 1.3 Määritelmät, termit ja lyhenteet
    - 1.4 Viitteet, muut tähän liittyvät dokumentit
    - 1.5 Yleiskatsaus dokumentteihin
2. Yleiskuvaus
    - 2.1 Ympäristö
    - 2.2 Toiminta
    - 2.3 Käyttäjät
    - 2.4 Yleiset rajoitteet
    - 2.5 Oletukset ja riippuvuudet
3. Tiedot ja tietokanta
4. Toiminnot
    - 4.n Toiminnon kuvaus
    - ...
5. Ulkoiset liittymät
    - 5.1 Käyttöliittymä
    - 5.2 Laitteistoliittymät
    - 5.3 Ohjelmistoliittymät
    - 5.4 Tietoliikenneliittymät
6. Muut ominaisuudet
    - 6.1 Suorituskyky
    - 6.2 Käytettävyys, toipuminen, turvallisuus ja suojaukset
    - 6.3 Ylläpidettävyys
    - 6.4 Siirrettävyys, yhteensopivuus
    - 6.5 Operointi
7. Suunnittelurajoitteet
    - 7.1 Standardit
    - 7.2 Laitteistorajoitteet
    - 7.3 Ohjelmistorajoitteet
    - 7.4 Muut rajoitteet

Kuva 4.2 Toiminnallisen määrittelyn sisältörunko (Haikala & Märijärvi, 2004)

Suunnitteluvaiheessa määrittelyssä kuvattujen toimintojen toteutus suunnitellaan. Suunnittelun tarkoituksena on kuvata järjestelmään liittyvät määritykset ja toiminnot teknillisellä kielellä. Suunnitteluvaihe jaetaan yleensä arkkitehtuurisuunnitteluun ja moduulisuunnitteluun. (Immonen, 2003; Haikala & Märijärvi, 2004.)

Arkkitehtuurisuunnittelussa järjestelmä jaetaan itsenäisiin osiin, moduuleihin. Samalla määritellään myös moduulien rajapinnat. Arkkitehtuurisuunnittelu tuottaa dokumenttina teknisen määrittelyn. Arkkitehtuurisuunnittelun jälkeen seuraa moduulisuunnittelu, siinä jokaisen moduulin sisäinen rakenne suunnitellaan. Moduulilla tarkoitetaan ohjelmasta erotettavissa olevaa loogista kokonaisuutta, joka sisältää tietomäärittelyitä ja joukon tietoa käsitteleviä funktioita. (Immonen, 2003; Haikala & Märijärvi, 2004.)

Toteutusvaihe koostuu ohjelman kirjoittamisesta teknisen määrittelyn mukaisesti, ohjelmadokumenttien tuottamisesta ja moduulitestauksen suorittamisesta. Ohjelmointivaihe sisältää moduulien sisäisten liitäntöjen ja tietorakenteiden suunnittelun, algoritmien valinnan ja suunnittelun, lähdekoodin kirjoittamisen ja lopulta ohjelman kääntämisen. (Immonen, 2003; Haikala & Märijärvi, 2004.)

Testauksella etsitään ohjelmistosta virheitä. Testaus jaetaan staattiseen ja dynaamiseen testaukseen. Staattisessa testauksessa ohjelmaa testataan suorittamatta sitä. Staattisen testauksen menetelmiä ovat katselmoinnit ja staattisen koodin analysoiminen. Staattisilla tekniikoilla voidaan jo varhaisessa vaiheessa löytää virheitä, jolloin niiden korjaaminen on vielä edullista. Dynaamisessa testauksessa testattavaa ohjelmaa suoritetaan ja selvitetään ohjelman ja ohjelmakoodin suorituksenaikaista käyttäytymistä. (Immonen, 2003; Haikala & Märijärvi, 2004.)

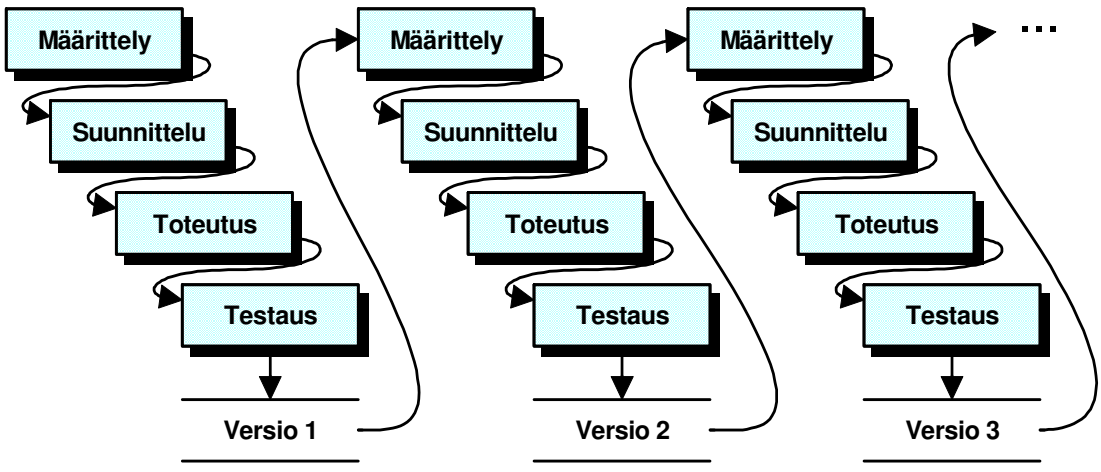
Kun ohjelman toteutus ja testaus on suoritettu loppuun, on vuorossa *käyttöönotto*. Käyttöönotossa toteutettu järjestelmä asennetaan asiakkaan ympäristöön ja suoritetaan tarvittavat konfiguroinnit. Käyttöönottoon liittyy yleensä myös asiakkaan koulutus uuden järjestelmän käyttöön.

Ylläpito koostuu asiakkaan ongelmien ratkomisesta, virheiden korjaamisesta, ohjelman muuttamisesta ja uuden piirteiden lisäämisestä. Ylläpito voidaan jakaa korjaavaan, adaptiiviseen ja täydentävään ylläpitoon. Korjaavassa ylläpidossa korjataan järjestelmästä löytyneitä virheitä, adaptiivisessa ylläpidossa järjestelmää muokataan vastaamaan muuttunutta ympäristöä ja täydentävässä ylläpidossa järjestelmää parannetaan muuttamalla tai lisäämällä sen ominaisuuksia. (Immonen, 2003; Haikala & Märijärvi, 2004.)

4.3 Inkrementaaliset mallit

Vesiputousmallin lisäksi yleisesti käytettyjä elinkaarimalleja ovat erilaiset inkrementaaliset mallit, muun muassa Evo-malli (*evolutionary delivery*), protoilumalli ja spiraalimalli.

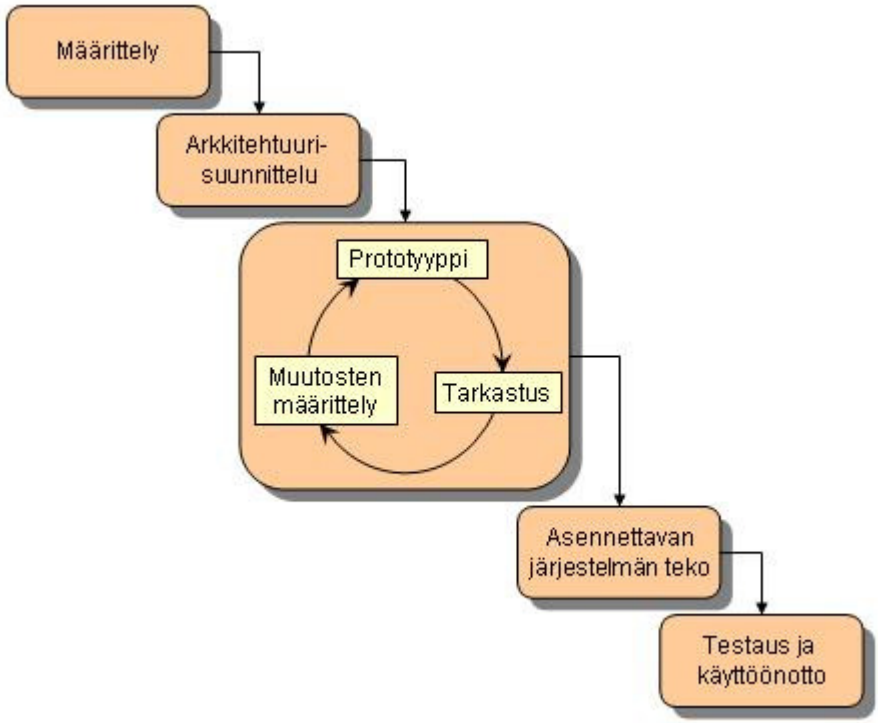
Evo-mallin ideana on rakentaa ensimmäisessä projektissa ydinjärjestelmä, jonka kehitystä seuraavissa projekteissa jatketaan. Jokaisen syklin eli iteraation tuloksena on uusilla ominaisuuksilla kasvatettu järjestelmä. Kuvassa 4.3 on esimerkki Evo-mallista. (Haikala & Märijärvi, 2004.)



Kuva 4.3 Evo-malli (Ilkko, 2008)

Prototyyppimallilla tarkoitetaan työskentelymallia, jossa jotain tuotteen piirrettä kokeillaan ennen varsinaisen tuotteen toteuttamista (kuva 4.4). Se soveltuu eri-

tyisesti uuden teknisen ratkaisun kokeilemiseen tai etsimään epäselviä asiakasvaatimuksia. Prototyypin valmistuttua sen perusteella voidaan määritellä toteutettava järjestelmä tai se voidaan kehittää valmiiksi järjestelmäksi. (Haikala & Märijärvi, 2004.)



Kuva 4.4 Protoilumalli (Ilkko, 2008)

Spiraalimallin perusajatuksena on, että prosessia tarkennetaan asteittain. Aluksi luodaan alkeellinen prototyyppi toteutettavasta ohjelmasta tai sen tärkeästä osasta. Sitä testataan ja mahdollisesti esitellään asiakkaalle. Testauksesta ja asiakkaalta saadusta palautteesta aloitetaan seuraava suunnittelun, toteutuksen ja testauksen kierros. Esimerkki spiraalimallista on kuvassa 4.5. (Saarinen, 2008.)

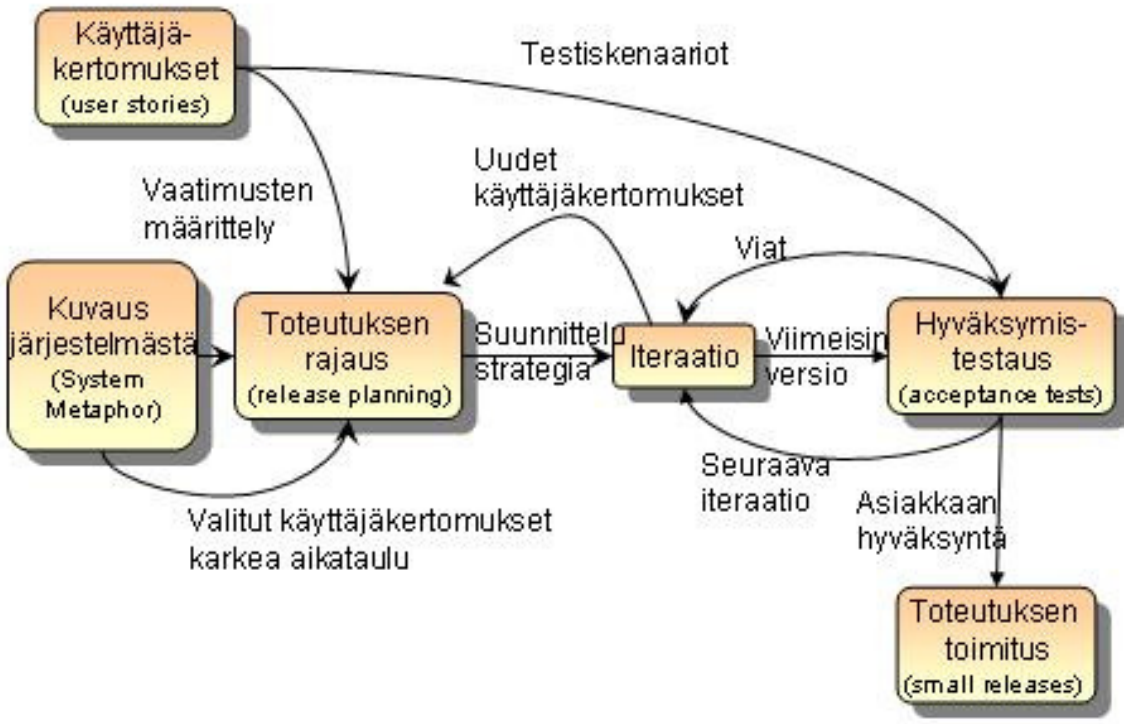
Kuva 4.5 Spiraalimalli (Ilkko, 2008)

4.4 Ketterät mallit

Ketterät elinkaarimallit ovat tyypiltään iteratiivisia menetelmiä, joissa projekti pyritään pilkkomaan pieniin erikseen suoritettaviin iteraatioihin. Iteraatiot ovat tyypillisesti hyvin lyhyitä, jopa niin lyhyitä, että kehittämisen voi ajatella olevan jatkuvaa. Jokaisen iteraation alussa selvitetään sen tavoitteet asiakkaan kanssa ja määritellään syklin puitteet. Iteraatioiden lopputuloksena saadaan aina toimiva tuote asiakkaan testattavaksi ja jatkojalostettavaksi. Yhteistä ketterille menetelmille ovat asiakkaan vahva läsnäolo, lyhyet kehityssyklit, pienet tiimit ja vahva kommunikaatio. (Haikala & Märijärvi, 2004; Wargh, 2009.)

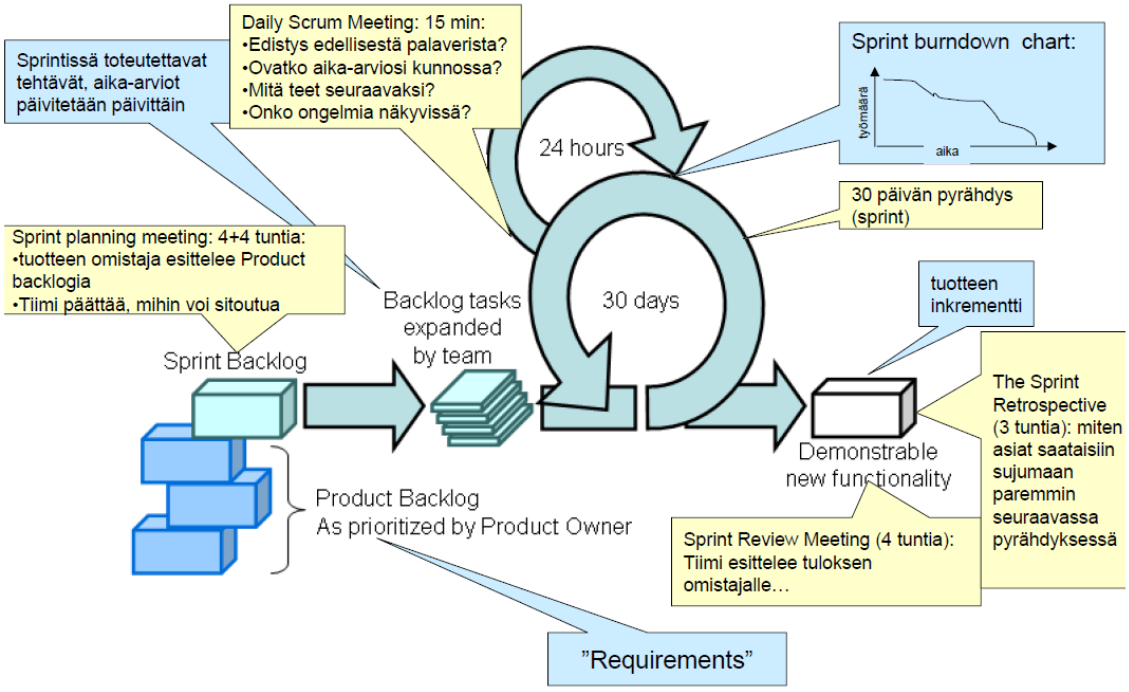
Tunnetuin ketterä elinkaarimalli on Kent Beckin kehittämä *XP-menetelmä* (*extreme programming*). Sen tunnusomaisia piirteitä ovat muun muassa jatkuva testaaminen ja pariohjelmointi. Pariohjelmoinnissa ohjelmistokehitys tapahtuu

pareittain, jolloin toinen kirjoittaa koodia ja toinen kommentoi tuotosta välittömästi. Kuvassa 4.6 on esimerkki XP-menetelmästä. (Haikala & Märijärvi, 2004; Wargh, 2009.)



