

Saimaan ammattikorkeakoulu
Tekniikka Imatra
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tuukka Pitkänen

Innoweb-toiminnanohjausjärjestelmä

Opinnäytetyö 2011

Tiivistelmä

Tuukka Pitkänen

Innoweb-toiminnanohjausjärjestelmä, 45 sivua

Saimaan ammattikorkeakoulu

Tekniikka Imatra

Tietotekniikan koulutusohjelma

Ohjelmistotekniikka

Opinnäytetyö 2011

Ohjaajat: Lehtori Martti Ylä-Jussila, Saimaan ammattikorkeakoulu

Toimitusjohtaja Markus Heikkinen, Innotek Oy

Asiakkaana opinnäytetyössä oli lieksalainen LVI-alan pienyritys Innotek Oy. Asiakas tarvitsee toiminnanohjausjärjestelmän ohjaamaan ja automatisoimaan liiketoimintaprosessia. Tämän opinnäytetyön tavoitteena oli töiden hallinnan määrittely, suunnittelu ja toteutus osana Innoweb-toiminnanohjausjärjestelmää.

Innoweb-järjestelmän (myös Innonet) tarkoitus on vähentää paperityön määrää yrityksessä sekä tehostaa liiketoimintaprosessia. Tarkoitus on myös helpottaa asentajien töiden hallintaa ja vähentää asentajien liikkumista paikkakuntien välillä. Lisäksi varastotietojen seuranta ja raportointi helpottuvat.

Työ aloitettiin tutustumalla asiakkaan liiketoimintaan sekä aiemmin toteutettuun järjestelmään sekä toiminnalliseen määrittelyyn ja kartoittamalla asiakkaan tarpeet mahdollisimman tarkasti.

Opinnäytetyön tuloksena toteutettiin Innoweb-järjestelmään töiden hallintaosio, jonka avulla tilauksista laaditaan työnannot, jotka määrätään asentajille ja aikataulutetaan. Lisäksi tämän työn puitteissa jatkokehitettiin järjestelmää laajemmin sekä suunniteltiin ja määriteltiin joitain muita osioita uudestaan.

Järjestelmän kehitys jatkuu vielä.

Asiasanat: Innonet, toiminnanohjausjärjestelmä, CodeIgniter, PHP, Energosäästöohjelma, Innotek Oy, määrittely, JavaScript, jQuery

Abstract

Tuukka Pitkänen
Innoweb Enterprise Resource Planning, 45 Pages
Saimaa University of Applied Sciences
Technology, Imatra
Degree Program in Information technology
Software Engineering
Bachelor's Thesis 2011
Instructor(s): Mr Markus Heikkinen, CEO, Innotek Oy
Mr Martti Ylä-Jussila, Senior Lecturer, Saimaa UAS

The client in this thesis was Innotek Oy from Lieksa, a company in the HVAC area of business. The client needs an enterprise resource planning system to control and manage their business process. The purpose of this thesis was to specify, design and implement the task management subsystem of the Innoweb ERP system.

The purpose of the Innoweb ERP system is to reduce the amount of paperwork in the company and speed up the business process by automating the information processing.

The thesis was started by assessing and analyzing the existing functional requirement document and the old Innonet system and by assessing requirements as carefully as possible. The business process had not been modeled or documented so the first thing to do was to understand the process.

The result of this thesis was the task management subsystem, which allows the creation of task assignments from orders, delegating them to the employees, and scheduling them. Other parts of the system were also developed further and some parts redesigned and reimplemented.

The development of the system will continue.

Keywords: Innonet, Enterprise Resource Planning, CodeIgniter, PHP, Energy Program, Innotek Oy, JavaScript, JQuery, Functional Specification

Termit ja lyhenteet

ACID	Atomicity, Consistency, Isolation, Durability. Tietokannan eheyden turvaamiseksi tarkoitettuja periaatteita.
AJAX	Asynchronous JavaScript And XML, kokoelma tekniikoita www-sivujen vuorovaikutteisuuden parantamiseksi.
Commit	Tehtyjen muutosten tallentamista versionhallinnan versioarkistoon.
ERP	Enterprise Resource Planning, toiminnanohjausjärjestelmä.
InnoDB	InnoDB on suomalaisen Innobasen kehittämä tallennusmoottori MySQL:n.
JSON	JavaScript Object Notation, tekstimuotoinen JavaScriptin käyttämä tiedonsiirtomuoto
MVC	Model – View – Control, arkkitehtuurimalli
MySQL	MySQL on avoimen lähdekoodin tietokantahallintajärjestelmä
PHP	PHP: Hypertext Preprocessor, ohjelmointikieli erityisesti dynaamisten web-sivujen toteuttamiseen
Repository	Versionhallintajärjestelmän versioarkisto ja muutoshistoria.
RUP	Rational Unified Process (myös Unified Process), ohjelmistokehityksen prosessimalli
SQL	Structured Query Language, kieli jolla tehdään kyselyitä tietokantaan.
SVN	Subversion, versionhallintajärjestelmä
UML	Unified Modeling Language

Sisältö

1	Johdanto	6
2	Asiakkaan toiminnan kuvaus	7
3	Toiminnanohjausjärjestelmä	8
4	Systeemyömallit	9
4.1	Vesiputousmalli	9
4.2	RUP	11
4.2.1	Työn kulku RUP:n mukaan	12
4.3	Ketterät menetelmät	17
5	Suunnittelumenetelmät	19
5.1	UML	20
5.2	Tietokannan suunnittelumenetelmät	21
6	Käytetyt tekniikat	24
6.1	PHP	25
6.2	CodeIgniter	25
6.3	JavaScript ja AJAX	25
6.3.1	jqGrid	26
6.3.2	FullCalendar	26
6.4	Relaatiotietokanta ja SQL	27
6.5	Etäkokousjärjestelmät ja Adobe Connect	28
6.6	Versionhallinta	29
6.6.1	Subversion ja keskitetty versionhallinta	29
6.6.2	Mercurial ja hajautettu versionhallinta	29
7	Projektin organisaatio ja vaiheet	30
8	Järjestelmän esittely	33
8.1	Asiakkaiden hallinta	35
8.2	Yritysten hallinta	36
8.3	Kiinteistöjen hallinta	37
8.4	Tilausten hallinta	38
8.5	Töiden hallinta	38
8.6	Raporttien hallinta ja laskujen hallinta	38
8.7	Varastojen hallinta	39
8.8	Ylläpito	39
9	Tehtyjä valintoja ja johtopäätöksiä	40
10	Yhteenveto	41

1 Johdanto

Tämän opinnäytetyön viitekehyksenä on yrityksen toiminnan kokonaisvaltainen kehittäminen. Tähän liittyy yrityksen tietojärjestelmän innovatiivinen kehittäminen yrityksen toimintaa tukemaan.

Asiakkaalla ei tällä hetkellä ole käytössään koko yrityksen toiminnan kattavaa toiminnanohjausjärjestelmää. Laskutukseen on käytössä ulkoinen järjestelmä, ja nyt toteutettava järjestelmä tulee käyttämään tuota laskutusjärjestelmää ja kommunikoimaan sen kanssa rajapinnan välityksellä. Töiden ohjaus tehdään käsin joko kynällä ja paperilla, Excel-taulukoilla tai Google Calendaria käyttämällä. Asentajat kirjoittavat mittaustulokset ja muut tiedot ylös käsin. Tämä aiheuttaa huomattavasti ylimääräistä työtä.

Tämän työn tarkoituksena on Innotek Oy:n toiminnanohjausjärjestelmän töiden hallinnan määrittely, suunnittelu ja toteutus. Töiden hallinta on osa yrityksen tilaus-toimitusketjua. Ketju koostuu useista toisiinsa liittyvistä yksittäisistä toiminnoista lähtien tavaroiden tilaamisesta yrityksen varastoon edeten tavaran toimittamiseen asiakkaalle ja päättyen asiakkaan laskuttamiseen. Tämä on liiketoiminnan ydinprosessi, jossa tarkoituksena on tarjota sellainen tuotepalvelukombinaatio, jota asiakkaat haluavat ja josta he ovat valmiita maksamaan. (Sakki 2001)

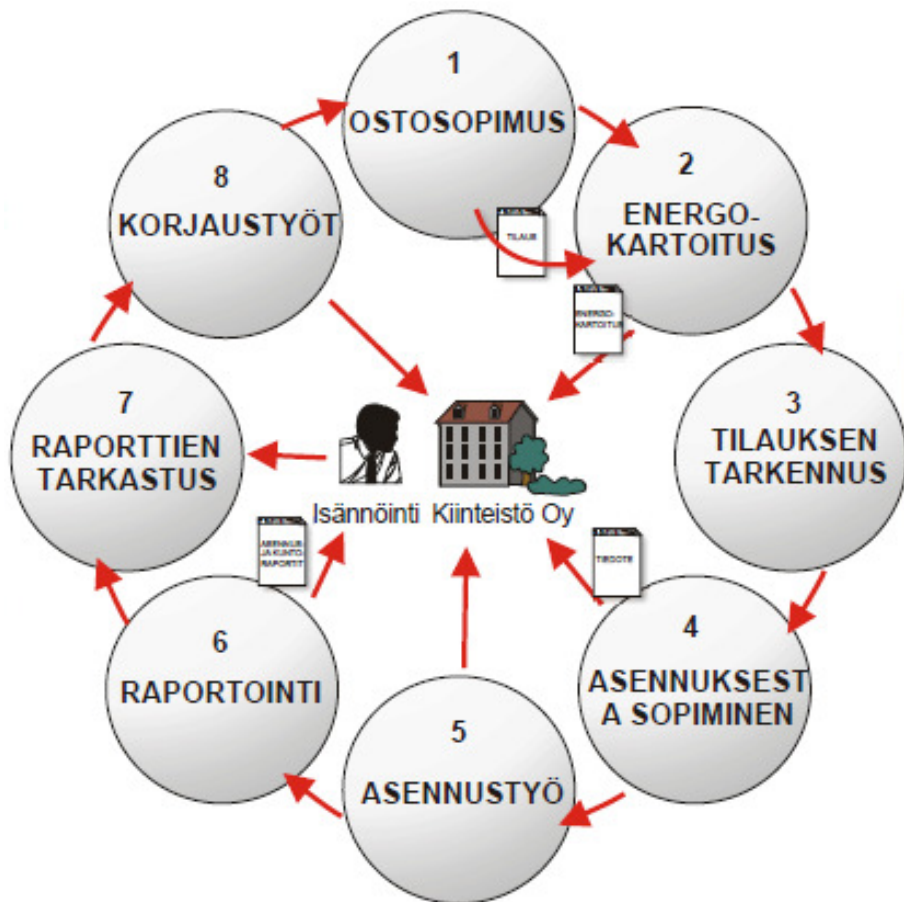
Järjestelmän tämän version kehitys on alkanut keväällä 2010 Saimaan ammattikorkeakoulun opiskelijaprojektina, ja tämän työn alkaessa se oli edelleen kesken.

Toteutettavan järjestelmän etuna on asentajien töiden lähes reaaliaikainen seuranta ja aikataulutus, mikä nopeuttaa tilausten toimitusta ja etenemistä laskutukseen. Lisäksi tämä vähentää asentajien matkustelua eri paikkakunnilla sijaitsevien asiakkaiden välillä, helpottaa varastotilanteen seurantaa ja nopeuttaa reklamaatioiden käsittelyä.

2 Asiakkaan toiminnan kuvaus

Innotek on Lieksassa sijaitseva LVI-alan pienyritys, jolla on noin 20 työntekijää ja noin 100 asiakasta. Yrityksellä on lisäksi toimipisteitä Kempeleellä ja Helsingissä sekä varastoja Lieksassa, Keravalla ja Kempeleessä (Rissanen 2010, 10).

Yrityksen Energo-ohjelman avulla asiakas voi saavuttaa jopa 35% säästön kokonaisvedenkulutuksessa ja 15% säästön energiankulutuksessa. Säästöohjelma toimii pääpiirteittäin seuraavasti: ensin asiakas tilaa Energo-kartoituksen kohdekiinteistöön, jonka jälkeen yrityksen työntekijät menevät kiinteistöön ja mittaavat noin viidestä huoneistosta suihkujen, hanojen, wc-pyttyjen yms. vedenkulutusta ja vedenpainetta. (Kuva 2.1)



Kuva 2.1 Energo-ohjelman eteneminen (Innotek Oy, 2000)

Mittaustulosten perusteella laaditaan kulutusanalyysi, johon sisältyvät suositukset korvattavista kalusteista, arvio tällä saavutettavasta säästöstä sekä asennustarjous. Tämän jälkeen asiakas voi halutessaan tilata asennukset näille kalusteille. Yrityksellä on valikoima erilaisista vettä säästävistä Energo-tuotteista. (Rissanen 2010)

Tämän lisäksi asentajat voivat kohteessa ollessaan havaita vikoja, joista laaditaan vikaraportti, joka saattaa sisältää myös kuvia tai videoita vioista. Raportoiduista vioista voidaan laatia työnanto korjausta varten.

3 Toiminnanohjausjärjestelmä

Toiminnanohjausjärjestelmät ovat modulaarisia liiketoimintaa ohjaavia järjestelmiä, joihin voi sisältyä alijärjestelmät esimerkiksi palkanlaskentaa, tilausten hallintaa, kirjanpitoa jne. varten. Modulaarisuuden ansiosta järjestelmän osia voidaan ottaa käyttöön tai poistaa käytöstä hallitusti, ja koko järjestelmä on laajennettavissa tarpeen mukaan. Järjestelmien hyötyinä pidetään liiketoiminnan tehostumista ja nopeuttamista rutiinitöiden karsimisen myötä sekä eri toimintojen reaaliaikaista seuranta, joka puolestaan nopeuttaa yrityksen toiminnan ohjaamista oikeaan suuntaan sekä helpottaa ennusteiden laskemista. (From, 2008; Tieke-kysely 2008)



Kuva 3.1 Kaavio toiminnanohjausjärjestelmästä ja sen yleisimmistä osista. (All About ERP and Business Softwares, 2011)

Kuvassa 3.1 on kuvattu toiminnanohjausjärjestelmien mahdollinen koostumus. Kuvassa näkyvät alijärjestelmät tilausketjun hallintaa, varastohallintaa, tuotannonhallintaa, kirjanpitoa, henkilöstöhallintaa, toimitusten hallintaa, tuotannon suunnittelua, liiketoimintatiedon hallintaa (business intelligence), teknistä suunnittelua sekä myyntiä varten. Käytössä olevat järjestelmät eivät välttämättä sisällä kaikkia näitä osia.

