Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikan koulutusohjelma
Viestintätekniikka

Ari Kultanen

Katsastusaseman työvuorojen suunnitteluohjelma

Opinnäytetyö 2013

Tiivistelmä

Ari Kultanen
Katsastusaseman työvuorojen suunnitteluohjelma, 69 sivua, 0 liitettä
Saimaan ammattikorkeakoulu
Tekniikan toimiala Lappeenranta
Tietotekniikan koulutusohjelma
Viestintätekniikka
Opinnäytetyö 2013
Ohjaajat: lehtori Martti Ylä-Jussila, Saimaan ammattikorkeakoulu, aluepäällikkö Jari Joenperä, A-Katsastus Oy Lappeenranta

Asiakkaana opinnäytetyössä on A-Katsastus Oy Lappeenranta ja mahdollisesti myöhemmin koko A-Katsastus-konserni. Työn kohteena on katsastusaseman työvuorojen suunnitteluohjelman (KATASO-ohjelman) toiminnallinen määrittely-dokumentti jatkokehitystyötä varten.

KATASO-ohjelman tarkoitus on automatisoida katsastusaseman työvuorojen suunnittelu ja jakelu, helpottaa suunnittelutyötä ja vähentää suunnitteluun käytettävää aikaa.

Opinnäytetyö aloitettiin tekemällä alustava tutkimus ja esiselvitys vanhasta työvuorojen suunnittelumenetelmästä. Esiselvityksen pohjalta päätettiin tehdä toiminnallinen määrittely uutta järjestelmää varten.

Opinnäytetyön tuloksena asiakkaalle tuotettiin KATASO-ohjelman järjestelmän määrittelydokumentti, yleisarkkitehtuurisuunnitelma sekä suunniteltiin tietokanta.

Järjestelmän toteutusta on mahdollista jatkaa muiden opinnäytetöiden puitteissa.

Avainsanat: työvuorojen suunnittelu, A-Katsastus Oy, toiminnallinen määrittely.

Abstract

Ari Kultanen
Rota planning software for A-Katsastus Oy, 69, Number of Appendices 0
Saimaa University of Applied Sciences
Engineering Lappeenranta
Information technology
Communication technology
Bachelor´s Thesis 2013
Instructor(s): Mr Martti Ylä-Jussila, lector, Saimaa University of Applied Sciences, Mr Jari Joenperä, Regional Manager, A-Katsastus Ltd.

The purpose of this thesis was to create functional specification for a rota planning for A-Katsastus Oy office in Lappeenranta. The software must be easy to use and its goal is to save time, make shifts management easier and make super-visor's work more efficient.

There are a little bit backgrounds helping to understand how the software has to be operated. There is different kind of rights to inspect vehicles of inspectors. There are also the different shifts of inspectors. First inspector has to receive a right to inspect small vehicles like passenger cars and vans. After that there is possibility to receive more rights by curriculums and work experience. Briefly, the scope of right the inspectors have may vary, which has to be considered in software, as well as part-time jobs, holidays and sick leaves.

The names and the rights of inspectors are stored in a database. There are also their personal calendars including their holidays, other vacations and training periods, which will be entered by superiors and/or root. In that way software can check who are available for a specific task.

The software will be generated shifts to workers automatically, but the shifts must be also manually modified by the superior.

There are eight main features in this software: the login, the workers management, the markings of workers calendars, the shift generation, the shift management, possibility to watch and print roster, the management of the marking types, management of the offices.

The software has been designed at first testing use in only one office (Lappeenranta), but in the future whole the corporation will be able to use it.
Execution of this software will be continued in another thesis.

Keywords: rota planning, A-Katsastus Oy, functional specification.