Kuva 4.6 XP-menetelmä (Ilkko, 2008)

Scrum on yksi vanhimmista ketteristä sovelluskehitysmalleista (kuva 4.7). Se on hyvin skaalautuva ja yleisesti käytetty malli. Scrumin ydinajatus on, että luoteeltaan monimutkaisia ohjelmistoprojekteja ei voida suunnitella etukäteen tai muutenkaan käyttää tuotantoteollisia oppeja sellaisenaan, vaan kehitysprosessin tulee olla empiirinen. Scrumissa toteutusvaiheita kutsutaan sprinteiksi. Yhden sprintin kestoksi on määritelty noin neljä viikkoa. (Haikala & Märijärvi, 2004; Wargh, 2009.)



Kuva 4.7 Scrum-menetelmä (Haikala, 2009)

5 TYÖSSÄ KÄYTETYT MÄÄRITTELY- JA SUUNNITTELUMENETELMÄT

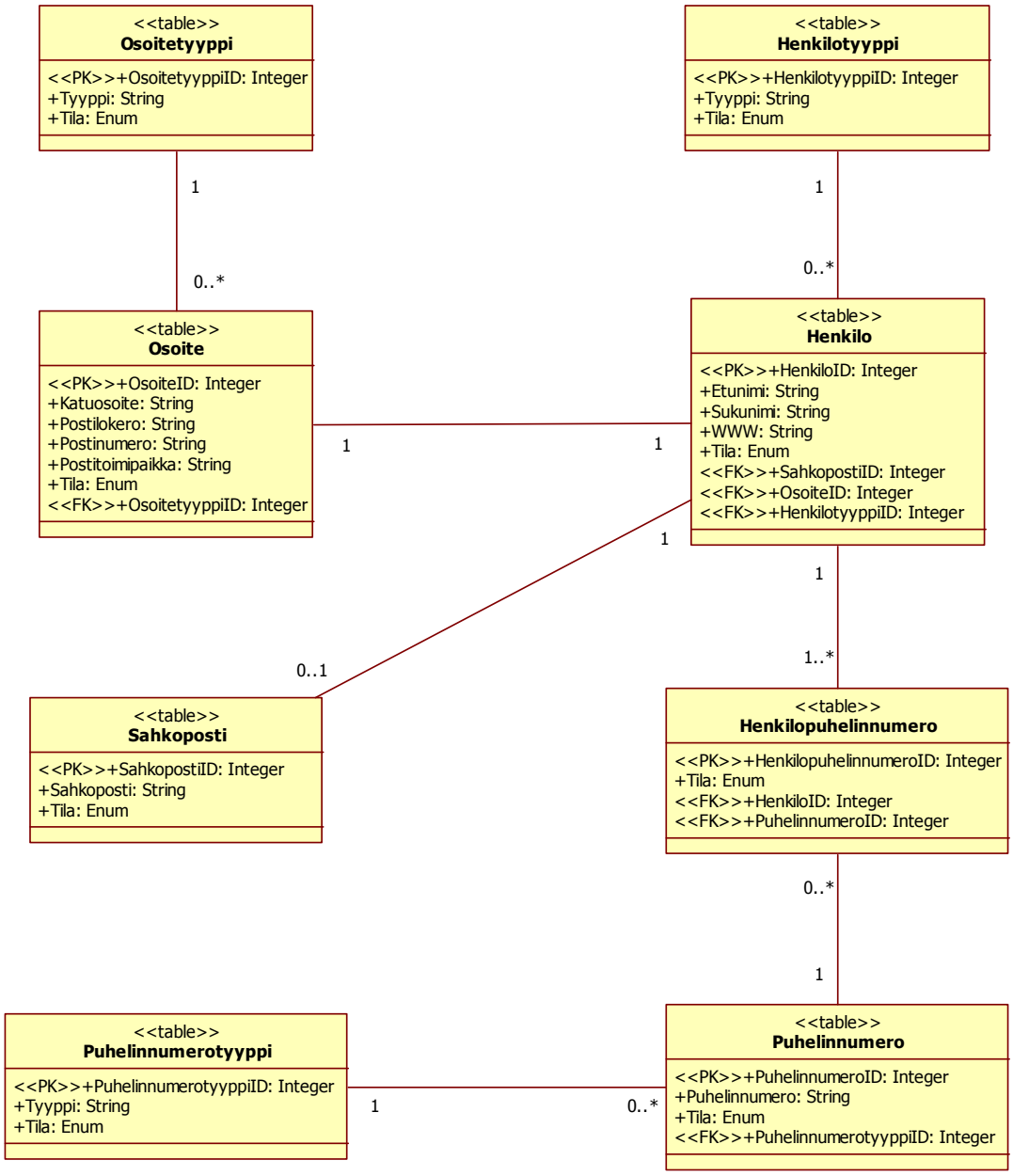
Tässä luvussa käydään läpi opinnäytetyössä käytetyt määrittely- ja suunnittelumenetelmät: käsiteanalyysi, käyttötapausanalyysi ja UML.

5.1 Käsiteanalyysi

Käsiteanalyysi on tietokantoihin johtavan suunnitteluprosessin ensimmäisiä vaiheita. Käsiteanalyysissä selvitetään, mitkä ovat tietokantaan talletettavat tiedot. Tietosisällöstä voidaan laatia eri tarkkuustason kuvauksia. (Laine, 2005.)

Käsiteanalyysin tarkoituksena on saada aikaan riittävän tarkka kuva tietosisällöstä, jotta tätä voisi käyttää lähtökohtana tietokannan tai muun tarkasteltavan tietokokoelman rakennetason suunnittelulle. Analyysiä joudutaan suorittamaan myös tilanteissa, joissa sovitetaan yhteen eri tietokantoja tai tietokokoelmia.

Käsiteanalyysin tuloksena on käsitekaavio. Kuvassa 5.1 on esitettynä *Henkilön* käsitekaavio, jossa näkyvät myös kaikki *Henkilöön* liittyvät taulut. Henkilö-taulun yksikkötyyppikuvaus näkyy kuvassa 5.2. (Laine, 2005.)



Kuva 5.1 Henkilön käsitekaavio

Henkilo

Sisältää kaikki tiedot henkilöistä, joita tarvitaan järjestelmässä.

Henkilo		
@HenkiloID	*INT(10)*	** Järjestelmän luoma yksilöivä arvo **
Etunimi	*VARCHAR (20)*	** Pakollinen kenttä **
Sukunimi	*VARCHAR(25)*	** Pakollinen kenttä **
WWW	*VARCHAR(80)*	** Vakiona tyhjä kenttä **
Tila	*ENUM*	** Pakollinen kenttä **
#SahkopostiID	*INT(10)*	** Viite, pakollinen **
#OsoiteID	*INT(10)*	** Viite, pakollinen **
#HenkilotyyppiID	*INT(10)*	** Viite, pakollinen **

Kuva 5.2 Henkilo-taulun yksilötyyppikuvaus

5.2 Käyttötapausanalyysi

Käyttötapausanalyysi on Ivar Jacobsonin kehittämä ja yleisesti käytetty järjestelmälle asetettujen toiminnallisten vaatimusten mallinnustekniikka. Sillä kuvataan, mitä järjestelmän tulisi tehdä tai mitä se tekee. Kyseessä on iteratiivinen työtapa, johon kuuluu neuvotteluita asiakkaan ja toimijoita vastaavien henkilöiden kanssa. (Tapola, 2004; Järvelä & Puusaari, 2005.)
Käyttötapaus kuvaa järjestelmän ja toimijan välisen vuorovaikutuksen halutun lopputuloksen aikaansaamiseksi. Vuorovaikutus ei aina ole tarkalleen samanlainen, eli samalla käyttötapauksella voi olla useita skenaarioita. (Järvelä & Puusaari, 2005.)

Järjestelmän käyttötapaukset muodostavat käyttötapausmallin, joka kuvaa täydellisesti järjestelmän ulkoisen käyttäytymisen ja ympäristön. Käyttötapausmallin tärkeimmät osat ovat käyttötapaukset, toimijat ja mallinnettava järjestelmä. Käyttötapausmallin tekemiseen kuuluu järjestelmän määrittely, toimijoiden ja käyttötapausten löytäminen, käyttötapausten ja niiden välisten suhteiden kuvaaminen ja lopulta mallin hyväksyntä. Käyttötapausmallia tarkennetaan vähitel-

len siten, että ensin laaditaan yleisluontoinen organisaation toimintaa kokonaisuutena valottava käyttötapausmalli ja tätä mallia tarkennetaan tiettyjä osa-alueita erikseen ja yksityiskohtaisemmin esittävillä malleilla. Näin ollen käyttötapausmalli tarkentuu koko järjestelmän kattavaksi kuvaukseksi. Käyttötapausmalli toimii ohjelmiston kehityksen perustana. Kuvassa 5.3 on nähtävillä kiinteistön haku -käyttötapauksen sanallinen kuvaus. (Korpimies, 2005; Järvelä & Puusaari, 2005.)

Kuvaus: Käyttäjä hakee kiinteistöjä järjestelmästä.

Suorittajat: Myyntineuvottelija, Ylläpitäjä

Liittyvät käyttötapaukset:
Selaa kiinteistöjä

Alkutila ja alkuehdot:
1. Käyttäjä on kirjautuneena järjestelmään
2. Käyttäjällä on käyttöoikeudet hakea kiinteistöjä

Tyypillinen käyttötapauksen kulku:
1. Käyttäjä avaa kiinteistöhaun
2. Järjestelmä näyttää kiinteistöhakusivun
3. Käyttäjä suorittaa kiinteistöhaun kiinteistönumerolla
4. Järjestelmä palauttaa onnistuneen haun tulokset

Vaihtoehtoinen käyttötapauksen kulku 1:
1. Käyttäjä avaa kiinteistöhaun
2. Järjestelmä näyttää kiinteistöhakusivun
3. Käyttäjä suorittaa kiinteistöhaun kiinteistön osoitteella
4. Järjestelmä palauttaa onnistuneen haun tulokset

Vaihtoehtoinen käyttötapauksen kulku 2:
1. Käyttäjä avaa kiinteistöhaun
2. Järjestelmä näyttää kiinteistöhakusivun
3. Käyttäjä suorittaa kiinteistöhaun asiakasnimellä
4. Järjestelmä palauttaa onnistuneen haun tulokset

Virhetilanteet:
1. Haettua kiinteistöä ei löydy järjestelmästä

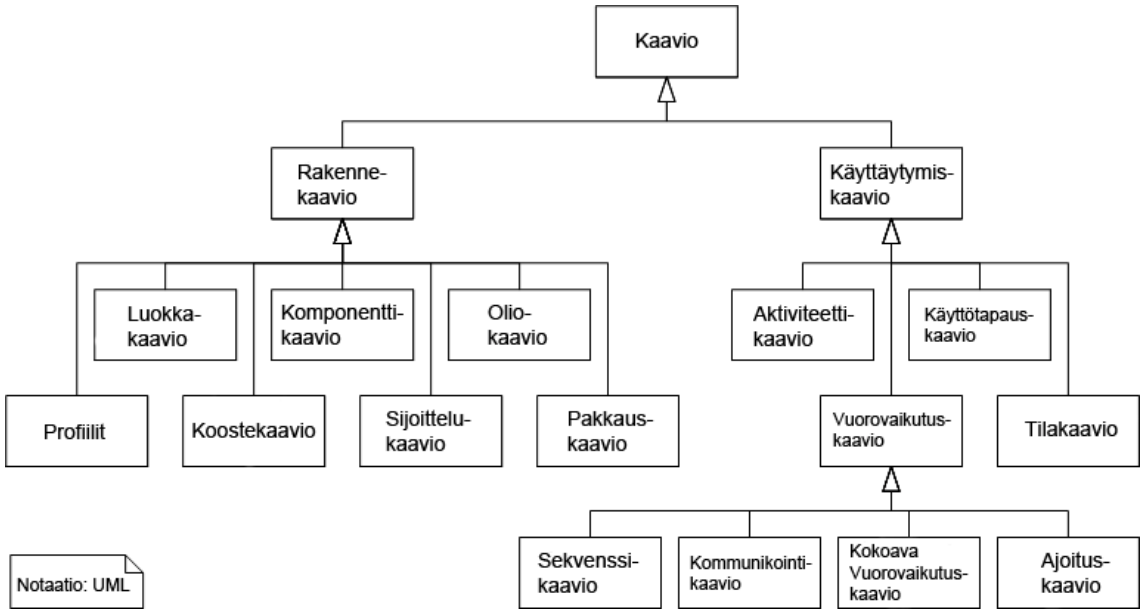
Lopputila ja jälkiehdot:
1. Kiinteistön haku on suoritettu onnistuneesti
2. Järjestelmä on palauttanut hakutulokset

Kuva 5.3 Kiinteistön haun sanallinen käyttötapauskuvaus

5.3 UML

UML (Unified Modelling Language) on graafinen ohjelmistojen mallinnuskieli oliopohjaiseen visualisointiin, määrittelyyn, rakentamiseen ja dokumentoimiseen. Se on kehitetty yhdistämällä kolmen tunnetuimman ns. ensimmäisen sukupolven oliomenetelmän, OMT:n, BOOCH:n ja OOSE:n, parhaat ominaisuudet yhdeksi mahdollisimman yleiskäyttöiseksi ja kattavaksi notaatioksi. UML määrittelee joukon ohjelmiston kuvaukseen käytettäviä kaaviotyyppejä mutta ei määrittele, missä ohjelmistonkehitysvaiheessa mitäkin kuvaustapaa tulisi käyttää. UML-mallinnuskieli hyväksyttiin standardiksi vuonna 1997 ja nykyisin sen kehittämisestä vastaa OMG (Object Management Group). (Äyrämö, 2002; Kylä-Nikkilä, 2008; Luukkainen & Laine, 2009.)

UML 2.2:ssa on 13 erilaista kaaviota, jotka on jaettu kahteen eri osioon, käyttäytymiskaavioihin ja rakennekaavioihin. Kaaviot on jaoteltu hierarkkisesti seuraavassa kuvassa (kuva 5.4).



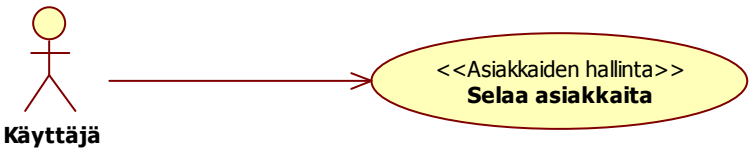
Kuva 5.4 UML-kaaviohierarkia

Seuraavassa kuvataan eri kaaviotyypit lyhyesti ja keskitytään tässä opinnäytetyössä käytettyihin kaaviotyyppeihin tarkemmin.