ERP-järjestelmät eivät ole vielä kovinkaan yleisiä. Tietoyhteiskunnan päättäjäindeksikyselyn mukaan noin puolella yrityksistä on käytössään jonkinlainen ERP-järjestelmä. Eniten niitä käytetään laskutuksessa, kirjanpidossa ja henkilöstöhallinnossa. Jakelussa, varastohallinnassa ja projektihallinnassa niiden käyttö on vähäisempää. (Tieke 2008)

Konsulttiyhtiö Gartnerin mukaan toiminnanohjausjärjestelmämarkkinoilla liikkuu vuosittain noin 12,25 miljardia euroa. Suuryritykset ovat tähän asti muodostaneet suurimman osan tästä markkinasta. PK-yritysten osuus on ollut noin yksi kolmasosa. Myös PK-yritysten toiminnanohjausjärjestelmien tuottajat ovat viime aikoihin asti olleet pieniä, erikoistuneita ja alueellisia yrityksiä. Perinteisesti suuryrityksiin keskittyneet Microsoft, Oracle ja SAP ovat huomanneet tämän markkinasegmentin ja suunnanneet sille tuotteita. (Weber 2007)

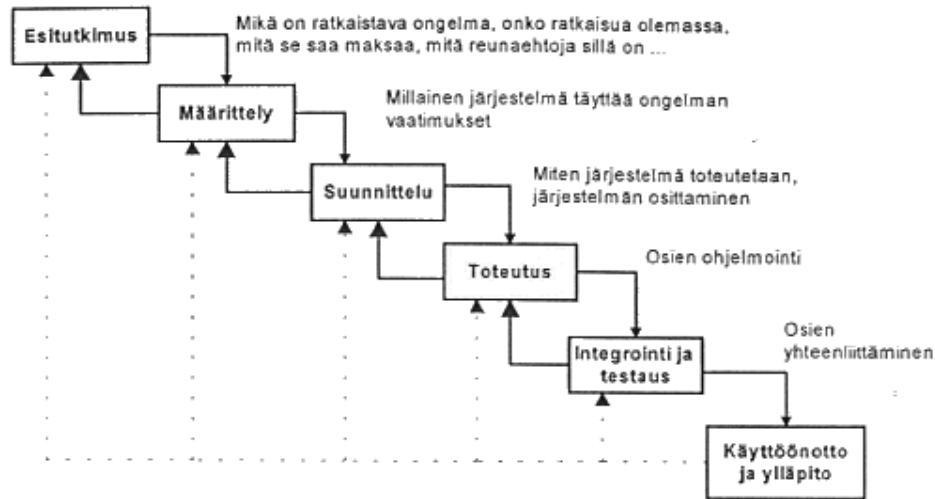
4 Systemityömallit

Systemityömallien tarkoituksena on toimia projektinhallinnan kehyksenä ja määrittellä prosessi, jonka mukaan projekti voi edetä. Tärkeitä haasteita ovat aikataulussa sekä budjetissa pysyminen, vaatimusten hallinta sekä laadunhallinta. Ohjelmistotuotteiden koon kasvaessa myös ongelmat ovat kasvaneet, ja erilaisia systemityömalleja on kehitetty vastaamaan näihin haasteisiin.

4.1 Vesiputousmalli

Vesiputousmalli on teoreettinen ohjelmiston elinkaarta kuvaava vaihejakomalli, joka ei koskaan toteudu sellaisenaan. Ohjelmiston elinkaarella tarkoitetaan aikaa, joka kuluu ohjelmiston kehittämisen alun ja ohjelmiston käytöstä poistamisen välissä. Vaiheita ovat tyypillisesti esitutkimus, määrittely, suunnittelu, toteu-

tus, integrointi ja testaus, sekä mahdollisesti käyttöönotto- ja ylläpitovaihe. (Haila & Märijärvi, 2004; Kainulainen 2008)



Kuva 4.1. Vesiputousmallin vaiheet. (Immonen 2001)

Vesiputousmallissa edetään peräkkäin vaiheesta toiseen, eli esimerkiksi ensin tehdään esitutkimus kokonaisuudessaan, jonka jälkeen siirrytään määrittelyyn ja niin edelleen. Jokainen vaihe tehdään kokonaan valmiiksi ja dokumentoidaan tarkasti ennen seuraavaan siirtymistä.

Vesiputousmallissa on useita ongelmia, minkä takia sitä ei juuri käytetä. Yksi ongelma on myöhäinen testaus, jonka takia ongelmat ilmenevät myöhään. Esimerkiksi suunnitteluvaiheen virheet tulevat esiin vasta testausvaiheessa.

Lisäksi vesiputousmallissa oletetaan, että vaatimukset pystytään määrittelemään kokonaisuudessaan heti projektin alussa, eivätkä ne tule muuttumaan myöhemmin. Usein asiakkaan vaatimukset selviävät vasta kun tuote on jo valmis.

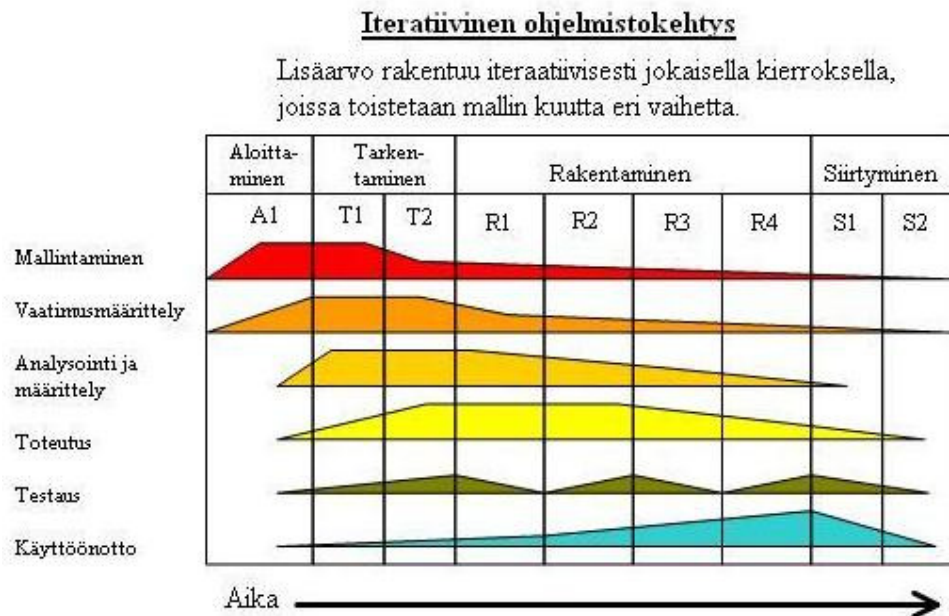
Vesiputousmallin etuna ovat selkeys ja yksinkertaisuus. Se on käyttökelpoinen lähinnä pienissä projekteissa, joissa määrittely on yksinkertainen ja yksiselitteinen ja kehittäjillä on jo alussa selkeä käsitys lopputuloksesta ja tavoitteista.

4.2 RUP

RUP (Rational Unified Process) on iteratiivisen ja inkrementaalisen ohjelmistokehityksen arkkitehtuurikeskeinen prosessikehys. (Eriksson & Penker, 2000) Ohjelmistokehityksen parhaat käytännöt on kirjattu ylös ja määritelty prosessiksi. (Kruchten, 1998) RUP on suunniteltu joustavaksi ja mukautettavaksi jokaisen projektin tarpeisiin, koska mikään prosessi ei sellaisenaan sovellu kaikille projekteille. Parhaat käytännöt RUP:n mukaan ovat

1. ohjelmiston kehittäminen iteratiivisesti
2. vaatimusten hallinta
3. komponenttipohjaisten arkkitehtuurien käyttäminen
4. ohjelmiston mallintaminen visuaalisesti
5. jatkuva laadunvalvonta
6. muutosten hallinta

RUP:n mukaan projekti jaetaan vaiheisiin, jotka ovat aloittaminen, tarkentaminen, rakentaminen ja siirtyminen. Jokaiseen vaiheeseen kuuluu tyypillisesti yhdestä kolmeen iteraatiota riippuen projektin koosta. (Kruchten 1998)



Kuva 4.2. RUP:n vaiheet ja eri työkulkujen suhteellinen painotus eri vaiheissa (Kruchten 1998, 45)

Kuvasta 4.2 nähdään, millä tavalla projektin osa-alueet painottuvat projektin vaiheissa. Kuvassa jokainen pystypalkki on yksi iteraatio. Alussa liiketoiminnan mallinnus ja vaatimusmäärittely ovat pääosassa, ja toteutus, testaus ja käyttöönotto jäävät ensimmäisessä iteraatiossa vähemmälle tai jopa kokonaan pois. Aloittamisvaiheen tärkeimmät tavoitteet ovat projektin rajaaminen, tärkeimpien käyttötapausten löytäminen, projektin aikataulutusta ja budjetointi sekä tärkeimpien riskien arviointi.

Tarkentamisvaiheen tavoitteita ovat järjestelmän arkkitehtuurin määrittely, näkemys lopullisesta tuotteesta sekä varmistaa että nämä ovat yhteensopivia. Lisäksi suunnitellaan toteutusvaihe.

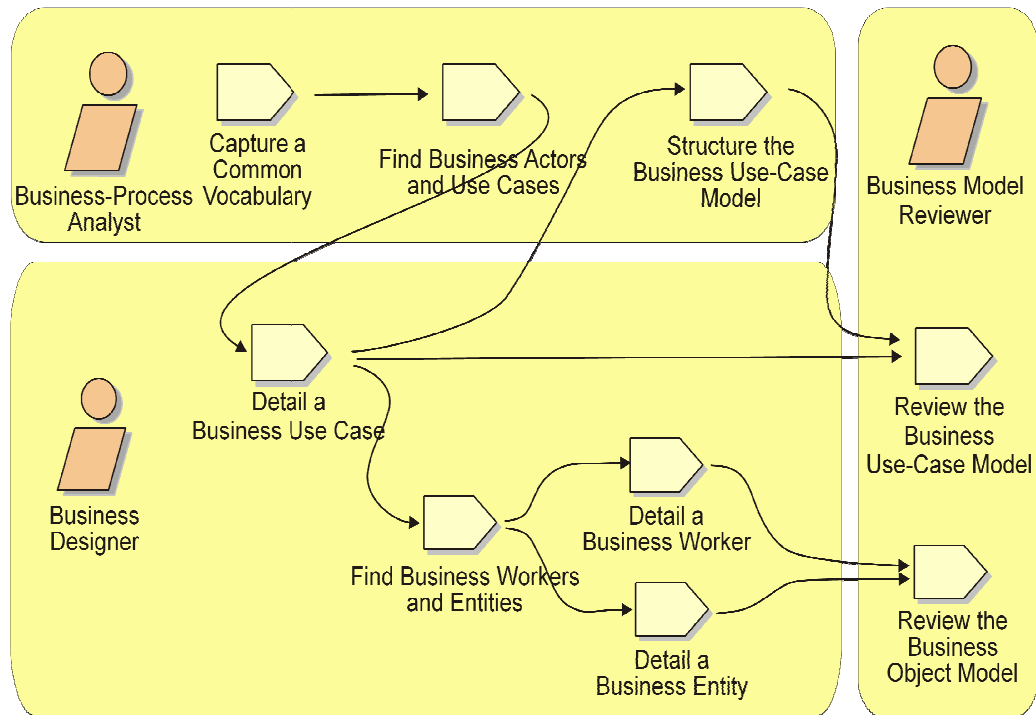
Toteutusvaiheessa pyritään toteuttamaan hyvälaatuisia testiversioita tuotteesta mahdollisimman nopeasti.

Siirtymisvaiheen alussa järjestelmä otetaan alustavasti tuotantokäyttöön. Tässä vaiheessa korjataan käyttäjien havaitsemat virheet sekä toteutetaan viimeiset vaatimukset. Tämän vaiheen jälkeen tuote on julkaisukunnossa. (Kruchten, 1998)

4.2.1 Työn kulku RUP:n mukaan

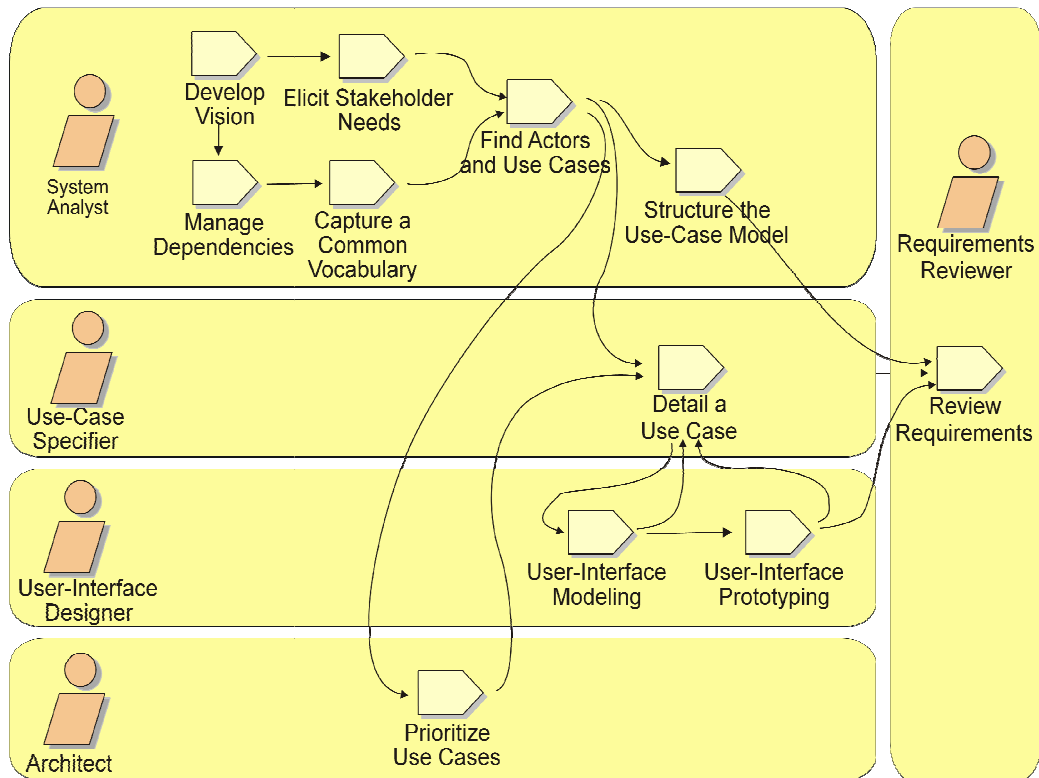
RUP:ssa työnkulku on jaettu osiin. Ensimmäinen osio on projektinhallinta, ja tähän tarkoitukseen RUP pyrkii tarjoamaan käytännöllisen kehyksen riskienhallintaan sekä projektin suunnitteluun, tarkkailuun ja läpivientiin. Se ei kata esimerkiksi budjetointia tai työvoiman palkkausta ja koulutusta. RUP:in pääpaino on riskienhallinnassa sekä iteratiivisen projektin suunnittelussa: iteraatioiden sisällöt, lukumäärä, niiden pituus sekä kehityksen tarkkailu. (Kruchten 1998)

Liiketoiminnan mallinnuksen tavoitteena on ymmärtää organisaation rakenne ja toiminta sekä selvittää, mikä tuotettavan ohjelmiston liiketoiminnallinen tavoite on. Tätä varten tuotetaan korkean tason vaatimusmäärittely ja tarvekartoitus. Tämän lisäksi tähän prosessiin kuuluu yhteisen sanaston luominen eli moniselitteisten termien täsmentäminen. (Kruchten 1998)



Kuva 4.3. Liiketoiminnan mallinnuksen työnkulku (Kruchten 1998, 134)

Vaatimusmäärittelyn työnkulun tavoitteisiin kuuluu asiakkaan ja tuottajan välinen yhteisymmärrys tuotettavan ohjelmiston toiminnallisuudesta eli toiminnallinen määrittely sekä käyttöliittymän määrittely. Lisäksi saadaan lähtökohdat projektin ajankäytön ja kustannusten arviointiin. Myös iteraatioiden sisällön arviointiin tarvitaan tuotettavia vaatimusmäärittelyjä. Vaatimusmäärittelyssä tarkennetaan aiemmin tuotettuja määrittelyjä niin tarkoiksi, että määrittelyjä voi käyttää ohjelmiston suunnittelun ja toteutuksen pohjana. (Kruchten 1998)



