

Saimaan ammattikorkeakoulu  
Tekniikka, Lappeenranta  
Tietotekniikan koulutusohjelma  
Tietojärjestelmien kehitys

Laura Maarnela

## **Kymecon Oy urakkatarjouslaskenta- ja kustannusseurantajärjestelmän toiminnallinen määrittely**

## **Tiivistelmä**

Laura Maarnela

Kymecon Oy:n urakkatarjouslaskenta- ja kustannusseurantajärjestelmän toiminnallinen määrittely, 66 sivua, 1 liitettä

Saimaan ammattikorkeakoulu

Tekniikka Lappeenranta

Tietotekniikan koulutusohjelma

Tietojärjestelmien kehitys

Opinnäytetyö 2013

Ohjaajat: Lehtori Martti Ylä-Jussila, Saimaan ammattikorkeakoulu, toimitusjohtaja Mikko Värjä, Kymecon Oy

Opinnäytetyön aiheena oli toiminnallinen määrittely Kymecon Oy nimiselle yritykselle urakkatarjouslaskenta- ja kustannusseurantajärjestelmästä. Järjestelmää toteutettiin ketterää menetelmää käyttäen, ja toiminnallinen määrittely tehtiin, koska järjestelmän toteuttajat vaihtuivat kesken projektin. Järjestelmän käyttöliittymän kuvat toteutettiin NetBeans-sovelluksella, jolla järjestelmää oli aikaisemmin toteutettu.

Työ aloitettiin tekemällä järjestelmän valmiista osasta toiminnallinen määrittely, jonka jälkeen asiakkaan kanssa kerättiin puuttuvasta osasta vaatimukset. Tämän jälkeen tästä puuttuvasta osasta pääsin tekemään käyttöliittymän kuvauksen ja käyttötapaukset kuvauksineen, jotka lisäsin määrittelydokumenttiin.

Asiasanat: ohjelmistotuotanto, toiminnallinen määrittely, ketterä menetelmä, prosessimalli

## **Abstract**

Laura Maarnela

Functional specification document of a tender calculation and cost follow-up system for Kymecon Oy, 66 of Pages, 1 of Appendices

Saimaa University of Applied Sciences

Technology Lappeenranta

Degree Programme in Information Technology

Advancement Information System

Bachelor's Thesis 2013

Instructor(s): Lecturer Martti Ylä-Jussila, Saimaa University of Applied Sciences, CEO Mikko Värjä, Kymecon Oy

The purpose of this thesis was to make a functional specification document of a tender calculation and cost follow-up system for Kymecon Oy. The system was created by using the agile method, and the functional specification document was made because the personnel changed in the middle of the project. Pictures of the user interface were created with the NetBeans application, the same one that was used before creating the system.

The thesis was started by making the functional specification of the ready part of the system. After that with a client we collected the requirements of the absent part of the system, and then I made the user interface description, a user case and user case descriptions and added them to the functional specification.

Keywords: software engineering, functional specification, agile methods, process model

## Sisältö

1	Johdanto.....	6
1.1	Kymecon Oy .....	7
1.2	Rakennushanke.....	7
1.3	Rakennushankkeen rakentamisvaihe .....	10
2	Ohjelmistotuotannon osa-alueet .....	13
2.1	Ohjelmistotuotannon kehitysmenetelmät .....	13
2.2	Ohjelmiston elinkaari.....	14
2.3	Vaihejakomallit.....	17
2.3.1	Vesiputous .....	18
2.3.2	Prototyypimalli.....	19
2.3.3	Spiraalimalli.....	21
2.3.4	RUP-malli .....	22
2.4	Ketterä ohjelmistokehitys.....	24
2.4.1	Extreme programming.....	26
2.4.2	Scrum.....	29
2.5	Laatu, laatujärjestelmä ja laadun varmistus .....	32
2.6	Dokumentointi ohjelmistotuotannossa .....	33
3	Toiminnallinen määrittely .....	34
3.1	Toiminnallisen määrittelyn dokumenttimallit ja standardit .....	35
3.2	Tietojen kuvaaminen.....	37
3.3	Käyttötapaukset .....	39
3.3.1	Käyttötapauksen kuvaus .....	40
3.3.2	Käyttäjätarina .....	41
4	Opinnäytetyöprojektin organisointi ja vaiheet.....	42
4.1	Projektin organisointi ja suunnittelu .....	42
4.2	Projektisuunnitelma .....	43
4.3	Määrittelyn suunnittelu .....	43
4.4	Toteutus.....	44
4.5	Määrittelyn katselmointi ja hyväksyminen .....	45
4.6	Riskit ja ongelmat .....	45
5	Lopputuloksen esittely .....	46
6	Pohdintaa.....	62
	Kuvat.....	64
	Lähteet.....	65

### Liitteet

Liite 1 Tietokantakaavio

## Termit ja lyhenteet

### Termit

Ohjelmistotuotanto	Ohjelmistotuotannolla tarkoitetaan ohjelmistotyötä, jonka tuloksena syntyvä järjestelmä toteuttaa käyttäjiensä kohdulliset toiveet ja odotukset.
Toiminnallinen määrittely	Ohjelmiston elinkaaren vaihe, missä kuvataan kaikki ohjelmiston toteuttamat toiminnot, ei-toiminnalliset vaatimukset ja liitännät järjestelmän ulkopuolelle.
Agile	Agile tarkoittaa ketterää ohjelmistokehitystä, jossa ohjelmisto toteutetaan pienissä osissa.
UML	Unified Modeling Language standardisoitu graafinen mallinnuskieli.
RUP	Rational Unified Process on iteratiivinen ja inkrementaalisen ohjelmistokehityksen prosessikehys
XP	Extreme programming on ketterän ohjelmistokehityksen menetelmä, jossa keskitytään ohjelmistokehityksen ohjelmointipuoleen
Scrum	Ketterän ohjelmistokehityksen menetelmä, joka on iteratiivinen ja inkrementaalinen suunnittelu- ja tuotantoprosessin menetelmä.
SQL	Structured Query Language on standardisoitu kieli tietokantakyselyä varten.
ISO	International Organization for Standardization on kansainvälinen standardisoimisjärjestö.

# 1 Johdanto

Opinnäytetyön aiheena oli Kymecon Oy:n urakkatarjouslaskenta- ja kustannus-seurantajärjestelmän toiminnallinen määrittely. Kymecon Oy on itsenäinen maa-, vesi- ja teollisuusrakennusalan yritys, joka on toteuttanut useita eri kohteita teollisuuden, kuntien ja valtion rakennuttajaorganisaatioille.

Ennen opinnäytetyön aloitusta kohteena olevaa järjestelmää oli tehty ketterää ohjelmistokehitysmenetelmää käyttäen – ei suunnitellusti vaan edeten osajärjestelmää kerrallaan syyskuusta 2011 lähtien. Kehitystyön aikana ei laadittu määrittely- eikä suunnitteludokumentteja, jotka päätettiin laatia vasta toteutuksen jälkeen.

Opettajani ehdotti minulle määrittelydokumentin laatimista opinnäytetyöksi, koska olen ollut sovelluksen teossa mukana alusta saakka. Opinnäytetyön tavoitteena on saada asiakkaalle kehitteillä olevasta sovelluksesta toiminnallinen määrittely, jossa kuvataan järjestelmän toiminnot, tiedot ja ulkoiset liitännät. Määrittelydokumentti on välttämätön, koska sovelluksen kehitystyö on kesken ja toteuttajat vaihtuvat. Määrittelydokumentti on myös sovelluksen käyttöohje asiakkaalle.

Toiminnallisessa määrittelyssä otetaan kantaa ohjelmointikieleen, koska järjestelmää aloitettiin koodaamaan Javalla ketterää menetelmää käyttäen.

Aluksi opinnäytetyössä kerrotaan ohjelmistotuotannosta, perinteisistä prosessimalleista, ja ketteristä ohjelmistokehitysmenetelmistä ja malleista. Luvussa 3 kerrotaan toiminnallisesta määrittelystä ja siinä käytetyistä dokumentointimenetelmistä.

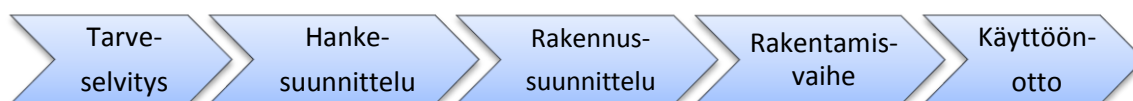
Tämän jälkeen kerrotaan toiminnallisen määrittelydokumentin luomisesta kyseiselle järjestelmälle.

## 1.1 Kymecon Oy

Kymecon Oy on vuonna 1996 perustettu itsenäinen maa-, vesi- ja teollisuusrakennusalan yritys. Vuodesta 1997 alkaen Kymecon Oy on toteuttanut useita eri kohteita teollisuuden, kuntien ja valtion rakennuttajaorganisaatioille. Yrityksen työntekijämäärä on ollut keskimäärin 10 henkilöä, jonka lisäksi sillä on useita yhteistyökumppaneita. Kymecon Oy:n vastuuhenkilöt ovat yrityksen perustaja ja toimitusjohtaja Mikko Värjä sekä työpäällikkö Mikko Kankkunen.

## 1.2 Rakennushanke

Talonrakennushanke on monivaiheinen prosessi, johon liittyy monta osapuolta. Suuri rakennushanke vaatii paljon erikoisosaamista ja siksi alalla toimii eri vaiheisiin ja tehtäviin erikoistuneita yrityksiä, konsultteja ja viranomaisia. Kuvassa 1 on esitetty rakennushankkeen vaiheet ja kuvassa 2 eri osapuolten tehtävien painottuminen eri vaiheisiin (RT 10-10387) .



Kuva 1. Rakennushankkeen vaiheet (RT 10-10387)

Rakennushanke etenee vaiheesta toiseen harkittujen päätösten kautta: hanke-suunnittelua edeltää *hankepääätös*, rakennussuunnittelua edeltää *investointipää-tös*, rakentamisvaihetta edeltää *rakentamispääätös*, käyttöönottoa edeltää *vas-taanottopääätös* ja hanke päättyy käyttöönottovaiheen jälkeen *takuutarkastuk-seen*.

	Hankkeen osapuolet	K	R	S	U	V
Hankkeen vaiheet		Käyttäjä	Rakennuttaja	Suunnittelija	Rakentaja	Viranomainen
<b>TS</b>	Tarveselvitys					
<b>HS</b>	Hankesuunnittelu					
<b>RS</b>	Rakennussuunnittelu					
<b>RA</b>	Rakentaminen					
<b>KO</b>	Käyttöönotto					

	Passiivinen
	Osa tehtävistä
	Suurin osa tehtävistä

Kuva 2. Rakennushankkeen vaiheet ja osapuolet (RT 10-10387)

### Rakennushankkeen vaiheet

TS Tarveselvitysvaiheessa selvitetään hankkeen tarpeellisuus, edellytykset ja toteuttamismahdollisuudet. Tuloksista kootaan tarveselvitys, jonka pohjalta tehdään hankesuunnittelupäätös.

HS Hankesuunnitteluvaiheessa arvioidaan hankkeen toteuttamismahdollisuudet ja toteutusvaihtoehdot. Tulokset kootaan hankesuunnitelmaksi, jossa asetetut laajuus- ja laatutavoitteet määrittävät hankkeen kustannustason ja aikataulun. Hankesuunnitelman pohjalta tehdään investointipäätös.

RS Rakennussuunnittelu jakaantuu kahteen vaiheeseen, luonnos- ja toteutus-suunnitteluun. Luonnossuunnitteluvaiheen tuloksena valitaan ja määritellään kohteen suunnitteluratkaisu, tekniset järjestelmät ja toteutustapa sekä tehdään päätös luonnossuunnitelmien hyväksymisestä. Toteutussuunnitteluvaiheessa määritellään hankkeen urakointitapa, laaditaan hankinta-asiakirjat ja piirustukset, valmistellaan hankinnat ja tehdään rakentamispäätös sekä solmitaan urakasopimukset. Tämän jälkeen alkaa rakentamisvaihe.



RA Rakentamisvaiheessa hankkeen suunniteltu lopputuote rakennetaan. Vaihe päättyy rakennuksen vastaanottopäätökseen.

KO Käyttöönottovaiheessa käynnistetään rakennuksen toiminta ja todetaan seurantatoimenpitein rakennuksen käyttövalmiudet. Rakennushanke päättyy takuutarkastukseen ja takuiden vapauttamiseen.

### **Rakennushankkeen osapuolet**

K Käyttäjä edustaa rakennushankkeen osapuolena sen toiminnan asiantunte-  
musta, jonka tilantarvetta varten hanke perustetaan. Käyttäjän esittämät toimin-  
nalliset ja laadulliset vaatimukset ja tavoitteet ovat lähtökohta hankkeelle. Hank-  
keen muiden osapuolten ammattitaito varmistaa käyttäjän tarpeiden toteutumisi-  
sen.

R Rakennuttaja on hankkeen toimeenpaneva osapuoli, joka käynnistää hank-  
keen ja hoitaa hankkeen läpiviennin. Rakennuttaja vastaa siitä, että käyttäjä saa  
käyttöön tarpeittensa mukaiset tilat.

S Suunnittelijaosapuoli vastaa rakennuksen tuotesuunnittelusta. Osapuoli muo-  
dostuu suunnittelijaryhmästä, jossa on edustettuna eri alojen suunnitteluasian-  
tuntemus (arkkitehtisuunnittelu, rakennuksen kokonaissuunnittelu, sisustus-  
suunnittelu, vihersuunnittelu, akustinen suunnittelu, rakennustekninen suunnit-  
telu, geotekninen suunnittelu, rakenne- ja elementtisuunnittelu, teknisten järjes-  
telmien suunnittelu, LVI-tekniikka suunnittelu, automaatiotekninen ja instrumen-  
tointisuunnittelu, sähkösuunnittelu, sähkötekniikka suunnittelu, teletekninen  
suunnittelu, muu teknisten järjestelmien suunnittelu, kiinteistönpidon suunnitte-  
lu, kustannussuunnittelu ja määrälaskenta). Suunnitteluryhmän työn koor-  
dinoinnista vastaa pääsuunnittelija.

U Rakentaja on rakennushankkeen osapuoli, joka rakennuttajan toimeksiannos-  
ta vastaa lopputuotteen konkreettisesta tuottamisesta, rakennuksen rakentami-  
sista (rakennusurakat, pääurakoitsija, muut rakennustekniset urakat, LVIS-  
urakat, erikoisurakat).

V Viranomaiset valvovat suunnittelua ja rakentamista lakien, asetusten, eri asteisten kaavojen, yleisten ja paikallisten määräysten, ohjeiden ja normien pohjalta (kaavoitusviranomainen, rakennusvalvontaviranomainen, rahoittava viranomainen).

### **1.3 Rakennushankkeen rakentamisvaihe**

Ennen rakentamispäätöstä rakennuttaja pyytää potentiaalisilta urakoitsijoilta urakkatarjoukset. Tarjouspyynnön tekemistä varten rakennuttaja on teettänyt tarvittavat suunnitelmat, urakkaohjelman, urakkapiirustukset, työselitykset sekä työpiirustukset ja muut tarjouspyyntöasiakirjat. (RT 10-10387.)

Kustannuslaskenta tehdään tarjouspyynnön teknisten ja kaupallisten asiakirjojen, sekä alustavien tuotantosuunnitelmien pohjalta. Kustannuslaskennan vaiheet ovat

- tarjouspyyntöasiakirjoihin perehtyminen
- hankkeen osittelu laskentaa varten
- rakennusosien määrämittaus
- hinnoittelu
- laskentamuistion teko
- kustannusarvion tarkistus.

Tarjous on tilaajalle sitoumus tarjouspyynnön ehtojen mukaisen urakan tekemisestä tai toimittamisesta tarjotulla hinnalla. Tarjous muodostetaan lisäämällä kustannusarvioon tarjouksen lisäerät. Tarjouksen lisäeriä ovat riskivaraus, kustannustason muutosvaraus ja työmaakate.

Mikäli tarjous hyväksytään, allekirjoitetaan urakkasopimus ja rakennuttaja tilaa pyydetyn kohteen ja urakoitsija viimeistelee kustannusarvio tavoitteelliseksi ohjeeksi eli tavoitearvioksi työn taloudellista toteutusta varten.

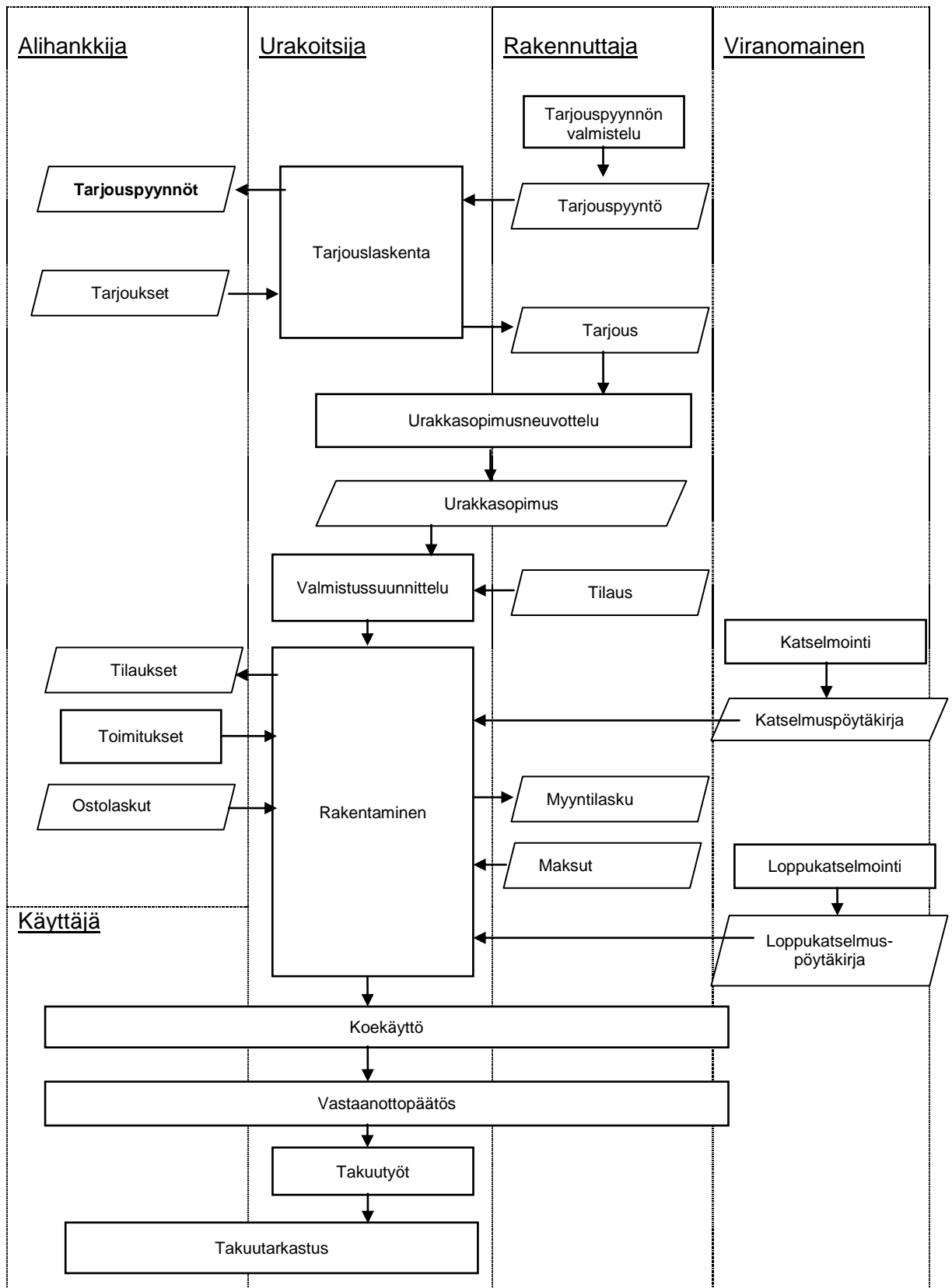
Rakentamisvaiheessa rakennetaan suunniteltu kohde. Rakentaminen toteutetaan valitun urakkamuodon puitteissa, solmitun urakkasopimuksen mukaan. Sopimusosapuolia koskevat velvoitteet ja vastuu käyvät ilmi urakka-asiakirjoista.