Termit ja lyhenteet

C#	Microsoft .NET Frameworkiä varten kehitetty ohjelmointikieli.
KATASO	Katsastusaseman työvuorojen suunnitteluohjelma.
Käyttöliittymä	Ohjelman osa, jonka avulla käyttäjä seuraa ja ohjaa ohjelman toimintaa ja saa tietoa ohjelman toiminnasta.
Microsoft Access	Relaatiotietokantaohjelma, joka soveltuu pienten ja keskisuurten aineistojen käsittelyyn.
SQL	Structured Query Language**.** IBM:n kehittämä standardoitu kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä.
TIR	Transport International Routier**.** Rahtiliikenteessä käytetty lyhenne Ranskassa tehdystä yleissopimuksesta, jonka tavoitteena on tavaroiden keskeytyksetön kuljettaminen maiden välillä samalla, kun kauttakulkumaan tulliviranomaisille suodaan maksimaaliset turvatakuut**.**
Työvuoro	Työvuoro on olennainen osa vuorotyötä, jossa työvuorot vaihtuvat säännöllisesti ja muuttuvat ennalta sovituin jaksoin. Työvuoro on tässä dokumentissa työntekijälle ennalta ilmoitettu työaika tiettynä ajanjaksona.
UML	Unified Modeling Language. Object Management Groupin (OMG) standardoima Graafinen mallinnuskieli
Windows 7	Microsoftin Windows-käyttöjärjestelmä henkilökohtaisiin koti- ja yritystietokoneisiin, kannettaviin tietokoneisiin sekä multimediatietokoneisiin.

Sisältö

1 Johdanto

1.1 Kohde

Opinnäytetyöni tavoitteena on tehdä katsastusaseman työvuorojen suunnitteluohjelman (KATASO-ohjelman) toiminnallinen vaatimusmäärittely.

Markkinoilla on saatavissa erilaisia työvuorojen suunnitteluun tarkoitettuja ohjelmia, mutta niiden soveltuvuus katsastusaseman käyttöön on epävarmaa – varsinkin, kun kaikki seikat, jotka työvuorojen suunnittelussa katsastusasemalla huomioidaan, esimerkiksi katsastajien erilaiset katsastusoikeudet ja tehtävät.

1.2 Tavoite

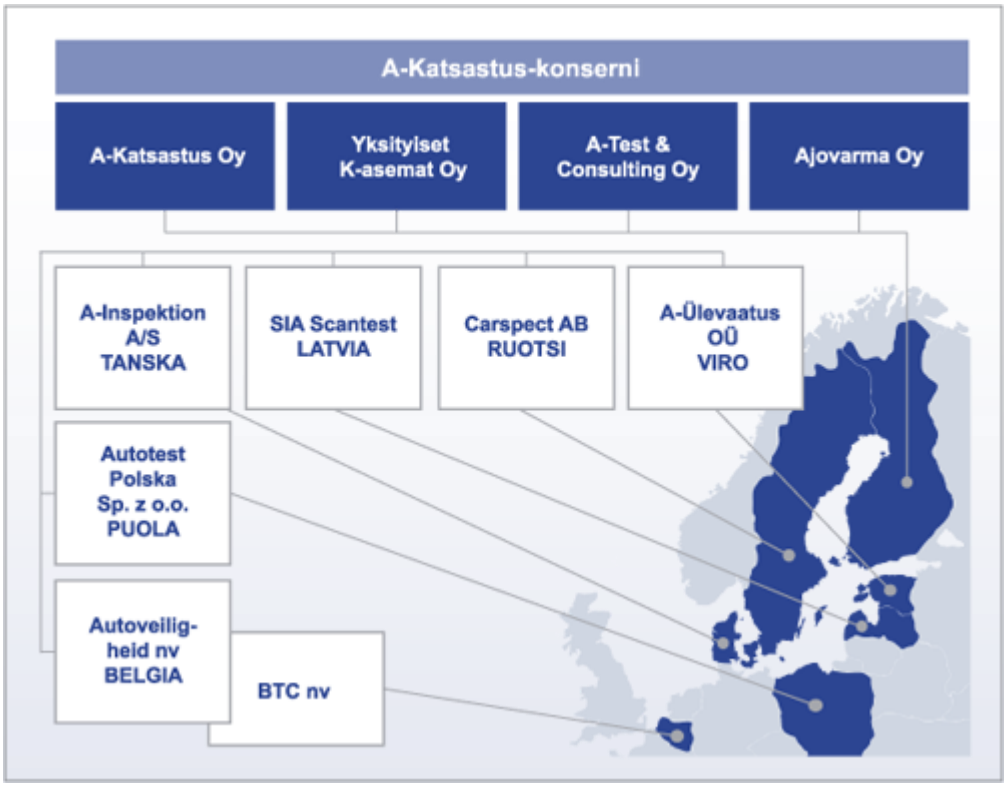
Ohjelman tarkoituksena on helpottaa esimiesten työtä heidän suunnitellessaan työntekijöille työvuoroja. Nykyisin työvuorot suunnitellaan ja merkitään manuaalisesti seitsemän viikkoa eteenpäin Excel-taulukkoon, joka tulostetaan työvuorolistaksi. Työvuorojen suunnittelu ja merkitseminen työvuorolistaan on aikaa vievää vaikka käytössä onkin ns. makroja, joten KATASO-ohjelman tarkoituksena on tehostaa ja nopeuttaa esimiesten ajankäyttöä automatisoimalla suunnittelua mahdollisimman pitkälle.