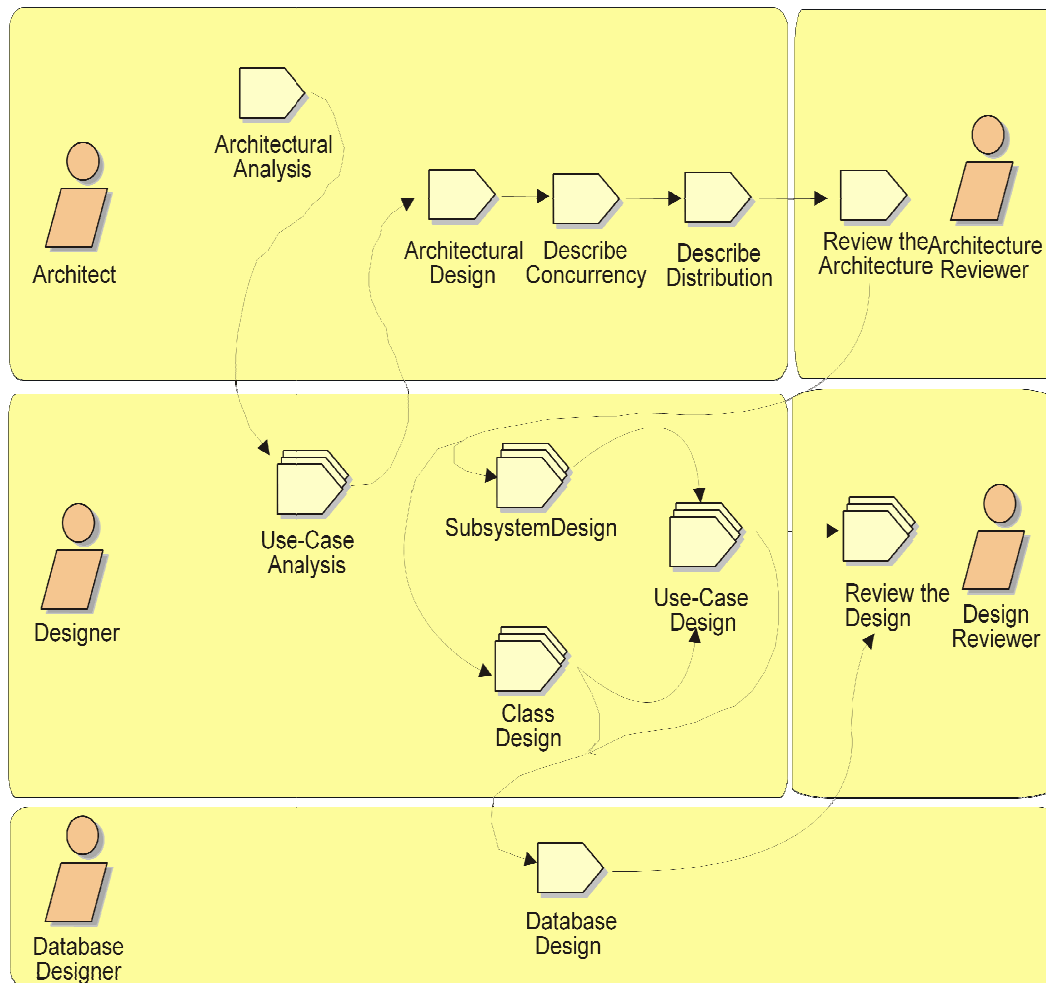
Kuva 4.4 Vaatimusmäärittelyn työnkulku (Kruchten 1998, 146)

Vaatimukset jaetaan toiminnallisiin ja ei-toiminnallisiin. Toiminnallisilla vaatimuksilla tarkoitetaan, mitä järjestelmän tulee tehdä. Ei-toiminnallisia vaatimuksia ovat mm. käytettävyyteen, luotettavuuteen, suorituskykyyn ja ylläpidettävyyteen liittyvät vaatimukset. Nämä ovat yhtä tärkeitä kuin toiminnalliset vaatimukset. (Kruchten 1998)

Vaatimusmäärittelyosioon kuuluu myös käytötapausten etsiminen, määrittely sekä kuvaaminen. Käyttötapauskuvaukset ovat sanallisia kuvauksia, joissa käydään läpi vaihe vaiheelta, miten käytötapaus etenee. Kuvauksen tulee olla riittävän tarkka jatkoa varten, mutta sen tulee myös olla asiakkaan ymmärrettävissä, jotta voidaan olla varmoja siitä, että se vastaa asiakkaan vaatimuksia. Näitä käytetään myös loppukäyttäjille tarkoitetun dokumentaation laatimiseen. (Kruchten 1998)

Analyysin ja suunnittelun tarkoituksena on muuntaa vaatimukset riittävän tarkaksi määrittelyksi että ohjelman toteuttaminen on mahdollista. Suunnittelun tarkkuus riippuu toteuttajan osaamistasosta ja suunniteltavan järjestelmän mo-

nimutkaisuudesta. Jos suunnittelu on riittävän tarkkaa, toteutus voidaan sitoa luonnokseen ja toteutuksen edetessä nämä voidaan synkronoida, jolloin välte-
tään yksi muunnosvaihe ja sitä kautta yksi virheen aiheuttaja. (Kruchten 1998)

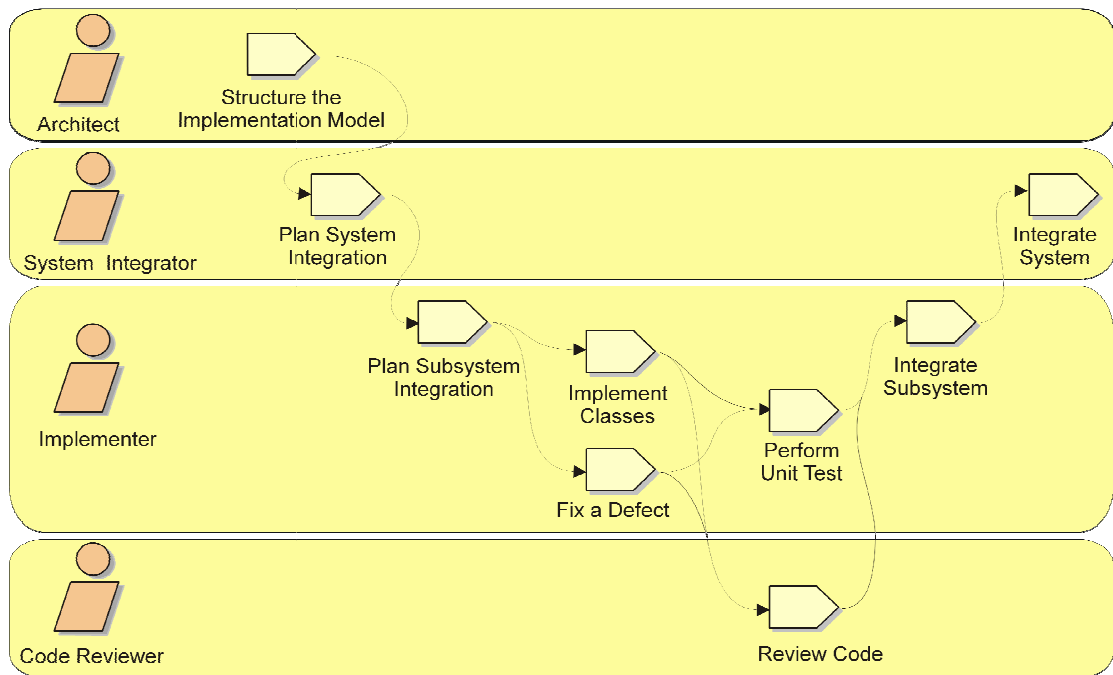


Kuva 4.5. Suunnitteluvaiheen työnkulku (Kruchten 1998, 154)

Suunnitteluvaiheessa määritellään arkkitehtuuri, kuhunkin käyttötapaukseen osallistuvat luokat ja niiden operaatiot, sekä luokkien väliset suhteet, alijärjestelmät, ja niiden rajapinnat. Suunnitteluvaihetta joudutaan toistamaan jatkossa, ja tämän vaiheen tuotoksia joudutaan muokkaamaan ja tarkentamaan projektin edetessä. Ensimmäisissä iteraatioissa ei kannata suunnitella järjestelmää liian yksityiskohtaisesti. (Kruchten 1998)

Toteutusvaiheen tavoitteena on toteuttaa luokat komponentteina sekä yksikkötestata nämä ja integroida ne suoritettavaksi järjestelmäksi. Integraatiotestaus

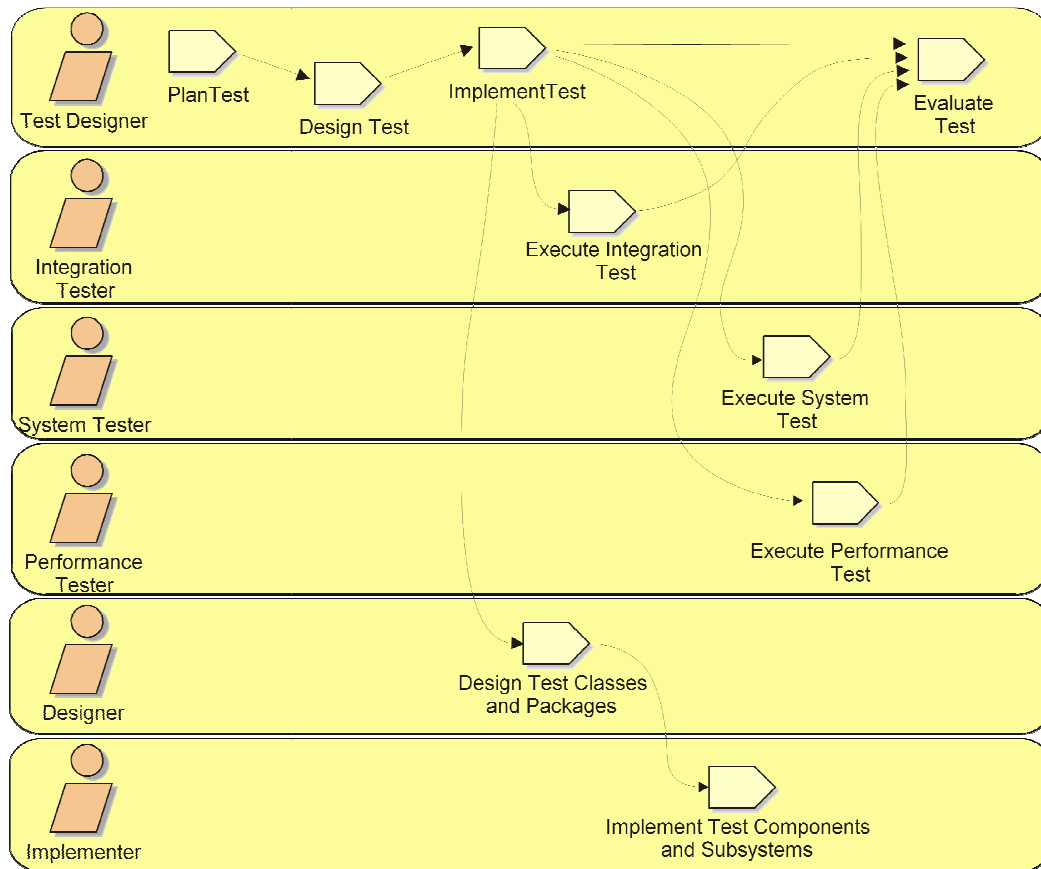
ja järjestelmän kokonaistestaus kuuluvat testausvaiheeseen. Yksikkötestaus kuuluu toteutusvaiheeseen. (Kruchten 1998)



Kuva 4.6 Toteutusvaiheen työkulku (Kruchten 1998, 163)

Arkkitehti määrittelee alijärjestelmän organisoinnin. Systemin integroija suunnittelee alijärjestelmien yhdistämisen järjestelmään. Toteuttaja suunnittelee luokkien yhdistämisen alijärjestelmäksi, toteuttaa luokat, yksikkötestaa ne, korjaa löytyneet virheet, ja lopulta yhdistää ne alijärjestelmäksi, joka palautetaan systeemin integroijalle, joka integroi ne osaksi järjestelmää. Luokkien toteutuksen ja alijärjestelmäintegroinnin välissä koodi katselmoidaan. Toteutus liittyy läheisesti suunnitteluvaiheeseen, ja tässä vaiheessa on myös mahdollista palata suunnitteluvaiheeseen ja esimerkiksi muuttaa luokkakaavioita vastaamaan toteutusta. (Kruchten 1998)

Testausvaiheessa tarkistetaan, että kaikki komponenttien ja olioiden väliset vuorovaikutukset toimivat niin kuin on suunniteltu, että kaikki komponentit integroituvat oikein ja että kaikki vaatimukset on toteutettu. Lisäksi varmistetaan, että kaikki havaitut virheet korjataan ennen käyttöönottoa. Kyse on ohjelmistotuotteen laadunvalvonnasta. (Kruchten 1998)



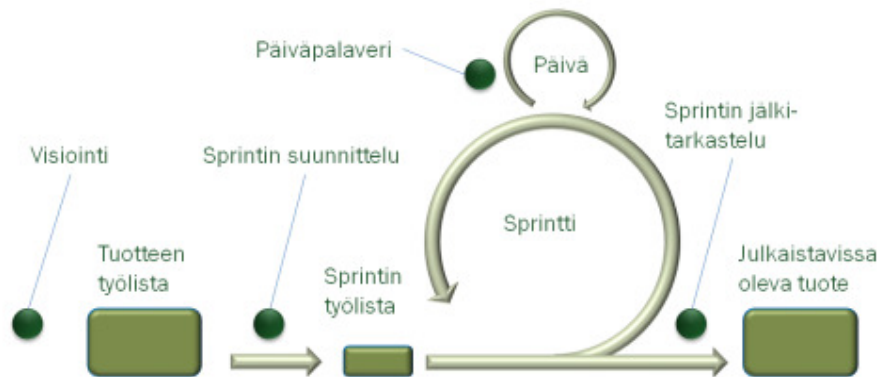
Kuva 4.7. Testausvaiheen työnkulku (Kruchten 1998, 176)

4.3 Ketterät menetelmät

Ketterät menetelmät ovat iteratiivisia kehitysmenetelmiä, joissa pyritään valmistautumaan muuttuviin vaatimuksiin ja minimoimaan riskit prosessin aikana. Iteraatiot ovat tyypillisesti hyvin lyhyitä, viikosta kuukauteen, ja yleensä tavoitteena on, että joka iteraation lopussa on toimiva ohjelma, johon on saatu lisättyä uusia ominaisuuksia, ja josta asiakas voi antaa palautetta. Asiakkaan palautteen avulla vaatimuksia voidaan tarkentaa ja muuttaa. Tyypillisesti ominaisuuksia valitaan toteutettavaksi tärkeysjärjestyksessä, asiakkaalle eniten lisäarvoa tuottavat ominaisuudet ensin. Tällä maksimoidaan asiakkaan projektista saama hyöty myös siinä tapauksessa, että projekti epäonnistuisi. (Kainulainen 2008; Haikala & Märijärvi 2004)

Yksi tunnetuimmista ketteristä menetelmistä on Scrum. Scrum pohjautuu pieniin (5-9 hengen) itseorganisoituviin tiimeihin sekä noin kuukauden mittaisiin sprint-

teihin. Sprintti tarkoittaa yhtä iteraatiota. Jokaisen sprintin alussa on suunnittelupalaveri, jonka alussa sovitaan, mitä ominaisuuksia tämän sprintin aikana pyritään toteuttamaan. Jokaisen sprintin tavoitteena on julkaisukelpoinen tuote. (Kainulainen 2008; Ketterät Käytännöt.fi 2011). Kuvassa 4.8 on kuvattu Scrumin prosessin kulku.



Kuva 4.8 Scrumin prosessi (Ketterät Käytännöt.fi, 2011)

Alussa tehdään työlista lopulliseen tuotteeseen vaadituista ominaisuuksista. Ominaisuuksia lähdetään toteuttamaan kiireellisyysjärjestyksessä. Etenemistä seurataan päivittäisissä lyhyissä, noin 15 minuutin pituisissa palavereissa, sekä joka sprintin jälkeisissä asiakkaan kanssa käytävissä katselmoineissa.