Toteutusvaiheen aikana hankkeen organisaatio on laajimmillaan. Siihen kuuluvat rakennuttajan edustaja, suunnittelijat, työmaan valvoja ja urakoitsijat. Lisäksi vaiheeseen kuuluu viranomaisen valvontatehtäviä.

Rakentamisvaiheen alussa urakoitsija kokoaa työorganisaation, nimeää vastuhenkilöt, laatii tarvittavat aikataulut, kuten työmaa-aikataulun ja teollisesti valmistettavien rakennusosien suunnittelu-aikataulun, ja rakentamisen yleissuunnitelman, josta käy selville keskeiset työmaa järjestelyt, kone- ja kalustosuunnitelma sekä työvoimatarve. Urakoitsija jatkokäsittelee saamaansa suunnittelutietoutta suunnitellessaan työmaata, työvaiheita ja suorittaessaan hankintoja. Rakennusurakan sopimusosapuolena urakoitsija vastaa asiakirjojen edellyttämästä työsuorituksesta urakkaehtojen mukaisesti, erikseen sovituista muutosten lisätoista sekä takuuajan velvoitteista. Urakoitsija huolehtii ajallaan rakennusluvan edellyttämät viranomaisilmoitukset, katselmukset ja tarkastukset. (RT 10-10387.)

Rakentamisvaihe päättyy vastaanottotarkastukseen, jossa rakennuttaja vastaanottaa valmiin rakennuksen urakoitsijalta. Urakoitsija korjaa loppukatselmuksen ja vastaanottotarkastuksen yhteydessä ilmenneet virheelliset työsuoritukset ja puutteet. Korjaustyöt valvotaan ja hyväksytään jälkitarkastuksessa. Takuuajan päätyttyä pidetään takuutarkastus, valvotaan takuun piiriin kuuluvat korjaustyöt ja hyväksytään ne.

Kuvassa 3 on pelkistetty toimintakaavio rakennusvaiheen työkulusta. Käytännössä rakennus vaiheeseen liittyy runsaasti dokumentteja, jotka on listattu ja kuvattu yksityiskohtaisesti viranomaisten rakennusmääräyksissä ja alan toimintaohjeissa. (RT 10-10387.)



Kuva 3. Rakennusvaiheen työnkulku

## 2 Ohjelmistotuotannon osa-alueet

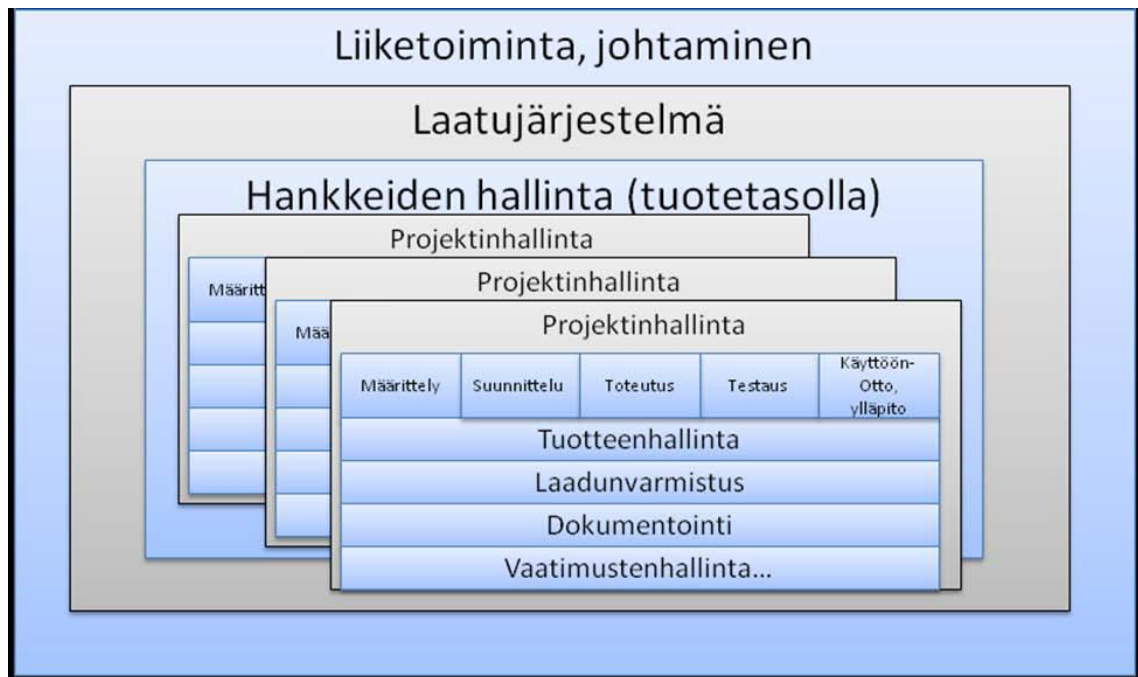
Ohjelmistotuotannolla (*Software engineering*) tarkoitetaan ohjelmistotyötä, jonka tuloksena syntyvä järjestelmä toteuttaa käyttäjiensä kohtuulliset toiveet ja odotukset. Lisäksi järjestelmä valmistuu laadittujen aikataulujen ja kustannusarvioiden mukaan.(Immonen 2003.)

### 2.1 Ohjelmistotuotannon kehitysmenetelmät

Kaikkea ohjelmistotyön tuloksena syntyvää materiaalia kutsutaan sanalla software, ja se yleensä suomennetaan termiksi ohjelmisto. Engineering tarkoittaa tekniikkaa, ja sillä tarkoitetaan tieteellisen tiedon soveltamista käytännön ongelmiin. Software engineering voidaan vapaasti suomentaa ohjelmistokehitykseksi tai ohjelmistotuotannoksi. Ohjelmistotuotantoon liittyvät dokumentaatiot kertovat, kuinka sovellus asennetaan, kuinka sitä käytetään ja kuinka sovellusta ylläpidetään. Software engineering terminä käsittääkin siis kaikki ohjelmiston tuotantoprosessiin liittyvät osa-alueet: laatujärjestelmä, projektinhallinta, dokumentointi, tuotteenhallinta, laadunvarmistus, testaus, määrittely, suunnittelu, toteutus, käyttöönotto ja ylläpito. Ohjelmistotuotanto käsittää siis kaikki ohjelmistotuotantoprosessiin liittyvät osa-alueet. Ohjelmistotuotanto voidaan jakaa eri osa-alueisiin kuvan 4 osoittamalla tavalla. (Haikala & Märijärvi 2004, 16.)

Laatujärjestelmä ohjaa yrityksen tuotantoa, eli laatujärjestelmä määrittelee ohjelmistoja tuottavan yrityksen toimintaprosessit eli toimintatavat. Ohjelmistoprojektit voivat yleensä olla osa laajempien tuotekehityshankkeiden osaprojekteja. Seuraavat vaiheet voidaan yleensä erottaa kehitysprosessissa: määrittely, suunnittelu, toteutus ja testaus. Näistä seuraavat vaiheet ovat ohjelmiston käyttöönotto ja ylläpito. Koko projektin ajan projektiin ja ohjelman elinkaareen liittyy tukitoimintoja. Tärkeimmät tukitoiminnot ovat laadunvarmistus, tuotehallinta ja dokumentointi. Isossa projektissa tukitoimintoja voi olla enemmänkin.

Vaatimustenhallintaa ja riskienhallintaa saatetaan käsitellä erillisinä tukitoimintoina. Pienemmässä projektissa ne sisältyvät esimerkiksi määrittelyyn, projektinhallintaan ja tuotteenhallintaan ja jokaiseen osa-alueeseen kuuluu omat erityisongelmat, menetelmät, käsitteistöt ja työkalut. (Haikala & Märijärvi 2004, 35-36.)



Kuva 4. Ohjelmistotuotannon osa-alueet (Haikala & Märijärvi 2004, 35)

## 2.2 Ohjelmiston elinkaari

Ohjelmiston elinkaari (*life cycle*) tarkoittaa ajanjaksoa, jossa ohjelmiston kehittämisen aloittamisesta kuluu aikaa ohjelmiston poistamiseen käytöstä. Vaihejakomalli on tapa, jolla ohjelmiston kehitystyö tai elinkaari jaetaan erilaisiin vaiheisiin. Jokaiseen vaiheisiin liittyy laadunvarmistustoimenpiteitä, kuten esimerkiksi katselmukset, tarkastukset ja testaus. Jokaisella toimenpiteellä pyritään etsimään ja poistamaan virheet järjestelmästä mahdollisimman aikaisessa vaiheessa, jolloin säästyy aikaa ja turvaa vaivaa edettäessä seuraavaan vaiheeseen. Jokaiseen vaiheen jälkeen pidetään katselmus, jossa katsotaan projektin tilanne, onko vaiheeseen tarvittavat tavoitteet saavutettu ja onko kaikki sovitut dokumentit laadittu. (Haikala & Märijärvi 2004, 37.)

Elinkaaren vaihejako määrittelee prosessin tärkeimmät osavaiheet ja määrittelee osavaiheiden suoritusjärjestyksen. Jokainen vaihe tuottaa jonkin määritellyn tuloksen, ja vaiheen tulos toimiikin syötteenä seuraavalle vaiheelle. Vaiheen tulokset ovat määritellyillä kriteereillä hylättävissä tai hyväksyttävissä. (Laine & Paakki.)

Ensimmäinen vaihe on esitutkimus, jonka tehtävä on asettaa yleiset järjestelmätason vaatimukset, ”*esimerkiksi varastonvalvontasovelluksen vaatimukseksi voidaan asettaa varaston kiertonopeuden kasvattaminen 10 prosentilla*”. Edellä mainittu teksti on asiakasvaatimus (*customer requirement, business requirement*). Asiakasvaatimus määrittelee asiakkaan tarpeet, mutta ei kerro vastausta siihen, millainen järjestelmä täyttää asiakkaan vaatimukset. Esitutkimuksessa vastataan kysymykseen, miksi järjestelmä tai ohjelmisto tulisi tehdä ja myös miksi järjestelmää tai ohjelmistoa ei tulisi tehdä. Elinkaaren tärkein vaihe on esitutkimus, sillä väärällä tiedolla ja väärin ymmärretyllä tiedolla ei voi saada aikaiseksi hyvää järjestelmää. Oikean tiedon kerääminen, sen ymmärtäminen ja asiakkaan todellisten tarpeiden ymmärtäminen ovat ehdottomia vaatimuksia hyvän ja toimivan järjestelmän toteuttamiseksi. (Haikala & Märjärvi 2004, 37.)

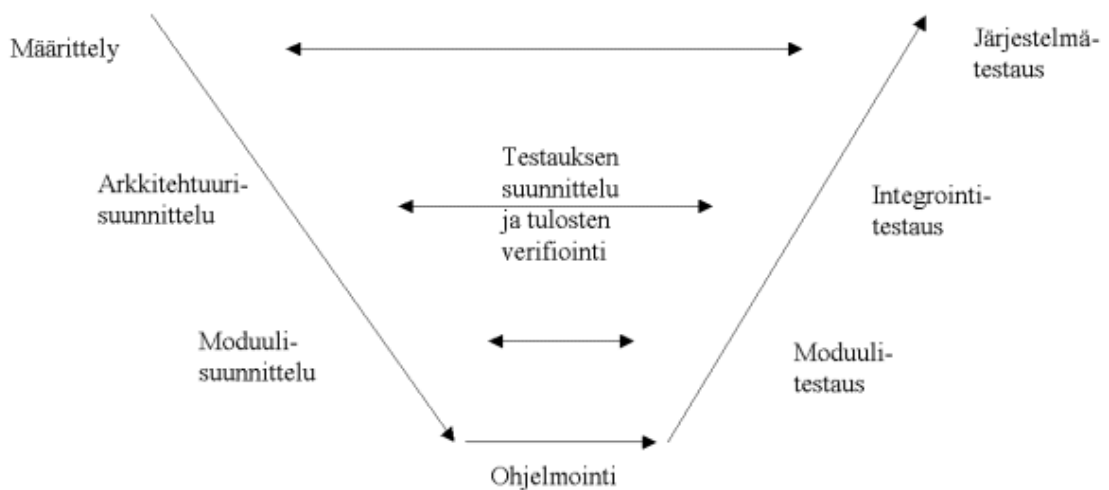
Määrittelyvaiheessa asiakasvaatimukset analysoidaan ohjelmistovaatimuksiksi, ja nämä vaatimukset määrittelevät toteutettavan järjestelmän. Määrittelyssä määritellään järjestelmän sisältämät toiminnot eli mitä järjestelmällä voi tehdä ja miten nimenomaan käyttäjä voi toiminnot tehdä. Määrittelyssä myös kuvataan järjestelmän käyttöliittymät ja järjestelmän ulkopuoliset liittymät. Määrittelyvaiheen tuotoksena syntyy toiminnallinen määrittelydokumentti, josta kerrotaan tarkemmin luvussa 3. (Haikala & Märjärvi 2004, 38.)

Suunnitteluvaiheessa (*design*) suunnitellaan määrittelyvaiheesta saatujen toimintojen tekninen toteutus. Suunnitteluvaiheessa toiminnot suunnitellaan eli päätetään, miten järjestelmä toiminnon tekee. Suunnitteluvaihe jaetaan vähintään kahteen tasoon; arkkitehtuurisuunnittelu (*architectural design*) ja moduulisuunnittelu (*module design*). Arkkitehtuurisuunnittelussa ohjelmisto jaetaan moduuleihin ja suunnitellaan järjestelmän yleinen rakenne. Arkkitehtuurisuunnittelun jälkeen tulee moduulisuunnittelu, jossa suunnitellaan jokaisen moduulin sisäinen rakenne ja jokaisen rakenneosan yksityiskohtainen kuvaus. Kuvaukses-

sa kuvataan rakenneosan tehtävä, rajapinta ja rakenneosan toiminta. Arkkitehtuurisuunnittelusta syntyvää dokumenttia kutsutaan tekniseksi määrittelyksi. (Kankaanpää 2013.)

Toteutusvaiheessa (*programming*) valitaan itse toteutusväline ja toteutustapa. Toteutuksella tarkoitetaan itse ohjelman kirjoitusvaihetta eli koodaamista. Koodatessa pitäisi muistaa kommentoida koodia dokumentoinnin kannalta.

Testausvaiheessa (*testing*) tarkoituksena on löytää ohjelmistosta erilaisia virheitä. Ohjelmistoja testataan yleensä niin sanotun V-mallin mukaisesti. V-malli testauksessa testaus jaetaan kolmeen osaan: moduulitestaus, integrointitestaus ja järjestelmätestaus. Moduulitestauksessa yksittäisistä moduuleista etsitään vikoja, integrointitestauksessa vikoja etsitään moduulien yhteistoiminnasta ja järjestelmätestauksessa vikoja etsitään koko järjestelmän toiminnoista ja suorituskyvystä. Alustavasti järjestelmätestauksen suunnittelu tehdään jo määrittelyvaiheessa ja testauksessa valmista järjestelmää verrataan aikaisempaan määrittelydokumenttiin. Testaus voi olla hyväksymistestaus, käytettävyydestestaus, alfa- tai betatestaus. (Haikala & Märjärvi 2004, 40.)



Kuva 5. Testauksen V-malli (Haikala & Märjärvi 2004, 289)



Käyttöönnotossa (*installation*) uusi järjestelmä otetaan käyttöön. Jos on olemassa aikaisemmasta järjestelmästä tiedostoja tai tietokantoja, niin käyttöönottovaiheessa vanhat tiedot, tiedostot ja tietokannat siirretään uuteen järjestelmään. Käyttöönottovaiheessa pidetään järjestelmän tilaajalle myös koulutus, jotta järjestelmän käyttäjät osaavat käyttää järjestelmää oikein, käyttöohjeet annetaan myös tässä vaiheessa.

Ylläpidossa (*maintenance*) ratkaistaan asiakkaan ongelmia, korjataan virheitä, muutetaan ohjelma järjestelmän ympäristön muuttuneiden vaatimusten mukaiseksi ja parannetaan ohjelmaa lisäämällä tai muuttamalla ohjelman toiminnallisuutta. Ylläpidon voi jakaa kolmeen osaan: korjaava (*corrective*), sopeutuva (*adaptive*) ja täydentävä (*perfective*). Ylläpidon korjaavassa vaiheessa korjataan ohjelman virheitä, sopeutuvassa ylläpidossa muutetaan itse ohjelmaa muuttuneen ympäristön vaatimusten mukaiseksi, ja täydentävässä ylläpidossa parannetaan ohjelmaa lisäämällä ja muokkaamalla ohjelman toiminnallisuutta. (Haikala & Märijärvi 2004, 41.)

### **2.3 Vaihejakomallit**

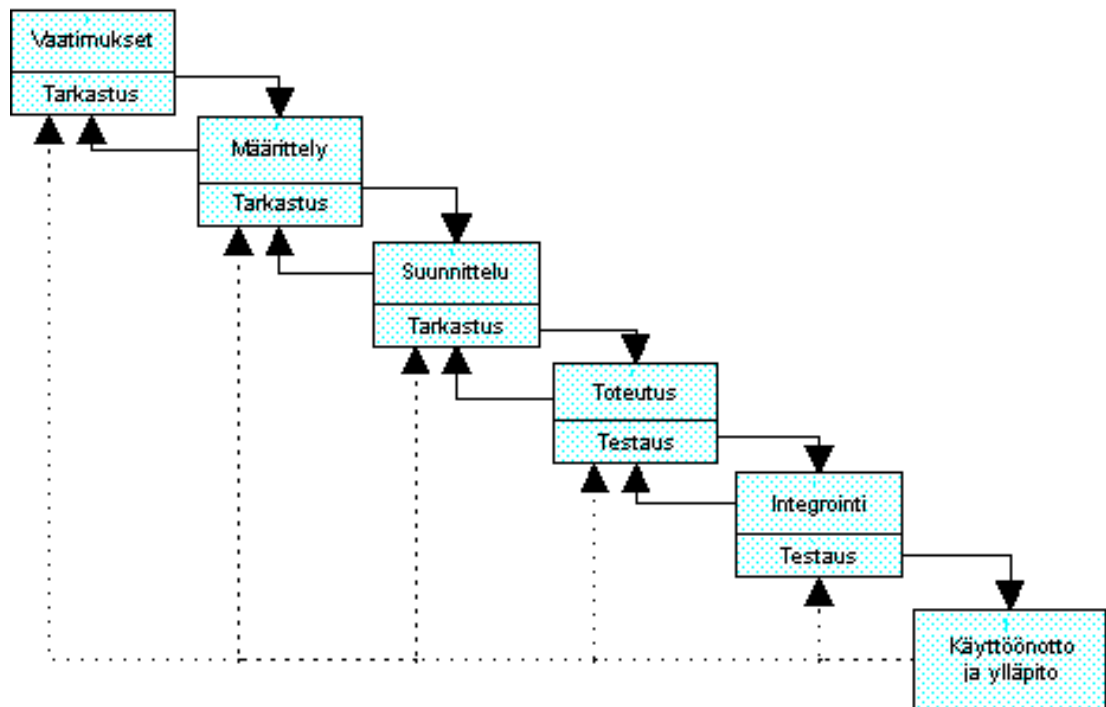
Tavallisin ja tunnetuin elinkaaren vaihejakomalli on niin sanottu vesiputousmalli. Siinä suunnittelu- ja toteutusprosessi etenee vaihe vaiheelta alaspäin, aivan kuten vesiputous. (Haikala & Märijärvi 2004, 36.) Muita elinkaarimalleja ovat vesiputouksen lisäksi spiraali, prototyyppi ja RUP, joista kerrotaan seuraavissa alaluvuissa.