5.3.1 Käyttötapauskaavio

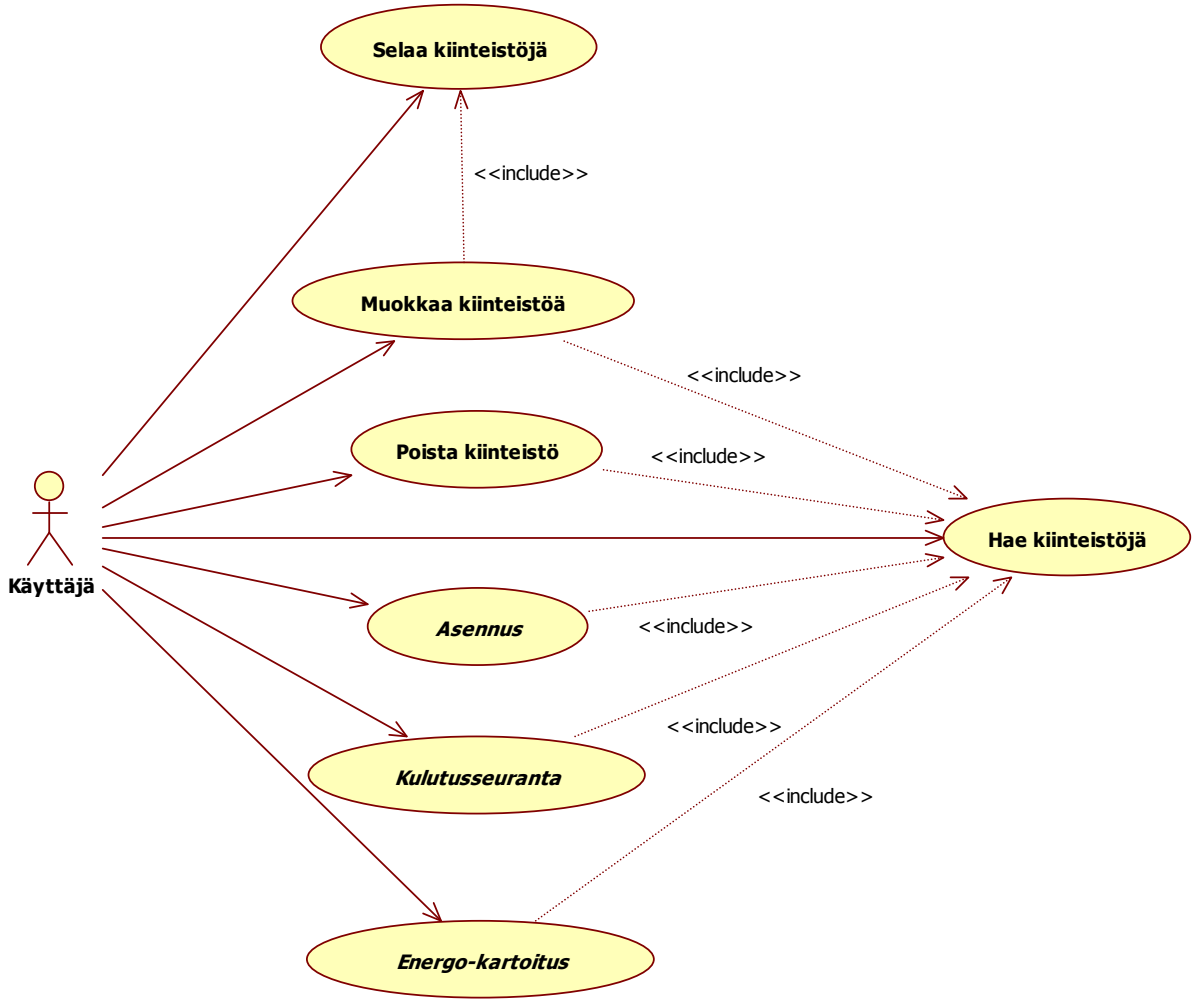
Käyttötapauskaavio on yksi UML:n käytetyimmistä kaaviotyypeistä. Se kertoo, mitä järjestelmä tekee tai mitä sillä voi tehdä. Käyttötapauskaavio on rakenteellisesti yksinkertainen kaavio, joka koostuu toimijoista, käyttötapauksista ja erilaisista suhteista. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)

Toimija on järjestelmän ulkopuolinen olio, joka toimii vuorovaikutuksessa järjestelmän kanssa. Se on järjestelmän käyttäjä ja käyttötapauksen suorittaja. Yhdellä toimijalla voi olla useita käyttötapauksia sekä päinvastoin yhdellä käyttötapauksella voi olla useita toimijoita. Toimija on yleensä henkilö, mutta se voi olla myös esimerkiksi ulkoinen järjestelmä. Toimijat kuvataan käyttötapauskaaviossa tikku-ukkoina (kuva 5.5). Toimijoiden rooleja voidaan periyttää (*generalization*). Periytyminen esitetään kolmiokärkisenä nuolena, joka osoittaa yleiseen toimijaan. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)



Kuva 5.5 Käyttötapauskaavion toimija

Käyttötapaus kuvaa järjestelmän toimintoa, joka palauttaa havaittavan vasteen toimijalle paljastamatta toiminnon toteutumisrakennetta. Visuaalisesti se kuvataan ellipsinä, jonka sisällä tai alapuolella lukee käyttötapauksen nimi. Käyttötapaus kytketään toimijaan assosiaatiolla, joka tarkoittaa sitä, että toimijalla on jokin rooli käyttötapauksen suorituksessa. Tätä assosiaatiota kuvataan kaaviossa viivalla ja se voidaan nimetä sen suorittaman toiminnon mukaan. Kuvassa 5.6 on kuvattu Innonet-järjestelmän kiinteistöjen hallinta käyttötapauskaaviolla. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)



Kuva 5.6 Kiinteistöjen hallinnan käyttötapauskaavio Innonet-järjestelmässä

Käyttötapausten välillä voi olla myös periytymis-, laajennus- ja sisällytystyyppisiä suhteita. Periytymissuhteessa käyttötapaus perii toisen käyttötapauksen, jolloin perivä käyttötapaus voi lisätä omaa toiminnallisuutta tai korvata vanhaa. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)

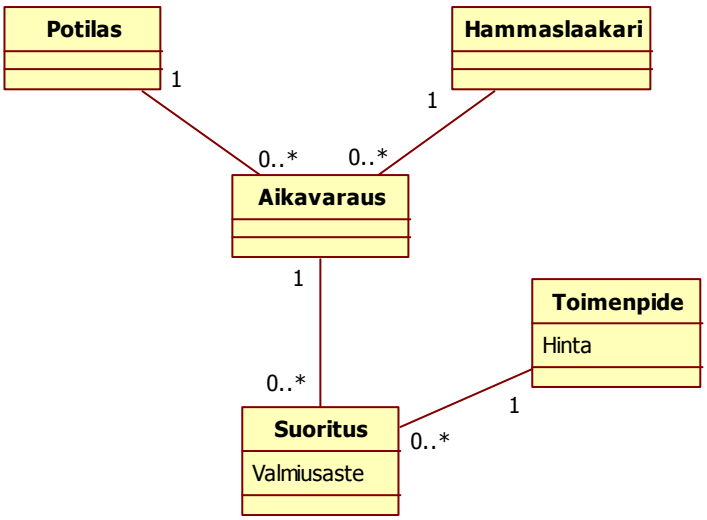
Laajennussuhde tarkoittaa, että määrätyissä tilanteissa toisen käyttötapauksen toiminnallisuudella voidaan laajentaa toista käyttötapausta. Laajennussuhde kuvataan katkoviivalla, jonka laajennettavan päässä on nuoli ja viivan yhteydessä stereotyyppi <<extend>>. (Kylä-Nikkilä, 2008.)

Sisällyssuhde tarkoittaa, että käyttötapauksen suorittaminen edellyttää myös toisen käyttötapauksen suorittamista. Tällainen käyttötapaus kuvataan katkovii-

vana, jonka käytettävän käyttötapauksen päässä on nuoli ja viivan yhteydessä stereotyyppi <<include>>. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)

5.3.2 Luokkakaavio

Luokkakaaviolla esitetään järjestelmän pysyvää rakennetta. Se kuvaa järjestelmässä olevia luokkia, niiden sisäistä rakennetta ja suhteita toisiin luokkiin. Luokkien väliset suhteet määrittelevät niiden keskinäisen riippuvuuden toisistaan. Luokka kuvataan kaaviossa laatikkona, jossa luetellaan luokan ominaisuudet, attribuutit ja operaatiot. Kuvassa 5.7 on esimerkki UML-luokkakaaviosta. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)



Kuva 5.7 Luokkakaavio hammaslääkärin toiminnasta

Luokkien väliset yhteydet kuvataan niiden välille piirretyillä viivoilla. Riippuvuus on kahden luokan välillä oleva suhde, jossa luokka tarvitsee toisen luokan oliota suorittaakseen omia toimintojaan. Riippuvuus kuvataan katkoviivalla varustetulla nuolella. Jos nuoli osoittaa luokasta A luokkaan B, on luokka A riippuvainen luokasta B. Kuvassa 5.8 on esimerkki, jossa luokkien *Ostoskori* ja *Tuote* välillä on riippuvuussuhde. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)

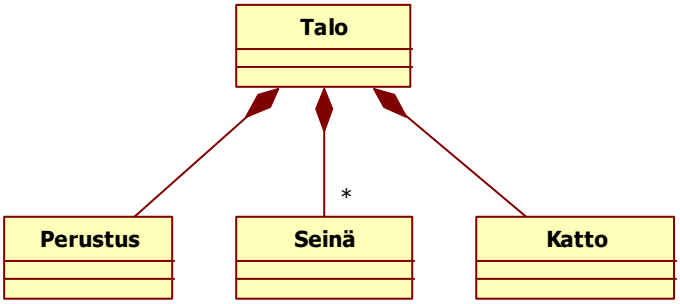
Kuva 5.8 Esimerkki riippuvuussuhteesta (Kylä-Nikkilä, 2008)

Assosiaatiossa luokat liittyvät ohjelmiston rakenteen kannalta toisiinsa. Se on riippuvuutta vahvempi suhde, koska luokat käyttävät toistensa rajapintapalveluita osana omia toimintojaan. Assosiaatio kuvataan yhtenäisenä viivana luokkien välillä. Assosiaatiota saadaan tarkennettua lisäämällä siihen esim. assosiaation nimi, luokkien roolinimet ja lukumääräsuhteet. Assosiaatio on yleensä kaksisuuntainen suhde, jossa luokat tietävät toistensa olemassaolon. Yksisuuntaisessa assosiaatiossa vain palveluita pyytävä luokka tietää palveluita tarjoavan luokan olemassaolosta. Tämä ilmaistaan piirtämällä nuoli kohti sitä luokkaa, joka ei tiedä assosiaatiosta (kuva 5.9). Kaksisuuntaisessa assosiaatiossa nuolia ei piirretä. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)

Kuva 5.9 Esimerkki assosiaatiosta kahden luokan välillä (Kylä-Nikkilä, 2008)

Koostesuhteet ovat assosiaation erikoistapauksia, joissa määritellään luokkien välisen yhteyden lisäksi niiden välille omistussuhde. Koostesuhteita on kolmea erilaista: normaali, jaettu sekä vahva kooste eli komposiitti. Normaali kooste kuvataan viivana, jonka kokonaisuutta kuvaavaan päähän piirretään tyhjä salmiakkikuvio. Jaettu kooste eroaa normaalista koosteesta siinä, että kokonaisuutta osoittavassa päässä kerrannaisuus on suurempi kuin yksi (1). (Äyrämö, 2002; Kylä-Nikkilä, 2008.)

Komposiitissa kaikki yhdistetyt luokat ovat kiinteästi osa kokonaisuutta. Tällöin, jos koosteolio poistetaan, poistuvat myös siitä riippuvaiset osaoliot. Komposiitti kuvataan täytetyllä salmiakkikuviolla. Kuvassa 5.10 on komposiitti esitettynä. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)



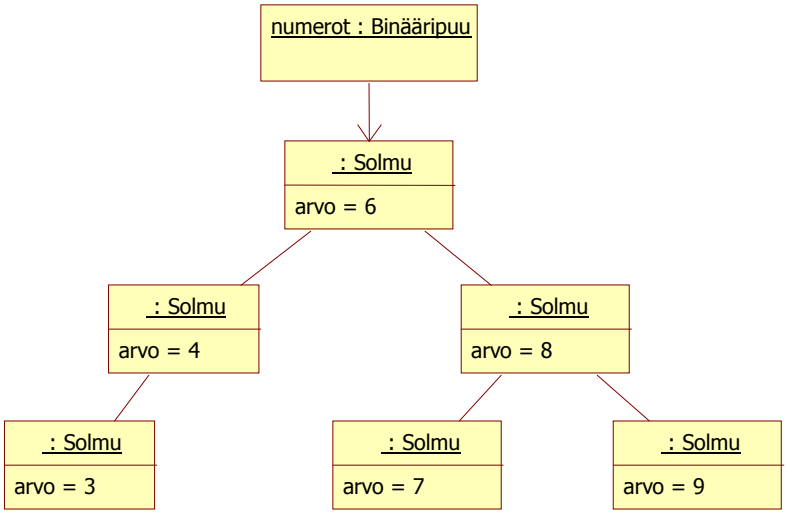
Kuva 5.10 Esimerkki komposiitista (Kylä-Nikkilä, 2008)

Periytyminen on suhde yleisen luokan ja siitä periytetyn luokan välillä. Periytettyä luokkaa kutsutaan aliluokaksi ja yleisempää luokkaa yliluokaksi. Aliluokka perii yliluokalta kaikki sen ominaisuudet, joita ovat attribuutit, assosiaatiot ja operaatiot. Periytyminen kuvataan nuoliviivalla, joka osoittaa aliluokasta yliluokkaan. Kuvassa 5.11 on yksinkertainen esimerkki perinnästä, jossa Henkilöluokasta periytetään Johtaja-luokka. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)

Kuva 5.11 Esimerkki periyttämisestä (Luukkainen & Laine, 2009)

5.3.3 Oliokaavio

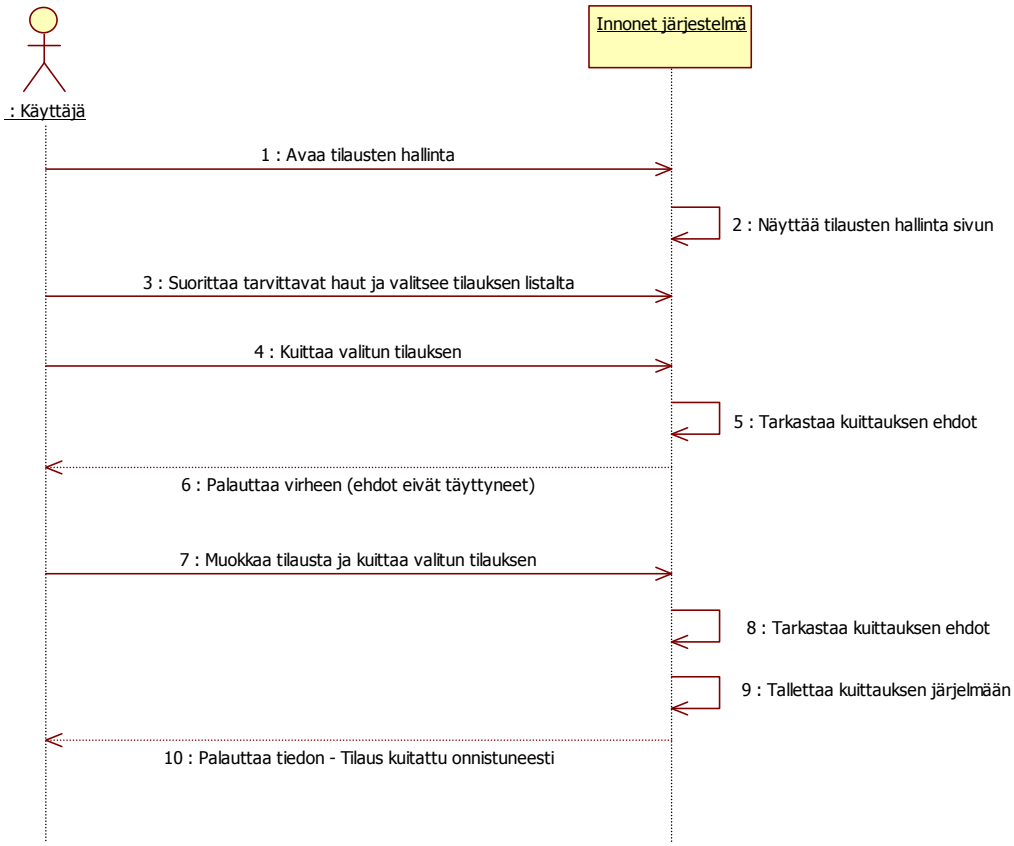
Oliokaavio kuvaa järjestelmän yksittäisten olioiden eli luokkien ilmentymien ja niiden välisten yhteyksien tilanteen tietyllä suoritushetkellä. Oliokaavion ja luokkakaavion suurin ero on se, että luokka voi esiintyä luokkakaaviossa vain kerran, mutta oliokaaviossa luokalla voi olla useita ilmentymiä kerralla näkyvissä. Oliokaavion ja luokkakaavion merkintätavat ovat muuten samanlaiset, mutta olio näytetään luokkana nimi alleviivattuna. Olion nimi näytetään ennen luokan nimeä kaksoispisteellä erotettuna, ja olion ollessa nimetön näytetään vain luokan nimi. Kuvassa 5.12 on esimerkki binääripuun oliokaaviosta. (Järvelä & Puusaari, 2005; Kylä-Nikkilä, 2008.)



Kuva 5.12 Oliokaavio binääripuusta (Kylä-Nikkilä, 2008)

5.3.4 Sekvenssikaavio

Sekvenssikaaviolla kuvataan olioiden välistä vuorovaikutusta ajallisessa järjestyksessä. Kaaviossa on kaksi akselia, joista pystyakseli määrittää ajan, joka kulkee ylhäältä alaspäin, ja vaaka-akseli määrittelee oliot, jotka ovat mukana vuorovaikutuksessa. Kaaviota luetaan yleensä ylhäältä alaspäin ja vasemmalta oikealle. Sekvenssikaaviossa kuvataan nimenomaan olioita eikä luokkia. Oliot instantioidaan luokista lisäämällä kaksoispiste luokan nimen eteen, jolloin tiedetään, että kyseessä on luokan ilmentymä. Sekvenssikaavio näyttää kerrallaan yhden toiminnan järjestelmästä. Kuvassa 5.13 on kuvattu Innonet-järjestelmän tilauksen kuittaus sekvenssikaaviolla. (Äyrämö, 2002; Kylä-Nikkilä, 2008.)