XP eli Extreme Programming on jatkuvaa testausta, jatkuvaa refaktorointia, lyhyitä iteraatioita ja pariohjelmointia painottava menetelmä. Yksikkötestit toteutetaan aina ennen varsinaista toteutusta, ja toteutusta kehitetään kunnes se läpäisee testit. Ohjelmaa kehitetään ominaisuus kerrallaan ja arkkitehtuuri pidetään niin yksinkertaisena kuin mahdollista. Tämä johtaa toistuvaan refaktorointiin (Kainulainen 2008).

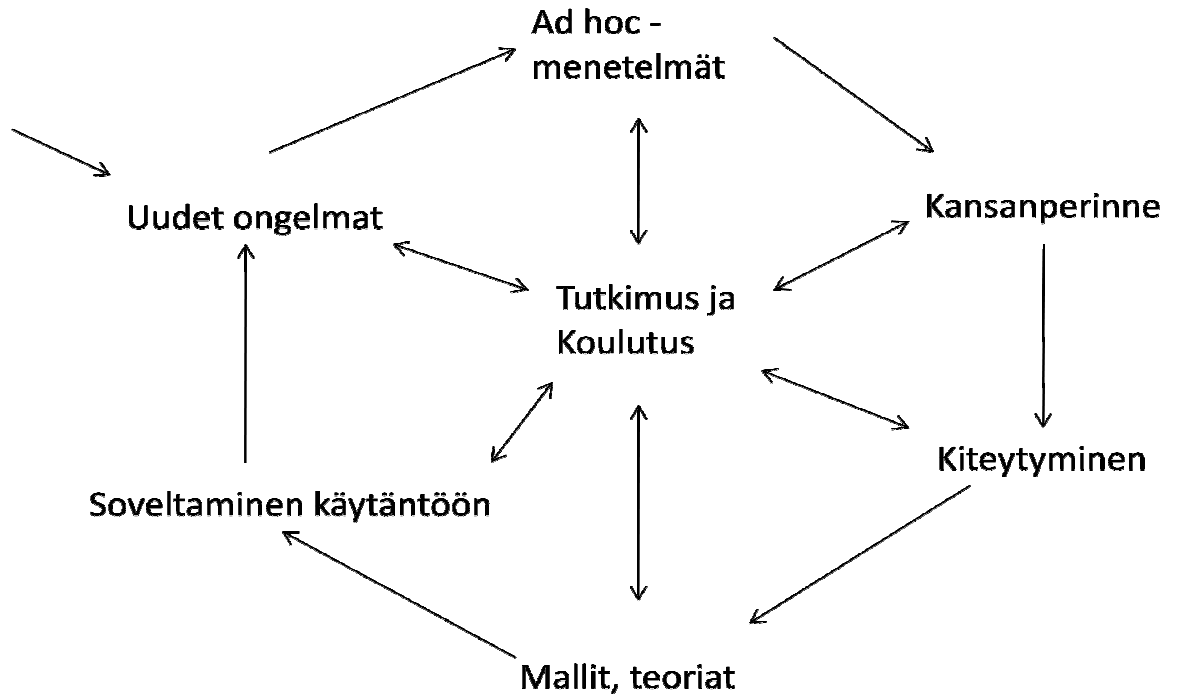
FDD, eli Feature-Driven Development tarkoittaa ominaisuuslähtöistä kehitystä. Ominaisuus tarkoittaa tässä sellaista kokonaisuutta, josta on asiakkaalle selvää liiketoiminnallista hyötyä ja joka voidaan toteuttaa itsenäisesti. FDD:lle tyypillistä on kahden viikon iteraatiot, joissa jokaisessa on uusia asiakkaalle lisäarvoa tuottavia ominaisuuksia. Ominaisuudet myös ohjaavat prosessin etenemisen seuranta: kun esimerkiksi 50 % kaikista ominaisuuksista on toteutettu, koko projekti on 50 %:sti valmis. (Kainulainen 2008) Perinteisillä menetelmillä projek-

tin valmiutta voidaan vain arvioida, ja yleensä arviot ovat liian optimistisia (Brooks 1995; Haikala & Märijärvi 2004, 29).

5 Suunnittelumenetelmät

Ohjelmistojen suunnittelumenetelmät ovat syntyneet ohjelmistoprojektien koon kasvaessa. Kun ratkottavat ongelmat kasvavat, tarvitaan välineitä ongelman jäsentämiseen, ymmärtämiseen ja kommunikointiin. Ymmärtämisen ja kommunikoinnin täytyy olla täysin yksiselitteistä väärinymmärrysten välttämiseksi. Jos ratkaistavaa ongelmaa ei ymmärretä oikein, voi sen ratkaiseminen onnistua vain hyvällä onnella. (Haikala & Märijärvi 2004, 31)

Menetelmien kehitys lähtee siitä, että vastaan tulee uudenlainen ongelma, jota ei ennen ole ratkaistu. Tällöin joudutaan keksimään ad hoc -menetelmä, eli keksimään mikä tahansa keino joka toimii. Kun vastaavia ongelmia ratkotaan enemmän, löydetään vähitellen yleisiä ratkaisuja, joista tulee eräänlaista kansanperinnettä. Tämä kansanperinne kiteytyy edelleen erilaisiksi malleiksi ja menetelmiksi, joille saadaan teoreettinen viitekehys. Näitä voidaan soveltaa käytäntöön systemaattisesti, kunnes taas törmätään ennennäkemättömään ongelmaan, ja prosessi alkaa jälleen alusta. (Haikala & Märijärvi 2004, 27)



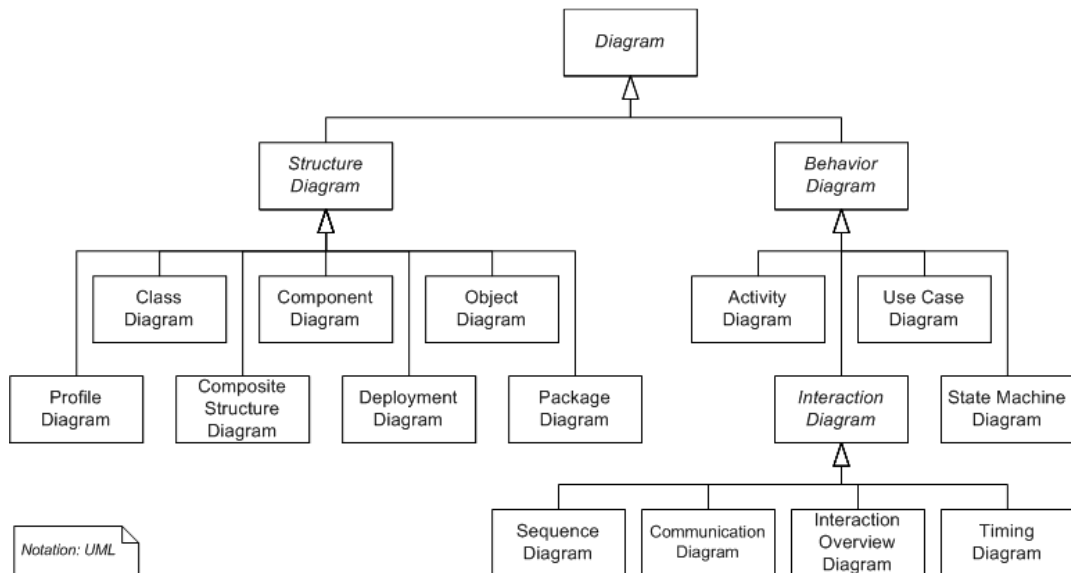
Kuva 5.1 Menetelmien kehitys (Haikala & Märijärvi 2004, 27.)

Ohjelmistotuotannon ongelma on toisaalta se, että ala on kohtuullisen nuori verrattuna vaikkapa siltojen suunnitteluun. Siltojen suunnittelu lähti siitä, kun ihminen ensimmäisen kerran kaatoi puunrungon puron ylle. Ohjelmistotuotannon historia taas ulottuu noin 1940-luvulle. Toinen ongelma on, että ratkaistavat ongelmat, eli tuotettavien ohjelmistojen koko, ovat koko tämän ajan kasvaneet eksponentiaalisesti. Tämän vuoksi ohjelmistotuotannossa ollaan jatkuvasti "ad hoc"-vaiheessa.

5.1 UML

UML (Unified Modeling Language) on oliosuuntautunut (object-oriented) mallinnusmenetelmä, joka syntyi 1990-luvun loppupuolella usean eri mallinnuskielten kehittäjien yhteistyön tuloksena. Tavoitteena oli tehdä yksi mallinnuskieli, joka olisi toisaalta riittävän laaja vastaamaan mahdollisimman monien kehittäjien tarpeisiin, mutta riittävän yksinkertainen ollakseen ymmärrettävä (Booch, Jacobson & Rumbaugh 1999).

UML pohjautuu vahvasti kaavioihin. Kaavioita on UML:n versiossa 2.2 neljätoista eri tyyppiä, joilla kaikilla on oma tarkoituksensa.



Kuva 5.2. Luokkakaavio UML-kaavioista (Rissanen 2010)

Kaaviotyyppiä ovat rakennekaaviot, käyttäytymiskaaviot ja vuorovaikutuskaaviot. Rakennekaavioihin kuuluvat luokkakaaviot, komponenttikaaviot, oliokaaviot, koostekaaviot, pakkauskaaviot ja sijoittelukaaviot. Vuorovaikutuskaavioita ovat ajoituskaaviot, kokoavat vuorovaikutuskaaviot, kommunikointikaaviot sekä sekvenssikaaviot. Käyttäytymiskaavioita ovat aktiviteettikaaviot, käyttötapauskaaviot ja tilakaaviot. (Fowler & Scott 2002; Erikson & Penker 2000)

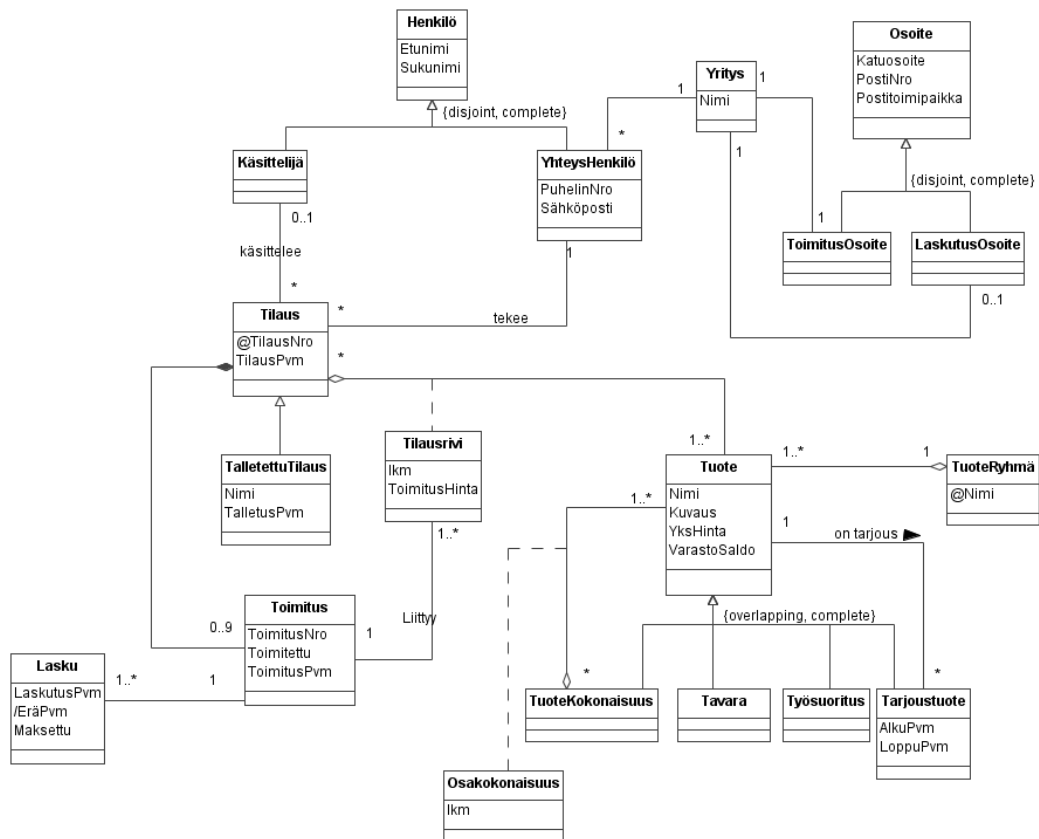
UML on alun perin luotu ohjelmistokehityksen tarpeisiin, mutta sitä on sittemmin sovellettu myös prosessitekniikkaan, työvirtojen kuvauksiin ja liiketoiminnan mallinnukseen.

5.2 Tietokannan suunnittelumenetelmät

Tietokannan suunnittelun tavoitteena on tietokanta, jossa on kaikki vaadittava tieto. Tiedon pitää olla jaettuna taulurakenteisiin siten, että kukin taulu kuvaa vain yhtä asiaa. Missään taulussa ei saa olla tietoa, joka ei liity suoraan kyseiseen käsitteeseen. Mikään tieto ei myöskään saa olla kannassa useaan kertaan tai useassa paikassa. (Hernandez 2000)

Hyvin suunniteltu tietokanta on laajennettavissa hallitusti tulevaisuuden tarpeita varten tai uusien vaatimusten ilmaantuessa. Tarpeelliset tiedot ovat helposti haettavissa ja muokattavissa. Sovellusten kehittäminen tietokantaa käyttäen tulisi olla mahdollisimman vaivatonta, eli ohjelmoitaessa ei tulisi joutua kiertämään tietokannan rakenteen aiheuttamia ongelmia. (Hernandez 2000)

Tietokannan suunnittelun ensimmäinen vaihe on tietosisällön kartoitus. Tarkoitus on kerätä tieto kaikesta tiedosta, jota tietokannassa pitää säilyttää. Tällöin selvitetään ja analysoidaan yrityksen liiketoiminnalliset tarpeet nyt ja jatkossa. (Hernandez 2000)



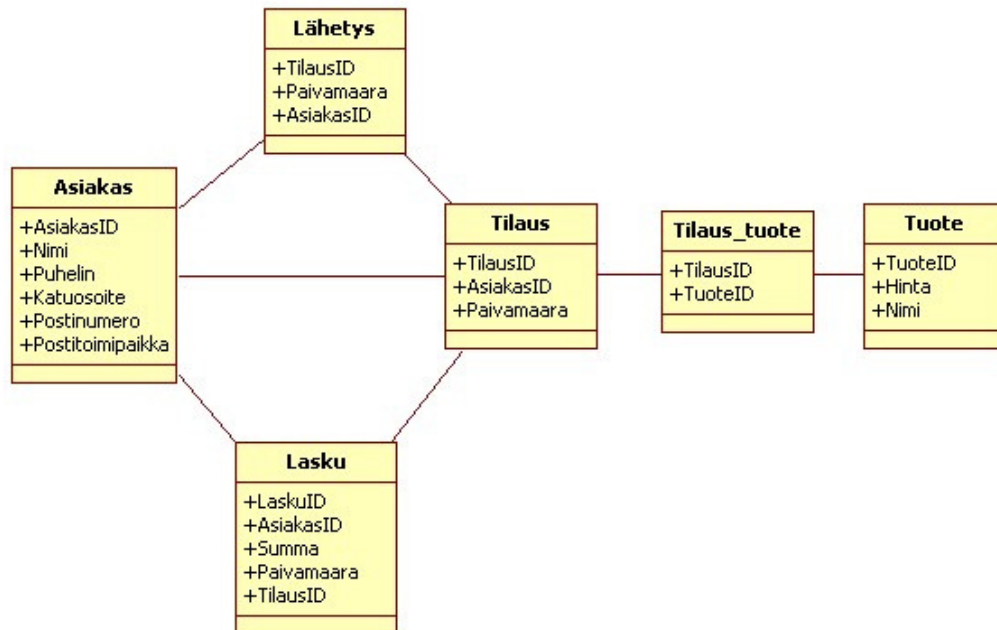
Kuva 5.3 Esimerkki käsitekaaviosta (Tampereen teknillinen yliopisto)

Tästä voidaan luoda käsitekaavio, jossa erotellaan käsitteet (esim. asiakas, tuote), näiden ominaisuudet (esim. nimi), ja käsitteiden väliset suhteet (esim. asiakas tilaa tuotteita). Käsitteistä tehdään tauluja, joihin tulee näiden ominaisuudet

sarakkeiksi. Suhteet muutetaan yhteyksiksi, jotka toteutetaan viiteavaimilla tai välitauluilla. (Hernandez 2000)

Tietokannan normalisointi on prosessi, jolla varmistetaan tietokannan eheys ja estetään tiedon toisto. Yleensä tietokanta saatetaan ensin ensimmäiseen, sitten toiseen, ja lopulta kolmanteen normaalimuotoon. Tämän jälkeen voi olla tarpeellista denormalisoida tietokantaa, koska kolmas normaalimuoto aiheuttaa monesti tarpeettoman monimutkaisen tietokantarakenteen, joka saattaa johtaa suorituskykyongelmiin tai hankaloittaa sovelluksen kehitystä.