### 2.3.1 Vesiputous

Vesiputousmalli (*waterfall model*) on yksi käytetyimmistä elinkaarimalleista. Vaikka vesiputousmallista onkin erilaisia muunnelmia, niin silti jokaisesta löytyvät ainakin määrittely-, suunnittelu- ja toteutusvaiheet. Laaduntarkastustoimenpiteet kuuluvat jokaiseen vaiheeseen. Laaduntarkastustoimenpiteitä ovat tarkastukset, katselmukset ja testaus. Tarkastuksella ja testaamisella pyritään löytämään virheet hyvissä ajoin, koska virheen korjauskustannukset kasvavat projektin edetessä. (Räsänen 2008.)

Kuvassa 6 esitetään, kuinka vesiputousmalli etenee ylhäältä esitutkimuksesta alaspäin aina ylläpitoon saakka. Jokaisen vaiheen lopussa pidetään katselmointitilaisuus, jossa tarkastetaan, onko kaikki sovitut tehtävät tehty, tavoitteet saavutettu ja dokumentit tuotettu, sekä päätetään, onko projekti valmis siirtymään seuraavaan vaiheeseen.

Vesiputousmallin etu on, että sillä saa aikaiseksi hyvän ja luotettavan järjestelmän perusrakenteen. Vesiputousmalli on myös erittäin selkeä. Haittoina taas ovat, että malli tarvitsee tarkan määrittelyn, ja arkkitehtuurin on oltava hyvin suunniteltu. Vasta testausvaiheessa huomataan virheet ja väärinkäsitykset, jotka ovat syntyneet vääristä olettamuksista määrittelyn aikana. Testausvaiheessa projekti on jo viimeistelyvaiheessa, joten virheiden korjaus on vaikeaa. Projektin edistymistä onkin vaikea nähdä ja tulokset projektista saadaan käyttöön vasta lopussa. (Räsänen 2008.)



Kuva 6. Vesiputousmalli (Haikala & Märijärvi 2004, 36)

### 2.3.2 Prototyypimalli

Prototyypimallilla (*prototypemodell*) eli protoilumallilla tarkoitetaan ihan mitä tahansa työskentelymallia, jossa ensimmäiseksi tehdään jonkinlainen kokeilumalli ennen kuin varsinaisen järjestelmän kehittäminen aloitetaan. Toisin sanoen, tuotteen jotakin piirrettä kokeillaan ennen varsinaisen tuotteen rakentamista. Prototyypimallilla voi testata uusia teknisiä ratkaisuja ennen kuin tehdään ratkaisu käyttöönotosta. Mallilla voi myös etsiä ja selvittää epäselviä asiakasvaatimuksia. Valmistuneen prototyypin voi jakaa kahteen pääkäyttövaihtoehtoon: määritellään toteutettava järjestelmä, jota lähdetään toteuttamaan alusta saakka uudelleen ja prototyypin kehittämistä valmiiksi järjestelmäksi. (Haikala & Märijärvi 2004, 42; Räsänen 2008.)

Kuvassa 7 esitetään prototyypimalli, jossa järjestelmä toteutetaan alusta alkaen.



Kuvassa 8 prototyyppi kehitetään valmiiksi järjestelmäksi. Prototyypissä ei esimerkiksi vielä ole virhetarkastuksia, opastuksia, tehokkaasti toimivaa tietokantaa, mutta se sisältää kaikki tärkeimmät toiminnot. Ennen järjestelmän lopullista toteutusta varmistetaan, että kaikki tarvittavat toiminnot löytyvät järjestelmästä ja asiakas on tyytyväinen käyttöliittymään. Protoilu on osoittautunut erittäin hyödylliseksi käyttöliittymien määrittelyssä. (Haikala & Märijärvi 2004, 43.)

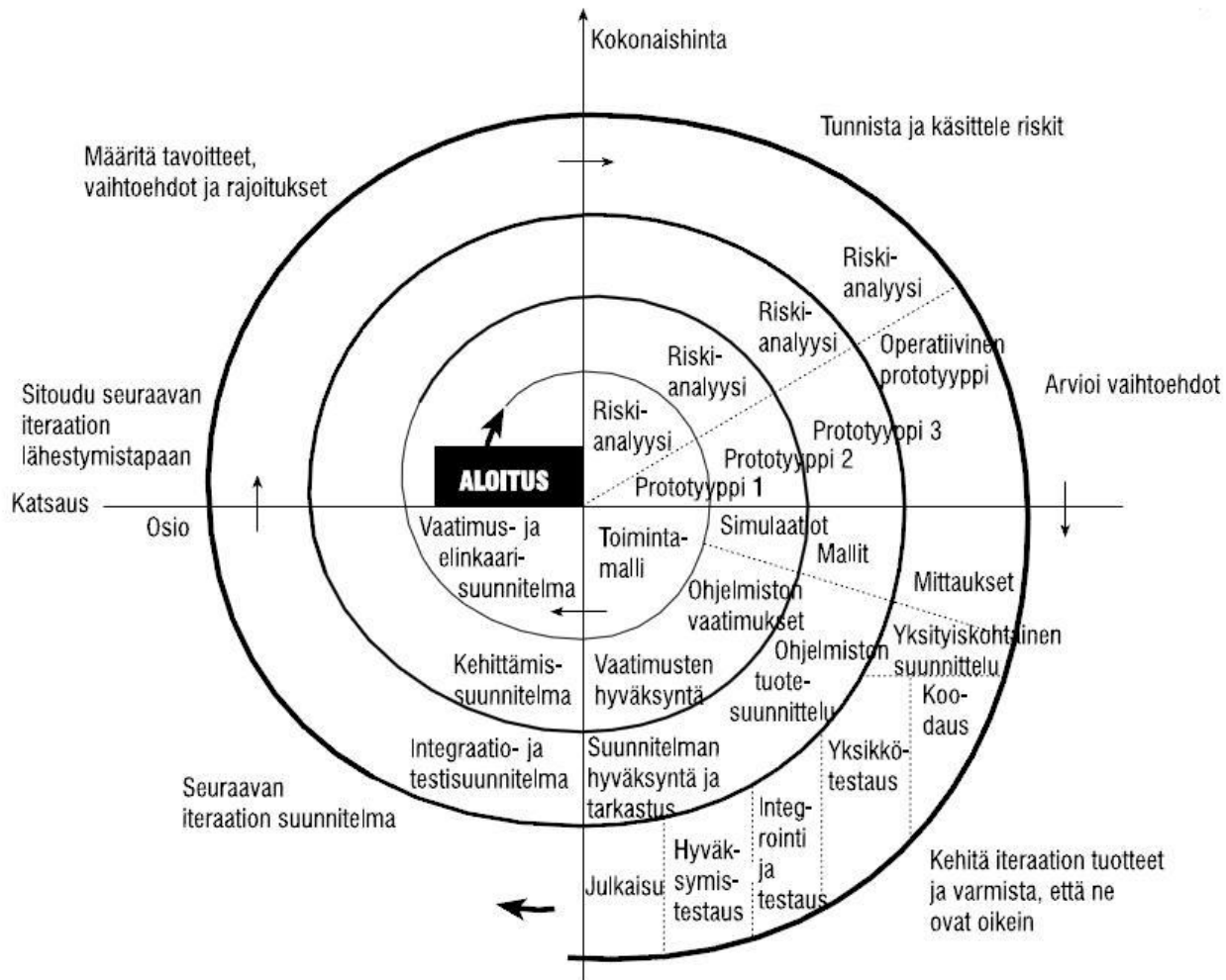
Protoilun ongelmaksi voi muodostua, että asiakas luulee järjestelmän, joka näyttää lähes valmiita, olevan jo valmis, vaikka suurin osa työstä on vielä tekevä. Asiakkaan täytyy nähdä, että järjestelmässä on vielä paljon tekemistä. Prototyypistä kannattaa siis tehdä myös sen näköinen ja tuntuinen, ettei se ole vielä valmis, eli hieman viimeistelemättömän näköinen, mutta asiakkaalle täytyy kuitenkin selvittää ohjelmiston tarkoitus. (Haikala & Märijärvi 2004, 43; Räsänen 2008.)

### **2.3.3 Spiraalimalli**

Spiraalimallissa (*spiral*) protoilu- ja elinkaarimalli yhdistetään. Spiraalimalli on suuntautunut riskeihin. Spiraalimallissa ohjelmistoprojekti pilkotaan pienemmiksi ohjelmistoprojekteiksi, joista jokainen keskittyy yhteen tai useampaan riskiin, kunnes kaikki tärkeimmät pääriskit on hoidettu. Riskejä voivat muun muassa olla väärin ja huonosti ymmärretyt ohjelmiston vaatimukset ja väärin ymmärretty arkkitehtuuri. Spiraalimalli korostaa riskien hallintaa: jos riskit kasvavat liian suuriksi, niin toteutus voidaan keskeyttää. Spiraalissa etuina ovatkin, että riskit ovat koko ajan täysin hallinnassa ja valmiina tuotteena on luotettava järjestelmä. (Pohjonen 2002, 42.)

Spiraalimalli perustuu neljään päävaiheeseen: suunnittelu, riskianalyysi, tuotanto ja arviointi. Näitä neljää vaihetta toistetaan tarkentaen niin kauan, että järjestelmä on valmis. Ensimmäinen vaihe on suunnittelu, jossa rakennettavalle ohjelmistolle määritellään tavoitteet, vaihtoehdot ja rajoitukset. Spiraalin seuraava vaihe on riskianalyysi, jossa arvioidaan eri vaihtoehtoihin liittyviä ongelmia. Seuraavassa vaiheessa on tuotanto, jossa tuotetaan seuraava prototyyppi eli ohjelmaversio. Arviointivaiheessa asiakas arvioi prototyyppiä, ja tämä arviointi

toimii tarkastuspisteenä. Tämän arvioinnin perusteella tehdään päätös ohjelmiston jatkamiselle tai lopettamiselle. (Pohjonen 2002, 42; Räsänen 2008.)

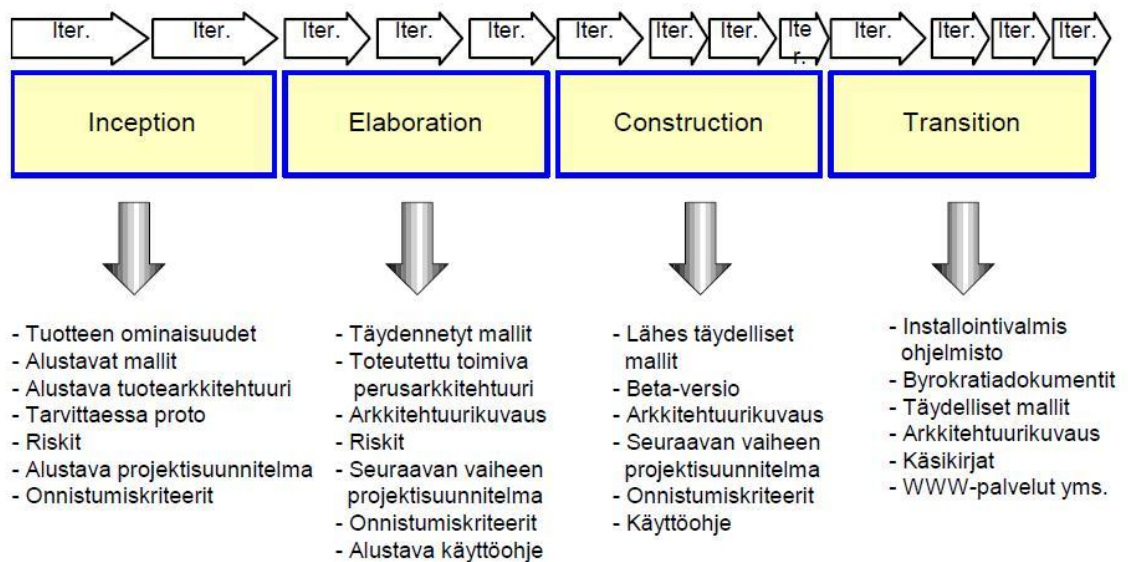


Kuva 9. Spiraalimalli (McConnell 2002, 142)

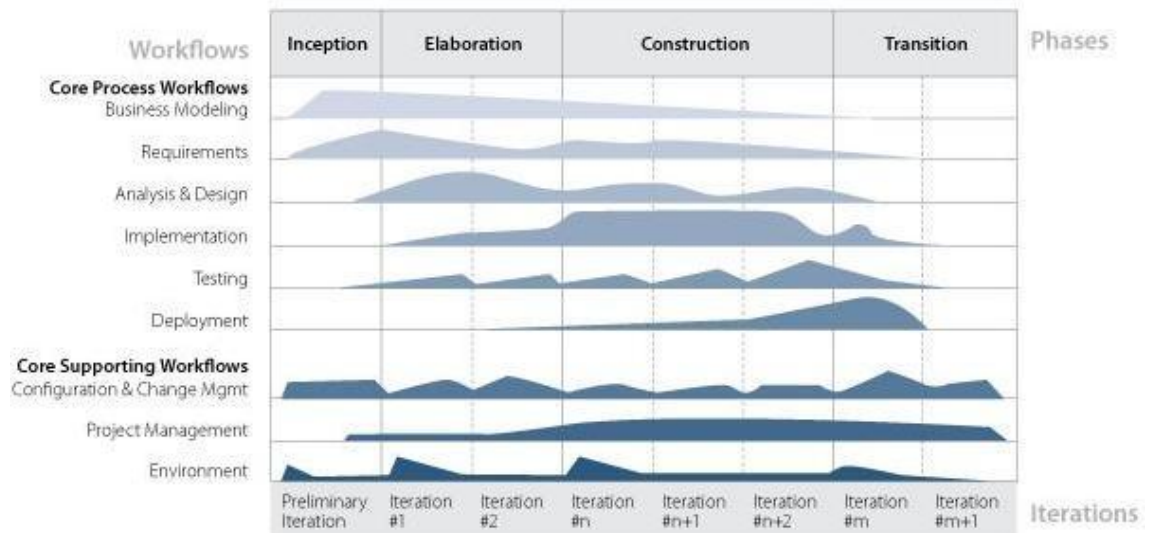
### 2.3.4 RUP-malli

RUP-prosessi (*Rational Unified Process*) perustuu peräkkäisiin iteraatioihin. Jokainen näistä iteraatioista muodostaa oman pienen vesiputouksensa: asiakasvaatimustenmäärittely, määrittely, suunnittelu, toteutus ja testaus. Ohjelmiston kehityksessä on neljä päävaihetta: aloitusvaihe (*inception*), valmisteluvaihe (*elaboration*), rakentamisvaihe (*construction*) ja siirtymävaihe (*transition*). Jokainen näistä vaihteista koostuu yhdestä tai useammasta iteraatiosta. Iteraatiot ovat tyypillisesti lyhyitä ja kestävät muutamasta viikosta muutamaa kuukautta. Iteraatiota kutsutaan myös sykliksi. Jokaisen iteraation tuloksena järjestelmä kasvaa uusilla ominaisuuksilla. Aloitusvaiheessa kartoitetaan erilaisia

vaihtoehtoja tuotekonseptiin. Aloitusvaiheen tehtävät painottuvat asiakasvaatimusten analysointiin, joka on samankaltainen kuin perinteinen esitutkimusvaihe. Valmisteluvaiheessa määritellään ominaisuuksia ja toteutetaan perusarkkitehtuuri. Rakentamisvaiheessa jokaisella iteraatiolla tuotetaan järjestelmästä uusi versio, eli niin sanottu beta-versio, joka annetaan valikoidulle käyttäjäkunnalle testikäyttöön. Siirtymävaiheessa asennusvalmis ohjelmisto siirretään käyttöön ja kaikki tarvittavat käsikirjat, ohjeet ja palvelut siirtyvät myös asiakkaalle. RUP-prosessimallin tapaiseen ohjelmistokehitykseen kuuluu vakava projektinhallinnallinen ongelma. Kun uusi versio ohjelmasta on saatu asiakkaalle, niin saattaa käydä, että virheiden korjaamiseen ja asiakkaan ongelmien selvittelyyn kuluu projektiryhmän aika. Näin ollen ohjelmistonkehityksessä ei päästä eteenpäin. (Haikala & Märijärvi 2004, 46.)



Kuva 10. RUP-prosessin päävaiheet (Haikala & Märijärvi 2004, 46)



Kuva 11. RUP-prosessissa työmäärän jakautuminen (Yumasoft 2003 – 2012)

## 2.4 Ketterä ohjelmistokehitys

Ketterässä (*agile*) menetelmässä ohjelmisto tuotetaan pienissä osissa, eli iteraatiot ovat lyhyitä. Jokainen iteraatio sisältää tehtävät, joita tarvitaan uusien toimintojen julkaisemiseen: projektisuunnittelun, vaatimusanalyysin, ohjelmistosuunnittelun, toteutuksen, testauksen ja dokumentoinnin. Ketterässä ohjelmistokehityksessä pyritään jokaisen iteraation lopussa julkaisukelpoiseen ohjelmiin, joten kun iteraatio on valmis, niin se toimitetaan asiakkaalle testattavaksi. Yksinkertaistetusti uuden ominaisuuden tai ohjelmistomuutoksen tekeminen etenee, että aloitetaan työ ohjelmoimalla muutokselle vähintään yksi testitapaus, ja testitapaukset ajetaan läpi ja huomataan, että testit epäonnistuivat. Epäonnistuneiden testien jälkeen aloitetaan näiden testitapausten ohjelmointi, ja ohjelmoidaan niin kauan, että kaikki testitapaukset saadaan virheettömästi suoritettua läpi. (Haikala – Märijärvi 2004, 47.)



Ketterässä menetelmässä liiketoiminnan ja teknisen puolen edustajan toivotaan tekevän yhteistyötä päivittäin, jotta molemmat osapuolet ymmärtävät ja saavuttavat projektin tavoitteet. Ohjelmistokehityksessä asiakkaalta vaaditaan tiivistä yhteydenpitoa projektin aikana, jolloin asiakas saa mahdollisuuden vaikuttaa enemmän lopputulokseen. Ketterässä menetelmässä käytetään tiimityötä, ja tiimit pidetään sopivan kokoisina ja työmäärä sopivana työmäärään nähden. Työtulos kärsii, jos tiimin henkilöstö on yllirasittunut. Tiimi suunnittelee itse arkkitehtuurin ja selvittää vaatimukset. Tiimit usein työskentelevät samassa tilassa, jolloin viestintä sujuu kasvotusten enemmän kuin dokumentteja lukemalla. Ketterässä menetelmässä pyritään parhaaseen ja yksinkertaisimpaan mahdolliseen ratkaisuun tinkimättä ohjelmiston teknisestä lopputuloksen tasosta. (Sainio 2010.)

Ketterissä menetelmissä noudatetaan seuraavaa 12 periaatetta:

1. ”Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.”
2. ”Otamme vastaan myös muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyyn edistämiseksi.”
3. ”Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.”
4. ”Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan.”
5. ”Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.”
6. ”Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.”
7. ”Toimiva ohjelmisto on edistymisen ensisijainen mittari.”
8. ”Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.”
9. ”Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.”

10. ”Yksinkertaisuus – tekemättä jätettävän työn maksimointi – on oleellista.”
  11. ”Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itse organisoituvissa tiimeissä.”
  12. ”Tiimi tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan, ja mukauttaa toimintansa sen mukaiseksi.”
- (Agile Manifesto 2001.)

#### **2.4.1 Extreme programming**

Extreme programming tunnetaan myös nimellä XP-menetelmä, joka on ketteristä menetelmistä se kaikkein tunnetuin ja keskittyy pääasiassa ohjelmistokehityksen ohjelmointipuoleen. XP-menetelmälle tunnusomaisia piireitä ovat esimerkiksi jatkuva testaaminen ja niin sanottu pariohjelmointi (pair programming). Pariohjelmoinnissa ohjelmistokehitys tapahtuu pareittain, toinen tuottaa koodia ja toinen kommentoi juuri tuotettua koodia. (Haikala & Märijärvi 2004, 47.)