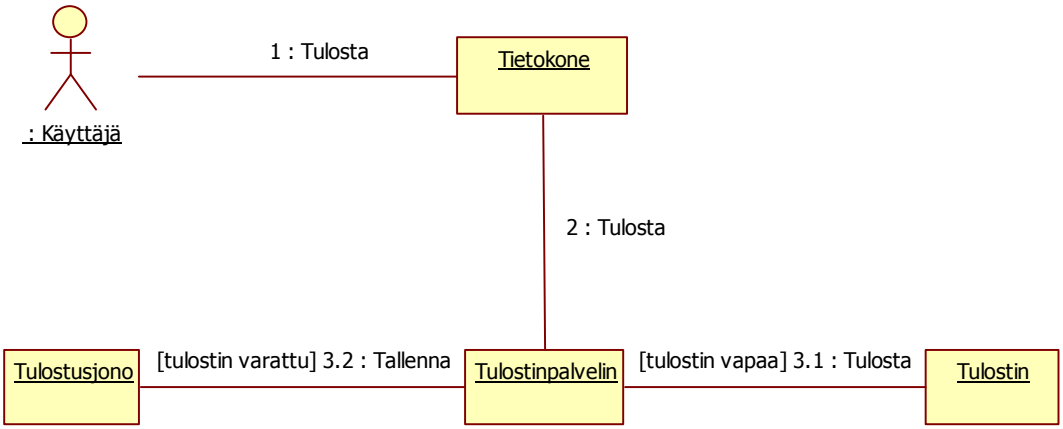
Kuva 5.13 Tilauksen kuittauksen sekvenssikaavio Innonet-järjestelmässä

Olioiden elinkaarta kuvataan pystysuuntaisella katkoviivalla, ns. elämänviivalla. Elämänviiva muutetaan paksummaksi palkiksi, kun olio tulee aktiiviseksi. Palkin pituus kuvaa toiminnon suorittamiseen kuluvaa aikaa. Vuorovaikutuksessa oleva toimija kuvataan yleensä kaavion vasempaan reunaan. Oliot kommunikoivat keskenään lähettämällä viestejä toisilleen. Viestit kuvataan nuolilla, jotka osoittavat yhteyden suuntaan. Täytetyllä nuolella kuvattu viesti tarkoittaa synkronista kutsua, eli kutsuja odottaa ja jatkaa vasta kun kutsu on suoritettu. Avoimella nuolella merkitty kutsu taas tarkoittaa asynkronista kutsua, jossa kutsuja ei jää odottamaan operaation suoritusta vaan jatkaa välittömästi. Aliohjelmasta paluu kuvataan katkoviivalla varustetulla nuolella. (Äyrämö, 2002; Kylä-Nikkilä, 2008; Luukkainen & Laine, 2009.)

Olion kutsuessa itseään rekursiivisesti lähtee kuvattu nuoli oliosta itsestään ja päättyy hieman alempana takaisin oliolle. Jos olio tuhotaan, merkitään se isolla rastilla elämänviivan loppuun. (Kylä-Nikkilä, 2008.)

5.3.5 Kommunikointikaavio

Kommunikointikaavio on sekvenssikaavion ohella toinen tapa kuvata olioiden suorituksenaikaista yhteistyötä. Se kuvaa olioiden vuorovaikutusta kuten sekvenssikaaviokin mutta tilakeskeisesti, kun taas sekvenssikaaviossa keskitytään olioiden ajalliseen järjestykseen. Myös komponentit ja käsitteet kommunikointikaaviossa ovat lähes samat kuin sekvenssikaaviossa, ainoastaan yhteyden esitysasu on erilainen. Koska kommunikointikaaviosta puuttuu sekvenssikaavion aikajana, kirjoitetaan yhteyden eteen juokseva numero, joka määrittää viestien järjestyksen. Jos samaan aikaan lähetetään useampia viestejä, lisätään tunnisteen perään kirjain (a-z). Toinen vaihtoehto on käyttää pistettä viestin tunnisteessa ja jakaa numerointi useampaan osaan. Kuvassa 5.14 on esitetty tulostinpalvelimen toiminta kommunikointikaavion avulla. (Kylä-Nikkilä, 2008.)

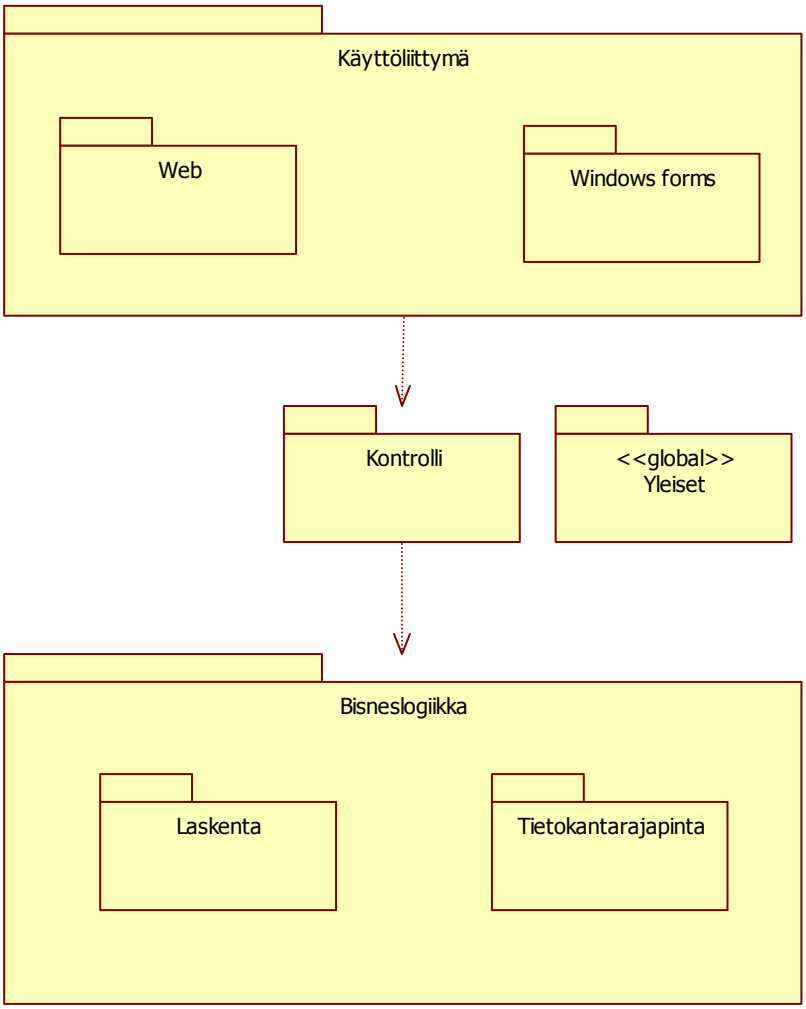


Kuva 5.14 Kommunikointikaavio tulostinpalvelimen toiminnasta (Kylä-Nikkilä, 2008)

5.3.6 Pakkauskaavio

Pakkauskaavio ryhmittelee ohjelman rakenteen suurempiin kokonaisuuksiin. Se voi sisältää lähes mitä tahansa UML:n elementtejä, myös toisia pakkauksia. Kaaviosta nähdään myös sen sisältämien elementtien väliset riippuvuudet ja näkyvyydet. Pakkauksen sisällä olevan luokan ollessa riippuvainen toisen pak-

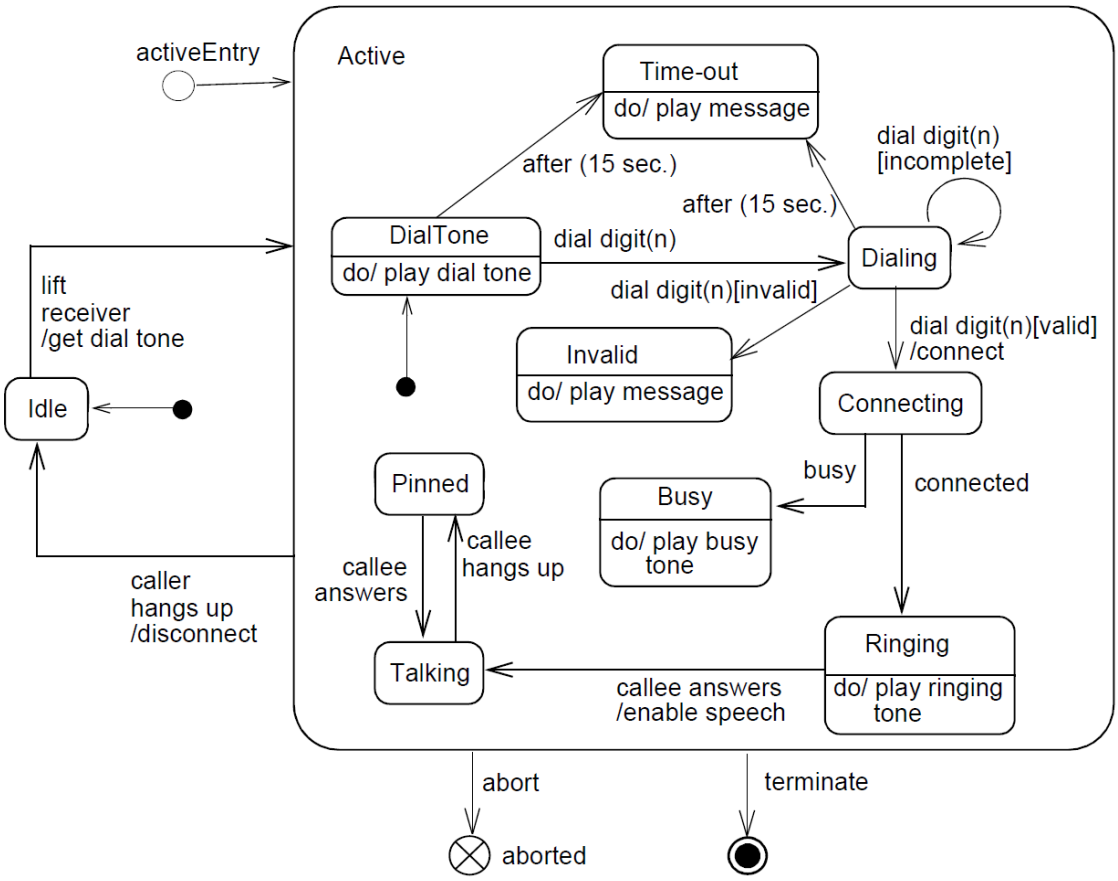
kauksen sisällä olevasta luokasta syntyy näiden kahden pakkauksen välille riippuvuus. Tätä riippuvuutta kuvataan katkoviivalla, joka suuntautuu siihen pakkaukseen, johon riippuvuus kohdistuu. Riippuvuuksissa voidaan käyttää myös stereotyyppejä tarpeen vaatiessa. Pakkaukset, joista kaikki muut pakkaukset ovat riippuvaisia, merkitään <<global>> -stereotyypillä. Esimerkki pakkauskaaviosta on kuvassa 5.15. (Kylä-Nikkilä, 2008; Luukkainen & Laine, 2009.)



Kuva 5.15 Pakkauskaavio (Kylä-Nikkilä, 2008)

5.3.7 Tilakaavio

Tilakaavioiden avulla kuvataan olion ja järjestelmän tilasta riippuvaa käyttäytymistä. Tilakaavioista ilmenee mallinnettavan kohteen tilat sekä eri tilojen väliset siirtymät. Tilakaavioita käytetään yleensä tosiaikaisten ja muiden tilakriittisten järjestelmien mallintamiseen. Esimerkki tilakaaviosta on kuvassa 5.16. (Kylä-Nikkilä, 2008; Luukkainen & Laine, 2009.)



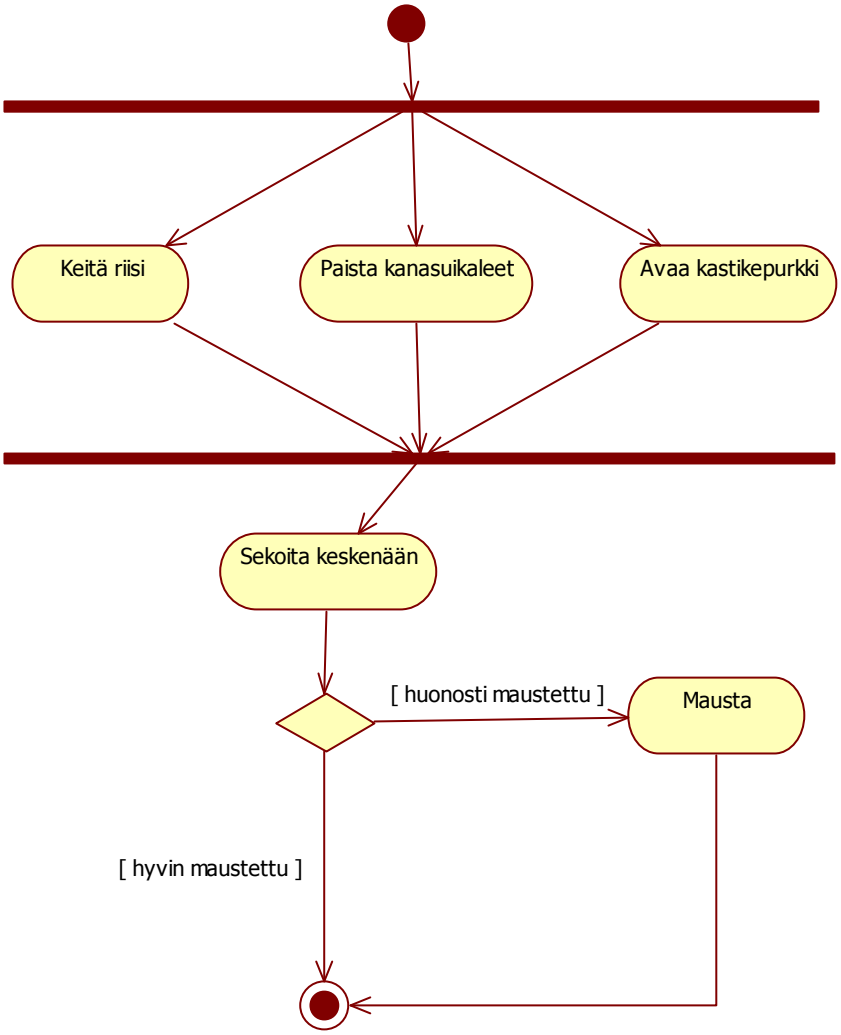
Kuva 5.16 Tilakaavio puhelimen toiminnasta (OMG, 2009)

5.3.8 Aktiviteettikaavio

Aktiviteettikaaviota käytetään tehtävien sisäisen logiikan ja niiden tulosten kuvaamiseen olioiden tilojen muutoksina. Aktiviteettikaavio muistuttaa paljon tilakaaviota. Tilakaavion kuvatessa yksittäisen olion toimintaa on aktiviteettikaaviolla mahdollisuus kuvata suurempaa toiminnallista kokonaisuutta. Jossain tapa-

uksissa aktiviteettikaavioita voidaan käyttää myös käyttötapausten kuvaamiseen. (Kylä-Nikkilä, 2008; Luukkainen & Laine, 2009.)

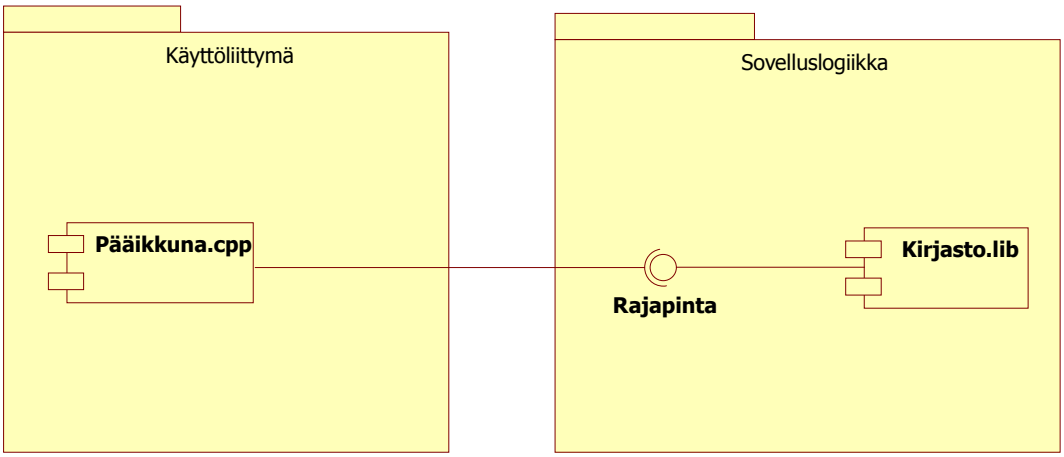
Kaaviossa tarkasteltava tehtävä voi olla yksittäinen operaatio tai useampi käyttötapaus. Tehtävään osallistuvat suorittajat erotellaan kaaviossa toisistaan sarakkeilla, joita kutsutaan uimaradoiksi. Uimaradat esitetään kaaviossa pystysuunnassa ja niillä on nimi. Kuvassa 5.17 on yksinkertainen aktiviteettikaavio kanakastikkeen valmistamisesta. (Kylä-Nikkilä, 2008; Luukkainen & Laine, 2009.)