Ensimmäinen normaalimuoto vaatii, että kussakin sarakkeessa on korkeintaan yksi arvo ja jokaisella rivillä on uniikki perusavain. Toinen normaalimuoto vaatii, että taulun sarakkeet eivät ole riippuvaisia mistään muusta kuin perusavaimesta.



Kuva 5.4 Käsitekaavio tilauksista

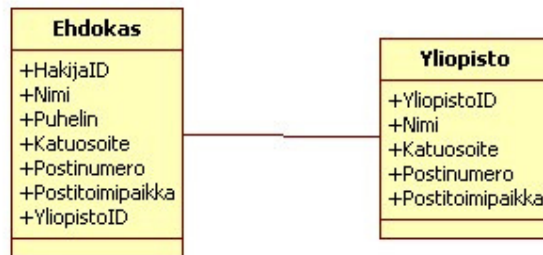
Microsoft antaa seuraavan esimerkin ensimmäisestä ja toisesta normaalimuodosta:

*Ajattele esimerkiksi asiakkaan osoitetta kirjanpitojärjestelmässä. Asiakkaat-
taulukko tarvitsee osoitetta, mutta niin tarvitsevat Tilaukset-, Lähetykset-, Laskut-,*

Myyntireskontra- ja Ryhmät-taulukot. Sen sijaan, että tallentaisit asiakkaan osoitteen erillisenä merkintänä jokaiseen näistä taulukoista, tallenna se yhteen sijaintiin joko Asiakkaat-taulukossa tai erillisessä Osoitteet-taulukossa.
(Microsoft 2008)

Tällöin joudutaan viittaamaan Asiakkaat-tauluun aina kun tarvitaan asiakkaan osoitetta, mutta osoite on vain yhdessä paikassa (katso kuva 5.4). Jos asiakkaan osoite muuttuu, muutos tarvitsee tehdä vain kerran eikä synny riskiä, että joku asiakkaan osoitteen kopiaista olisi jäänyt muuttamatta. Tässä tilanteessa tilaukset ja laskut samalle asiakkaalle voisivat mennä eri osoitteisiin.

Kolmas normaalimuoto tarkoittaa, että jokaisella rivillä on vain siitä avaimesta riippuvaisia tietoja.



Kuva 5.5 Ehdokkaan yliopiston tiedot eri taulukossa

Microsoftin esimerkki kolmannesta normaalimuodosta:

Esimerkiksi hakijan yliopiston nimi ja osoite saattaa sisältyä Työntekijöiden työhönotto -taulukkoon. Tarvitset kuitenkin täydellisen luettelon yliopistoista ryhmäpostituksia varten. Jos yliopistojen tiedot on tallennettu Ehdokas-taulukkoon, ei ole mitään tapaa näyttää luettelo yliopistoista, joista ei ole nykyisiä hakijoita. Luo erillinen Yliopisto-taulukko ja linkitä se Ehdokas-taulukkoon käyttäen yliopistokoodiavainta.
(Microsoft 2008)

6 Käytetyt tekniikat

Melkein kaikkia tässä projektissa käytettyjä tekniikoita on jo käsitelty laajasti aiemmissa tähän hankkeeseen liittyvissä opinnäytetöissä, esimerkiksi Paajanen 2011 ja Rissanen 2010, joten tässä niitä ei käsitellä kovin syvällisesti. JavaSc-

ript-pohjaisia komponentteja eli jqGridiä ja FullCalendaria ei ole käsitelty aiemmissa töissä, koska ne otettiin käyttöön myöhemmässä vaiheessa projektia.

6.1 PHP

PHP on avoimen lähdekoodin palvelimella suoritettava ohjelmointikieli, joka soveltuu erityisesti web-sivujen luomiseen. Käytetyssä http-palvelimessa tulee olla PHP-tulkki, joka tulkitsee sivun html-koodin sisään upotetun PHP-koodin ja lähettää näin luodun sivun asiakkaalle. (Paajanen 2011, Rissanen 2010)

6.2 CodeIgniter

CodeIgniter on MVC-mallia käyttävä PHP-sovelluskehys. Se on suhteellisen kevyt ja sen dokumentaatio on kattava ja helppolukuinen (Paajanen 2011). Sovelluskehysten käyttö on perusteltua, sillä se sisältää yleisimmät tarvittavat toiminnot, jotka tulisi muuten toteuttaa itse. Kehyksen avulla voidaan uudelleenkäyttää valmista koodia. Sovelluskehystä käytettäessä 60-80 % sovelluksen koodista voi olla saatu kehyksestä (Koskimies 2000, 262).

6.3 JavaScript ja AJAX

JavaScript on Netscape Communications Corporationin kehittämä selaimessa suoritettava komentosarjakieli, jonka pääasiallinen tarkoitus on lisätä toiminnallisuutta web-sivuille. JavaScriptin virallinen nimi on ECMAScript.

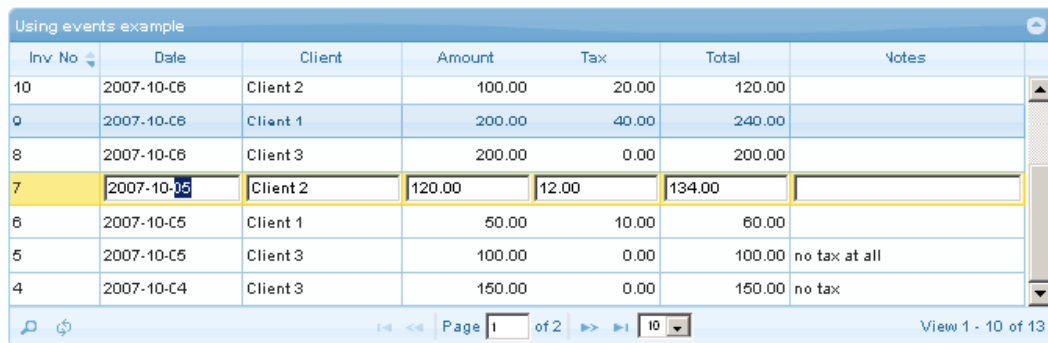
AJAX on lyhenne sanoista Asynchronous JavaScript And XML ja tarkoittaa koelmaa tekniikoita, joilla web-sivuista saadaan vuorovaikutteisempia. Ajaxin avulla selaimessa ajettava JavaScript-ohjelma voi vaihtaa pieniä määriä dataa palvelimen kanssa (esim. XML- tai JSON-muodossa) ilman että koko sivua tarvitsee päivittää. Käyttäjä voi siten esimerkiksi täyttää lomakkeen ja lähettää sen palvelimelle ilman koko sivun lataamista uudelleen. (Paajanen 2011)

JSON (JavaScript Object Notation) on tekstipohjainen tiedonsiirtomuoto, jota käytetään paljon AJAX-ohjelmissa tiedon siirtoon selaimen ja palvelimen välillä. Tyypillisesti palvelinpuolen ohjelma hakee pyydetyn tiedon tietokannasta, muotoilee sen JSON-muotoon ja lähettää sen asiakkaan päässä suoritettavalle JavaScript-koodille, joka muotoilee sen käyttäjälle näytettävään muotoon web-

sivulle. JavaScript-koodi voi myös ottaa käyttäjän syöttämää tietoa, muotoilla sen JSON-muotoon ja lähettää palvelimelle, jossa palvelinpuolen ohjelma muotoilee sen tietokantaan sopivaan muotoon ja tallentaa sen tietokantaan.

6.3.1 jqGrid

jqGrid on ilmainen AJAXia hyödyntävä komponentti, jolla voi näyttää ja muokata taulukkomuotoista dataa web-selaimessa. jqGrid käyttää jQuery-JavaScript-kirjastoa. Komponentilla on helppoa luoda taulukko, joka hakee näytettävän datan palvelimelta, ja jolla voi muokata näytettyä dataa. Kuvassa 6.1 on esimerkki jqGrid-tilistä ja riviä muokkauksesta.



The screenshot shows a web browser window titled "Using events example" containing a jqGrid table. The table has columns: Inv No, Date, Client, Amount, Tax, Total, and Notes. The data is as follows:

Inv No	Date	Client	Amount	Tax	Total	Notes
10	2007-10-C6	Client 2	100.00	20.00	120.00	
9	2007-10-C6	Client 1	200.00	40.00	240.00	
8	2007-10-C6	Client 3	200.00	0.00	200.00	
7	2007-10-C6	Client 2	120.00	12.00	134.00	
6	2007-10-C5	Client 1	50.00	10.00	60.00	
5	2007-10-C5	Client 3	100.00	0.00	100.00	no tax at all
4	2007-10-C4	Client 3	150.00	0.00	150.00	no tax

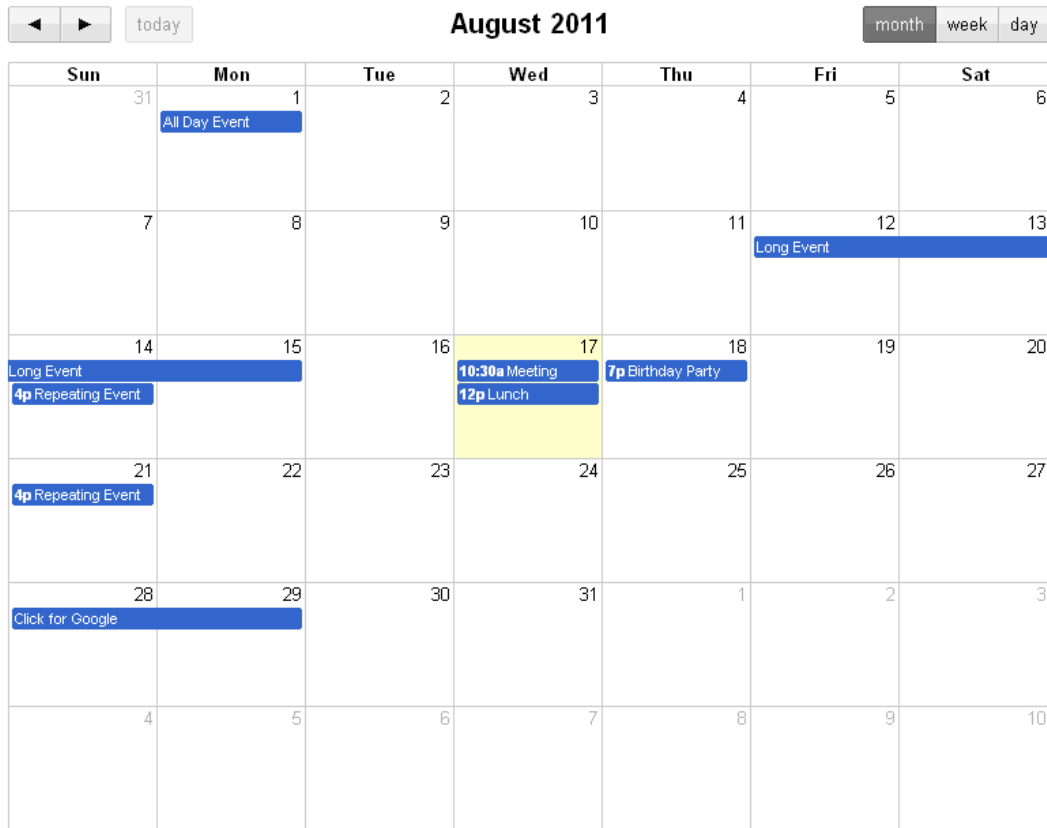
The row with Inv No 7 is highlighted in yellow, indicating it is selected for editing. The Date field in this row is currently set to 2007-10-C6. The bottom of the window shows a pagination control: "Page 1 of 2" and "View 1 - 10 of 13".

Kuva 6.1 Esimerkki jqGrid-tilistä (jqGrid demos)

Kuvasta 6.1 näkee, kuinka riviä voi muokata suoraan taulukosta. Muokkaamisen jälkeen pelkästään muutettu rivi lähetetään palvelimelle. Tämä vähentää liikennettä selaimen ja palvelimen välillä ja yksinkertaistaa käyttöliittymää.

6.3.2 FullCalendar

FullCalendar on ilmainen avoimen lähdekoodin jQuery-kirjastoa käyttävä kalenterikomponentti, joka hakee kalenterissa näytettävät tapahtumat palvelimelta AJAX-tekniikalla JSON-muodossa. FullCalendar osaa myös hakea tapahtumia esimerkiksi Google Calendar -palvelusta. Tapahtumia voi venyttää sekä raahata ja pudottaa kalenterissa.



Kuva 6.2 FullCalendarin kuukausinäkömä (Adam Shaw 2011)

FullCalendarissa on kuukausi- viikko- ja päivänäkymät. Kuvassa 6.2 nähdään kuukausinäkömä sekä muutamia tapahtumia.

6.4 Relaatietietokanta ja SQL

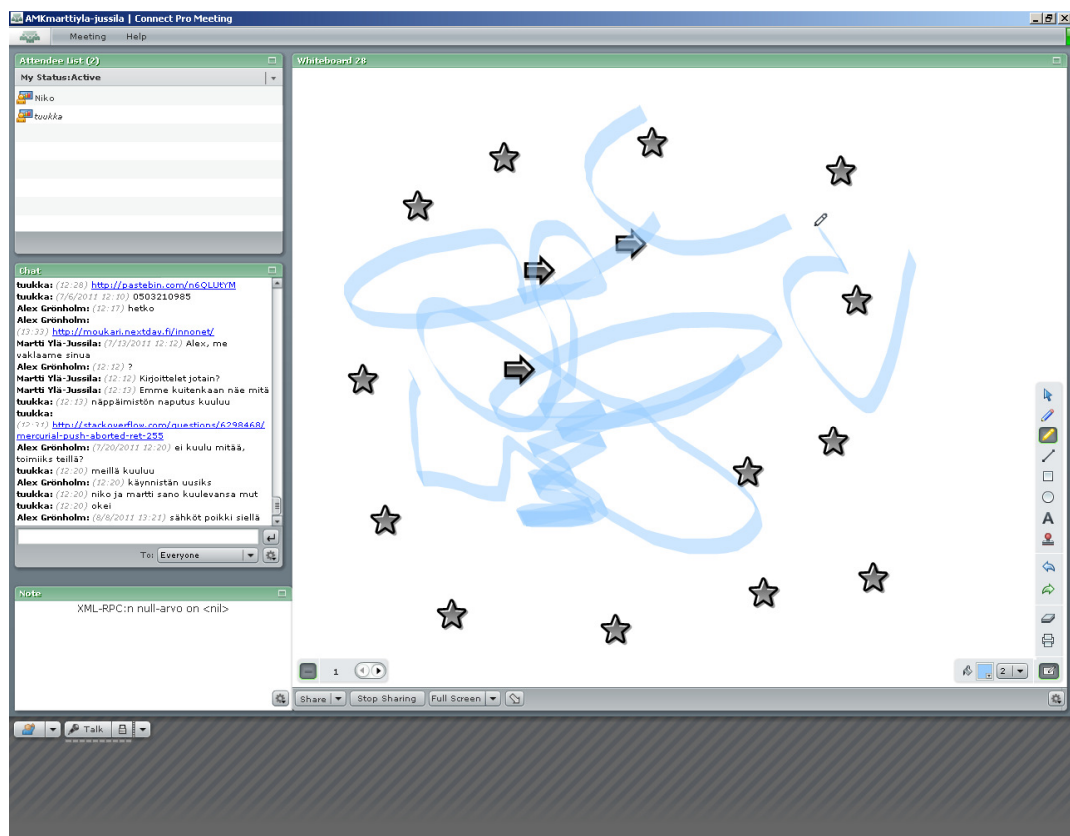
Relaatietietokanta on tiedon tallentamisen muoto, jossa tieto on järjestetty tauluiksi ja niiden suhteiksi. Taulut koostuvat riveistä ja sarakkeista. Jokaisella taululla tulee olla yhdestä tai useammasta sarakkeesta koostuva primääriavain, joka yksilöi jokaisen siihen tauluun kuuluvan rivin. (Rissanen 2010, Paajanen 2011)

SQL on lyhenne sanoista Structured Query Language. Se on standardoitu kieli relaatietietokantojen käsittelyyn ja sisältää kaikki tarpeelliset toiminnot tietokantojen luomista, muokkausta, poistoa sekä niiden sisältämän tiedon käsittelyä varten. (Rissanen 2010, Paajanen 2011.)