Extreme programming -menetelmällä on perustana neljä arvoa: kommunikaatio, yksinkertaisuus, palaute ja rohkeus. XP-menetelmässä kommunikoidaan keskustelemalla kasvotusten ja tuotetaan vähemmän dokumentteja. Yksinkertaisuudella tarkoitetaan, että järjestelmä toteutetaan mahdollisimman yksinkertaisella tavalla. Suunnitteluvaiheessa täytyy kaikkien vaatimuksien olla selvillä, jolloin järjestelmää ei lähdetä myöhemmin laajentamaan. Tämä arvo hieman sotii muiden ohjelmistokehitysmallien kanssa, koska muut kehitysmallit kannustavat rakentamaan järjestelmän, joka on helposti laajennettavissa. Palaute kerätään heti, kun työ on tehty, ja se halutaan asiakkaalta, toimittajalta ja rakennettavalta ohjelmistolta. Järjestelmästä saadaan palautetta integroinneista ja testeistä päivittäin. XP-menetelmä on varsin radikaali periaatteiltaan, joten rohkeutta tarvitaan. Menetelmän takia asiakas ei voi saada tietoa ohjelmiston kustannuksista ja työmäärästä, joten rohkeutta vaaditaan myös asiakkaalta. (Lindberg, 22.)

XP-menetelmän käytännöt:

1. *Suunnittelupeli (planning game)*

Tarkoittaa tiivistä yhteistyötä asiakkaan ja ohjelmoijien välillä. Ohjelmoijat tekevät tehtävistä työtarpeista arvoon työtarpeista ja asiakas päättää julkaisujen ajankohdan ja laajuuden.

2. *Pienet julkaisut (small releases)*

Pieniä julkaisuja järjestelmästä julkaistaan tiheästi, koska kaikkia ongelmia ei ratkaista kerralla. Julkaisuja järjestelmän versioista voi tulla päivittäin, mutta vähintään kerran kuukaudessa.

3. *Metafora (metaphor)*

Tarkoittaa vertauskuvaa, joka tarjoaa kehitettävästä järjestelmästä ja sen toiminnasta kokonaiskuvan. Monimutkaiset asiat voidaan selittää metaforan avulla niin, että asiakaskin ymmärtää.

4. *Yksinkertainen suunnittelu (simple design)*

Mahdollisimman yksinkertainen suunnittelu, jossa tarpeeton kompleksisuus ja tarpeeton koodi poistetaan.

5. *Testaus (test)*

Tehdään ohjelmalle asiakkaan määrittelemät toiminnalliset testit ja kehittäjien vaatimat yksikkötestit.

6. *Refaktorointi (refactoring)*

Muokataan koodia paremmin ylläpidettäväksi ja selkeämpään muotoon ohjelmiston toiminnallisuuteen puuttumatta.

7. *Pariohjelmointi (pair programming)*

Ohjelmistoa kehitetään pareittain yhdellä tietokoneella, toinen kirjoittaa koodia ja toinen kommentoi tuotettua koodia.

8. *Koodin kollektiivinen omistaminen (collective ownership)*

Tarkoittaa koodin yhteisomistusta eli jokainen ohjelmoija voi muokata koodia milloin vain.

9. *Jatkuva integrointi (continuous integration)*

Ohjelmistoa koostetaan ja integroidaan jatkuvasti, määritetyt testit suoritetaan ja kaikkien testien täytyy mennä läpi, jotta ne voidaan hyväksyä.

### 10.40 tuntinen työviikko (40 hour week)

Viikossa tehdään 40 työtuntia, joka on maksimi eikä ylitöitä sallita, ettei henkilöstö yllirasitu ja työtulos kärsi.

### 11. Asiakkaan läsnäolo (on-site customer)

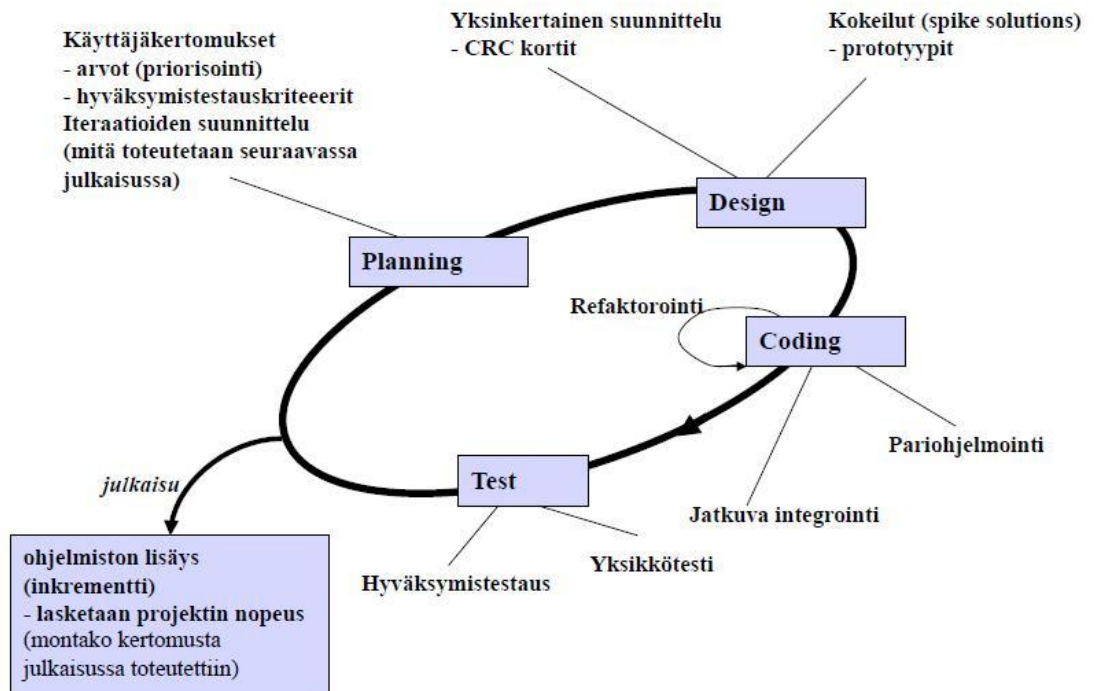
Asiakas tekee tiivistä yhteistyötä tiimin kanssa koko projektin ajan.

### 12. Ohjelmointistandardit (coding standards)

Kehitystiimi omaksuu vapaaehtoisesti standardit, koska projekteissa ei käsketä käyttää ohjelmointistandardeja. Ohjelmakoodia tuottaessa on kuitenkin noudatettava siihen liittyviä sääntöjä.

(Tervonen 2003, 25.)

## XP prosessi



Kuva 12. XP-prosessin kuvaus (Tervonen 2003, 16)

## 2.4.2 Scrum

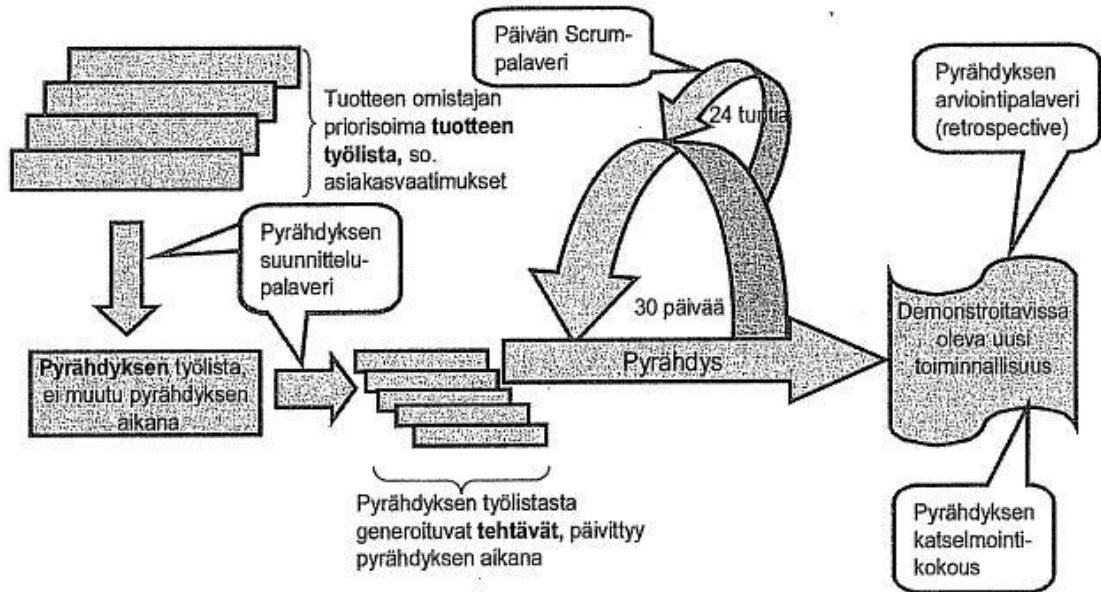
Scrum on ketterän ohjelmistokehityksen malli, joka on iteratiivinen ja inkrementaalinen suunnittelu- ja tuotantoprosessimenetelmä. Ohjelmisto rakentuu vähitellen täydellisemmäksi ja valmiimmaksi useiden toteutuskierrosten eli kehitysjaksojen aikana. Kehitysjaksoa kutsutaan pyrähdykseksi (*sprint*). Jokainen pyrähdys kestää noin 30 kalenteripäivää. Pyrähdyn suunnittelupalaveri kestää koko päivän ja siihen osallistuvat tuotteen omistaja, Scrum-mestari ja tiimi. Jokainen pyrähdys alkaa kyseessä olevalla palaverilla. Kokouksessa tuotteen omistaja esittää tuotteen työlistan, jonka jälkeen sovitaan, mitkä alkiot tuotteen tehtävälästä otetaan mukaan seuraavaan pyrähdykseen. Alkiot pilkotaan tehtäviin, jotka yleensä kestävät 4 – 16 tuntia. Pyrähdyn tehtävälästä ei saa muuttaa pyrähdyn aikana. Päiväpalaveri (*daily scrum*) pidetään joka päivä, ja sen pituus on noin 15 minuuttia, jonka aikana jokainen tiimin jäsen vastaa kolmeen kysymykseen:

- ”Mitä olet tehnyt edellisen kokouksen jälkeen?”
- ”Mitä aiot tehdä seuraavaksi?”
- ”Onko tiedossasi esteitä, jotka hidastavat työtäsi?”

Tähän kokoukseen osallistuvat Scrum-mestari ja tiimi. Pyrähdyn jälkeen pidetään pyrähdyn katselmointikokous, johon osallistuu tuotteen omistaja, Scrum-mestari, tiimi ja haluttaessa sidosryhmien edustajia. Tämän jälkeen on pyrähdyn arviointipalaveri (*retrospective*), jossa arvioidaan pyrähdyn onnistumista ja mietitään toimintatapojen kehittämistä. Palaveriin osallistuvat tuotteen omistaja, Scrum-mestari ja tiimi. Palaveri kestää noin 2 – 3 tuntia. Jokainen palaveriin osallistuja vastaa kahteen kysymykseen:

- ”Mikä meni hyvin?”
- ”Mitä pitäisi parantaa seuraavassa pyrähdyksessä?”

(Haikala & Mikkonen 2011, 49 – 51; Hypermedian opetus 2011.)



Kuva 13. Scrum-prosessi (Haikala & Mikkonen 2011, 48)

Scrum-projektissa esiintyy kolme roolia; tuotteen omistaja (*product owner*), Scrum-mestari (*Scrum master*) ja tiimi (*team*). Perinteisiä nimityksiä käyttäen tuotteen omistaja muistuttaa tuotepäällikköä, Scrum-mestari projektipäällikköä ja tiimi projektiryhmää. (Haikala & Mikkonen 2011, 48.)

Tuotteen omistaja (*product owner*) toimii rajapintana projektin kaikkiin sidosryhmiin ja vastaa projektin taloudellisesta tuloksesta. Hän kerää kaikilta sidosryhmiltä järjestelmän vaatimukset tuotteen työlistaksi (*product backlog*) ja ylläpitää listaa prioriteettijärjestyksessä. Työlistan alkiot (*item*) voivat olla melkein mitä tahansa, esimerkiksi tuotteen ominaisuuksia, käyttötapauksia, käyttäjätarinoita, vaatimuksia, virheraportteja, dokumentaation kehittämistä ja arkkitehtuurin parantamista. Jokaiseen työlistan alkioon liittyy aika-arvio ja arvio sen liike-toiminta-arvosta. Projektin epäonnistumisen syyt johtuvat yleensä vaatimustenhallinnasta. Scrum-malli ei ota tähän mitään kantaa, ja vaatimustenhallinta jätetäänkin kokonaan tuotteen omistajan vastuulle ja tehtäväksi. (Haikala & Mikkonen 2011, 48 - 49.)

Scrum-mestari (*Scrum master*) vastaa ja huolehtii, että noudatetaan Scrum-prosessia, toimii valmentajana tiimille ja tuotteen omistajalle ja vastaa pyrhdyksen tuloksesta ja varmistaa, että valmistumisen toteamiseen määritellyt ehdot (*definition – of – done, DoD*) on täytetty ennen kuin tehtävä voidaan merkitä valmiiksi. Ehtoja voivat olla esimerkiksi, että koodi kirjoitettu, testitapaukset ovat olemassa ja ajettu onnistuneesti läpi sekä dokumentaatio on päivitetty. Scrum-mestari vastaa myös tiimistä ja sen hyvin voinnista sekä poistaa esteet (*impediment*) jotka haittaavat ja hidastavat tiimin työskentelyä. Lisäksi Scrum-mesterin tehtävänä on poistaa sopimaton tiimin jäsen. (Haikala & Mikkonen 2011, 49.)

Tiimiin (*team*) kuuluvat kaikki, jotka ovat tekemässä projektia, ja tiimin ihanne koko on noin 7 kokopäiväistä henkilöä, jotka taustojen perusteella saisivat olla erilaisia, kuten esimerkiksi testaajia, ohjelmoijia, käyttöliittymäsuunnittelijoita, joilta löytyy tarvittava osaaminen. Tiimi on itseorganisoituva ja päättää itse työskentelykäytännöistä, eli varsinaista projektipäällikköä ei ole vaikka Scrum-mestari yleensä nimitetään projektipäälliköksi. Tiimi itse vastaa yhteisesti pyrhdyksen työlistan paloittelusta tehtäviksi ja jakaa ne keskenään. Ihanne on, että tiimi työskentelee samassa tilassa, johon tehtävätaulu sijoitetaan. Tehtävätaulusta tiimin jäsenet näkevät kaikki tehtävät ja niiden tilat, joita ovat esimerkiksi ei-aloitettu, kesken, valmis ja hyväksytty. Tiimi ja Scrum-mestari ylläpitävät tehtävätaulua yhdessä. (Haikala & Mikkonen 2011, 49.)

Scrum-menetelmällä on perustana viisi arvoa: sitoutuminen, keskittyminen, avoimuus, kunnioitus ja rohkeus. Sitoutumisella tarkoitetaan tiimiä, sillä tiimi sitoutuu yhteiseen päämäärään. Tiimillä on vapaat kädet valita, miten edetään ja miten työ tehdään, joten näihin päätöksiin täytyy sitoutua. Keskittyminen tarkoittaa, että keskitytään vain siihen, mitä on luvattu tehdä ja mitä ollaan tekemässä, sillä jokaisella pyrhdyksellä on selkeä tavoite ja tämän tavoitteen saavuttaminen on tiimin tehtävä, joka vaatii keskittymistä. Avoimuus on prosessissa tärkeää, koska kaikki projektin tiedot ovat näkyvillä. Projektin ja pyrhdyksen työlistat ovat julkisia, päiväpalaveri on kaikille avoin ja jokaisen pyrhdyksen tulokset esitetään julkisesti. Kunnioitus on tärkeää tiimissä, sillä tiimin rauhan ylläpitämiseksi jokaista tiimiläistä täytyy kunnioittaa: yksilön osaamista ja itse

ihmistä sellaisenaan. Rohkeutta tiimiltä vaaditaan sitoutumisessa ja avoimuudessa. Omat virheet täytyy myöntää ja kertoa siitä tiimiläisille. (Sininen Meteoritti 2011.)

## **2.5 Laatu, laatujärjestelmä ja laadun varmistus**

ISO-standardin mukaan laadulla tarkoitetaan ohjelmistotuotteen kykyä täyttää asiakkaan eli käyttäjensä kohtuulliset toiveet ja odotukset: se valmistuu sovittoon aikaan mennessä sovitulla budjetilla, on hinnalta kohtuullinen, virheetön ja tehokas ja dokumentit ovat käyttökelpoisia. Näin ollen laatu on subjektiivinen eli se on käyttäjästä ja käyttöympäristöstä riippuvainen käsite. Termillä laatu ei siis tarkoiteta huippulaatua vaan laadukkaalla tuotteella tarkoitetaan, että ohjelmisto täyttää sille vaaditut toiminnalliset ja ei-toiminnalliset vaatimukset. (Haikala & Märijärvi 2004, 48; Immonen 2003.)

Laatujärjestelmäksi kutsutaan yrityksen toimintatapaa, jota käytetään tuotteen tekemisessä. ISO-standardeissa käytetään termiä laadunhallintajärjestelmä, mutta tässä raportissa käytetään lyhyempää termiä laatujärjestelmä. Laatujärjestelmä takaa, että tuotantoprosessissa tuotetaan sovittua laatua sovitulla budjetilla sovitussa ajassa, eli laatujärjestelmä muodostuu yrityksen työprosesseista, jolloin asetetut laatutavoitteet pyritään saavuttamaan. (Haikala & Märijärvi 2004, 48; Immonen 2003.)

Laadunvarmistus voidaan karkeasti jakaa kahteen osaan: toiminnan laadunvarmistus ja tuotteen laadunvarmistus. Toiminnan laadunvarmistuksella tarkoitetaan tuotantoprosessien laadunvarmistamista. Laatua voidaan varmistaa esimerkiksi auditioinneilla. Auditioinnissa laatujärjestelmä tai jokin laatujärjestelmän osa käydään systemaattisesti läpi, ja varmistetaan laatujärjestelmän ja toiminnan yhdenmukaisuus. Tuotteen laadunvarmistuksessa estetään virheiden pääseminen tuotteeseen mutta jos tuotteeseen on päässyt virhe, niin se pyritään etsimään mahdollisimman aikaisin.



## 2.6 Dokumentointi ohjelmistotuotannossa

Dokumentoinnissa tuotetaan asiakirjoja, eli projektin aikana kirjattu ja kerätty tieto saatetaan dokumentti muotoon. Dokumentointi on iso osa ohjelmistotuotantoa, joka valitettavasti jää yleensä aikataulun ja laiskuuden takia tekemättä. Erilaisia tehtävään ja projektiin liittyviä dokumentteja voi olla kymmeniä. Kolme dokumenttia ohjelmistoprojektista on vähintään löydyttävä: projektisuunnitelma, määrittelydokumentti eli toiminnallinen määrittely ja suunnitteludokumentti eli tekninen määrittely, sekä mielellään testaussuunnitelma testausvaihetta varten. Tämänkaltaista dokumentointia kutsutaan minimidokumentoinniksi, koska edellä mainitut dokumentit ovat niin sanotusti pakollisia. Dokumenteista kannattaa tehdä yhdenmukaiset, jolloin dokumenteista kannattaa tehdä dokumentti-pohjat. (Haikala & Märijärvi 2004, 51.)

Huonosti tuotetut dokumentit muuttuvat ajan kuluessa täysin hyödyttömiksi, joten dokumentit kannattaa tuottaa laadukkaiksi. Jokainen tehtävä muutos ohjelmistoon täytyy dokumentoida kunnolla, jotta dokumenteista olisi hyötyä jopa tulevaisuudessakin. Projektin edetessä dokumentointi jää unohduksiin aikataulupaineiden takia, joten dokumentointia tehdään myös jälkikäteen. Minimidokumentaatio tuotteesta on oltava, sillä näiden dokumenttien puuttuessa jossakin vaiheessa projektia seuraa koko tuotteen uudelleenkodeaus. (Haikala & Märijärvi 2004, 70 – 71.)