Kuva 5.17 Aktiviteettikaavio kanakastikkeen valmistamisesta (Kylä-Nikkilä, 2008)

5.3.9 Komponenttikaavio

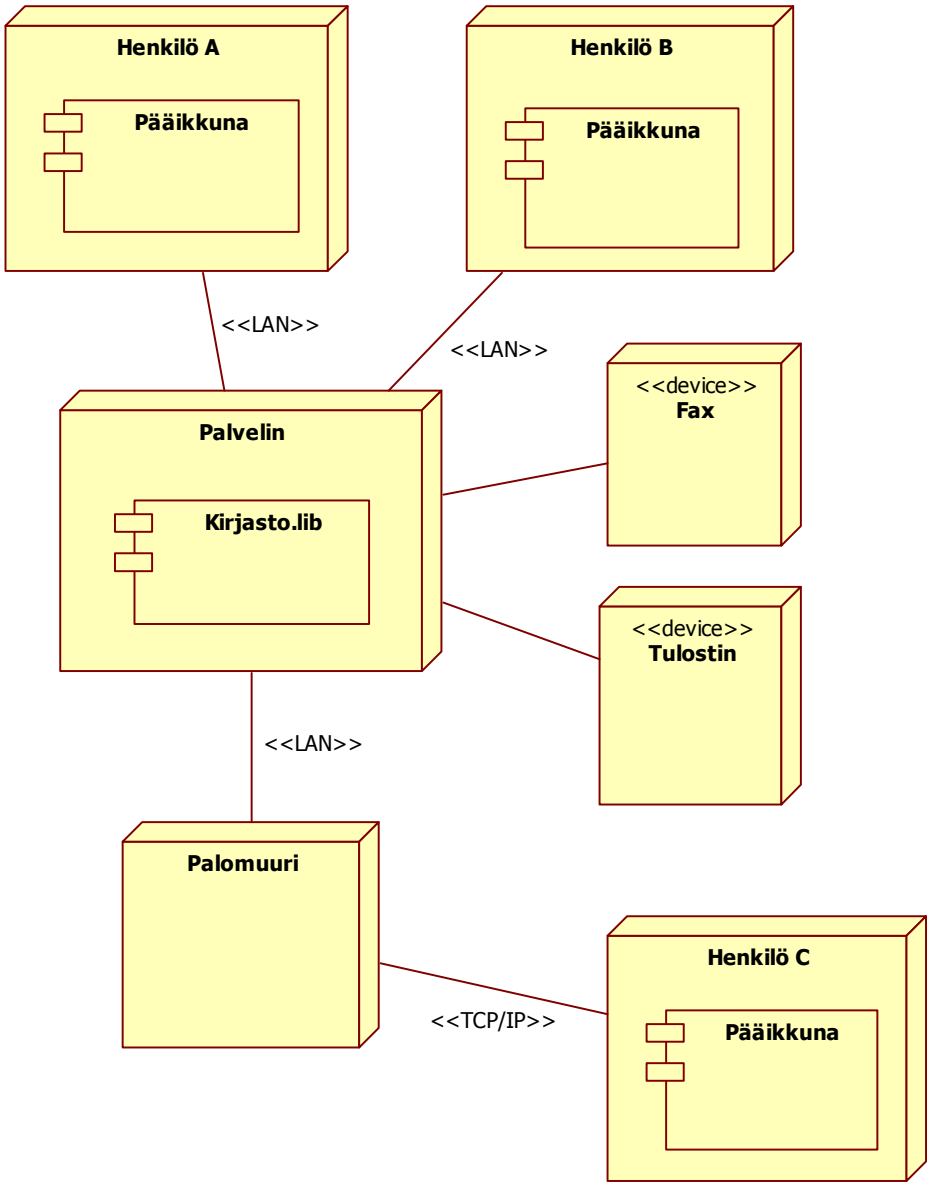
Komponenttikaavion tarkoitus on helpottaa kokonaisuuksien hallintaa kuten pakettikaavionkin. Komponenttikaaviolla voidaan esittää järjestelmän eri komponentit ja niiden väliset suhteet. Komponentilla tarkoitetaan loogisessa kaaviossa kuvatun käsitteen fyysistä toteutusta, kuten luokkaa. Kuvassa 5.18 on esimerkki komponenttikaaviosta. (Kylä-Nikkilä, 2008.)



Kuva 5.18 Komponenttikaavio (Kylä-Nikkilä, 2008)

5.3.10 Sijoittelukaavio

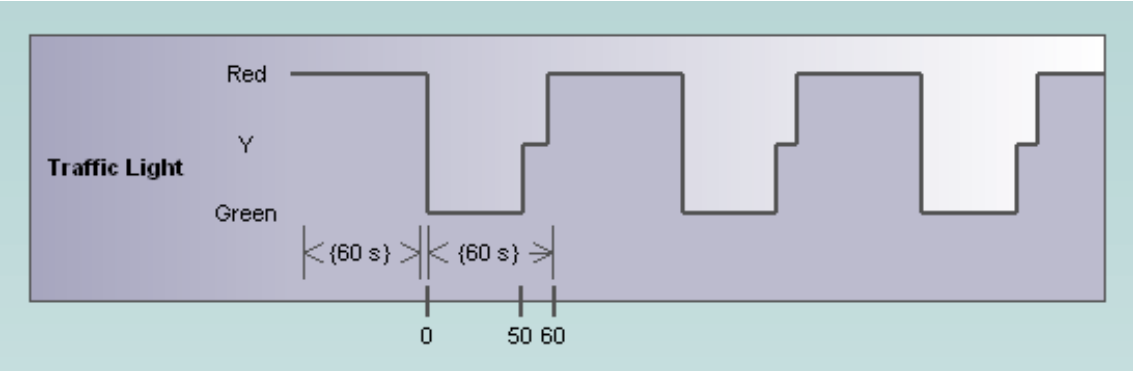
Sijoittelukaaviota käytetään kuvaamaan järjestelmän ajonaikaista tilannetta. Siinä kuvataan laitteiden ja ohjelmistokomponenttien välistä arkkitehtuuria. Sijoittelukaavion avulla voidaan kuvata järjestelmän ajoympäristö fyysisellä tasolla. Kuvassa 5.19 on sijoittelukaaviosta yksinkertainen esimerkki, joka kuvaa yrityksen laitteistoarkkitehtuuria. (Kylä-Nikkilä, 2008.)



Kuva 5.19 Sijoittelukaavio yrityksen laitteistoarkkitehtuurista (Kylä-Nikkilä, 2008)

5.3.11 Ajoituskaavio

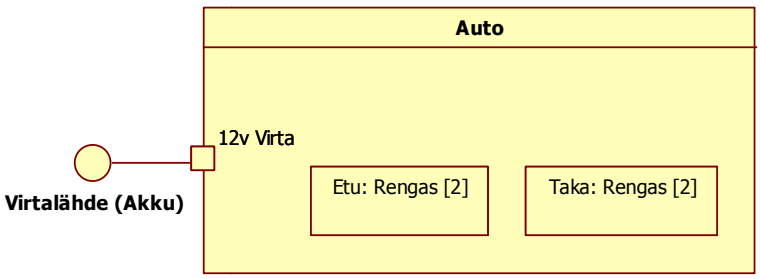
Ajoituskaaviolla kuvataan aktiivisia objekteja ja niiden tiloja vuorovaikutuksen aikana painottuen toimintojen ajoitukseen. Aika kulkee kaaviossa x-akselilla, ja objektit ja niiden tilat on kuvattuna y-akselilla. Kaaviota käytetään lähinnä aika-kriittisissä järjestelmissä, kuten sulautetuissa ympäristöissä ja tosiaikaisissa järjestelmissä. Kuvassa 5.20 on esimerkki liikennevalojen toimintaa kuvaavasta ajoituskaaviosta. (Kylä-Nikkilä, 2008.)



Kuva 5.20 Ajoituskaavio liikennevalojen toiminnasta (Altova, 2010)

5.3.12 Koostekaavio

Koostekaavio on tarkoitettu monimutkaisissa järjestelmissä olevien elementtien välisten riippuvuuksien kuvaamiseen. Koostekaavioilla voidaan kuvata luokkien sisäistä rakennetta sekä rajapintojen ja komponenttien välisiä toiminnallisia riippuvuuksia. Koostekaaviolla pystytään myös kuvaamaan kontekstiin sidottuja assosiaatioita toisin kuin olio- ja luokkakaavioilla. Kuvassa 5.21 on yksinkertainen koostekaavio auton rakenteesta. (Kylä-Nikkilä, 2008.)

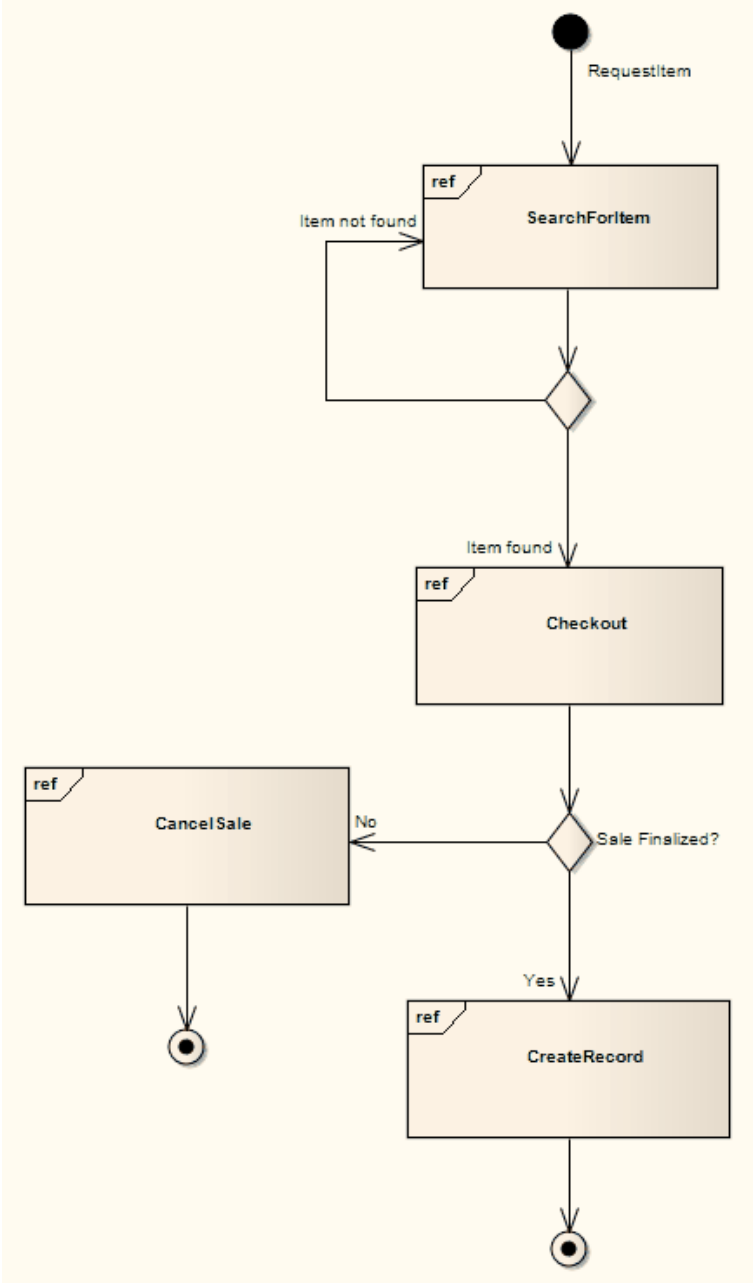


Kuva 5.21 Koostekaavio auton rakenteesta (Kylä-Nikkilä, 2008)

5.3.13 Kokoava vuorovaikutuskaavio

Kokoava vuorovaikutuskaavio voidaan yksinkertaisimmillaan mieltää aktiviteettikaavioksi, jossa aktiviteettina käytetään jotakin vuorovaikutuskaaviota. Sitä käytetään tilanteissa, joissa halutaan saada hyvä kokonaiskuva mallinnettavas-

ta järjestelmästä. Kokoavassa vuorovaikutuskaaviossa voidaan käyttää ajoitus-, kommunikointi- ja sekvenssikaavioita. Jokainen näistä kaavioista keskittyy tiettyyn toimintaan ja siihen liittyvään vuorovaikutukseen hyvin yksityiskohtaisella tasolla. Kokoavalla vuorovaikutuskaaviolla nämä kaaviot saadaan niputettua yhteen isommaksi kokonaisuudeksi (kuva 5.22). (Kylä-Nikkilä, 2008.)



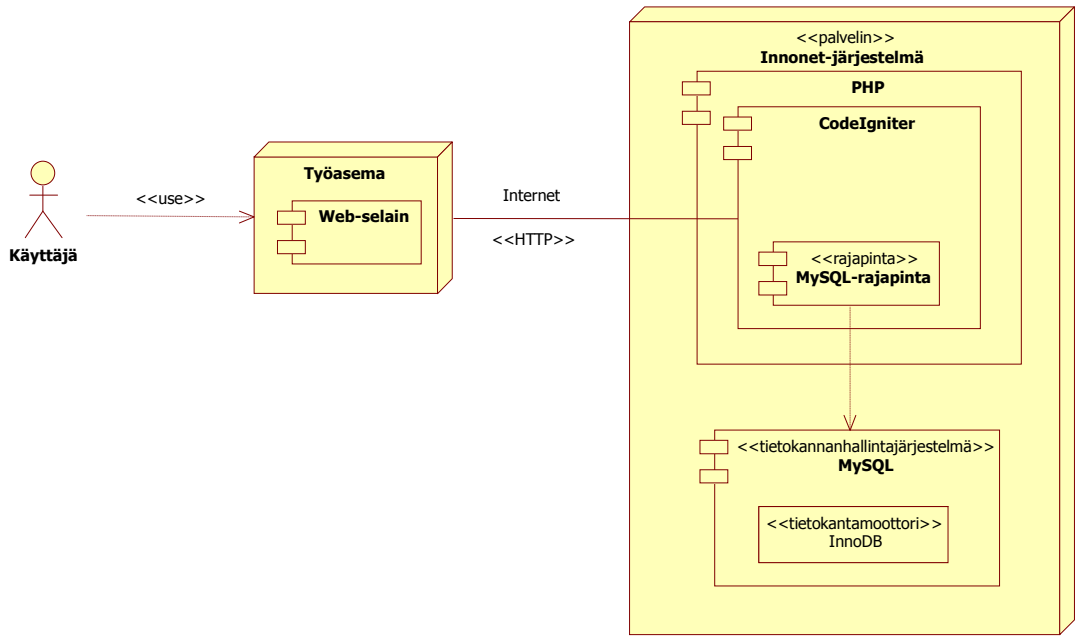
Kuva 5.22 Kokoava vuorovaikutuskaavio myyntiprosessista (Sparx Systems, 2010)

5.3.14 Profiilit

Profiilit ovat tapa laajentaa UML:ää käyttäen sen sisäänrakennettuja laajennusmekanismeja, joita ovat stereotyypit, rajoitukset, nimetyt arvot ja ominaisuudet. Ne ovat kevyt tapa sopeuttaa UML-mallinnuskieltä tiettyyn alustaan tai toimialaan. (Turpeinen, 2004.)

6 TYÖSSÄ KÄYTETTÄVÄT TEKNIIKAT

Innonet-toiminnanohjausjärjestelmää ei ole vielä toteutettu ja tekniikat, joilla järjestelmä toteutetaan, on valittu asiakasvaatimusten pohjalta. Innonet on selainpohjainen toiminnanohjausjärjestelmä. Järjestelmä toteutetaan PHP-kielellä käyttäen hyväksi CodeIgniter PHP MVC -ohjelmistokehystä. Innonetin tietosisällöstä vastaa MySQL-tietokannan-hallintajärjestelmällä ja InnoDB-tietokantamoottorilla varustettu tietokanta. Innonet-järjestelmä sekä tietokanta sijaitsevat Innotek Oy:n toimitilojen ulkopuolella olevalla palvelimella ja Innonetiä käytetään Internetin yli selaimen välityksellä. Innonet-järjestelmän käyttöönottokaavio on hahmoteltu kuvassa 6.1.



Kuva 6.1 Innonet-järjestelmän käyttöönottokaavio

Seuraavaksi käydään läpi tässä opinnäytetyössä käytetyt tekniikat tarkemmin.