6.5 Etäkokousjärjestelmät ja Adobe Connect

Etäkokousjärjestelmät vapauttavat työntekijät hajautettuja projekteja vaivaavista maantieteellisistä kommunikaatorajoituksista.

Adobe Connect (entinen Acrobat Connect Pro) on Flash-tekniikalla toimiva selainkäyttöinen etäkokousjärjestelmä. Se mahdollistaa puheen ja videon käytön etäkokouksissa sekä tiedostojen jakamisen ja työpöydän näyttämisen kokoukseen osallistuvien kesken. Lisäksi siinä on tekstipohjainen keskustelu ja virtuaalinen liitutaulu.



Kuva 6.3 ACP-etäkokousjärjestelmän liitutaulu hyötykäytössä

Liitutaulua voi käyttää esimerkiksi järjestelmän arkkitehtuuria havainnollistavien kaavioiden piirtämiseen.

6.6 Versionhallinta

Versionhallinta on ohjelmistoprojektin tuotosten hallintaa. Tavoitteena on ohjelmiston kehityshistorian seuranta. Tarkoituksena on tallettaa kaikki tiedostot sekä niiden muutoshistoria siten, että mikä tahansa muutos on nähtävissä ja peruttavissa myöhemmin, ja että mihin tahansa aiempaan versioon voidaan palata. Tiedetään myös, kuka teki valitun muutoksen ja koska. Versionhallinnan käytöllä pyritään mahdollistamaan myös usean kehittäjän yhtäaikainen työskentely samassa projektissa sekä estämään mahdollisia samanaikaisesta kehittämisestä johtuvia ristiriitoja. Versionhallinta on havaittu välttämättömäksi, kun ohjelmistoprojektien koko on ajan myötä kasvanut. Yksinkertaisimmillaan versionhallinta vaatisi, että jokainen kehittäjä tekisi tiedostosta kopion jokaisen muutoksen yhteydessä ja nimeäisi sen uudelleen siten, että nimessä näkyisi kehittäjän nimi sekä ajankohta. Tämä lisäisi kuitenkin kehittäjien työtä ja olisi hyvin altis virheil- le. Tämän takia ensimmäiset versionhallintajärjestelmät kehitettiin 1970-luvulla.

6.6.1 Subversion ja keskitetty versionhallinta

Subversion eli SVN on keskitetty versionhallintajärjestelmä, jossa on yksi keskuspalvelin, jota kaikki kehittäjät käyttävät. Tällä palvelimella on versioarkisto (repository). Arkisto sisältää koko hakemistorakenteen sekä muutoshistorian. Käyttäjät hakevat arkiston palvelimelta (update), tekevät omat muutoksensa tiedostoihin, tallentavat ne ja lähettävät ne keskuspalvelimelle (commit), jolloin ne ovat myös muiden käyttäjien saatavilla.

6.6.2 Mercurial ja hajautettu versionhallinta

Mercurial on hajautettu versionhallintajärjestelmä. Se eroaa SVN:stä siten, että jokainen käyttäjä luo oman arkistonsa (repository) omalle työasemalleen ja commit-komento tallentaa muutokset tähän paikalliseen arkistoon. Mitään keskusarkistoa ei ole, vaan kaikki arkistot ovat samanarvoisia.

Subversionin ja muiden keskitettyjen versiohallintajärjestelmien kanssa ongelmana on usein se, että commit-käsky pakottaa muutokset suoraan kaikille käyttäjille keskuspalvelimen kautta. Versiohallinnan tarkoitus on kuitenkin se, että voidaan tehdä muutoksia, ja koska tahansa palata aiempaan versioon siinä ta-

pauksessa, että muutokset eivät toimineetkaan. Tämä onnistuu vain, jos kaikki muutokset commitoidaan. Koska keskitetyt versionhallintajärjestelmät pakottavat kaikki commitoidut muutokset kaikille käyttäjille, käyttäjät eivät commitoi muutoksia mielellään, ja lopputuloksena versionhallinnasta ei saada kaikkea hyötyä irti. Mercurialissa ja muissa hajautetuissa versionhallintajärjestelmissä jokaisella käyttäjällä on oma versioarkistonsa, jolloin omat muutokset saa talteen ja niiden välillä voi liikkua vapaasti, joten muutokset tulee commitoitua useammin.

Toinen tärkeä ero Mercurialin ja Subversionin välillä on, että Subversion seuraa koko hakemistorakenteen versioita (revision), eli miltä hakemistorakenne näytti jollain hetkellä, kun taas Mercurial seuraa muutoksia (changeset). Jos kaksi Subversionin käyttäjää muutti samaa tiedostoa commitien välissä, Subversion ei välttämättä pysty yhdistämään muutoksia, koska se ei seurannut kuka muutti ja mitä. Mercurial seuraa juuri tätä, jolloin muutosten yhdistäminen on yleensä suhteellisen vaivatonta.

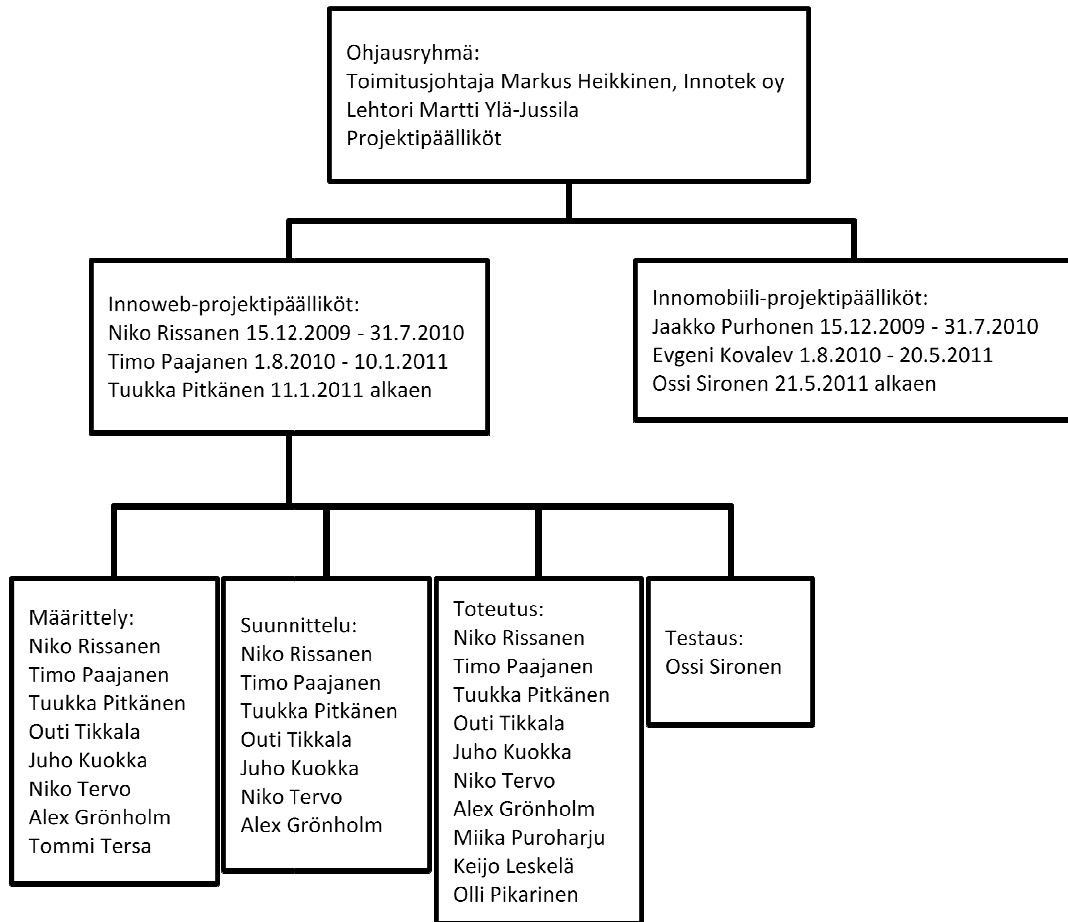
Kun kaikki arkistot ovat keskenään samanarvoisia ja yksittäiset muutokset tiedostoissa on arkistoitu, järjestelmä on todella joustava verrattuna keskitettyyn versionhallintaan. Kuka tahansa kehittäjä voi ”vetää” (pull) omaan arkistoonsa keneltä tahansa toiselta käyttäjältä minkä tahansa talletetun muutoksen.

Toisaalta, kun kehittäjille annetaan oma versioarkistonsa, jonne he pystyvät commitoimaan ja tallentamaan versiohistoriansa jakamatta sitä muille, riskinä on, että kehittäjät eivät ole tietoisia toistensa töistä eivätkä pääse testaamaan tai arvioimaan sitä. Joku kehittäjä saattaa haluta tehdä jonkun ominaisuuden tai osion alusta loppuun valmiiksi ennen kuin näyttää sen muille eikä halua näyttää keskeneräistä työtä.

7 Projektin organisaatio ja vaiheet

Projekti oli lähtenyt liikkeelle alun perin jo vuosia aiemmin, ja tuolloin oli toteutettu alustava versio järjestelmästä. Tämä versio ei kuitenkaan vastannut asiakkaan tarpeita, joten asiakas antoi keväällä 2010 Saimaan ammattikorkeakoulun opiskelijoille tehtäväksi jatkokehittää tätä järjestelmää. Pian kuitenkin selvisi,

että tämän järjestelmän jatkokehitys olisi toivotonta, joten silloin katsottiin parhaaksi aloittaa puhtaalta pöydältä ja määrittellä, suunnitella ja toteuttaa järjestelmä kokonaan alusta käyttäen vanhaa järjestelmää avuksi asiakkaan tarpeiden kartoittamisessa.



Kuva 7.1 Innonet-hankkeen organisaatiokaavio

Projekti lähti Saimaan ammattikorkeakoulussa liikkeelle projektin aloittajien (Rissanen ja Purhonen) tekemästä esitutkimuksesta, jonka pohjalta tehtiin projektisuunnitelma. Projektisuunnitelmassa projekti jaettiin seuraaviin vaiheisiin: vanhaan järjestelmään tutustuminen, PHP-kieleen tutustuminen, tekniikoiden kartoittaminen, toiminnallinen määrittely, uusien tekijöiden opastus, toiminnallisen määrittelyn opiskelu, tekninen määrittely, järjestelmän toteutus, moduulitestaus, integraatiotestaus, käyttöönottotestaus, käyttöohjeen laatiminen, käyttöönotto ja ylläpito. Nämä vaiheet olivat jossain määrin myös delegoitu senhetkisille projektiryhmäläisille.

Projektin etenemistä seurattiin viikoittaisissa palavereissa, joissa käytiin läpi edellisen viikon tehdyt työt, seuraavan viikon suunnitellut työt, eteen tulleet ongelmat sekä esiin nousseet riskit. Asiakaspalavereita pidettiin satunnaisesti, yleensä asiakkaan aikataulun mukaan. Fyysisten välimatkojen takia palaverit pidettiin Adobe Connect Pro-ohjelman välityksellä.

Liityin projektiin marraskuussa 2010. Aluksi oma tehtäväni oli määrittellä, suunnitella ja toteuttaa töiden hallinta ensin työharjoitteluna ja sitten jatkaa opinnäytetyönä. Projektipäällikkönä oli tällöin Timo Paajanen. Ryhmässä olivat myös Juho Kuokka, Niko Tervo, Outi Tikkala, Miika Puroharju, Ossi Sironen sekä Keijo Leskelä. Koulun ulkopuolelta mukaan tuli DI Alex Grönholm, toiminimi Nextday Solutions. Innomobiili-projekti oli Evgeni Kovalevin vastuulla.

Timo Paajanen sai opinnäytetyönsä valmiiksi ja jäi pois projektista tammikuussa 2011, jolloin minut valittiin projektipäälliköksi. Tällöin koulun projektiryhmään kuuluivat minun lisäksi käytännössä Niko Tervo, Juho Kuokka ja Outi Tikkala. Tikkala jäi helmikuussa pois.

Pyrimme jakamaan projektin osiin eri tekijöille. Omalla vastuullani oli töiden hallinta, Kuokan vastuualueena oli asiakkaiden hallinta, Tervon kiinteistöjen hallinta, ja Tikkalan yritysten hallinta. Käytännössä kaikki joutuivat ensin määrittelemään ja suunnittelemaan oman osionsa, ja tässä pyrimme mahdollisuuksien mukaan tekemään yhteistyötä. Käytännössä myös toteutus tapahtui välillä ryhmätöinä, välillä pariohjelmointina, ja välillä yksin. Työnjako ei siis ollut niin järkevä, että se olisi estänyt auttamasta muita.

Tiedotus ja kommunikaatio tapahtuivat suullisesti niiltä osin kuin se oli mahdollista ja sähköpostilla silloin kun suullinen kommunikaatio ei ollut mahdollista. Keväällä otimme käyttöön Bitbucketin, jota käytettiin työnjakoon, projektiseurantaan ja ongelmien seurantaan. Bitbucketista ja sen käytöstä kerrotaan lisää luvussa 9.