Ylläpidossa ohjelmistoa korjataan ja muutetaan, jolloin dokumentaatiota pitäisi päivittää. Valitettavasti tämä päivitys jää usein tekemättä. Laadukas dokumentointi auttaa tulevaisuudessa paljon. Ohjelmiston laajennus- tai muutosvaiheessa tarvitaan kaikki tuotetut dokumentit aina vaatimustenmäärittelystä aina ylläpitoon. (Haikala & Märijärvi 2004, 51 – 52.)

### 3 Toiminnallinen määrittely

Toiminnallinen määrittely (*functional specification*) on yksi ohjelmiston elinkaaren vaihe, jossa kuvataan kaikki ohjelmiston toteuttamat toiminnot (*operations, functions*), toteutukselle asetettavat ei-toiminnalliset vaatimukset sekä rajoitukset ja kaikki liitännät järjestelmän ulkopuolelle. Ei-toiminnallisia vaatimuksia ovat muun muassa suoritusteho, käytettävyyys ja vasteaika. Rajoituksia ovat esimerkiksi käytettävissä oleva muistitila ja ohjelmiston toteutus tietyllä ohjelmointikielellä. Toiminnalliseen määrittelyyn dokumentoidaan järjestelmän vaatimukset sekä vaatimukset täyttävän järjestelmän kuvaus. Toimintojen yhteydessä määritellään ohjelmistolla toteutettavat ominaisuudet, käyttöliittymä ja kommunikointi toisten järjestelmien kanssa. (Haikala & Märijärvi 2004, 39, 79 – 81.)

Toiminnallinen määrittelydokumentti kuvaa järjestelmän kaikki erilaiset näytöt, valikot sekä niiden asetukset ja kaikki käyttöliittymäkontrollit. Järjestelmän tuottaessa tulosteita määrittelydokumentissa tulisi olla kaikista järjestelmän eri näytöistä visuaalinen ja tekstikuvaus. Määrittelydokumentin tulisi olla tarpeeksi kattava, että teknisessä suunnitteluvaiheessa ja siitä eteenpäin ei ole enää missään tulevassa vaiheessa epäselvää, miten järjestelmän tulee toimia missäkin vaiheessa ja tilanteessa. (Reaktor, Laakso & Mäkinen 2011.)

Onnistunut toiminnallinen määrittely vähentää järjestelmän käyttäjän työaikaa tai poistaa turhia työvaiheita, kuvaa tarkasti toteutettavan ohjelmiston, jolloin selviää, mitä toimintoja järjestelmään tulee ja mitä toimintoja siihen ei tule, tuottaa yksinkertaisen, tehokkaan ja käyttäjäystävällisen käyttöliittymän ja vähentää virheellisestä tai puuttuvasta toiminnallisuudesta aiheutuvat kalliit muutostyöt. (Reaktor, Laakso & Mäkinen 2011.)

### 3.1 Toiminnallisen määrittelyn dokumenttimallit ja standardit

<b>1. Johdanto</b> 1.1 Tarkoitus 1.2 Tuote 1.3 Määritelmät, termit ja lyhenteet 1.4 Viitteet 1.5 Yleiskatsausdokumenttiin	<b>6. Muut ominaisuudet</b> 6.1 Suorituskyky ja vasteajat 6.2 Käytettävyys, toipuminen, turvallisuus ja suojaukset 6.3 Ylläpidettävyys 6.4 Siirrettävyys ja yhteensopivuus 6.5 Operointi
<b>2. Yleiskuvaus</b> 2.1 Ympäristö 2.2 Toiminta 2.3 Käyttäjät 2.4 Yleiset rajoitteet 2.5 Oletukset ja riippuvuudet	<b>7. Suunnittelurajoitteet</b> 7.1 Standardit 7.2 Laitteistorajoitteet 7.3 Ohjelmistorajoitteet 7.4 Muut rajoitteet
<b>3. Tiedot ja tietokanta</b> 3.1 Tietosisältö	<b>8. Hylätyt ratkaisuvaihtoehdot</b>
<b>4. Toiminnot</b> 4.1 Yleistä 4.2 Toiminnot n Toiminnon kuvaus	<b>9. Jatkokehitysajatuksia</b>
<b>5. Ulkoiset liittymät</b> 5.1 Käyttöliittymä 5.2 Laitteistoliittymä 5.3 Ohjelmistoliittymät 5.4 Tietoliikenneliittymät	<b>10. Vielä avoimet asiat</b>

Kuva 14. Toiminnallisen määrittelyn sisältörunko (Haikala & Märijärvi 2004, 80)

Johdannossa kerrotaan, kenelle järjestelmä on tehty, miksi järjestelmä on tehty sekä tavoitteet, ja jos on tarvetta, niin määritelmiä, termejä ja lyhenteitä. Mikäli on dokumentteja, jotka liittyvät järjestelmään, ne eritellään myös johdannossa. Johdannon yleiskatsausdokumenttiin kohdassa kuvataan dokumentin rakenne, jonka tarkoituksena on antaa lukijalle tarpeeksi tietoa siitä, mitkä kohdat dokumentista kannattaa lukea. (Haikala & Märijärvi 2004, 80.)

Dokumentin toisessa luvussa on yleiskuvaus järjestelmän toiminnasta. Yleiskuvauksessa kuvataan järjestelmään liittyvä ympäristö, toiminta, käyttäjät, käyttöympäristö ja yleiset rajoitteet. Yleiset rajoitteet kohdalla tarkoitetaan esimerkiksi toteutustyökaluihin tai lainsäädäntöön liittyviä rajoitteita. Viimeisessä kohdassa esitetään oletukset, joiden voimassa ollessa määrittely on voimassa. Voimassa olevat oletukset voivat koskea esimerkiksi käytettävissä olevan lait-

teiston tehoa tai muiden osaprojektin tuottamia tuotteita. (Haikala & Märijärvi 2004, 80.)

Kolmannessa luvussa kuvataan järjestelmän käsittelemistä tiedoista ja tietokannoista niiden tietosisältö, tiedon pysyvyysvaatimukset, kapasiteetti- ja saantivaatimukset ja niin edelleen. Tietojen attribuuteista kuvataan niiden esitystapa, tarkkuus, käytettävät yksiköt ja rakenne, sekä haluttaessa kuvauksessa voi käyttää sanallisia kommentteja. (Haikala & Märijärvi 2004, 80 – 110.)

Dokumentin neljännessä luvussa kuvataan toiminnot. Jokaisessa alaluvussa täsmennetään aina yksi järjestelmän toiminto. Jokaisesta toiminnosta kuvataan sen tarkoitus, mitä syötteitä toiminto tarvitsee, miten toiminnon käsittely tapahtuu ja mitä vaikutuksia ja mitä tulosteita toiminnolla on. (Haikala & Märijärvi 2004, 80 – 81.)

Viidennessä luvussa täsmennetään toisessa luvussa yleisesti kuvatut liittymät järjestelmän ympäristöstä. Tässä vaiheessa voidaan jo tehdä tarkka käyttöliittymän kuvaus tai se voidaan vielä esittää yleisellä tasolla, jolloin käyttöliittymän tarkentaminen jää suunnitteluvaiheen tehtäväksi. Liittymät oheislaitteisiin ja tietoliikenneyhteyksiin kuvataan myös tässä luvussa. (Haikala & Märijärvi 2004, 80 – 81.)

Kuudennessä luvussa kuvataan järjestelmän ei-toiminnalliset ominaisuudet, kuten suorituskyky, luotettavuus, turvallisuus, käytettävyys, ylläpidettävyys ja siirrettävyys (Haikala & Märijärvi 2004, 81).

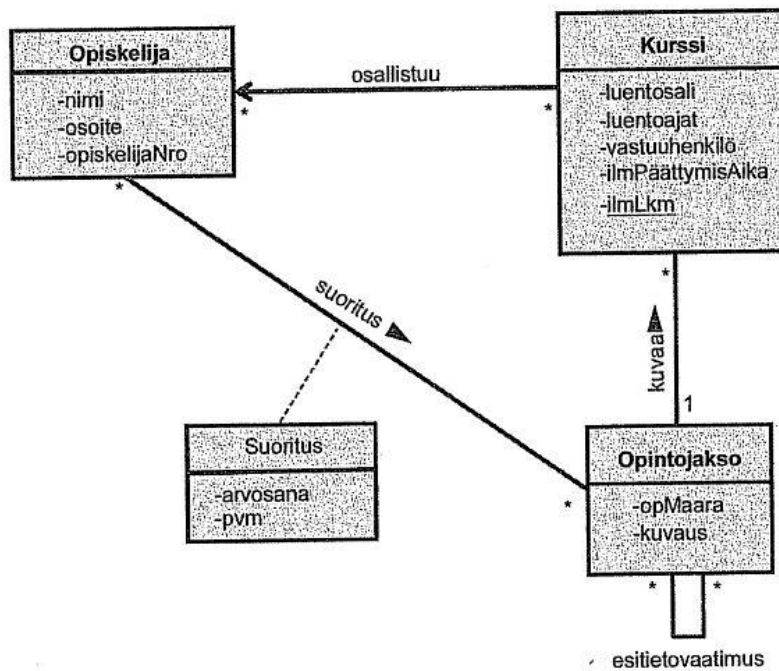
Seitsemännessä luvussa kuvataan suunnittelurajoitteet. Suunnittelurajoitteita ovat standardit, laitteisto- ja ohjelmistorajoitteet. Kahdeksas, yhdeksäs ja kymmenes luku eivät ole pakollisia, mutta kaikki aikaisemmat luvut ovat. (Haikala & Märijärvi 2004, 81.)

Toiminnallinen määrittely kuvitetaan eri notaatiolla tehdyillä kuvauksilla. Esimerkiksi dokumentin yleiskuvausluvussa voidaan käyttää käyttötapauksia, liittymäkaavioita, ylimmän tason tietovirtakaavioita, luokkakaavioita sekä erilaisia käyt-

töliittymäkuvauksia havainnollistamaan yleiskuvausta. Dokumentin kolmannessa luvussa käytettäviä kuvaustekniikoita voivat esimerkiksi olla tietohakemistokuvaukset ja luokkakaaviot. Toimintojen määrittelyssä apuna voidaan käyttää esimerkiksi tietovuokavioita, tila-automaatteja ja toiminnanmäärittelytekniikoita, joita ovat esimerkiksi kulkukaaviot, päätöstaulut ja kommunikointikaaviot ja niin edelleen. (Haikala & Märijärvi 2004, 81.)

### 3.2 Tietojen kuvaaminen

Tietojen kuvaaminen aloitetaan käsitekaavion piirtämisellä. UML:ssä käsitekaavio piirretään luokkakaavion muotoon (*Domain model*). Tietojen kuvauksessa kiinnostavia ovat tietoluokat (*entity*), joita kutsutaan myös yksilötyypeiksi. Jokaisesta yksilötyyppistä eli tietoluokasta on kuvattava sen ominaisuustyytit eli attributit. Attribuuttien kuvauksessa on hyvä käyttää taulukoita. (Kuva 15.)



Kuva 15. Luokkakaavio (Haikala & Mikkonen, 86)

Kaikki järjestelmässä tarvittavat tiedot kuvataan tietohakemistossa (*data dictionary*), ja tietojen kuvaustarkkuus riippuu tietohakemiston esitystavasta, mutta pyritään niin tarkkaan kuvaukseen, jonka perusteella voidaan tuottaa esimerkiksi ohjelmointikielen syntaksin mukaiset määrittelyt. Tiedoista voidaan kuvata esimerkiksi esitystapa, tarkkuus, käytettävät yksiköt, rakenne ja tietojen yhteyteen voi liittää sanallisia kommentteja. Tarvittavat tiedot tietojen kuvaamiseen saadaan esimerkiksi luokkakaaviosta, jossa näkyvät kaikki järjestelmässä tarvittavat taulut ja niiden attribuutit eli kuvattavat tiedot, ja ne kuvataan tietohakemistonotaation avulla. Yleisesti käytetyn tietohakemistonotaation merkinnät näkyvät alla olevassa kuvassa (kuva 16) ja niitä käytetään tietosisällön kuvaamisessa. Henkilötaulun henkilötiedot voidaan kuvata esimerkiksi näin:

- ”henkilötiedot = nimi + @henkilötunnus + aviosääty
- nimi = 1 {etunimi} 3 + sukunimi
- henkilötunnus = 6 {numero} 6 + 3 {numero} 3 + 1 {numero|kirjain} 1
- aviosääty = [naimaton|naimisissa|eronnut|leski]”

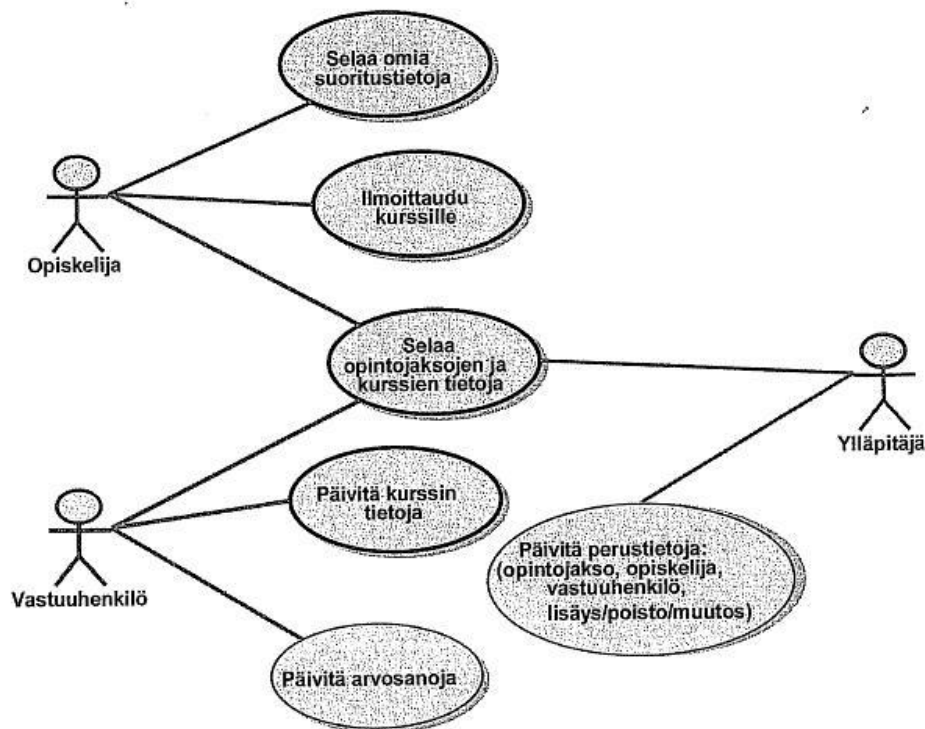
(Haikala & Mikkonen 2011, 131 – 132; Haikala & Märijärvi 2004, 110.)

merkki	merkitys
+	ja
n{<...>}m	toisto n–m kertaa
(<...>)	optionaalinen (voi puuttua), siis oikeastaan 0{<...>}1
{<...>}	toisto (0–n kertaa), siis oikeastaan 0{<...>}n
[<...> <...> <...>]	vaihtoehdot esitetään hakasulkeissa
	vaihtoehtojen erotin
@	avainkenttä
*	* kommentti *

Kuva 16. Tietohakemistonotaatio (Haikala & Mikkonen 2011, 131)

### 3.3 Käyttötapaukset

Käyttötapauskaavio (*use case diagram*) on yksi UML-mallinnuksessa käytettävä kaavio, joka kuvaa toimijan (*actor*) toimintoja ohjelmistossa tai järjestelmässä. Hahmottaessa ja rajattaessa järjestelmää käyttötapauskaavio toimii apuna ja kaavio auttaa tunnistamaan järjestelmää käyttäviä sidosryhmiä eli kuka tai mikä käyttää järjestelmää. Kuvassa 17 esitetään kaikki tärkeimmät elementit, joita kaaviossa käytetään: käyttötapaukset (*use case*) toimijat (*actor*) ja niitä yhdistävät viivat. Kuvassa näkyvät soikiot kuvaavat käyttötapauksia. Toimijat edustavat järjestelmän käyttäjiä, mutta toimijoita voivat olla myös toiset tietokoneet, tulostimet ja tietokannat. Viiva käyttötapauksen ja toimijan välillä tarkoittaa, että toimija osallistuu jollakin tavalla kyseiseen käyttötapaukseen. Käyttötapauskaavio antaa selkeän yleiskuvan järjestelmästä. Käyttötapaukset täytyy ensin selvittää, jonka jälkeen ne voidaan kuvata yksityiskohtaisesti. Käyttötapauksia voi kuvata esimerkiksi käyttötapauksen kuvauksella tai käyttäjätarinalla. Lisäksi jokainen toiminto voidaan kuvata UML:n tapahtumasekvenssikaaviolla tai toimintokaaviolla. ( Haikala & Mikkonen 2011, 77 – 78.)



Kuva 17. Selkeä esimerkki käyttötapauskaaviosta (Haikala & Mikkonen 2011, 77)

### 3.3.1 Käyttötapausten kuvaus

Käyttötapausten tyypillinen kuvaustapa on esitetty kuvassa 18. Käyttötapausten tärkein osa on kuvaus, jossa kerrotaan, mitä käyttötapausten pitäisi oikein tehtynä tapahtua eli kuvataan onnistunut suoritus. Ongelmatilanteet voi merkitä esimerkiksi, hakasulkeisiin tarinaan [Esitietovaatimukset eivät täyty]. Poikkeusten käsittely voidaan esittää itse käyttötapausten tai siihen muussa liittyvässä tekstissä. Käyttötapausten kuvausta ei ole mitenkään standardisoitu UML:ssä, joten siihen ei ole oikeaa tai väärää tapaa. Ei-toiminnallisten vaatimusten dokumentointia ei ole myöskään mitenkään standardisoitu UML:ssä. Niitä voi haluttaessa kirjata esimerkiksi *Muut ominaisuudet* yhteyteen käyttötapausten kanssa. (Haikala & Mikkonen 2011, 79.)

**Nimi:** Ilmoittaudu kurssille.

**Versiohistoria:** versio 1.0/ijh.

**Osallistujat:** Opiskelija

**Tuloehdot:** Opiskelija, kurssi ja opintojakso on syötetty järjestelmään, kurssille ilmoittautuminen on avattu, opiskelija on kirjautuneena järjestelmään (KT:t Päivitä perustietoja, Päivitä kurssin tietoja)

**Kuvaus:** Opiskelija seuraa WWW-linkkiä, joka johtaa kurssin sivulle. Hän valitsee vaihtoehdon ilmoittaudu. Järjestelmä tarkastaa, että opiskelijalla on tarvittavat esitiedot [Poikkeus: esitietovaatimukset eivät täyty]. Järjestelmä varmistaa vielä OK/CANCEL-kyselyllä, että opiskelija todella haluaa ilmoittautua. Tämän jälkeen järjestelmä lisää opiskelijan kurssin osallistujaksi.