6.1 PHP

PHP (PHP: Hypertext Preprocessor) on suosittu skriptikieli, joka sopii erityisesti WWW-sovelluskehitykseen. PHP eroaa monista muista ohjelmointikielistä siinä, että sitä voidaan upottaa suoraan HTML-dokumentin sisälle. Tämä mahdollistaa monipuolisten ja helppokäyttöisten verkkosovellusten luonnin PHP:llä. Syntaksiltaan PHP muistuttaa hyvin paljon C-, Java- ja Perl-kieliä. PHP on palvelinpuolella suoritettava ja tulkattava ohjelmointikieli. Tämä tarkoittaa, että PHP-sivut ja niiden sisältämä koodi käsitellään WWW-palvelimella ennen kuin sivu ladataan asiakkaan selaimelle. PHP:n, kuten myös muiden upotettavien komentosarjakielien ensisijaisena tavoitteena on tarjota nopea ja monipuolinen kehitysympäristö WWW-kehittäjille. PHP perustuu avoimeen lähdekoodiin ja on täysin ilmainen. PHP:llä on myös erittäin kattava tuki eri käyttöjärjestelmille ja laitealustoille. Ohessa on koodiesimerkki PHP-kielestä upotettuna HTML-sivuun (kuva 6.2). (Heinisuo 2004; Heikkilä, 2006; Rundberg, 2009.)

```
1  <html>
2  <head>
3  <title>Ensimmäinen PHP-sivuni</title>
4  </head>
5  <body>
6  <?php
7  echo "Hello World! ";
8  ?>
9  </body>
10 </html>
```

Kuva 6.2 Koodiesimerkki PHP:stä upotettuna HTML-sivuun

6.2 Relaatiotietokannat ja SQL

Relaatiotietokantamallin kehitti IBM:n tutkija Edgar F. Codd 1960-luvun lopulla. Relaatiotietokantamalli perustuu joukkoteoriaan, matematiikan oppeihin ja tietorakenteiden käyttöön. Relaatiotietokanta koostuu tauluista, jotka jakautuvat riveihin ja sarakkeisiin. Jokaisella rivillä on yhtä monta kenttää ja yksikäsitteinen perusavain. Jokainen yksittäinen tieto relaatiotietokannassa voidaan hakea ilmoittamalla taulun nimi, perusavaimen sarakenimi ja avaimen arvo sekä haettavan tiedon sarakenimi. (Hernandez, 2000.)

SQL (Structured Query Language) on kieli relaatiotietokantojen käsittelyyn. ANSI (American National Standards Institute) määritti ensimmäisen SQL-standardin vuonna 1986 (SQL/ANSI-86). Kansainväliseksi ISO-standardiksi (International Organization for Standardization) se hyväksyttiin vuonna 1987. Uusin SQL-standardi on vuonna 2008 standardoitu SQL-2008. (Kolehmainen, 2006.)

6.2.1 MySQL

MySQL on suosittu avoimen lähdekoodin SQL-tietokannan hallintajärjestelmä. MySQL on hyvin nopea ja yksinkertainen verrattuna järeisiin järjestelmiin, kuten esimerkiksi ORACLEen. MySQL:n tärkeimmät piirteet ovatkin nopeus, siirrettävyys, yhteensopivuus eri ohjelmointikielien kanssa ja hinta. MySQL on suosittu WWW-sivujen tietokantana helppoutensa ja hinnan takia. Osittain syynä on myös PHP:n suosio ja MySQL:n yhteensopivuus sen kanssa. MySQL:n kyselykielenä on SQL. MySQL on kuitenkin tehnyt omia muutoksiaan SQL-standardiin. (Heinisuo, 2004; Patosuo, 2009.)

MySQL noudattaa asiakas-palvelin-arkkitehtuuria, jossa sovellukset eivät koskaan käsittele tietokantaa suoraan vaan käsittely tapahtuu aina palvelinohjelman kautta (Heinisuo, 2004).

MySQL-palvelimella voi olla useita tietokantoja, joissa on vaihteleva määrä tauluja. Tietokannan sisäisten käyttöoikeuksien määritykset mahdollistavat eri vaikeustasoisten käyttöoikeuksien luomisen. (Heinisuo, 2004.)

6.2.2 MyISAM

MyISAM on MySQL:n oletustietokantamoottori. Se on tehokas ja vähän levytilaa käyttävä kovalevypohjainen tietokantamoottori. Sen suurin heikkous on transaktioiden ja viiteavaimien puute. Transaktiot huolehtivat tietokannan eheydestä ilman ulkopuolista ohjelmallista tukea. (Oracle, 2010.)

6.2.3 InnoDB

InnoDB-tietokantamoottorin on kehittänyt suomalainen Innobase Oy. InnoDB on suunniteltu raskaisiin tuotantojärjestelmiin ja prosessoimaan suuria tietomääriä. InnoDB on kovalevypohjainen tietokantamoottori ja tarjoaa täyden tuen ACID-transaktioille (Atomicity, Consistency, Isolation, Durability). InnoDB vaatii enemmän kovalevytilaa kuin MyISAM varastoidakseen datansa, mutta tämä lisääntynyt kuormitus korvataan tehokkaalla välimuistin käytöllä. (Oracle, 2010.)

6.3 PHP-ohjelmistokehykset

Ohjelmistokehys helpottaa sovellusten kehitystä, koska se sisältää ratkaisut moniin yleisesti toistuviin tehtäviin. Se sisältää myös valmiin rakenteen koodille, mikä auttaa suunnittelijaa kirjoittamaan hyvää, selkeää ja helposti hallittavaa koodia. Ohjelmistokehyksen avulla suunnittelija voi tehdä monimutkaisia toteutuksia yksinkertaisilla kutsuilla.

PHP-ohjelmistokehykset vähentävät WWW-sovelluksen suunnittelijan tarvetta tehdä samalle operaatiolle koodia moneen kertaan. PHP-ohjelmistokehys helpottaa suunnittelijan työtä yksinkertaistamalla komentoja, joilla esimerkiksi otetaan yhteys tietokantaan.

Suosituimmat MVC-arkkitehtuuriin (Model–View–Controller) perustuvat PHP-ohjelmistokehykset ovat Zend, CodeIgniter, Symfony, Yii ja CakePHP.

Seuraavassa luvussa käsitellään tarkemmin tässä opinnäytetyössä käytettyä PHP-ohjelmistokehystä, CodeIgniteriä.

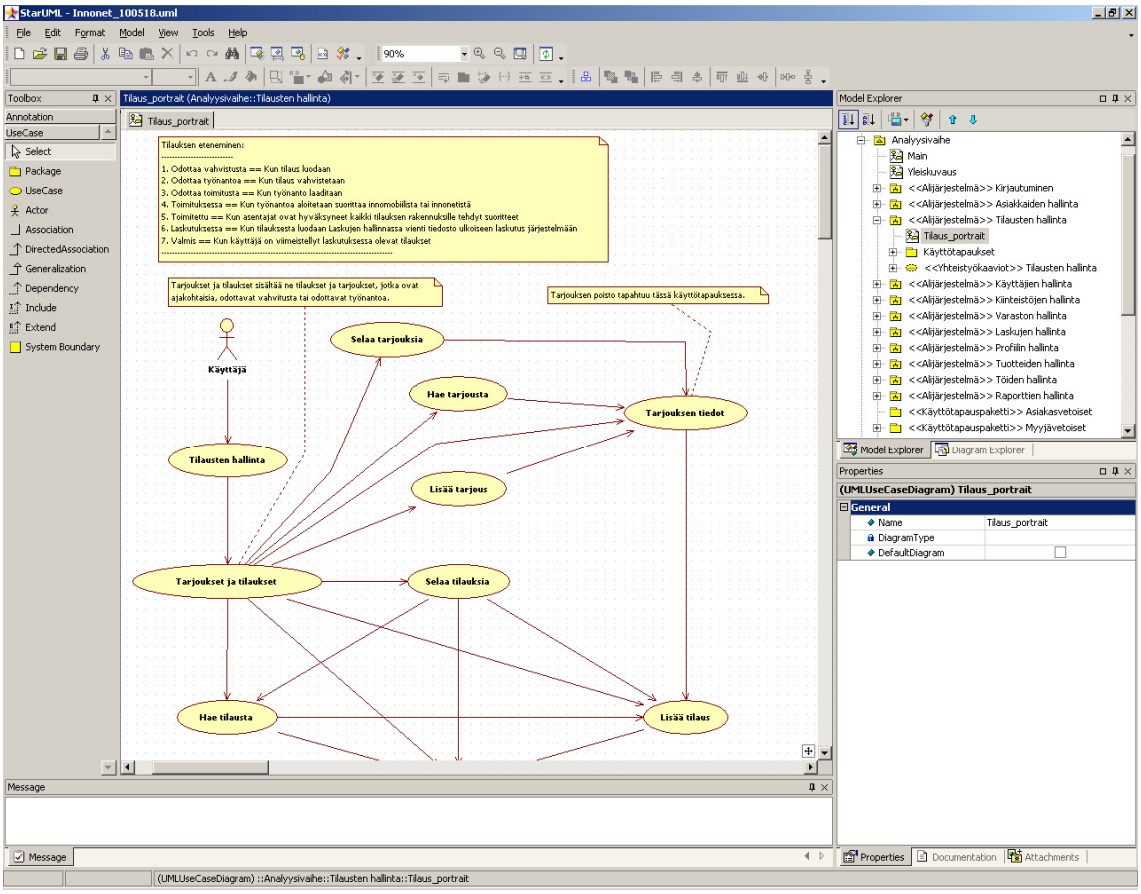
6.3.1 CodeIgniter

CodeIgniter on ilmainen PHP MVC -ohjelmistokehys. Se on kevyt ja erittäin nopea ohjelmistokehys. CodeIgniterissä on matala oppimiskynnys ja kattava dokumentaatio. CodeIgniter tukee PHP-ohjelmointikielen versioita 4 ja 5. CodeIgniterissä on suora tuki yleisimmille tietokantojen hallintajärjestelmille, kuten MySQL, MS SQL, PostgreSQL ja Oracle.

6.4 StarUML

StarUML on vapaan lähdekoodin vastine kaupallisille UML-mallinnusohjelmistoille. Se on nopea, joustava ja laajennettava UML-työkalu Windows-alustalle. StarUML:ssä kaikki projektin kaaviot tallennetaan yhteen projektitiedostoon.
StarUML tukee UML:n versiota 2.0, mutta se ei kuitenkaan tue kaikkia UML:n kaaviotyyppejä. Puuttuvia kaaviotyyppejä ovat ajoitus-, olio-, kommunikointi- ja pakettikaaviot. StarUML:ssä on myös tuki MDA-teknologialle (Model-Driven Architecture). (Kylä-Nikkilä, 2008.)

StarUML:n käyttöliittymä on selkeä ja helppokäyttöinen. Käyttöliittymän rakenne selviää alla olevasta kuvasta (kuva 6.3), jossa on avattuna käyttötapauskaavio.



Kuva 6.3 StarUML:n käyttöliittymä

7 PROJEKTIN KULKU

7.1 Projektin organisaatio

Projektin johtamisesta on vastannut ohjausryhmä. Ohjausryhmä on koostunut asiakkaasta, projektin ohjaajasta ja projektiryhmästä. Projektiryhmän puheenjohtajana ja sihteerinä on toiminut projektipäällikkö Niko Rissanen. Projektiryhmä on kokoontunut säännöllisesti viikkopalavereihin. Ohjausryhmä on pitänyt myös palavereja noin kerran kuukaudessa. Alla olevasta taulukosta (taulukko 7.1) ilmenee ohjausryhmän kokoonpano ja vastuualueet projektissa.

Taulukko 7.1 Ohjausryhmän kokoonpano

Nimi	Nimike	Tehtävä
Niko Rissanen	projektipäällikkö	Innonetin määrittely ja suunnittelu
Timo Paajanen	suunnittelija	Innonetin määrittely ja suunnittelu
Olli Pikarinen	toteuttaja	Innonetin toteutus
Outi Tikkala	käyttöliittymäsuunnittelija	Innonetin käyttöliittymän suunnittelu ja toteutus
Ossi Sironen	testaaja	Innonetin testaus
Markus Heikkinen	asiakas (Innotek Oy)	Innonet-projektin asiakas ja opinnäytetyön ohjaaja
Martti Ylä-Jussila	opinnäytetyöohjaaja (Saimaan ammattikorkeakoulu)	Opinnäytetyön ohjaaja

7.2 Esitutkimus ja alustus

Opinnäytetyöprosessi aloitettiin valitsemalla aihe. Opinnäytetyön aiheeksi valittiin Innotek Oy:n tarjoama aihe, Innonet-web-käyttöliittymä. Kun opinnäytetyön aihe oli valittu, tehtiin aiheesta esitutkimus ja pidettiin aloituspalaveri. Aloituspalaverin jälkeen laadittiin projektisuunnitelma, jossa määriteltiin projektin tavoitteet, aikataulutus sekä sovittiin tulevat palaverit.

7.3 Vanhan Innonet-järjestelmän opiskelu ja dokumentointi

Itse opinnäytetyö aloitettiin tutustumalla jatkokehitettävään Innonet-järjestelmään. Ensin omalle koneelle asennettiin XAMPP-serveriohjelmisto, joka mahdollistaa Innonet-järjestelmän ajamisen paikallisesti. Kun Innonet-järjestelmä oli saatu asennettua, aloitettiin itse järjestelmään tutustuminen toiminnallisuuden kartoittamisella ja lähdekoodien lukemisella.

Koska opinnäytetyön tekijällä ei ollut aikaisempaa kokemusta PHP-ohjelmointikielestä, se opiskeltiin opinnäytetyöprojektin aikana. Kielen opetteleminen ei kuitenkaan tuottanut suuria vaikeuksia, koska kielen syntaksi oli samankaltainen Java- ja C-kieliin verrattuna.

Projektin edetessä selvisi, ettei järjestelmästä oltu tehty minkäänlaista dokumentaatiota. Joten ennen kuin järjestelmää alettaisiin jatkokehittää, päätettiin laatia jo toteutetusta osiosta toiminnallinen määrittelydokumentti. Toiminnallisen määrittelyn kirjoittaminen aloitettiin tutustumalla tietokantaan ja järjestelmän toiminnallisuuteen.

Tietokantaa tutkittaessa huomattiin, ettei se vastaa asiakkaan järjestelmälle asettamiin vaatimuksiin. Tietokannasta löytyi paljon epäjohdonmukaisuutta ja rakenneongelmia. Pitkän harkinnan tuloksena päätettiin tietokanta suunnitella ja toteuttaa kokonaan uusiksi puhtaalta pöydältä. Samalla päätettiin vaihtaa tietokantamoottori vanhanaikaisesta MyISAM:sta uuteen InnoDB-moottoriin. InnoDB-tallennusmoottori tukee muun muassa transaktioita.

Tässä vaiheessa Timo Paajanen tuli mukaan projektiin. Paajanen toi mukanaan projektiin tärkeää tietotaitoa mutta ennen kaikkea rohkeutta tehdä projektin suhteen isoja ja ratkaisevia päätöksiä, kuten tietokannan uusiminen.

Projekti jatkui vanhan järjestelmän määrittelemisellä ja asiakaspalavereilla. Palavereissa tuli ilmi monia muutosehdotuksia, myös vanhaan järjestelmään. Muutosehdotuksia tuli niin paljon, että oli toisen suuren päätöksen aika. Tämä toinen päätös oli määritellä koko järjestelmä alusta asti täysin uudestaan.

7.4 Innonet-järjestelmän uudelleen määrittely

Tässä vaiheessa olivat opinnäytetyön vaatimukset muuttuneet alkuperäisestä Innonet-web-käyttöliittymän ja raportointityökalujen jatkokehityksestä koko toiminnanohjausjärjestelmän kattavan määrittelydokumentaation tuottamiseen ja uuden tietokannan määrittelyyn, suunnitteluun ja toteutukseen.

Järjestelmän määrittely sujui ilman suurempia ongelmia ja vaatimuksia tarkennettiin asiakkaan kanssa pidettävissä palavereissa. Kokonaiskuva järjestelmän suuruudesta alkoi pikku hiljaa hahmottua ja ymmärrettiin, että näin iso järjestelmä tarvitsisi erittäin vakaan rungon vastatakseen sille asetettuihin vaatimuksiin. Vastaus tähän ongelmaan löytyi PHP MVC -ohjelmistokehyksestä. Seuraavat viikot kuluivat eri PHP MVC -ohjelmistokehysten vertailuissa ja vihdoin löydettiin projektille sopiva ohjelmistokehys, CodeIgniter.