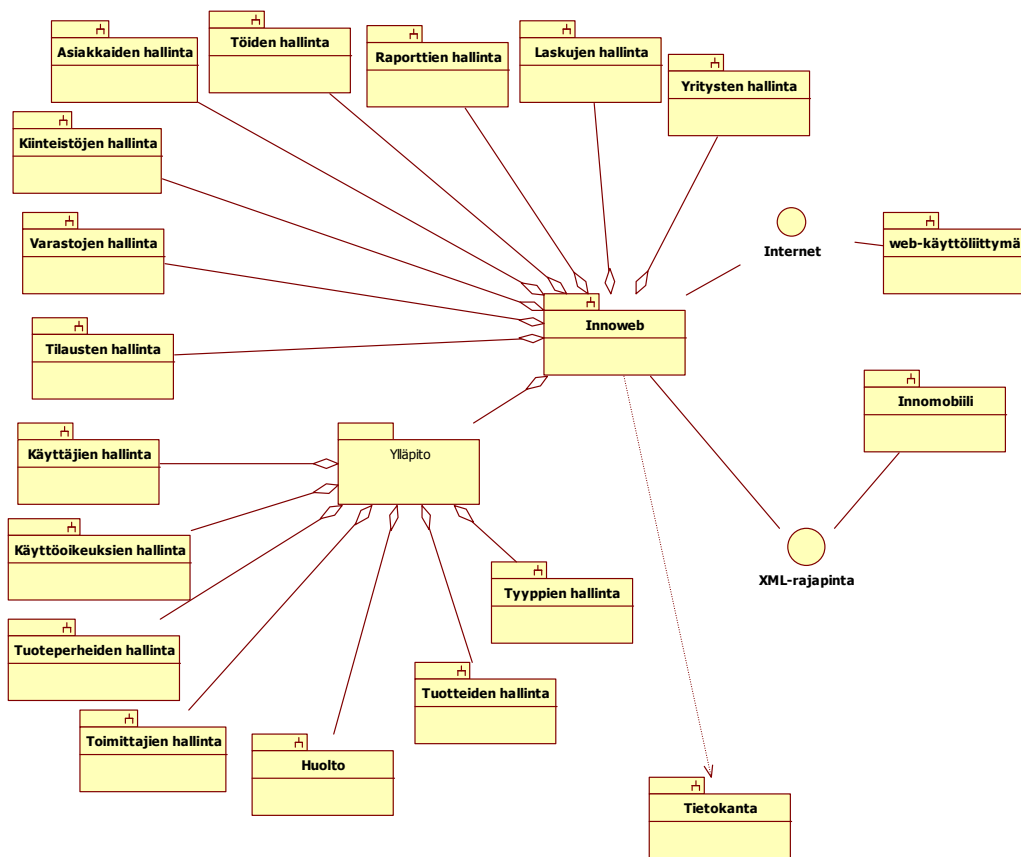
Kehitys tapahtui jossain määrin iteratiivisesti. Usein ryhdyimme toteuttamaan jotain huomataksemme vain, että määrittely oli puutteellinen, ja jouduimme palaamaan määrittelyvaiheeseen ja sen jälkeen suunnittelemaan osion uudestaan

uudemman määrittelyversion pohjalta. Tämän jälkeen palasimme taas toteutusvaiheeseen. Toteutuksen jälkeen oli moduuli- ja integraatiotestauksen vuoro.

Kesällä saimme lisää työvoimaa lähinnä testaamaan ja määrittelemään järjestelmää.

8 Järjestelmän esittely

Innoweb-järjestelmä on PHP:llä ja JavaScriptillä toteutettu selainkäyttöinen toiminnanohjausjärjestelmä, jonka tarkoituksena on automatisoida tiedonkulkua ja käsittelyä yrityksen sisällä ja näin tehostaa toimintaa ja vähentää inhimillisiä virheitä. Yrityksen päivittäisissä toiminnoissa liikkuu ja syntyy huomattava määrä tietoa lähtien uuden asiakkaan lisäämisestä mittausraportteihin, laskun lähettämiseen tai varaston saldon ylläpitoon.

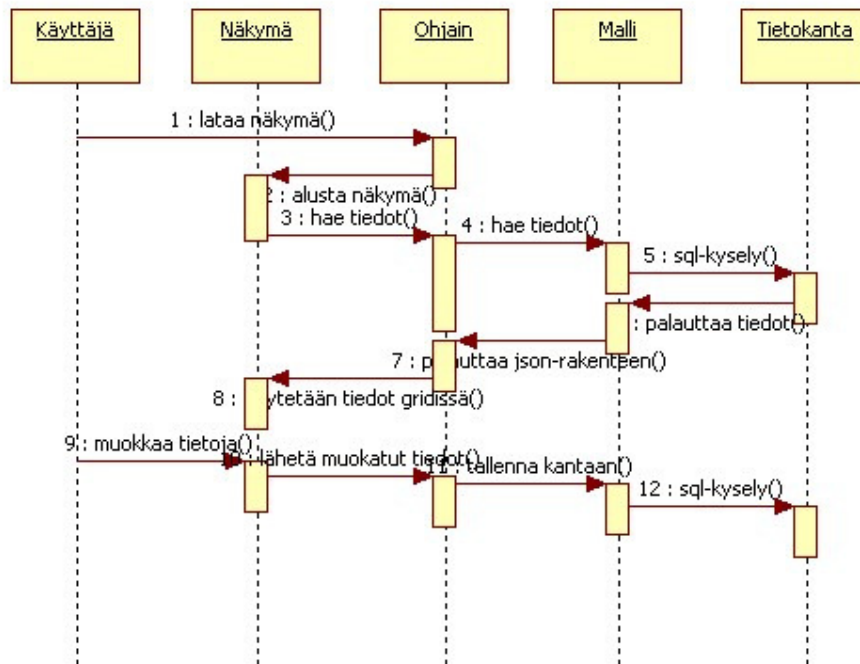


Kuva 8.1 Innoweb-järjestelmän looginen näkymä.

Kuvassa 8.1 nähdään järjestelmän jako alijärjestelmiin. Osa alijärjestelmistä on niputettu ylläpitoon. Innoweb-järjestelmää käytetään internetin välityksellä selainpohjaisesti.

Innoweb-järjestelmän lisäksi järjestelmään kuuluu Innomobiili-sovellus, joka on Innotek Oy:n asentajien käyttöön tarkoitettu puhelimesta toimiva tiedonkeruu- ja raportointisovellus. Innomobiili käyttää palvelimen kanssa kommunikointiin XML-rajapintaa. Innomobiiliin ladataan päivän alussa työnannot, kohteiden osoitteet, tuotetiedot ynnä muuta töiden tekemiseen vaadittavaa tietoa. Työpäivän aikana asentajat tekevät Energo-mittauksia tai asennuksia, havaitsevat kohteissa vikoja yms., ja päivän lopussa he lähettävät työraportit, mittaustulokset ja vikaraportit palvelimelle.

Innoweb-järjestelmä käyttää löyhähkösti MVC, eli malli-näkymä-ohjain-arkkitehtuuria (model-view-controller). MVC-arkkitehtuurissa ohjelma on jaettu kolmeen osaan, jossa näkymä huolehtii käyttäjälle näkyvästä käyttöliittymästä, malli huolehtii tiedon varastoinnista eli tässä tapauksessa tietokannasta, ja ohjain on vastuussa ohjelman toimintalogiikasta.



Kuva 8.2 Sekvenssikaavio järjestelmän yleisarkkitehtuurista

Käyttäjä kutsuu selaimella ohjainta (controller), joka puolestaan lataa käyttäjälle selaimessa näkyvän näkymän. Näkymässä oleva JavaScript-koodi pyytää ohjainluokalta näytettävän datan. Ohjain tyypillisesti pyytää näitä malliluokalta, joka hakee ne tietokannasta, ja välittää ne edelleen näkymän JavaScript-komponentille. Lopuksi näkymä muotoilee datan käyttäjälle näytettäväksi ja muokattavaksi.

Seuraavassa käydään läpi lyhyesti Innoweb-järjestelmän tällä hetkellä valmiit sekä suunnitellut osiot.

8.1 Asiakkaiden hallinta

Asiakkaiden hallinnassa lisätään ja poistetaan asiakkaita sekä heidän yhteyshenkilöitään.

Myös ei-aktiiviset asiakkaat

Asiakkaat						Osoitetiedot			
Asiakasnumerc	Nimi	Y-Tunnus	Puhelin	WWW-osoite	Aktiivinen	Osoitetyyppi	Katuosoite	Postinumero	Toimipaikka
19	Boris Jeltsin	1234567-8	1234	jeltsin.lollerball	<input checked="" type="checkbox"/>	käyntiosoite	jokukatu 13	00400	mikämikämaa
22	Megaman Y	1234567-4	234980	www.tua.cc	<input checked="" type="checkbox"/>				
27	rakuuna mies				<input checked="" type="checkbox"/>				
29	muumipappa	123678		www.muumit.f	<input checked="" type="checkbox"/>				
30	aku ankka	-	07020678	www.ankkalinr	<input checked="" type="checkbox"/>				
31	Testi Asiakas	13345-6	050667788	www.testiasia	<input checked="" type="checkbox"/>				
32	erkki esimerki	123456	0502345123	www.example.	<input checked="" type="checkbox"/>				

+ Lisää Muokkaa Poista Etsi Rivit 1 - 7 / 7

Yhteyshenkilöt									
Sukunimi	Etunimi	Nimike	Puhelinnumero	Osoite	Postinumero	Toimipaikka	Email	Lisätiedot	
peppi	pitkatossu	sihteeri	0800123123	huvikumpu	00900	huvikumpu	peppi@pitkatossu.	*Isoilla ihmisillä ei	

+ Lisää yhteyshenkilö Muokkaa yhteyshenkilöä Poista yhteyshenkilö Etsi

Kuva 8.3. Ruutukaappaus asiakkaiden hallinnasta.

Kuvassa 8.3 nähdään ylhäällä vasemmalla lista asiakkaista. Kun listasta valitaan joku asiakas, oikeanpuoleiseen listaan tulee lista asiakkaan osoitteista. Alas tulee lista yhteyshenkilöistä. Yritysassiakkailla on tyypillisesti pääkonttori, useita haarakonttoreita tai toimipisteitä sekä mahdollisesti eri laskutusosoite. Yksityisasiakkailla on yleensä vain yksi osoite. Kuvassa näkyvät tiedot eivät ole oikeiden asiakkaiden tietoja.

8.2 Yritysten hallinta

Yritysten hallinnassa hallitaan isännöitsijöitä, isännöintitoimistoja ja huoltoyhtiöitä.

Huoltoyhtiöt							
Nimi	Y-tunnus	WWW-osoite	Sähköpostiosoite	Puhelinnumero	Katuosoite	Postinumero	Postitoimipaikka
huoltoyhtio	8685754	www.huoltoyhtio.fi	info@huoltoyhtio.fi	0700111111			
Talotakurit	12345678	www.talotakurit.cc	seppo@talotakurit.	0500123456	Pihlajakatu 23	00140	Helsinki

+ Lisää Muokkaa Poista Etsi Rivit 1 - 2 / 2

Isännöintitoimistot							
Nimi	Y-tunnus	WWW-osoite	Sähköpostiosoite	Puhelinnumero	Katuosoite	Postinumero	Postitoimipaikka
lyijynen	y6565354	www.lyijynen.fi	esa@lyijynen.fi	0800123123	Pohjolankatu 23	53101	lapeen Rata
oy firma ab	251256	www.firma.fi	info@firma.fi	0600888888			

+ Lisää Muokkaa Poista Etsi Rivit 1 - 2 / 2

Isännöitsijät					
Etunimi	Sukunimi	Sähköpostiosoite	Katuosoite	Postinumero	Postitoimipaikka
ismo	isännöitsijä	ismo@isannointitoim.kotikatu 13		53100	lepra

Kuva 8.4 Ruutukaappaus yritysten hallinnasta

Kuvassa näkyvät taulukot huoltoyhtiöitä, isännöintitoimistoja ja isännöitsijöitä varten sekä lisää-, muokkaa-, poista- ja etsi-painikkeet.

8.3 Kiinteistöjen hallinta

Kiinteistöjen hallinnassa hallitaan kiinteistöjä sekä niihin kuuluvia rakennuksia. Tiedot voivat muuttua, ja voi olla tarpeellista päästä näkemään myös aiempia tietoja, joten tietojen muuttuessa ne säilytetään kiinteistöhistoria- ja rakennushistoriatauluissa.

The screenshot displays a web application interface for property management. It is divided into several sections:

- Kiinteistöt (Properties):** A table with columns 'Kiinteistötunnus' and 'Nimi'. The last row is highlighted in yellow, showing '567ach' and 'Testinimi2'. Below the table are buttons for '+ Lisää', 'Muokkaa', 'Poista', and 'Etsi', along with 'Rivit 1 - 4 / 4'.
- Kiinteistön tiedot (Property Details):** A panel with input fields for: Tunnus (567ach), Nimi (Testinimi2), Tyyppi (toinen esimerkkikiin), Rakennuksia (2), Omistaja (Kalervo Uurna), Isännöintitoimisto, Isännöitsijä (ismo isännöitsijä), and Huoltoyhtiö (huoltoyhtio).
- Rakennukset (Buildings):** A table with columns 'Tunnus', 'Katuosoite', and 'Paikkakunta'. The first two rows are highlighted in yellow, showing '456' with 'Testikatu #' and 'Täälläpäs', and '9999' with 'Testikatu 2' and 'Tuolla'. Below the table are buttons for '+ Lisää', 'Muokkaa', 'Poista', and 'Etsi', along with 'Rivit 1 - 2 / 2'.
- Rakennuksen tiedot (Building Details):** A panel with input fields for: Tunnus (456), Tyyppi (kerrostalo), Katuosoite (Testikatu #), Postitoimipaikka (01234 Täälläpäs), Rakennusvuosi (1999), Peruskorjausvuosi (2000), Kustannuspaikka (asdf), Huoneistoja (40), Asukkaita (40), Rakennuskuutiot (400), Vesikuutiot (400), Vesikuutiot / vuosi (400), Toteutunut lämpö (400), Normeerattu lämpö (400), Lämmitys / vuosi (12), and Lämmitystyyppi (puu).
- Kiinteistön aiemmat tiedot (Previous property details):** A collapsed section at the bottom.
- Rakennuksen aiemmat tiedot (Previous building details):** A collapsed section at the bottom.

Kuva 8.5. Kiinteistöjen hallinta

Kuvassa 8.5 näkyy vasemmalla ylhäällä lista kiinteistöistä. Kun joku kiinteistö valitaan, oikealla näkyy kiinteistön tarkemmat tiedot, ja vasemmalla alhaalla lista kiinteistöön kuuluvista rakennuksista. Vastaavasti rakennuksen tarkemmat tiedot näkyvät alhaalla oikealla. Lisäksi näiden alapuolella on valitun kiinteistön ja rakennuksen aiemmat talletettavat tiedot supistettuina.

8.4 Tilausten hallinta

Tilausprosessi lähtee tarjouksesta. Kun asiakas on hyväksynyt myyjän tekemän tarjouksen, tarjouksesta tehdään tilaus. Tällöin myös laaditaan työnanto, joka voidaan ladata Innomobiiliin suoritettavaksi. Kun työnanto on tehty, tilaus siirtyy laskutukseen. Tässä osiossa laaditaan tilaukset sekä työnannot.

8.5 Töiden hallinta

Töiden hallinnassa aikataulutetaan tilausten hallinnassa laaditut työnannot ja delegoidaan ne asentajille. Kalenterissa näytetään työnannot värikoodattuna niiden tilan mukaan. Työnannon tarkemmat tiedot -näkyvässä näytetään työnantoon kuuluvat työsuoritteet ja niiden tila, tuotteet, kohteen, asiakkaan ja tilauksen tiedot sekä työnantoon määrätyt asentajat.

Tarkoituksena olisi myös toteuttaa raportointityökaluja, joilla voitaisiin seurata esimerkiksi sitä, kuinka kauan jonkun tietyn tuotteen asentamiseen keskimäärin kuluu aikaa. Tällainen seuranta voi pitkällä aikavälillä tuottaa yrityksen toiminnan kehittämisen kannalta arvokasta informaatiota.

8.6 Raporttien hallinta ja laskujen hallinta

Raporttien hallinnassa tuotetaan Energo-kartoitusraportteja ja asennusraportteja asiakkaille asentajien tekemien kartoitusten ja asennusten pohjalta.

Laskujen hallinnassa käsitellään valmiit tilaukset ja muotoillaan ne ulkoisen laskutusjärjestelmän rajapinnan vaatimaan muotoon.

Nämä osiot ovat toistaiseksi kesken.

8.7 Varastojen hallinta

Varastojen hallinnassa nähdään kaikki varastot ja niissä olevat tuotteet ja saldot. Lisäksi tavaran siirrot sisään ja ulos hoidetaan täällä.