**Poikkeukset:** Esitietovaatimukset eivät täyty: opiskelijalle annetaan luettelo puuttuvista esitietokursseista.

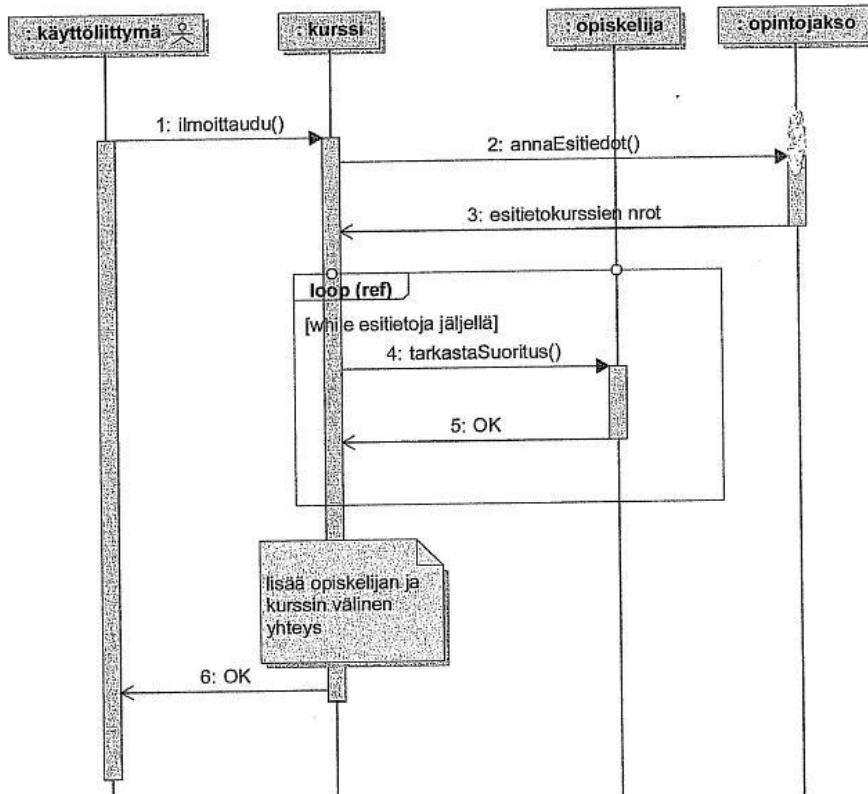
**Lopputulokset:** Opiskelija on rekisteröity kurssin osallistujaksi.

**Muut vaatimukset:** Päivittäin käsitellään kiireisimpänäkin tuntina enintään noin 50 varausta. Vastausajan on aina oltava alle 5 sekuntia.

Kuva 18. Yksi mahdollisuus käyttötapausten kuvaamisesta (Haikala & Mikkonen 2011, 80)

Käyttötapausten toimintoa havainnollistetaan tapahtumasekvenssikaaviolla (*communication diagram, sequence diagram*), jossa näytetään oliot ja kuvataan olioiden välinen kommunikointi. Kaaviossa näkyvät nuolet ilmoittavat missä järjestyksessä metodit eli toiminnot etenevät, pystyviivoilla (*lifeline*) kuvataan elämänlanka, joka on joukko kommunikoivia oliota. Kaaviota sanotaan myös skenaarioksi, koska skenaario on yleisnimitys tapahtumasarjan kuvaukselle, ja tapahtumasekvenssikaavio (kuva 19.) on yksi mahdollinen skenaariokuvaustekniikka. (Haikala & Mikkonen 2011, 97 – 100.)





Kuva 19. Tapahtumasekvenssikaavio (Haikala & Mikkonen 2011, 98)

### 3.3.2 Käyttäjätarina

Käyttäjätarina (*user story*) on tapa, jolla dokumentoidaan vaatimuksia ketterissä menetelmissä. Yksi järjestelmän toiminnoista kuvataan lauseena, jonka asiakas ymmärtää, joten teksti ei sisällä teknistä termistöä. Käyttäjätarina kuvaa kuka tekee (aktori) toiminnon, mitä hän tekee toiminnossa sekä joskus kuvataan mitä lisäarvoa tästä toiminnon tekemisestä on aktorille. Esimerkki käyttäjätarinasta:

*"Kurssin vastuuhenkilönä (aktori) pystyn tekemään kaikki yhden kurssin luentosalivaraukset yhdellä varausoperaatiolla silloin kun luentoajat ovat koko kurssin ajan samoina viikonpäivinä samaan kellonaikaan (mitä tehdään). Tämä säästää paljon työtä verrattuna nykyiseen ratkaisuun, jossa tilan joutuu varaamaan jokaista opetustapahtumaa varten erikseen (lisäarvo)."*

Scrumissa tuotteen työlliställä olevat käyttäjätarinat puretaan pyrhdyksen tehtäviksi työlliställe. (Haikala & Mikkonen 2011, 83 – 84.)

## 4 Opinnäytetyöprojektin organisointi ja vaiheet

Tässä luvussa kerrotaan opinnäytetyön vaiheet. Aluksi käydään läpi projektin organisointi ja siihen liittyvät asiat. Seuraavana kerrotaan opinnäytetyön vaiheista. Lopuksi kerrotaan työn riskit ja ongelmat.

### 4.1 Projektin organisointi ja suunnittelu

Kymecon Oy urakkatarjouslaskenta- ja kustannusseurantajärjestelmäprojekti alkoi syyskuussa 2011 opettaja Jouni Könösen ohjauksessa. Projektin tekijät Sami Anttonen ja minä teimme projektia niin kauan ennen kuin aloitimme opinnäytetyömme. Opinnäytetyöprojektissa minä teen toiminnallisen määrittelyn järjestelmästä Martti Ylä-Jussilan ohjauksessa. Asiakkaan edustajana toimi Mikko Värjä, joka on Kymecon Oy:n toimitusjohtaja. Hän kertoi järjestelmän vaatimukset. Opinnäytetyön projektisuunnitelma oli ensimmäinen vaihe tämän projektin aloituksessa ja se aloitettiin marraskuussa 2012.

Henkilö	Tehtävät ja vastuu	
Martti Ylä-Jussila	Opinnäytetyön ohjaaja, konsultti	Projektin ohjaus ja konsultointi
Mikko Värjä	Kymecon Oy:n toimitusjohtaja, asiakas	Tavoitteiden ja vaatimusten esittäminen
Laura Maarnela	Toiminnallisen määrittelyn toteuttaja, projektipäällikkö	Suunnittelu, toteutus, toiminnallinen määrittely

Taulukko 1. Toiminnallisen määrittelyn organisaatio

## 4.2 Projektisuunnitelma

Projektin suunnittelu aloitettiin tekemällä projektisuunnitelma, jossa ilmenivät projektin tarkoitus, tavoitteet, tehtävät, aikataulu ja tulokset. Projektisuunnitelma hyväksyttiin ensimmäisessä tapaamisessa asiakkaan kanssa, johon osallistui koko projektiorganisaation henkilöt eli toiminnallisen määrittelyn toteuttaja, asiakas ja ohjaava opettaja.

### 4.2.1 Tehtävät, työmäärä arvio ja aikataulu

Projektin tehtävät projektisuunnitelman mukaan, näiden tehtävien työmäärä arvio ja projektin aikataulu on kuvattu taulukossa 2. Projektin edetessä mukaan tehtäviksi tulivat myös jatkokehitysajatuksia ja avoimet asiat. Työmäärä arviosta näkee, kuinka paljon jokaiselle tehtävälle oli alustavasti varattu työtunteja ja tehtävien aikataulu.

Tehtävä	Työmäärä arvio (h)	Aikataulu
Projektisuunnitelma	5	20.11.2012
Johdanto	5	07.01.2012 - 15.02.2013
Yleiskuvaus	20	07.01.2012 - 15.02.2013
Tietosisältö	70	07.01.2012 - 15.04.2013
Käyttötapaukset ja käyttöliittymä	100	07.01.2012 - 15.04.2013
Ulkoiset liittymät	20	22.01.2013 - 15.04.2013
Muut ominaisuudet	20	22.01.2013 - 15.04.2013
Suunnittelurajoitteet	10	22.01.2013 - 15.04.2013
Opinnäytetyö	140	30.01.2012 - 19.04.2013

Taulukko 2. Työmäärä arvio ja aikataulu

## 4.3 Määrittelyn suunnittelu

Määrittelyn aloitin muiden tekemien toiminnallisten määrittelyjen lukemisella, jotta sain paremman käsityksen, mitä toimivan toiminnallisen määrittelyn pitäisi sisältää sekä minkälainen sen tulisi olla. Sain suunniteltua, mitä otsikoita määrittelyssä tulisi olla. Näistä otsikoita tuli tehtävälista, jonka laitoin myös projektisuunnitelmaan tehtäväluetteloksi.

#### 4.4 Toteutus

##### *Määrittelyn aloitus ja eteneminen*

Projektisuunnitelman hyväksymisen jälkeen saatoin aloittaa toiminnallisen määrittelyn kirjoittamisen. Järjestelmästä oli tehty hallintaosio, tarjouksen hinnoittelu ja kassa, ja dokumentoinnissa hyödynsin teknistä raporttia, jonka olimme Sami Anttosen kanssa tehneet. Toiminnallisesta määrittelystä ensimmäiseksi kirjoitin johdannon ja yleiskuvauksen, jonka jälkeen kävin tekemään käyttötapauskäviötä järjestelmän valmiista osasta, josta näki, mitä toimintoja käyttäjän on pysyttävä järjestelmällä tekemään, jotta pääsin kirjoittamaan käyttötapausten kuvauksia. Sen jälkeen katsoin jo järjestelmän käyttämää tietokantakaaviota, josta näin tietokantataulujen attribuutit, jotta pääsin kuvaamaan järjestelmän käyttämien tietosisällön. Käyttötapausten ja tietosisällön kuvaaminen oli aikaa vievää, joten kesti ennen kuin pääsin seuraavaan vaiheeseen eli käyttöliittymän suunnitteluun ja toteutukseen järjestelmän puuttuvasta osuudesta, tavoitearvio ja kustannusseuranta. Tein tästä alustavan käyttöliittymän, jonka jälkeen muistimme, että kustannustenseurantaan kuuluu vielä kustannusarvion muodostaminen ja raportointi. Tämän jälkeen kirjoitin ulkoiset liittymät, muut ominaisuudet ja suunnittelurajoitteet. Näiden jälkeen päätin kirjoittaa vielä jatkokehitysajatuksia ja avoimet asiat, johon voitiin kirjoittaa tuntikortin puuttumisen järjestelmästä, johon ei oteta kantaa tässä toiminnallisessa määrittelyssä, sekä muita avoimia asioita. Projektin uudet tekijät jatkavat projektin toteuttamista ja testaamista omalla aikataulullaan, sekä päivittävät minun tekemää toiminnallista määrittelyä projektin edetessä lehtori Mikko Huhtasen johdolla.

Projektissa pidettiin kerran kuukaudessa ohjausryhmän virallinen palaveri, jossa käytiin läpi edellisen seurantajakson tehdyt työt, projektin eteneminen aikatauluun nähden, riskit ja ongelmat sekä seuraavan jakson tehtävät ennen seuraavaa ohjausryhmän kokousta. Kun pääsin järjestelmän puuttuvan osuuden määrittelyyn, pidimme useita kokouksia asiakkaan kanssa, jotta puuttuvasta osuudesta saatiin vaatimukset kirjattua ylös. Näistä vaatimuksista tein käyttöliittymät järjestelmään, käyttötapaukset ja niiden kuvaukset.

#### 4.5 Määrittelyn katselmointi ja hyväksyminen

Toiminnallista määrittelyä katselmoi opinnäytetyön ohjaava opettaja Martti Ylä-Jussila ja asiakas eli Kymecon Oy:n toimitusjohtaja Mikko Värjä. He yhdessä katsoivat omalla aikataulullaan, että toiminnallinen määrittely vastaa odotuksia ja kaikki on määrittelyssä oikein. Hyväksyminen tapahtuu viimeisessä palaverissa asiakkaan kanssa.

#### 4.6 Riskit ja ongelmat

Taulukossa 3 on kuvattu projektin riskejä.

Nro	Kuvaus	Toden- näköisyys	Seuraus
R1	Sami Anttonen ei jatka järjestelmän toteutusta	50 %	Järjestelmän kehitys viivästyy heti vähintään 1 kuukaudella
R2	Lauran kone rikkoontuu	50 %	Dokumentit ja ohjelmat menetetään

Taulukko 3. Projektin riskit ja ongelmat

R1 eli riski numero 1 toteutui, sillä Sami Anttonen ei ollut enää mukana toteuttamassa järjestelmää. Martti Ylä-Jussila etsi kuitenkin järjestelmälle neljä kappaletta uusia koodaajia, jotka jatkavat järjestelmän toteuttamista ja testaamista.

## **5 Lopputuloksen esittely**

Lopputuloksen esittelyssä esitetään määrittelydokumentin sisältöluettelo, järjestelmän tietokantakaavio, työntekijästä tietosisällön kuvaus, käyttötapauskaavio, Sisäänkirjautuminen - käyttötapauksen kuvaus sekä järjestelmän pääsivusta, tarjouksen laskennasta ja kustannustenseurannasta käyttöliittymälomake ja lyhyt sanallinen kuvaus niiden toiminnasta (kuva 20.).

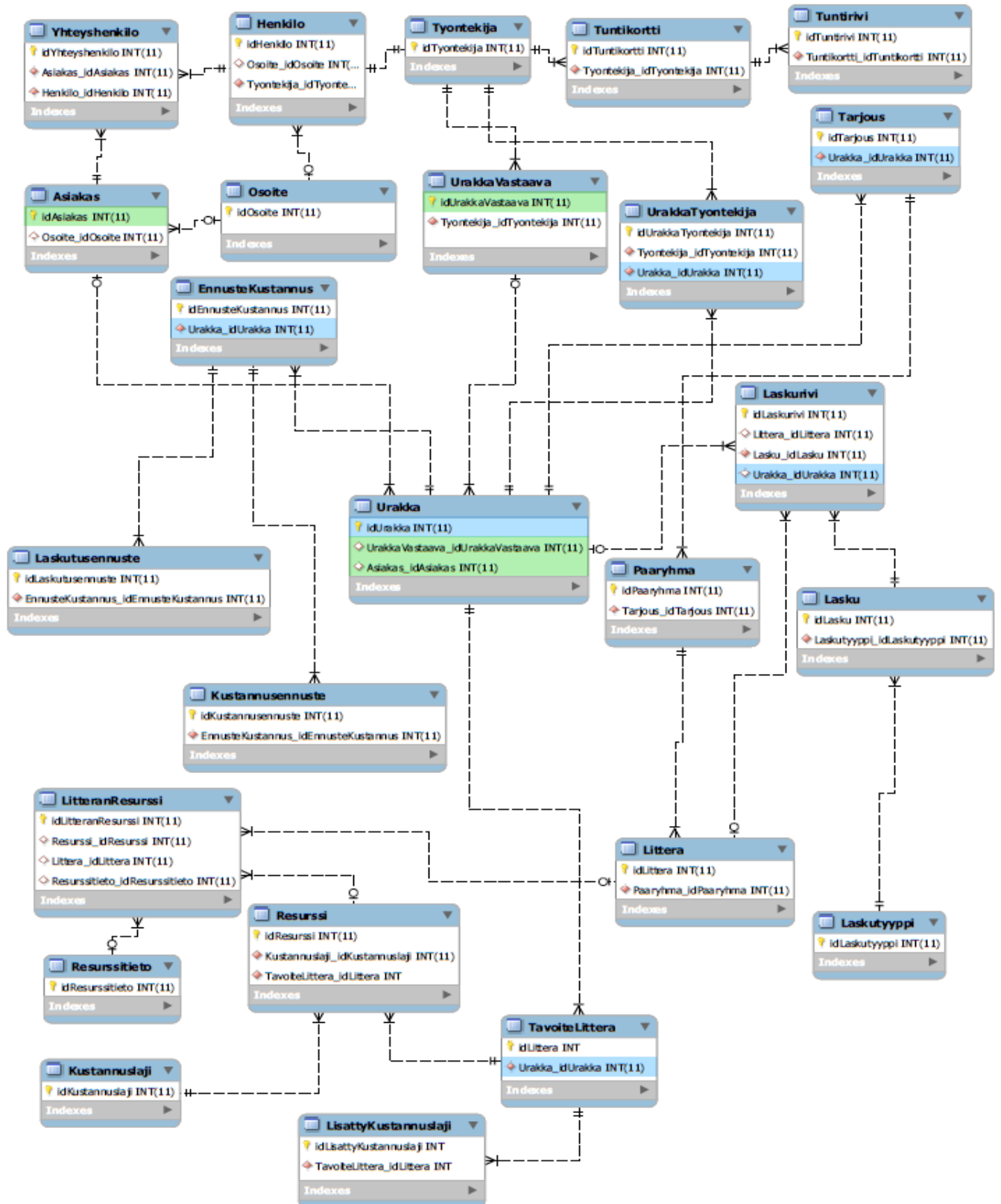
## Sisältö

1	Johdanto.....	6
1.1	Tarkoitus ja kattavuus.....	6
1.2	Tuotekuvaus.....	6
1.3	Tuoteympäristö.....	6
1.4	Määritelmät, termit, lyhenteet.....	7
1.5	Viitteet.....	8
1.6	Yleiskatsaus dokumenttiin.....	9
2	Yleiskuvaus.....	10
2.1	Kymecön Oy.....	10
2.2	Rakennushanke.....	10
2.3	Kymecön Oy:n tarjous – tilaus – toimitus – prosessi.....	13
2.4	Järjestelmän toiminta.....	16
2.5	Käyttäjät.....	20
2.6	Yleiset rajoitteet.....	20
2.7	Oletukset ja riippuvuudet.....	20
3	Järjestelmän tietosisältö.....	21
3.1	Tietosisältö.....	22
4	Käyttötapaukset ja käyttöliittymä.....	34
4.1	Käyttöliittymän yleiskuvaus.....	35
4.2	Käyttötapausten kuvaus.....	39
4.2.1	Sisäänkirjautuminen.....	39
4.2.2	Tarjouksen hinnoittelu.....	41
4.2.3	Pääryhmän lisääminen.....	42
4.2.4	Litteran lisääminen.....	45
4.2.5	Resurssin lisääminen.....	46
4.2.6	Tarjouksen yhteenvetotietojen lisääminen.....	47
4.2.7	Raporttien esikatselu/tulostus.....	48
4.2.8	Tarjousten hallinta.....	54
4.2.9	Urakoiden hallinta.....	56
4.2.10	Asiakkaiden hallinta.....	58
4.2.11	Työntekijöiden hallinta.....	60
4.2.12	Tavoitearvion hallinta.....	62
4.2.13	Kassanseurantaa.....	66
4.2.14	Kustannusseuranta.....	71
4.2.15	Raportointi.....	74
5	Ulkoiset liittymät.....	77
5.1	Laitteistoliittymät.....	77
5.2	Ohjelmistoliittymät.....	77
5.3	Tietoliikenneliittymät.....	77
6	Muut ominaisuudet.....	78
6.1	Käytettävyys.....	78
6.2	Turvallisuus ja suojaukset.....	78
6.3	Ylläpidettävyys.....	78
6.4	Siirrettävyys ja yhteensopivuus.....	79
6.5	Operoitavuus.....	79
7	Suunnittelurajoitteet.....	80
7.1	Laitteistorajoitteet.....	80
7.2	Ohjelmistorajoitteet.....	80
8	Jatkokehitysjatoksia.....	80
9	Avoimet asiat.....	80
9.1	Työtuntien seuranta.....	80
9.2	Peruuta - painike.....	80
9.3	Kustannusennuste.....	80
9.4	Tapahtumaloki.....	81
	Liitteet.....	81
	Liite 1 Tietokantakaavio.....	81

Kuva 20. Sisällysluettelo valmiista toiminnallisesta määrittelydokumentista

Tietosisällön tiedot ovat tietokantataulujen attribuutteja, ja jokainen tietokannan taulu kuvataan erikseen toiminnallisessa määrittelyssä. Esimerkkinä tietosisällön kuvauksessa käytetään Työntekijä-taulun tietoja. Ennen Työntekijä-taulun tietojen kuvausta on lyhyt sanallinen kuvaus taulusta, jossa kuvataan, mitä virkaa taululla on eli miksi se on tehty. (Kuva 21.)