Päätös CodeIgniterin käyttämisestä projektissa oli helppo. Se on kevyt ja nopea ohjelmistokehys mutta myös syntaksiltaan looginen ja helppo oppia. Tässä vaiheessa tiedettiin jo, ettei projektissa oleva työpanos riittäisi projektin läpivientiin, joten ohjelmistokehyksen tuli olla sellainen, että myös uudet tekijät pääsevät sisälle järjestelmään nopeasti.

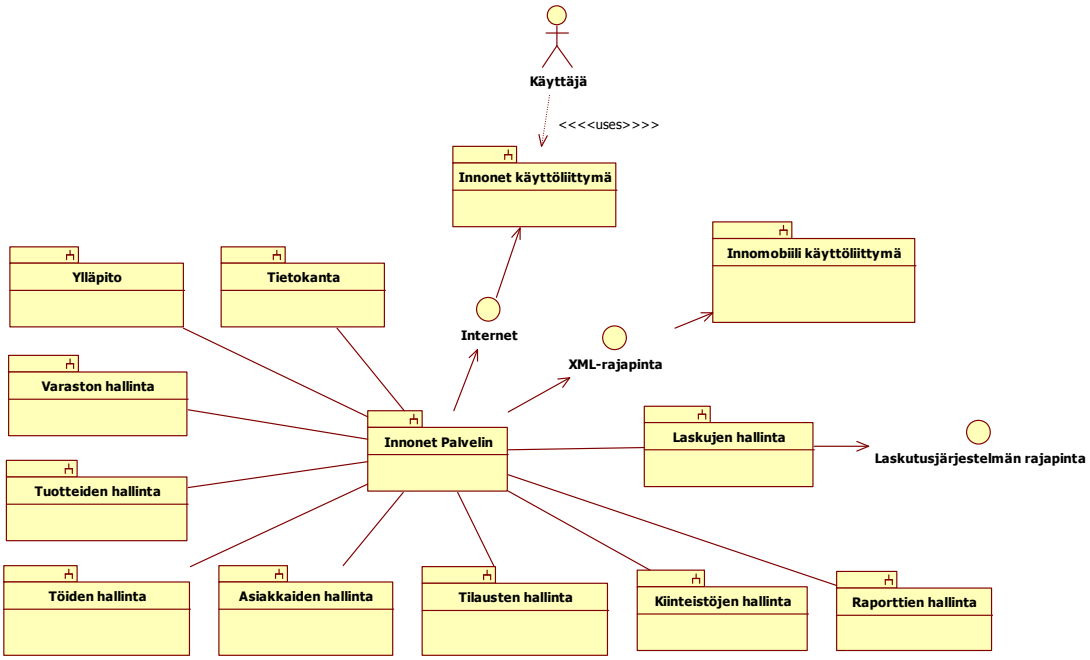
8 UUDEN INNONET-JÄRJESTELMÄN MÄÄRITTELYN ESITTELY

Innonet on keskisuuren LVI-alalla toimivan Innotek Oy:n Energo-ohjelmaan keskittyvä toiminnanohjausjärjestelmä. Järjestelmä käsittää asiakkaiden, kiinteistöjen, tilausten, töiden, raporttien, laskujen, tuotteiden ja varaston hallinnan sekä ylläpidon.

Innonet on tarkoitettu Innotek Oy:n työntekijöiden käyttöön. Tulevaisuudessa järjestelmää aiotaan laajentaa myös Innotek Oy:n asiakkaiden käyttöön.

Innonet-toiminnanohjausjärjestelmä toteutetaan PHP-ohjelmointikielellä käyttäen hyväksi CodeIgniter PHP MVC -ohjelmistokehystä. Innonetin valtavasta tietosisällöstä vastaa MySQL-tietokannanhallintajärjestelmällä ja InnoDB-tietokantamoottorilla toimiva 85 taulua käsittävä tietokanta. Innonetiä käytetään Internet-selaimen välityksellä.

Innotek Oy:n Energo-asentajat käyttävät apunaan Innomobiili-järjestelmää. Innomobiili on Pocket PC:ssä toimiva sovellus, jolla Energo-asentajat hallitsevat päivittäisiä työtehtäviä, tekevät kartoitus- ja asennuskohteissa Energo-kartoituksia sekä asennuksia, hallitsevat työtuntiseurantaa ja suorittavat vika- ja työraportointia. Innomobiili käyttää tietojen tallentamiseen väliaikaisesti Pocket PC:n muistia, mutta lopullinen tiedon säilytyspaikka on sama tietokanta, jota Innonet-järjestelmä käyttää hyväkseen. Innomobiilin määrittely ja toteutus toteutetaan omana opinnäytetyönään eikä siihen tässä opinnäytetyössä puututa tämän enempää. Kuvassa 8.1 on kuvattu Innonet-järjestelmän looginen näkymä, joka sisältää kaikki Innonetiin liittyvät komponentit.

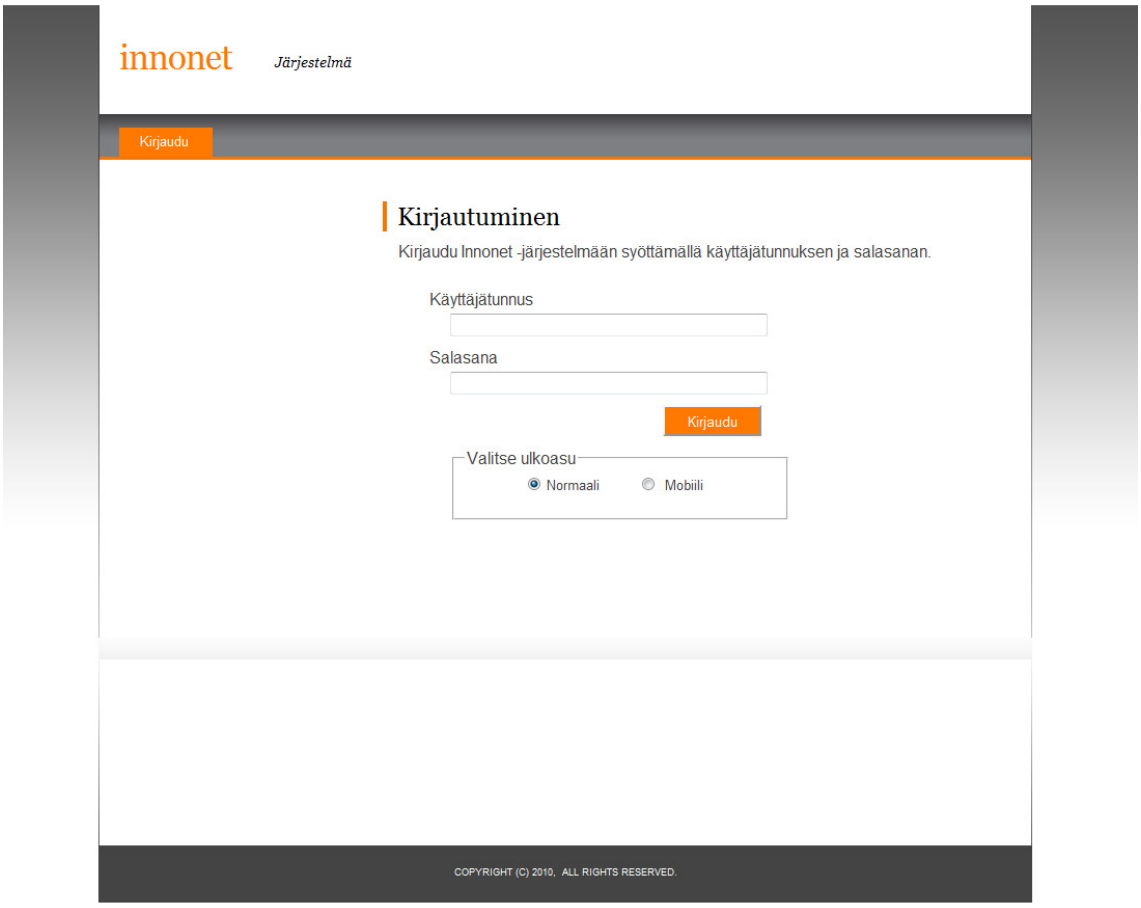


Kuva 8.1 Innonet-järjestelmän looginen näkymä

Seuraavaksi käydään läpi lyhyesti Innonetin eri osa-alueiden toiminta.

8.1 Kirjautuminen

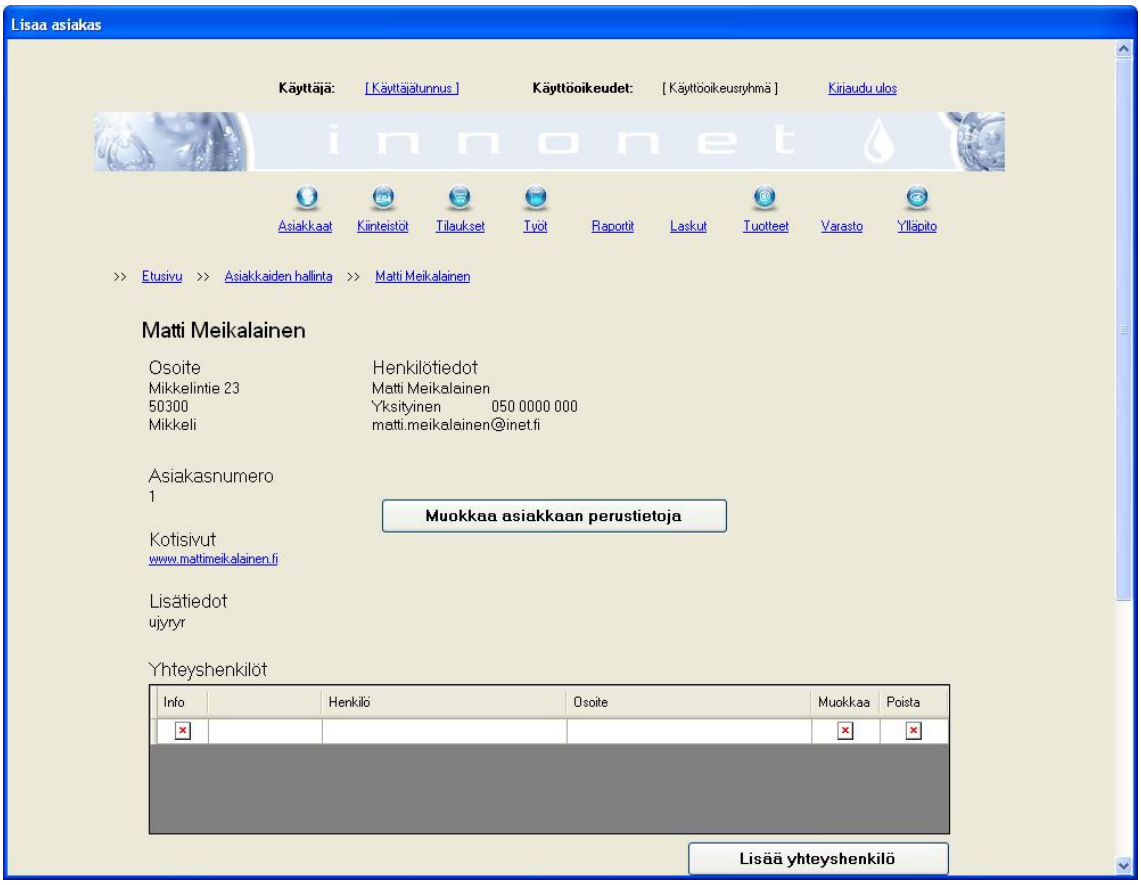
Kirjautuminen käsittää kaksi eri toimintoa, sisäänkirjautumisen ja uloskirjautumisen. Sisäänkirjautumisessa syötetään käyttäjätunnus ja salasana, jolloin järjestelmä suorittaa tunnusten tarkastamisen ja onnistuneen sisäänkirjautumisen jälkeen aloittaa käyttöistunnon. Uloskirjautumisessa käyttöistunto lopetetaan ja istuntoon sidotut resurssit vapautetaan. Kuvassa 8.2 on hahmoteltu kirjautumisen käyttöliittymäkuvausta. Nämä käyttöliittymäkuvaukset ovat vasta alustavia ja tulevat luultavasti muuttumaan projektin edetessä.



Kuva 8.2 Kirjautumisen käyttöliittymähahmotelma

8.2 Asiakkaiden hallinta

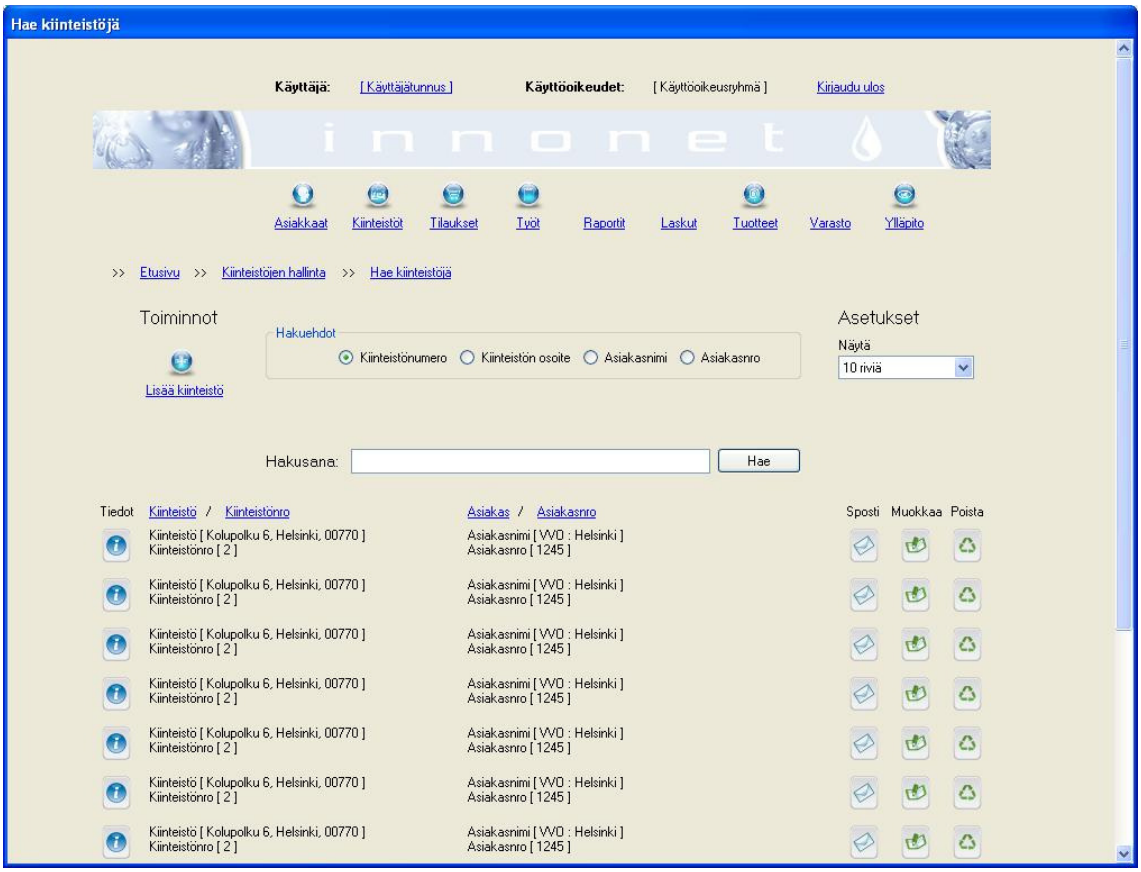
Asiakkaiden hallinnassa pidetään kirjaa Innotek Oy:n Energo-ohjelman asiakkaista ja yhteyshenkilöistä. Asiakkaat jaetaan eri tyyppeihin, esimerkiksi kerta-asiakas, kanta-asiakas ja suurasiakas. Yhteyshenkilöitä voi olla myös erityyppisiä, esimerkiksi isännöitsijä, huoltoyhtiö ja asiakkaan edustaja. Kuvassa 8.3 on käyttöliittymähahmotelma siitä, kuinka järjestelmässä näytetään asiakkaan tiedot.



Kuva 8.3 Asiakkaan tietojen käyttöliittymähahmotelma

8.3 Kiinteistöjen hallinta

Kiinteistöjen hallinnassa pidetään kirjaa niistä kiinteistöistä, jotka liittyvät asiakkaan tekemiin tilauksiin. Kiinteistöillä on myös historiatietoja kuten peruskorjausvuosi. Kiinteistöihin liittyvät henkilöt ja yritykset, kuten isännöitsijä ja huoltoyhtiö, näkyvät kiinteistöjen hallinnasta. Kiinteistöjen haun käyttöliittymähahmotelma näkyy kuvassa 8.4.