Myös käytöstä poistetut

Varastot

Nimi	Tunnus	Katuosoite	Postinro	Toimipaikka	Vastuhenkilö	Tyyppi	Aktiivinen
abcd	1234	abcde 1	12345	abcd	admin	Auto	<input checked="" type="checkbox"/>
Maken nissan	45	Seppäkatu 5	45671	Lohja	tuukka	Takakontti	<input checked="" type="checkbox"/>

+ Lisää Muokkaa Poista Rivit 1 - 2 / 2

Tuotteet

<input type="checkbox"/>	Nimi	Koodi	Hälytysraja	Saldo	Varaukset
<input checked="" type="checkbox"/>	tuote	567	45	56	0

+ Lisää Muokkaa Poista Siirrä

Erät

Lisätty	Määrä	Toimittaja	Ostohinta
2011-08-05 10:13:24	56	Toivo Toimittaja	56.00

+ Lisää Muokkaa Poista

Kuva 8.6 Varastojen hallinta

Kuvassa 8.6 näkyy ylhäällä taulukko varastoista. Alhaalla vasemmalla näkyy lista valitussa varastossa olevista tuotteista. Alhaalla oikealla on luettelo valitun tuotteen tuote-eristä.

8.8 Ylläpito

Ylläpidossa on toisaalta järjestelmän toiminnan kannalta välttämättömiä toimintoja ja toisaalta yrityksen liiketoiminnallisia toimintoja, joita ei tarvita päivittäisessä toiminnassa ja joihin pääsyä pitää rajoittaa jo pelkästään turvallisuussyistä.

Ylläpidosta löytyvät muun muassa tietokannan varmuuskopiointi ja palautus varmuuskopiosta, käyttäjien, käyttäjäryhmien ja käyttöoikeuksien hallinta sekä tuotteiden ja toimittajien hallinta.

9 Tehtyjä valintoja ja johtopäätöksiä

Järjestelmän toteutus oli aloitettu käyttämällä Codelgniter-kehystä ilman muita komponentteja. Tämän takia käyttöliittymää oli jouduttu toteuttamaan käsin HTML:llä ja CSS:llä, mikä oli varsin työlästä. Alex Grönholmin ehdotuksesta ryhdyimme käyttämään jqGrid-komponenttia projektissa. Koska juuri kellään meistä ei ollut aiempaa kokemusta JavaScriptistä tai jQuery:stä, nämä tekniikat vaativat muutaman kuukauden opiskelun, mutta sen jälkeen työ nopeutui todella paljon. Komponentti sekä Alex Grönholmin kirjoittama rajapinta jqGridin ja Codelgniterin välille automatisoivat ison osan tietokantasovelluksen rutiinointimenpiteiden (lisää, selaa, muokkaa, poista) sekä käyttöliittymän toteutuksesta, ja kun sitä kerran oppi käyttämään, työ eteni nopeasti. Koska uutta koodia tarvitsi kirjoittaa vähemmän, myös virheitä tuli vastaavasti vähemmän.

Alex Grönholm ehdotti myös versionhallinnan käyttöönottoa, mikä osoittautuikin välttämättömäksi. Tähän asti projekti oli edennyt sen verran hitaasti ja pienellä panostuksella, ettei sitä ollut tarvittu. Aluksi käytössä oli Subversion. Tämä oli parempi kuin ei mitään, mutta se aiheutti myös huomattavasti ongelmia esimerkiksi niissä tilanteissa, joissa kaksi kehittäjää oli samaan aikaan muuttanut samaa tiedosta, tai jos joku oli unohtanut päivittää oman kopionsa versioarkistosta. Myöhemmin käyttöön otettiin Mercurial, joka oli hajautettuna versionhallintajärjestelmänä huomattavasti monimutkaisempi oppia, mutta jonka kanssa tuli käytännössä vähemmän ongelmia. Lisäksi ryhdyimme käyttämään Bitbucket-palvelua versioarkistointiin sekä projektinhallintaan. Bitbucket tarjoaa Mercurial-versioarkistoinnin lisäksi ”issue tracker” eli ongelmanseurantatoiminnon, jota on käytetty projektissa virheraportointiin ja tekemättömien töiden delegointiin ja seurantaan. Tämä on selkeyttänyt projektin koordinoitua merkittävästi.

Töiden hallintaan tarvittiin asiakkaan toiveesta kalenteri töiden aikataulutusta varten. Valmiita kalenterikomponentteja on olemassa useampia, mutta ongelmana oli löytää yksi ainoa, joka täyttäisi vaatimukset. Kalenterin tulee olla suunniteltu useiden käyttäjien töiden aikataulutukseen eikä yhden ihmisen henkilökohtaiseen käyttöön. Sen tulee olla sovitettavissa tähän järjestelmään mahdollisimman vaivattomasti. Työnantaja pitää pystyä raahaamaan ja pudottamaan päivältä toiselle. Kalenterikomponentin lisenssin tulee sallia käyttö tässä

järjestelmässä. Lisäksi sen tulee olla mahdollisimman helppokäyttöinen sekä kehittäjälle, eli minulle, että loppukäyttäjälle, eli asiakkaalle. Osa vaatimuksista oli asiakkaan vaatimia, osa minun. Lopulta löysimme FullCalendar-komponentin, joka täytti kaikki vaatimukset riittävän hyvin. FullCalendarin käytön opiskelun jälkeen töiden hallinta eteni nopeasti.

Koska projektiryhmäläisillä ei tammikuusta 2011 eteenpäin ollut juuri muita opintoja tämän projektin lisäksi, katsoimme aiheelliseksi järjestää projektille pysyvän työtilan, mikä edisti kommunikaatiota ja paransi viihtyvyyttä. Valitettavasti se myös ajoittain häiritsi keskittymistä huomattavasti ja saattoi tätä kautta hidastaa projektin etenemistä jonkin verran. Viime kädessä tämä johtui motivaation ja itsekurin puutteesta. Yhteisellä työtilalla oli kuitenkin hyvätkin puolensa, vaikka työ ei aina edennytäkään kovin nopeasti. Ongelmien tai epäselvyyksien ilmetessä oli helppoa kysyä kaverilta. Lisäksi projektin koordinointi oli helppoa, koska tiesin koko ajan mitä muut olivat tekemässä ja miten heidän työnsä eteni.

10 Yhteenveto

Asiakkaan tavoitteena on kehittää liiketoimintaansa, mikä synnytti tarpeen ohjelmistolle, joka tukee liiketoimintaa. Tällaisen ohjelmiston kehitys oli alkanut jo vuosia ennen tämän projektin alkua, mutta tämä ei ollut vastannut asiakkaan tarpeita, joten tämän projektin puitteissa työ alkoi alusta. Vanha järjestelmä kuitenkin auttoi kartoittamaan tavoitteita ja vaatimuksia.

Ohjelmiston hyöty on mielestäni selkeä. Innomobiili ja Innoweb yhdessä helpottavat ja automatisoivat koko liiketoimintaprosessia huomattavasti, selkeyttävät asentajien töiden koordinointia sekä vähentävät asentajien paperityötä kentällä. Tämä tulee nopeuttamaan tilausten etenemistä toimitukseen ja edelleen laskutukseen. Tästä saatava liiketoiminnallinen hyöty on kiistämätön. Pitkällä tähtäimellä järjestelmä tulee myös helpottamaan yrityksen liiketoiminnan kehittämistä edelleen.

Kun itse liityin projektiin, henkilökohtaisena tavoitteenani oli saada töiden hallinta valmiiksi opinnäytetyön puitteissa. Jouduin kuitenkin ottamaan vastuulleni muitakin projektin osa-alueita. Projektipäällikkönä jouduin koordinoimaan mui-

den tekemisiä, seuraamaan projektin etenemistä, hoitamaan yhteydenpidon asiakkaan kanssa sekä huolehtimaan työskentelyedellytyksistä, esimerkiksi että työasemille on asennettu tarpeelliset ohjelmistot. Koska tunsin järjestelmän laajemmin kuin useimmat muut projektiryhmäläiset laajemman tuntimääräisen panostukseni takia, sain jossain vaiheessa tietokannan muutosten suunnittelun ja toteutuksen vastuulleni. Lisäksi määrittelydokumentin ylläpito vei aikaani.

Töiden hallinnassa pystyy nyt laatimaan työnantoja, aikatauluttamaan ne, määrittämään ne asentajille sekä seuraamaan niiden etenemistä. Käytännössä osio vaatisi vielä huomattavaa panostusta, että kaikki asiakkaan vaatimukset täyttyisivät, mutta tärkeimmät toiminnot ovat valmiit.

Asiakkaan kiireet sekä sijainti toisella paikkakunnalla ovat rajoittaneet kommunikaatiota, mikä on vaikeuttanut ohjelmiston määrittelyä ja toteutusta. Havaitsin, että iteratiivinen, prototyyppilähtöinen lähestymistapa on toimiva keino päästä asiakasta tyydyttävään lopputulokseen. Vaatimuksia ei voida saada kirjattua kerralla ylös ennen kuin jotain konkreettista on toteutettu. Kun asiakas näkee prototyypin, hänen on helppo huomata, mitä siitä puuttuu tai mikä siinä on turhaa. Viimeistään tällöin saadaan korjattua puutteellisesta määrittelystä johtuneet virheet.

Vaatimusten määrittelyssä yksi ongelma oli, että asiakkaan liiketoimintaa ei ollut kuvattu juuri lainkaan. Tämä vaikeutti toiminnallisen määrittelyn ymmärtämistä. Määrittely kertoi, mitä asiakas halusi, mutta ei kertonut, mitä merkitystä jollain vaatimuksella oli liiketoiminnan kannalta. Loppujen lopuksi asiakas joutui kuvailemaan liiketoimintaprosessia jokaisen toteutettavan osion osalta vielä senkin jälkeen, kun se oli jo kerran määritelty ja mahdollisesti ensimmäinen prototyyppi oli jo toteutettu.

Useat projektiin osallistuneet opiskelijat eivät olleet ennen ohjelmoineet PHP:llä tai JavaScriptillä. Monet projektiryhmäläiset joutuivat opettelemaan nämä kielet sekä käytetyt komponentit ohjelmistoa toteuttaessaan sekä pyytämään neuvoa muilta, mikä hidasti kaikkien työskentelyä.

Ongelmana oli myös se, että opiskelijoiden aika on rajallinen. Useimmat projektiryhmäläiset suorittivat esimerkiksi projektityökurssin, eli 10 opintopistettä, eli

270 tuntia tässä projektissa ja lähtivät. Tästä ajasta suuri osa meni opiskeluun. Juuri kun opiskelija alkoi oppia käytetyt tekniikat ja tuntemaan järjestelmän ja kun hänellä oli edellytykset tuottavaan työhön, hänellä tuli tunnit täyteen.

Toteutusta vaikeutti myös järjestelmän laajuus, koska mitään osa-aluetta ei voi lähteä toteuttamaan muista irrallisena, vaan koko järjestelmä pitää ymmärtää jollain tasolla kokonaisuutena ennen kuin voi aloittaa toteuttamisen. Lisäksi kaikkia yksityiskohtia ei voi mitenkään muistaa kirjoittaa ylös tulevia projekti-ryhmäläisiä varten. Tässä projektissa asiakas joutui selvästi toistamaan itseään, koska aiemmat projektiryhmäläiset eivät olleet dokumentoineet kaikkea. Toisaalta dokumentaatiota oli olemassa valtavasti, eikä kaikkea sitä voinut mitenkään käydä läpi. Joistain asiakkaan vaatimuksista sanottiin, että se on käyty läpi *aiemmassa palaverissa ... joskus kesällä*.

Nämä tekijät aiheuttivat myös rajoituksia työvoiman rekrytoinnille. Kun uusia työntekijöitä saatiin projektiin, heidän tehtävänsä piti harkita tarkkaan vastamaan heidän osaamistaan ja ajallista panostustaan. Lisäksi tuli pyrkiä minimoimaan heidän opastamiseensa kuluva aika. Brooksin laki sanoo, että uusien työntekijöiden lisääminen myöhästyneeseen ohjelmistoprojektiin myöhästyttää sitä entisestään (Brooks 1995). Tämän takia kevään ja kesän aikana projektiin liittyneet työntekijät saivat tehtävikseen lähinnä testausta ja määrittelyn siistimistä.

Tällä hetkellä näyttää siltä, että projekti kokonaisuudessaan voisi olla valmis tämän vuoden (2011) loppuun mennessä. Olisi mielenkiintoista päästä näkemään järjestelmän käyttöönotto sekä järjestelmän vaikutus yrityksen toimintaan. Pidemmällä aikavälillä voitaisiin tarkastella toiminnan tehostumisen taloudellisia vaikutuksia. Todennäköisesti vaikutukset tulevat käyttöönoton ongelmien jälkeen olemaan positiiviset.

Lähteet

Adam Shaw, FullCalendar, <http://arshaw.com/fullcalendar/> (luettu 8.11.2011)

All About ERP and Business Softwares
<http://www.abouterp.com/> (luettu 30.8.2011)

Booch, G., Rumbaugh, J., Jacobson, I. 1999. The Unified Modeling Language User Guide. Massachusetts: Addison-Wesley

Brooks, F. 1995. The Mythical Man-Month: Essays on Software Engineering. Massachusetts: Addison-Wesley

Erikkson, H. & Penker M. 2000. UML. Jyväskylä: IT Press.

Fowler, S. & Scott, K. 2002. UML. Jyväskylä: Docendo.

From, M. 2008. ERP luultua tärkeämpi pk-yritykselle. TIEKEN tiedotteet 2008. http://www.tieke.fi/tieke/tieken_tiedotteet_2008/erp_luultua_tarkeampi_pk-yrityks/ (Luettu 11.11.2010).

Hernandez, M. 2000. Tietokannat – suunnittelu käytännössä. Jyväskylä: Gummerus Kirjapaino Oy.

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

Immonen, J. Johdatus Ohjelmistotuotantoon.
http://cs.joensuu.fi/~jimmonen/jot_moniste/jot_moniste_121.html (luettu 12.9.2011)

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

jqGrid Demos. <http://www.trirand.com/blog/jqgrid/jqgrid.html> (luettu 13.10.2011)

Kainulainen, A. 2008. Agile-menetelmät. Jyväskylän Ammattikorkeakoulu. Teknologiaosaamisen johtamisen koulutusohjelma. Opinnäytetyö.

Ketterät Käytännöt.fi, Scrum – mahdollisuuksien taide
<http://www.ketteratkaytannot.fi/fi-FI/Menetelmat/Scrum/>

Koskimies, K. 2000. Oliokirja. Jyväskylä: Satku.

Kruchten, P. 1998. The Rational Unified Process: An Introduction. Reading: Addison-Wesley

Microsoft, 2008, Tietokannan normalisoinnin perusteiden kuvaus,
<http://support.microsoft.com/kb/283878> (luettu 30.8.2011)

Paajanen, T. 2011. Innonet-toiminnanohjausjärjestelmä. Opinnäytetyö. Saimaan ammattikorkeakoulu.

Rissanen, N. 2010. Innonet-toiminnanohjausjärjestelmä. Opinnäytetyö. Saimaan ammattikorkeakoulu.

Sakki, J. 2001. Tilaus-toimitusketjun hallinta: logistinen b to b -prosessi. Espoo: Jouni Sakki Oy.

Spolsky, J. Subversion Re-education
<http://hginit.com/00.html> (luettu 30.8.2011)

Tampereen teknillinen yliopisto, 2004. Tietokantajärjestelmien suunnittelu
http://www.cs.tut.fi/~tikas/arkisto/harjoitustyo2002/osa_1/ratkaisu/kasitekaavio/ (luettu 30.9.2011)

TIEKE kysely 2008
<http://www.tieke.fi/liiketoimintapalvelut/paattajaindeksi/2008/> (Luettu 11.11.2010).

Rational Unified Process: Best Practices for Software Development Teams
http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf (luettu 20.8.2011)

Weber, T. 2007. Do small firms need big software?
<http://news.bbc.co.uk/2/hi/6509943.stm> (luettu 30.8.2011)