Kuva 21. Tietokantakaavio

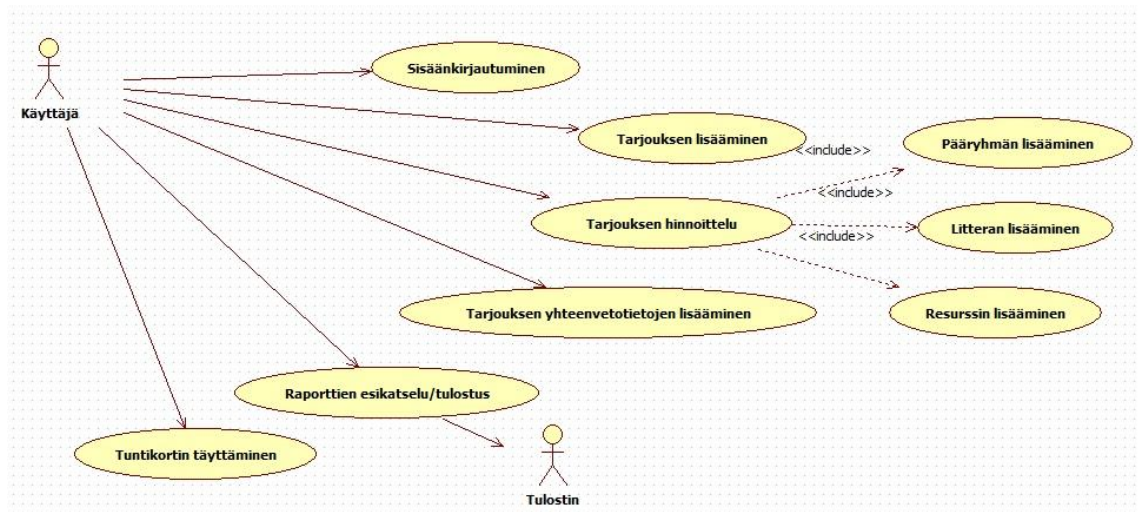
Työntekijä on Kymecon Oy:ssä työskentelevä henkilö, jolle maksetaan palkkaa tehdystä työstä. Työntekijä ei itse käytä järjestelmää, joten tiedot järjestelmään syöttää käyttäjä. Työntekijälle voi kuitenkin antaa käyttäjätunnuksen ja salasanan, jos työntekijän halutaan pääsevän sisään järjestelmään.

Työntekijä = @työntekijä\_id + tehtävä + tuntipalkka + työkalukorvaus + työjohtolisä + käyttäjätunnus + salasana + poistettu (Taulukko 4)

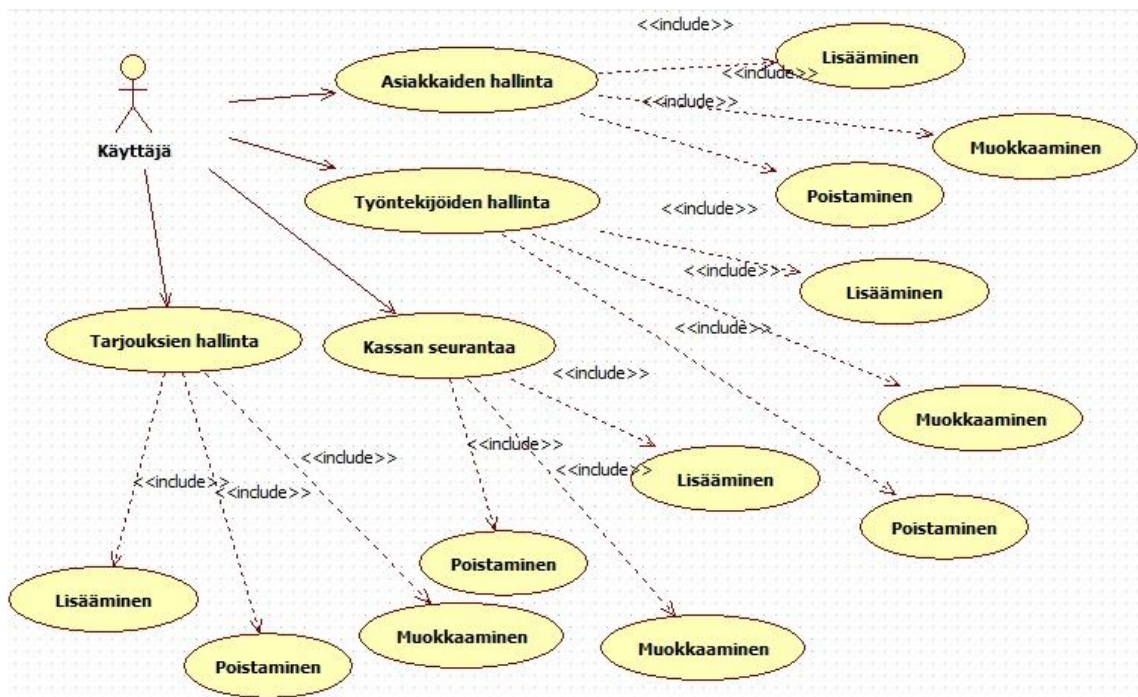
Tieto	Tyyppi	Selite
Työntekijä_id	@/	Perusavain, taulukon avainominaisuus.
Tehtävä	3{M}100	Työntekijän työtehtävä Esim. laatoittaja
Tuntipalkka	2{M}6	Työntekijän tuntipalkka Esim. 15,02 (euroa)
Työkalukorvaus	2{M}6	Työntekijän työkalukorvaus Esim. 12,00 (euroa)
Työjohtolisä	2{M}6	Työntekijän työjohtolisä Esim. 10,00 (euroa)
Käyttäjätunnus	3{M}15	Työntekijän henkilökohtainen käyttäjätunnus. Esim. matmei
Salasana	6{M}15	Työntekijän henkilökohtainen salasana. Esim. mattiM
Poistettu	{N}	Onko työntekijä poistettu vai ei 0 = Ei poistettu 1 = Poistettu

Taulukko 4. Työntekijän tiedot

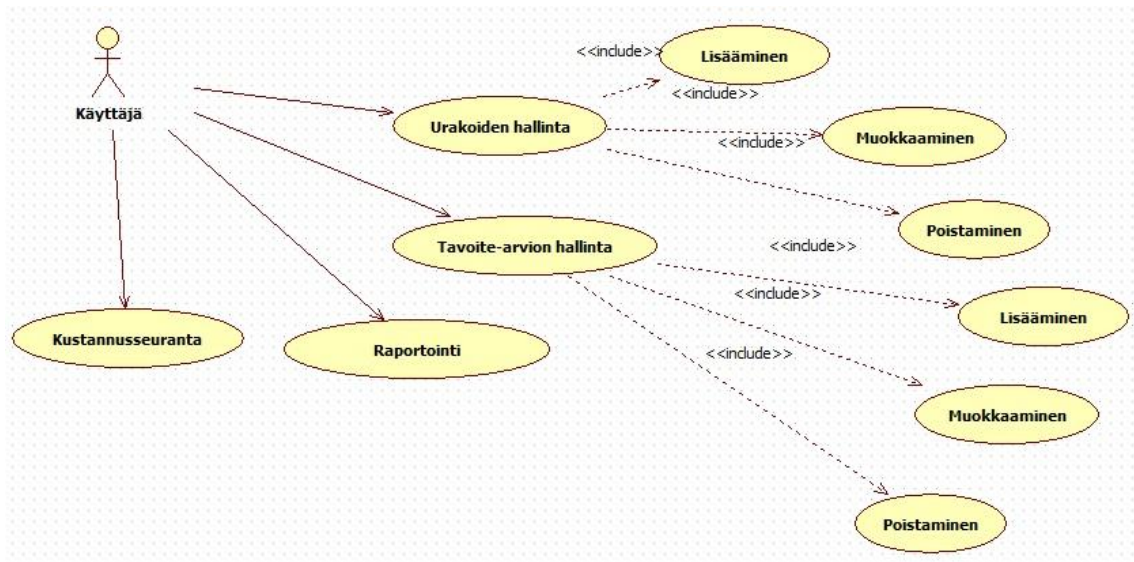
Käyttötapauskaavio piti tehdä, jotta nähdään, mitä toimintoja järjestelmässä pitää olla, ja mitä toimijoita järjestelmällä on. Kaavion luomisen jälkeen jokainen käyttötapaus kuvataan. Käyttötapausten tärkein osa on kuvaus, jossa kerrotaan mitä käyttötapauksessa pitäisi oikein tehtynä tapahtua eli kuvataan onnistunut suoritus. Havainnollistamaan kuvausta voidaan lisätä kuva käyttäjästä ennen tai jälkeen kuvausta. (Ks. kuvat 22 – 24.)



Kuva 22. Käyttötapauskaavio 1



Kuva 23. Käyttötapauskaavio 2



Kuva 24. Käyttötapauskaavio 3

Esimerkkinä käyttötapausten kuvauksesta esitetään Sisäänkirjautuminen (kuva 25.).



Kuva 25. Järjestelmän Sisäänkirjautumisen-ikkuna

<b>Nimi</b>	Sisäänkirjautuminen
<b>Kuvaus</b>	Käyttäjä kirjautuu käyttäjätunnuksella ja salasanalla sisään järjestelmään.
<b>Toimijat</b>	Toimijoita ei ole tässä käyttötapauksessa muita kuin käyttäjä.
<b>Liittyvät käyttötapaukset</b>	Muita käyttötapauksia ei liity kyseiseen käyttötapaukseen.
<b>Alkutila ja alkuehdot</b>	Työntekijällä on sisäänkirjautumiseen tarvittava käyttäjätunnus ja salasana.

## **Käyttötapausten kulku**

1. Käyttäjä käynnistää järjestelmän, jolloin ensimmäiseksi aukeaa sisäänkirjautumisnäkyvä.
2. Käyttäjä syöttää käyttäjätunnuksen ja salasanan, jonka jälkeen klikataan Kirjaudu-painiketta.
3. Järjestelmä tarkistaa, että käyttäjätunnus ja salausna ovat molemmat oikein. Samalla järjestelmä luo yhteyden ja hakee tietoja tietokannasta (virhe 1).
4. Käyttötapausta päättyy, kun tunnukset on todettu oikeiksi. Tämän jälkeen aukeaa järjestelmän päälomake.

Jos käyttäjä antaa virheellisen käyttäjätunnuksen tai salasanan, järjestelmä ei päästä kirjautujaa luomaan yhteyttä ja hakemaan tietoa tietokannasta eikä laske kirjautujaa sisään järjestelmään.

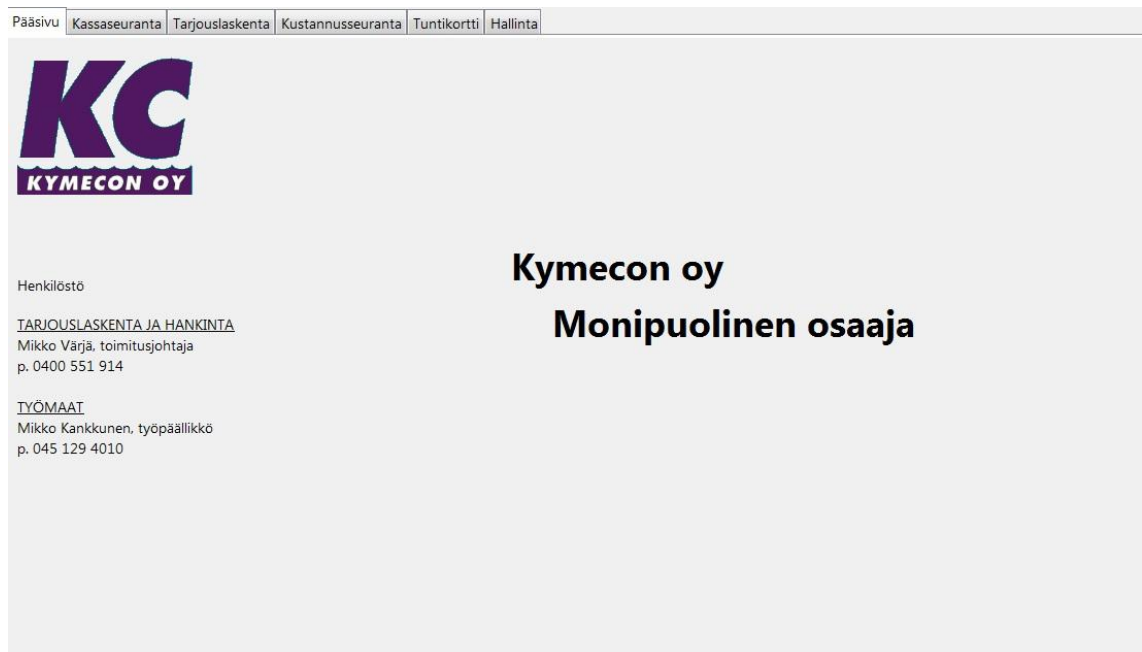
### **Virhe 1**

Jos Internet-yhteys ei ole kunnossa, niin järjestelmä ei pysty sisäänkirjautuessa luomaan yhteyttä tietokantaan. Järjestelmä ilmoittaa sisäänkirjautumisen epäonnistuessa.

### **Lopputulos**

Käyttäjä on kirjautunut sisään järjestelmään ja hänellä on näkyvillä sovelluksen päälomake.

Sisäänkirjautumisen jälkeen aukeaa kuvassa 26 esitetty järjestelmän pääsivu, jolta käyttäjä valitsee halutun välilehden ja klikkaa sitä.



Kuva 26. Järjestelmän pääsivu

Hallintavälilehdellä käyttäjä hallinnoi työntekijöitä, asiakkaita, tarjouksia ja urakoita. Tuntikorttivälilehdeltä pääsee tuntikorttiin, jossa voidaan esimerkiksi syöttää työntekijälle tehtyjä työtunteja. Kustannusseurannassa käyttäjä tekee urakasta tavoite-arvion ja seuraa kustannuksia vertaamalla toteutuneita kustannuksia tavoitekustannuksiin sekä tulostaa raporteja. Tarjouslaskennassa käyttäjä laskee tarjouksen. Kassaseurannassa lisätään laskuja järjestelmään ja seurataan kassan tilannetta päivittäin.

Tarjouksen laskenta on yksi järjestelmän tärkeimmistä toiminnoista (kuva 27).

Tunnus	Nimi	Määrä	Yks	€/Yks	Yhteensä
		XXX	XXX	XXXX	XXXX


  

Tunnus	Nimi	Kaava	Määrä	Yksikkö	e/Yks	Yhteensä
--------	------	-------	-------	---------	-------	----------

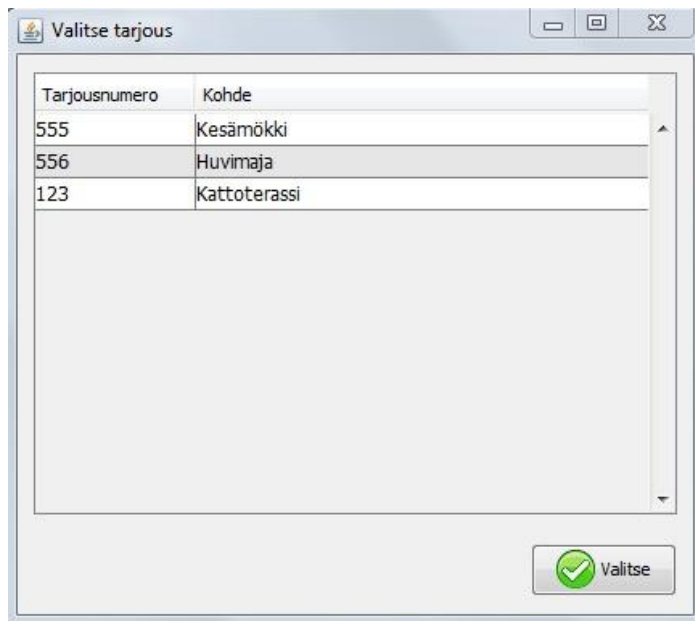
  

Nimi	Klaji	Määrä	Yks	Teho	Menekki	A-hinta	€/Yks	Yhteensä
------	-------	-------	-----	------	---------	---------	-------	----------

Kuva 27. Tarjouksen hinnoittelun näkymä

Tarjouslaskennassa käyttäjä hinnoittelee valitsemansa tarjouksen ja lisää tarjoukselle pääryhmät, litterat ja resurssit. Tarjouksen hinnoittelu aloitetaan lisäämällä uusi pääryhmä.  - kuvakkeesta klikkaamalla käyttäjälle aukeaa kuvassa 29 esitetty ikkuna, josta käyttäjä pääsee lisäämään, muokkaamaan ja poistamaan pääryhmiä, eli pääryhmän hallintaan. Jos käyttäjä ei ole valinnut hinnoiteltavaa tarjousta ennen pääryhmän lisäämistä, niin käyttäjälle avautuu kuvassa 28 esitetty näkymä eli Valitse tarjous -ikkuna. Valitse tarjous -kuvassa näkyvät esimerkit tarjouksesta tai sen kohteesta ovat ihan pelkkiä esimerkkejä, tämä ikkuna ei voi koskaan olla tyhjä vaan aina on oltava tarjous, jota käyttäjä lähtee hinnoittelemaan. Jos käyttäjä ei ole lisännyt järjestelmään tarjousta niin, ei voi myöskään hinnoitella tarjousta.










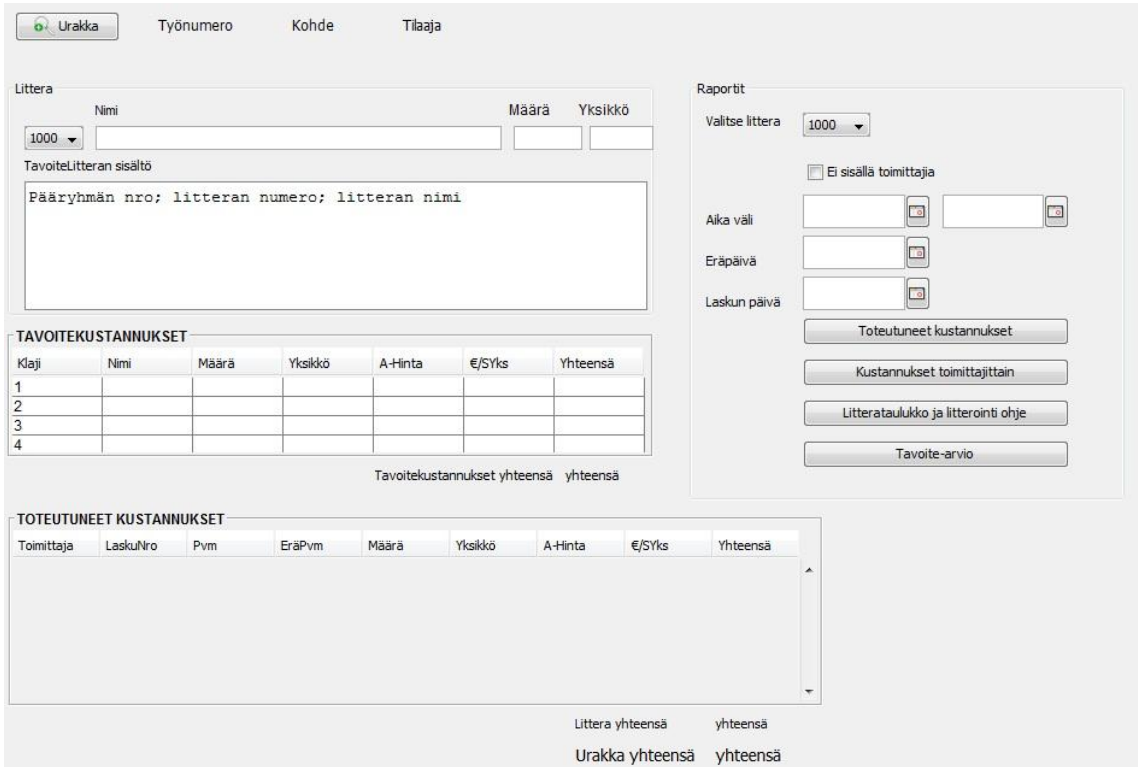
Kuva 28. Valitse tarjous - ikkuna



Kuva 29. Pääryhmien hallinta - ikkuna

Pääryhmän lisäämisen jälkeen käyttäjä lisää pääryhmälle litteroita halutun määrän klikkaamalla  - kuvakkeesta, jolloin ilmestyy uusi tyhjä littera rivi, jolle käyttäjä pääsee syöttämään tietoja.  - kuvakkeesta klikkaamalla käyttäjä poistaa valitun litteran, jolloin tältä litteralta poistetaan myös kaikki sen sisältämät resurssit. Litteran lisäämisen jälkeen käyttäjä syöttää litteralle resursseja halutun määrän klikkaamalla  - kuvaketta, jolloin ilmestyy uusi resurssirivi jolle käyttäjä pääsee syöttämään tietoja.  - kuvakkeesta klikkaamalla käyttäjä poistaa valitun resurssin.  - kuvakkeesta klikkaamalla käyttäjä tallentaa kaikki lisätyt, muokatut ja poistetut pääryhmät, litterat ja resurssit tietokantaan.