Kuva 8.4 Kiinteistöjen haun käyttöliittymähahmotelma

8.4 Tilauksien hallinta

Tilausten hallinnassa luodaan, seurataan ja käsitellään tilauksia sekä tarjouksia. Normaali tilausprosessi alkaa asiakaskontaktilla, josta seuraa tarjous asiakkaalle. Hyväksytystä tarjouksesta syntyvä tilaus lisätään järjestelmään. Tilaukset jaetaan Energo-kartoitustilauksiin ja -asennustilauksiin. Tilauksen voi myös tehdä ilman tarjousta.

8.5 Töiden hallinta

Jokaisesta järjestelmään lisätystä ja vahvistetusta tilauksesta tehdään työnanto. Työnannoissa määritellään tehtävät työt, asennettavat tuotteet ja tarkastettavat kohteet kiinteistöissä. Työnannot tehdään ensisijaisesti Innomobiili-järjestelmästä, mutta ne voidaan suorittaa myös Innonetin puolelta. Töiden hallinnasta seurataan myös Energo-asentajien työaikoja sekä työnantoihin liittyviä

asennus- ja tuntitöitä. Myös asentajien työaikaraportit tuotetaan töiden hallinnassa.

8.6 Raporttien hallinta

Raporttien hallinnassa käsitellään kaikki järjestelmässä tuotettavat raportit. Niitä ovat muun muassa Energo-kartoituksiin liittyvät kiinteistökohtaiset Energo-kulutusanalyysit ja vikaraportit sekä asiakaskohtaiset yhteenvetoraportit. Energo-asennuksista tuotetaan asennuserittely ja asennuksen yhteenvetoraportti. Laajasta kuntoraportoinnista tuotetaan laaja kuntoraportti.

8.7 Laskujen hallinta

Laskujen hallinta tuottaa tehtävästä laskusta vientitiedostot määritellyn rajapinnan pohjalta. Vientitiedostot siirretään ulkoiseen laskutusjärjestelmään jatkotoimenpiteitä varten. Innonet-järjestelmässä ei syvennytä laskutusprosessiin edellä mainittua enempää.

8.8 Tuotteiden hallinta

Tuotteiden hallinnassa käsitellään kaikkia järjestelmään liittyviä tuotteita. Tuotteita, jotka eivät sidoksissa varaston hallintaan, käsitellään abstraktisti. Tuotteet voidaan jaotella tuoteperheisiin, tuoteryhmiin ja kategorioihin. Tuotteiden hallinnasta voidaan luoda uusia tuoteperheitä ja tuoteryhmiä sekä jaotella tuotteita niihin.

8.9 Varaston hallinta

Varaston hallinta sisältää järjestelmään kuuluvien varastojen logistiikan. Varaston hallinnasta voidaan seurata saapuvia, lähteviä, asennettavia sekä poistettavia tuotteita. Lisäksi varaston hallinnassa voidaan luoda uusia varastoja sekä poistaa vanhoja ja siirrellä tuotteita varastojen välillä. Tuotteet kuvataan varas-

toissa tuote-erinä, ja jokaisella tuote-erällä on oma sisäänostohintansa ja kappalemääränsä. Tuote-eriä käsitellään varastoissa tapahtumapohjaisesti.

8.10 Ylläpito

Ylläpidolla hallitaan käyttäjätilejä, käyttäjäryhmiä sekä käyttäjäryhmien käyttöoikeuksia. Tämän lisäksi ylläpidosta voidaan muokata kaikkia järjestelmässä esiintyviä tyyppitietoja, kuten asiakastyyppejä, kiinteistötyyppejä ja tilaustyyppejä.

8.11 Tietokanta

Innonetin tietokanta käyttää hyväkseen MySQL-tietokannanhallintajärjestelmää. Tietokannan taulut toimivat InnoDB-tietokantamoottorilla ja näin ollen tukevat transaktioneita. Koska PHP:ssä on natiivituki MySQL:lle, on tietokanta integroitu suoraan kiinni itse toiminnanohjausjärjestelmään.

9 YHTEENVETO

Ryhdyin tekemään opinnäytetyötä tästä aiheesta, koska aihe oli kiinnostava ja samalla oppisin uuden ja nykyisillä työmarkkinoilla tärkeän ohjelmointikielen, PHP:n. Opinnäytetyön aikana en kuitenkaan koodannut riviäkään koodia, mutta jatkan vielä projektissa tämän opinnäytetyön päätyttyä, joten saan tärkeää kokemusta PHP-ohjelmoinnista. Vaikka opinnäytetyö ei lopulta vastannut alkuperäistä aihetta, olen kuitenkin tyytyväinen projektissa tehtyihin päätöksiin.

Mielestäni koko järjestelmän uudelleen määrittely sekä tietokannan uusiminen oli oikea päätös. Toiminnanohjausjärjestelmä on yrityksen selkäranka, yksi tärkeimmistä yrityksessä olevista järjestelmistä. Se on tehtävä kunnolla. Nyt järjestelmällä on hyvä perusta, josta projektia voidaan jatkaa turvallisin mielin.

Tämän opinnäytetyön aikana Innonet-toiminnanohjausjärjestelmän toiminnallinen määrittely saatiin viimeistä silausta vaille valmiiksi. Myös Innonetin tietokanta saatiin kokonaisuudessaan määriteltyä, suunniteltua ja toteutettua.

Projektissa riittää työtä vielä pitkäksi aikaa, ja jatkan projektin parissa tämän opinnäytetyön päätyttyä toiminnallisen määrittelyn viimeistelyllä, järjestelmän suunnittelulla ja mahdollisesti myös toteutuksella. Toiminnallisen määrittelyn valmistuttua on edessä teknillisen määrittelydokumentin tuottaminen. Teknillisessä määrittelydokumentissa suunnitellaan järjestelmän käyttämä arkkitehtuuri, moduulit sekä muut tekniset ratkaisut. Tätä vaihetta kutsutaan myös suunnitteluvaiheeksi. Suunnitteluvaiheen jälkeen seuraa koko järjestelmän toteutus. Toteutusvaiheessa itse järjestelmä ohjelmoidaan PHP:llä käyttäen CodeIgniter-ohjelmistokehystä. Kun järjestelmä on saatu toteutettua, seuraa järjestelmän testaus. Testauksen tarkoituksena on löytää järjestelmästä virheitä. Lopulta tapahtuu järjestelmän käyttöönotto. Käyttöönotossa järjestelmä asennetaan asiakkaan palvelimelle ja opastetaan asiakasta järjestelmän käytössä. Siihen on kuitenkin vielä pitkä ja kivinen tie, joka tuo varmasti paljon uusia haasteita.

KUVAT

Kuva 1.1 Vuorokauden vedenkulutuksen jakaantuminen (Innotek Oy, 2000a), s. 8

Kuva 2.1 Energo-ohjelman vaiheet (Innotek Oy, 2000a), s. 11

Kuva 4.1 Tyypillinen vesiputousmalli (Haikala & Märijärvi, 2004), s. 16

Kuva 4.2 Toiminnallisen määrittelyn sisältörunko (Haikala & Märijärvi, 2004), s. 17

Kuva 4.3 Evo-malli (Ilkko, 2008), s. 19

Kuva 4.4 Protoilumalli (Ilkko, 2008), s. 20

Kuva 4.5 Spiraalimalli (Ilkko, 2008), s. 21

Kuva 4.6 XP-menetelmä (Ilkko, 2008), s. 22

Kuva 4.7 Scrum-menetelmä (Haikala, 2009), s. 23

Kuva 5.1 Henkilön käsitekaavio, s. 24

Kuva 5.2 Henkilo-taulun yksilötyyppikuvaus, s. 25

Kuva 5.3 Kiinteistön haun sanallinen käyttötapauskuvaus, s. 26

Kuva 5.4 UML-kaaviohierarkia, s. 27

Kuva 5.5 Käyttötapauskaavion toimija, s. 28

Kuva 5.6 Kiinteistöjen hallinnan käyttötapauskaavio Innonet-järjestelmässä, s. 29

Kuva 5.7 Luokkakaavio hammaslääkärin toiminnasta, s. 30

Kuva 5.8 Esimerkki riippuvuussuhteesta (Kylä-Nikkilä, 2008), s. 31

Kuva 5.9 Esimerkki assosiaatiosta kahden luokan välillä (Kylä-Nikkilä, 2008), s. 31

Kuva 5.10 Esimerkki komposiitista (Kylä-Nikkilä, 2008), s. 32

Kuva 5.11 Esimerkki periyttämisestä (Luukkainen & Laine, 2009), s. 32

Kuva 5.12 Oliokaavio binääripuusta (Kylä-Nikkilä, 2008), s. 33

Kuva 5.13 Tilauksen kuittauksen sekvenssikaavio Innonet-järjestelmässä, s. 34

Kuva 5.14 Kommunikointikaavio tulostinpalvelimen toiminnasta (Kylä-Nikkilä, 2008), s. 35

Kuva 5.15 Pakkauskaavio (Kylä-Nikkilä, 2008), s. 36

Kuva 5.16 Tilakaavio puhelimen toiminnasta (OMG, 2009), s. 37

Kuva 5.17 Aktiviteettikaavio kanakastikkeen valmistamisesta (Kylä-Nikkilä, 2008), s. 38

Kuva 5.18 Komponenttikaavio (Kylä-Nikkilä, 2008), s. 39

Kuva 5.19 Sijoittelukaavio yrityksen laitteistoarkkitehtuurista (Kylä-Nikkilä, 2008), s. 40

Kuva 5.20 Ajoituskaavio liikennevalojen toiminnasta (Altova, 2010), s. 41

Kuva 5.21 Koostekaavio auton rakenteesta (Kylä-Nikkilä, 2008), s. 41

Kuva 5.22 Kokoava vuorovaikutuskaavio myyntiprosessista (Sparx Systems, 2010), s. 42

Kuva 6.1 Innonet-järjestelmän käyttöönottokaavio, s. 43

Kuva 6.2 Koodiesimerkki PHP:stä upotettuna HTML-sivuun, s. 44

Kuva 6.3 StarUML:n käyttöliittymä, s. 48

Kuva 8.1 Innonet-järjestelmän looginen näkymä, s. 53

Kuva 8.2 Kirjautumisen käyttöliittymähahmotelma, s. 54

Kuva 8.3 Asiakkaan tietojen käyttöliittymähahmotelma, s. 55

Kuva 8.4 Kiinteistöjen haun käyttöliittymähahmotelma, s. 56

TAULUKOT

Taulukko 7.1 Ohjausryhmän kokoonpano, s. 49

LÄHTEET

Altova. 2010. UML Timing Diagrams.
http://www.altova.com/umodel/timing-diagrams.html (luettu 26.5.2010)

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto. Helsinki.
Talentum Media Oy.

Haikala, I. 2009. Ohjelmistotuotannon perusteet.
Tampereen teknillinen yliopisto.
http://www.cs.tut.fi/~otupk/luennot/kalvot/alku.pdf (luettu 3.6.2010)

Heikkilä, H. 2006. Tekstiviestipalvelun kehittäminen PHP:n ja XML:n avulla.
Opinnäytetyö, Tampereen ammattikorkeakoulu.
http://urn.fi/URN:NBN:fi:amk-201003064904 (luettu 9.5.2010)

Heinisuo, R. 2004. PHP ja MySQL: Tietokantapohjaiset verkkopalvelut. Helsinki.
Talentum Media Oy.

Hernandez, M. 2000. Tietokannat: Suunnittelu ja toteutus käytännössä.
Helsinki. IT Press.

Ilkko, L. 2008. Ohjelmistotekniikka.
Oulun ammattikorkeakoulu.
http://www.students.oamk.fi/~s6matu00/sekalaista/ot/teema2.ppt (luettu
2.6.2010)

Immonen, J. 2003. Johdatus ohjelmistotuotantoon.
Joensuun yliopisto.
http://cs.joensuu.fi/~jimmonen/jot_moniste/jot_moniste_121.html (luettu
8.5.2010)

Innotek Oy. 2000a. Energo-ohjelma esityskalvot.
http://www.innotek.fi/datapankki/esityskalvot1428kt.pdf (luettu 3.5.2010)

Innotek Oy. 2000b. Lehdistötiedote.
http://www.innotek.fi/datapankki/lehdistotiedote.pdf (luettu 3.5.2010)

Järvelä, E. & Puusaari, E. 2005. UML-käsikirja.
Jyväskylän yliopisto.
http://urn.fi/URN:ISBN:951-39-2153-0 (luettu 17.5.2010)

Kettunen, J. & Simons, M. 2001. Toiminnanohjausjärjestelmän käyttöönotto pk-
yrityksessä.
Valtion teknillinen tutkimuskeskus.
http://www.vtt.fi/inf/pdf/julkaisut/2001/J854.pdf (luettu 24.5.2010)

Kolehmainen, K. 2006. PHP ja MySQL: Teoriasta käytäntöön. Helsinki.
Readme.fi

Korpimies, K. 2005. Johdatus ohjelmistotuotantoon ja UML-kieleen.
Helsingin yliopisto.
http://www.helsinki.fi/~korpimie/ohjelmistotekniikka/ (luettu 2.6.2010)

Kylä-Nikkilä, J. 2008. UML-kaaviot.
Pro Gradu-tutkielma, Tampereen yliopisto.
http://www.cs.uta.fi/research/thesis/masters/Kyla-Nikkila_Jouni.pdf (luettu 17.5.2010)

Laine, H. 2005. Tietokantojen perusteet: Käsiteanalyysi.
http://www.cs.helsinki.fi/u/laine/tkp/tietomallit/kasiteanalyysi.html (luettu 17.5.2010)

Luukkainen, M. & Laine, H. 2009. Ohjelmistojen mallintaminen.
Helsingin yliopisto.
http://www.cs.helsinki.fi/u/mluukkai/ohmas09/ohma.pdf (luettu 17.5.2010)

Maailma.net. Vesi – katoava luonnonvara.
http://maailma.net/artikkelit/vesi_katoava_luonnonvara (luettu 4.5.2010)

Motiva Oy. 2009. Vedenkulutus.
http://www.motiva.fi/koti_ja_asuminen/mihin_energiaa_kuluu/vedenkulutus (luettu 4.5.2010)

Object Management Group. 2009. UML Superstructure Specification, v2.2.
http://www.omg.org/spec/UML/2.2/Superstructure/PDF/ (luettu 26.5.2010)

Oracle. 2010. MySQL 5.5 Reference Manual.
http://dev.mysql.com/doc/refman/5.5/en/index.html (luettu 16.5.2010)

Patosuo, M. 2009. MySQL tietokantavarastointimoottori pienyrityksen tietovarastointiin.
Opinnäytetyö, Lahden ammattikorkeakoulu.
http://urn.fi/URN:NBN:fi:amk-200905072546 (luettu 9.5.2010)

Rundberg, M. 2009. PHP:n ja ASP.NETin soveltuvuus toiminnanohjausjärjestelmäprojektin toteutukseen : case: SF-Data Osuuskunta.
Opinnäytetyö, Lahden ammattikorkeakoulu.
http://urn.fi/URN:NBN:fi:amk-200905072547 (luettu 9.5.2010)

Saarinen, J. 2008. Testaus osana ohjelmistoprojektia.
Opinnäytetyö, Tampereen ammattikorkeakoulu.
http://urn.fi/URN:NBN:fi:amk-201003065011 (luettu 1.6.2010)

Sparx Systems Pty Ltd. 2010. Interaction Overview Diagram.
http://www.sparxsystems.com/uml_tool_guide/uml_dictionary/interactionoverviewdiagram.htm (luettu 26.5.2010)

Tapola, V. 2004. Käyttötapausanalyysi ja testaus.
Joensuun yliopisto.
http://cs.joensuu.fi/tSoft/dokumentit/tSoft20040915_KTA_et_testaus.pdf (luettu 17.5.2010)

TIEKE. 2008. ERP luultua tärkeämpi pk-yritykselle.
http://www.tieke.fi/tieke/tieken_tiedotteet_2008/erp_luultua_tarkeampi_pk-yrityks/ (luettu 4.5.2010)

Turpeinen, J. 2004. UML-laajennusten arviointi: sovellusalueena WWW-sovellukset.
Pro Gradu-tutkielma, Jyväskylän yliopisto.
https://jyx.jyu.fi/dspace/bitstream/handle/123456789/12477/G0000655.pdf (luettu 23.5.2010)

Wargh, M. 2009. Siirtyminen elinkaarimallista ketterään ohjelmistoliiketoimintaan: hallinnoinnin haasteet.
Pro Gradu -tutkielma, Helsingin yliopisto.
http://www.cs.helsinki.fi/u/paakki/Semik09-Wargh.pdf (luettu 1.6.2010)

Äyrämö, S. 2002. Reaaliaikajärjestelmän mallintaminen UML-kuvausmenetelmän avulla.
Pro Gradu -tutkielma, Jyväskylän yliopisto.
http://users.jyu.fi/~samiayr/pdf/gradu_ayramo.pdf (luettu 8.5.2010)