Kustannusseuranta on yksi järjestelmän toiminnoista (kuva 30).



The screenshot shows a web application interface for cost tracking. At the top, there are tabs for 'Urakka' (selected), 'Työnumero', 'Kohde', and 'Tilaja'. The main area is divided into several sections:

- Littera:** A form with fields for 'Nimi', 'Määrä', and 'Yksikkö'. A dropdown menu shows '1000'. Below is a text area for 'Tavoitelitteran sisältö' containing 'Pääryhmän nro; litteran numero; litteran nimi'.
- TAVOITEKUSTANNUKSET:** A table with columns: Klaji, Nimi, Määrä, Yksikkö, A-Hinta, €/SYks, Yhteensä. It has 4 rows.
- TOTEUTUNEET KUSTANNUKSET:** A table with columns: Toimittaja, LaskuNro, Pvm, EräPvm, Määrä, Yksikkö, A-Hinta, €/SYks, Yhteensä. It is currently empty.
- Raportit:** A sidebar with a dropdown for 'Valitse littera' (set to 1000), a checkbox for 'Ei sisällä toimittaja', and date pickers for 'Aika väli', 'Eräpäivä', and 'Laskun päivä'. Below are buttons for 'Toteutuneet kustannukset', 'Kustannukset toimittajittain', 'Litterataulukko ja litterointi ohje', and 'Tavoite-arvio'.

At the bottom right, there are summary labels: 'Littera yhteensä yhteensä' and 'Urakka yhteensä yhteensä'.

Kuva 30. Kustannusseurannan näkymä



malla niiden painikkeista. Toteutuneet kustannukset –raportin (kuva 35.) käyttäjä voi rajata litteran, aikaväli, eräpäivän ja/tai laskupäivän mukaan. Raportti voi myös sisältää toimittajat jos käyttäjä klikkaa merkin Sisältää toimittajat valintaan. Kolmea muuta raporttia (kuva 32, 33 ja 34) ei voi rajata litteralla.

KYMECON OY		KUSTANNUKSET TOIMITTAJITTAIN								sivu	
										26.3.2013	1/3
										11:44	
Toimittaja	Laskunro	LaskuPvm	MaksuPvm	Selite	Määrä	Yks	Ahinta	Yhteensä	Littera		
K-rauta	125	15.3.2013	31.3.2013		2 kpl		1 500,00	3 000,00	9000 TAVOITELITTERAN NIMI		
K-rauta	865	15.3.2013	31.3.2013					1 500,00	9600 TYÖMAA		
K-rauta	215	15.3.2013	31.3.2013					3 540,00	1000 SEINÄN PURKU		
K-rauta	569	15.3.2013	31.3.2013					567,00	2000 KATON KORJAUS		
				K-rauta				<b>8 607,00</b>	5000 TAVOITELITTERAN NIMI		
								<b>12 423,00</b>	1000 SEINÄN PURKU		
Starkki	2565	20.5.2013	30.5.2013					156,00	2000 KATON KORJAUS		
Starkki	5989	20.5.2013	30.5.2013					542,00	3000 TAVOITELITTERAN NIMI		
Starkki	5347	20.5.2013	30.5.2013					5 623,00	5000 TAVOITELITTERAN NIMI		
Starkki	2327	20.5.2013	30.5.2013								
				Starkki				<b>18 744,00</b>			
								<b>27 351,00</b>			
								<b>Kustannukset yhteensä</b>	<b>27 351,00</b>		

Kuva 32. Kustannukset toimittajittain- raportti

KYMECON OY		LITTERATAULUKKO					sivu	
							26.3.2013	1/2
							11:44	
145 Mökki								
Littera	Nimi						Määrä	Yks
1000	SEINÄN PURKU						1	ERÄ
	Litteranro	Litteranimi	Kaava	Litteramäärä	Litterayksikkö			
	110	sora	5+5+5	1,5	m <sup>2</sup>			
2000	KATON KORJAUS						1	ERÄ
	Litteranro	Litteranimi	Kaava	Litteramäärä	Litterayksikkö			
	214	sora	5+5+5	1,5	m <sup>2</sup>			
9600	TYÖMAA						1	KK
	960	koppi						
	961	työkalut						

Kuva 33. Litterataulukko- raportti

KYMECON OY		TAVOITEARVIO		sivu		1/1	
				26.3.2013		11:44	
145 Mökki							
Littera Nimi		Määrä Yks		Yhteensä			
1000 SEINÄN PURKU		1 ERÄ		8 900,00			
	1 Miestyö	h		150	30,00	4 500,00	
	2 Materiaali	h		30	20,00	600,00	
	3 Alihankinta					600,00	
	4 Konetyö	h		80	40,00	3 200,00	
2000 KATON KORJAUS		1 ERÄ		19 960,00			
	1 Miestyö	h		230	80,00	18 400,00	
	3 Alihankinta					1 560,00	
9600 TYÖMAA		1 KK		7 000,00			
	1 Miestyö	h		200	35,00	7 000,00	
				<b>Tavoitearvio yhteensä</b>		<b>35 860,00</b>	

Kuva 34. Tavoitearvioraportti

KYMECON OY		TOTEUTUNEET KUSTANNUKSET LITTEROITAIN		sivu		1/3		
				26.3.2013		11:44		
Toimittaja	Nro	Pvm	EräPvm	Selite	Määrä	Yks	Ahinta	Yhteensä
1000	SEINÄN PURKU				1 ERÄ			88 465,00
K-rauta	565	26.3.2013	31.3.2013					1 320,00
K-rauta	268	26.3.2013	31.3.2013		2 kpl		1 500,00	3 000,00
K-rauta	523	26.3.2013	31.3.2013					16 516,00
K-rauta	725	26.3.2013	31.3.2013					565,00
K-rauta	72	26.3.2013	31.3.2013					25,00
K-rauta	8625	26.3.2013	31.3.2013					52,00
K-rauta	726	26.3.2013	31.3.2013					62,00
K-rauta	7257	26.3.2013	31.3.2013					62 689,00
Starkki	267	26.3.2013	31.3.2013					365,00
Starkki	573	26.3.2013	31.3.2013					259,00
Starkki	67	26.3.2013	31.3.2013					234,00
Starkki	65	26.3.2013	31.3.2013					526,00
Starkki	27	26.3.2013	31.3.2013					2 852,00
2000	KATON KORJAUS				1 ERÄ			1 027,00
Starkki	258	26.3.2013	31.3.2013					654,00
K-rauta	2354	26.3.2013	31.3.2013					247,00
Tuntipalkka	2000	26.3.2013	31.3.2013		58 h			126,00
9600	TYÖMAA				1 KK			7 076,00
Mittausta	152	26.3.2013	31.3.2013					1 563,00
Mittausta	452	26.3.2013	31.3.2013					5 513,00
				<b>Yhteensä</b>				<b>96 568,00</b>

Kuva 35. Toteutuneet kustannukset litteroittain-raportti

## 6 Pohdintaa

Projektin tavoitteena oli saada asiakkaalle, Kymecon Oy:lle, kehitteillä olevasta järjestelmästä toiminnallinen määrittely, jossa kuvataan kaikki järjestelmän toteuttamat toiminnot, toteutukselle asetettavat ei-toiminnalliset vaatimukset sekä rajoitukset ja kaikki liitännät järjestelmän ulkopuolelle. Toisena tavoitteena oli, että määrittelydokumentti olisi ollut järjestelmän käyttöohje.

Toiminnallisen määrittelyn suhteen projekti saavutti tavoitteensa, mutta käyttöohjeen suhteen ei, koska määrittely ei ole hyvä käyttöohje, vaan käyttöohje olisi kannattanut tehdä erikseen. Lopullisen käyttöohjeen voi tehdä vasta, kun järjestelmä on lähestulkoon valmis. Haikalan mukaan minimidokumentaatioon kuuluu alustava käyttöohje ja toiminnallinen määrittely, joka sopii tähän kuvaukseen erittäin hyvin.

Toiminnallisesta määrittelystä on hyötyä seuraaville projektin tekijöille, sillä määrittelydokumentin luettuaan uudet projektin tekijät tietävät, miksi järjestelmää on lähdetty tekemään, mitä sillä tehdään, miten järjestelmä toimii sekä sen liitännät ulkopuolelle.

Koska projektin ensimmäisillä tekijöillä ei ollut aikaisempaa käytännön kokemusta toiminnallisen määrittelydokumentin teosta, niin määrittely päätettiin jättää tekemättä ja edetä ketterää kehitysmenetelmää käyttäen. Kuitenkin määrittelydokumentin tekeminen vasta projektin puolivälissä oli erittäin haastavaa, koska osa vaatimuksista oli jo unohtumassa. Palavereissa ilmenevät asiat, kuten uudet vaatimukset ja muutokset määrittelyyn, kannattaa kirjoittaa heti ylös, sillä Haikalan mukaan niitä ei muista viikkoa kauempaa.

Urakkatarjouslaskenta- ja kustannusseurantajärjestelmän tekeminen rakennusyritykselle oli haastavampaa kuin olisin aluksi uskonut. Termit ovat outoja ja liiketoimintaprosessi täytyy ymmärtää alusta loppuun ja olenkin sitä mieltä, että projektissa olisi pitänyt olla mukana myös rakennuspuolen opiskelija. Asiakkaalla ja minulla itselläni ei ollut aikaisempaa kokemusta järjestelmän suunnittelusta ja toteuttamisesta.

Järjestelmää tehtiin ketterällä XP-menetelmällä, jossa dokumentointi oli vähäistä. Toiminnallista määrittelyä kirjoittaessa huomasi, että näistä puuttuvista dokumenteista olisi ollut hyötyä kirjoitettaessa järjestelmän erilaisista vaatimuksista, joista silloin syyskuussa 2011 oli asiakkaan kanssa puhetta. Nyt kun projektin tiimin koko on kasvanut, niin dokumentointia kannattaisi tehdä enemmän ja kaikkien dokumenttien tulisi olla kaikkien tiimiläisten saatavilla. Kunnollisella dokumentoinnilla säästetään aikaa, koska jokaiselle tiimin jäsenelle ei tarvitse selittää samoja asioita erikseen.

Kaikkia haluttuja toimintoja ei ajan puutteen vuoksi ehtinyt toiminnalliseen määrittelyyn tekemään, sillä projektissa on vielä useita satoja työtunteja jäljellä. Nämä puuttuvat toiminnot lisättiin määrittelyn yhdeksänteen lukuun, avoimet asiat. Järjestelmään jäi vielä määrittelemättä tapahtumaloki, kustannusennuste, työtuntien seuranta ja Peru-toiminto. Esimerkiksi työtuntien seuranta, joka on olennainen osa järjestelmää, jää kokonaan uusien tekijöiden tehtäväksi. Käyttöliittymien luomisessa käytettiin NetBeans-ohjelmaa, jolla itse järjestelmää koodataan, mutta käyttöliittymien tekemiseen kyseinen ohjelma ei minusta ollut hyvä. Ohjelma heittää komponentteja käyttöliittymä näkymässä pois omilta paikoiltaan useaan kertaan ja niiden takaisin vetämiseen kuluu aikaa, jonka voisi käyttää tehokkaammin ja järkevämmiin.

Toiminnallista määrittelyä täytyy päivittää projektin edetessä ja määrittelydokumenttia päivittää uuden projektiporukan projektipäällikkö Kimmo Viinanen.

## Kuvat

- Kuva 1. Rakennushankkeen vaiheet, s. 7
- Kuva 2. Rakennushankkeen vaiheet ja osapuolet, s. 8
- Kuva 3. Rakennusvaiheen työnkulku, s. 12
- Kuva 4. Ohjelmistotuotannon osa-alueet, s. 14
- Kuva 5. Testauksen V-malli, s. 16
- Kuva 6. Vesiputousmalli, s. 19
- Kuva 7. Prototyypimalli, jossa järjestelmä toteutetaan alusta, s. 20
- Kuva 8. Prototyypimalli, jossa prototyyppi kehitetään valmiiksi järjestelmäksi, s. 20
- Kuva 9. Spiraalimalli, s. 22
- Kuva 10. RUP-prosessin päävaiheet, s. 23
- Kuva 11. RUP-prosessissa työmäärän jakautuminen, s. 24
- Kuva 12. XP-prosessin kuvaus, s. 28
- Kuva 13. Scrum-prosessi, s. 30
- Kuva 14. Toiminnallisen määrittelyn sisältörunko, s. 35
- Kuva 15. Luokkakaavio, s. 37
- Kuva 16. Tietohakemistonotaatio, s. 38
- Kuva 17. Selkeä esimerkki käyttötapauskaaviosta, s. 39
- Kuva 18. Yksi mahdollisuus käyttötapausten kuvaamisesta, s. 40
- Kuva 19. Tapahtumasekvenssikaavio, s. 41
- Kuva 20. Sisällysluettelo valmiista toiminnallisesta määrittelydokumentista, s. 47
- Kuva 21. Tietokantakaavio, s. 49
- Kuva 22. Käyttötapauskaavio 1, s. 51
- Kuva 23. Käyttötapauskaavio 2, s. 51
- Kuva 24. Käyttötapauskaavio 3, s. 52
- Kuva 25. Järjestelmän Sisäänkirjautumisen-ikkuna, s. 53
- Kuva 26. Järjestelmän pääsivu, s. 55
- Kuva 27. Tarjouksen hinnoittelun näkymä, s. 56
- Kuva 28. Valitse tarjous – ikkuna, s. 57
- Kuva 29. Pääryhmien hallinta – ikkuna, s. 57
- Kuva 30. Kustannusseurannan näkymä, s. 58
- Kuva 31. Kustannusseurannan Valitse urakka –ikkuna, s. 59
- Kuva 32. Kustannukset toimittajittain- raportti, s. 60
- Kuva 33. Litterataulukko- raportti, s. 60
- Kuva 34. Tavoitearvioraportti, s. 61
- Kuva 35. Toteutuneet kustannukset litteroittain-raportti, s. 61

## Taulukot

- Taulukko 1. Toiminnallisen määrittelyn organisaatio, s. 42
- Taulukko 2. Työmäärä arvio ja aikataulu, s. 43
- Taulukko 3. Projektin riskit ja ongelmat, s. 45
- Taulukko 4. Työntekijän tiedot, s. 50



## Lähteet

- Agile Manifesto. 2001. Ketterän ohjelmisto kehityksen periaatteet.  
<http://www.agilemanifesto.org/iso/fi/principles.html>. Luettu 18.2.2013.
- Haikala, Ilkka & Mikkonen, Tommi. 2011. Ohjelmistotuotannon käytännöt.  
Helsinki: Talentum Oyj.
- Haikala, Ilkka & Märijärvi, Jukka. 2004. Ohjelmistotuotanto. Helsinki: Talentum Oyj.
- Hypermedian opetus. 2011. Ketterät menetelmät, Scrum.  
<http://hlab.ee.tut.fi/hmopetus/vpsist-oppimateriaali/4-menetelmia-ja-malleja/4-4-ketterat-menetelmat/4-4-1-scrum>. Luettu 5.3.2013.
- Immonen, Jarkko. 2003. Johdatus ohjelmistotuotantoon. Joensuun Yliopisto.  
[http://cs.joensuu.fi/~jimmonen/jot\\_moniste/jot\\_moniste\\_121.html](http://cs.joensuu.fi/~jimmonen/jot_moniste/jot_moniste_121.html).  
Luettu 7.2.2013.
- Kankaanpää, Timo. Ohjelmistomäärittely.  
[http://www.cc.puv.fi/~tka/kurssit/Ohjelmiston\\_maarittely/luennot.htm](http://www.cc.puv.fi/~tka/kurssit/Ohjelmiston_maarittely/luennot.htm).  
Luettu 30.1.2013.
- Laine, Harri & Paakki, Jukka. Ohjelmistotuotanto. Tietojenkäsittelytieteen laitos.  
Helsingin Yliopisto.  
<http://www.cs.helsinki.fi/u/paakki/ohtuk03-luento2-bw.pdf>. Luettu 4.1.2013.
- Lindberg, Harri. 2003. Extreme programming. Tietojenkäsittelytieteen laitos.  
Tampereen Yliopisto.  
[http://www.cs.uta.fi/research/thesis/masters/Lindberg\\_Harri.pdf](http://www.cs.uta.fi/research/thesis/masters/Lindberg_Harri.pdf).  
Luettu 20.2.2013.
- McConnell, Steve. 2002. Ohjelmistotuotannon hallinta. Helsinki: Edita IT Press
- Pohjonen, Risto. 2002. Tietojärjestelmien kehittäminen. Jyväskylä: Docendo Oy
- Reaktor, Laakso & Mäkinen 2011. Toiminnallinen määrittely.  
[http://reaktor.fi/osaaminen/toiminnallinen\\_maarittely/](http://reaktor.fi/osaaminen/toiminnallinen_maarittely/). Luettu 14.2.2013.
- Räsänen, Seppo. 2008. Ohjelmistojen elinkaarimallit. Savonia ammattikorkeakoulu  
[http://www.google.fi/url?sa=t&rct=j&q=vaihejakomallit%20spiraali&source=web&cd=3&ved=0CDgQFjAC&url=http%3A%2F%2Fwebd.savonia.fi%2Fhome%2Fktrse%2Fmat\\_tiedostot%2Fthy8s%2FTHY22\\_elinkaari.ppt&ei=Pf0UUeHnNYv4sgb-3IGwCg&usg=AFQjCNGWVXiwpJmq8COVUZdlifjrj2zE8lw&cad=rja](http://www.google.fi/url?sa=t&rct=j&q=vaihejakomallit%20spiraali&source=web&cd=3&ved=0CDgQFjAC&url=http%3A%2F%2Fwebd.savonia.fi%2Fhome%2Fktrse%2Fmat_tiedostot%2Fthy8s%2FTHY22_elinkaari.ppt&ei=Pf0UUeHnNYv4sgb-3IGwCg&usg=AFQjCNGWVXiwpJmq8COVUZdlifjrj2zE8lw&cad=rja).  
Luettu 7.1.2013.

Sainio, Laura. 2010. Ohjelmistotestauksen menetelmät ja työvälineet. Saimaan ammattikorkea koulu. Tietotekniikan koulutusohjelma.  
[https://publications.theseus.fi/bitstream/handle/10024/12297/Sainio\\_Laura\\_liite1.pdf?sequence=1](https://publications.theseus.fi/bitstream/handle/10024/12297/Sainio_Laura_liite1.pdf?sequence=1). Luettu 18.2.2013.

Sininen Meteoriiitti, 2011. Ketteryys haltuun: Scrum.  
<http://www.meteoriiitti.com/fi-FI/tiedotteet/ajankohtaista/ketteryys-haltuun-scrum-pahkinankuoressa>. Luettu 5.3.2013.

Talonrakennushanke RT 10-10387. 2001. Rakennustieto Oy. Luettu 26.3.2013.

Tervonen, Ilkka. Ohjelmistotekniikka. Oulun Yliopisto. Tietojenkäsittelytieteenlaitos.  
[http://www.tol.oulu.fi/users/ilkka.tervonen/Ote\\_2\\_11.pdf](http://www.tol.oulu.fi/users/ilkka.tervonen/Ote_2_11.pdf). Luettu 18.2.2013.

Yumasoft 2003 – 2012. RUP-menetelmä.  
<http://www.yumasoft.com/development/methodology>. Luettu 14.2.2013.