2 Asiakas

2.1 Yleistä A-Katsastus-konsernista

A-Katsastus on yksityinen ajoneuvokatsastusten, rekisteröintien, kuljettajantutkintojen ja ajoneuvontestauspalveluiden tarjoaja Pohjois-Euroopassa. Ensisijainen tavoite on parantaa liikenneturvallisuutta ja säästää ympäristö haitallisilta päästöiltä. A-Katsastus-konsernin palveluksessa Euroopassa on yhteensä noin 2000 työntekijää. Toiminta-alueena on Suomi, Belgia, Latvia, Puola, Ruotsi, Tanska ja Viro, joissa on 300 katsastusasemaa. Konsernin pääkonttori sijaitsee Helsingissä. Vuonna 2012 konsernin liikevaihto oli 170,6 miljoonaa euroa. Konserni suorittaa noin 3 miljoonaa suoritettua määräaikaiskatsastusta vuodessa. Päätoimialat ovat ajoneuvojen katsastukset, kuljettajantutkintojen vastaanotto, ajoneuvojen rekisteröinnit ja autoalan laatu- ja testauspalvelut. A-

Katsastuksen historia ulottuu aina 1900-luvun alkuun asti, jolloin ensimmäiset ajoneuvojen katsastukset ja kuljettajantutkinnot suoritettiin Suomessa (A-Katsastus Oy 2013.) Kuvassa 2.1 on esitetty A-Katsastus-konsernin rakenne.



Kuva 2.1 A-Katsastus Group konsernikaavio (A-Katsastus Oy 2013)

Asiakkaana on A-Katsastus Oy:n Lappeenrannan toimipiste, joka ohjelman valmistuttua toimii KATASO-ohjelman pilottitoimipaikkana. Muut toimipisteet rajataan ulos vielä tässä vaiheessa.

2.2 Toiminta ja toimipisteet

A-Katsastuksen Lappeenrannan toimipisteen toimialana on ajoneuvojen katsastukset, kuljettajantutkintojen vastaanotto (Ajovarma) ja ajoneuvojen rekisteröinnit. Lappeenrannan ja Imatran alueella sijaitsee viisi A-Katsastus-konserniin kuuluvaa katsastusasemaa sekä Lappeenrannan Auto-Kilta Trucksin tiloissa tiistai-torstai-iltapäivisin toimiva pelkästään raskaan kaluston katsastuksia suorittava Mustola. Tarvittaessa työntekijöitä kierrätetään näiden toimipisteiden välillä. A-Katsastus-konserniin kuuluvat katsastusasemat ovat Lappeenranta, Lauritsala ja Imatra. Lisäksi Lappeenrannan ja Imatran alueella toimivat Leirin Kat-

sastus sekä Mansikkalan katsastus kuuluvat Yksityiset K-Asemat ketjuun, joka on A-Katsastus-konserniin kuuluva tytäryhtiö.

2.3 Autokatsastus ja katsastusoikeudet

Tutuin kaikille ajoneuvokatsastuksista lienee määräaikaiskatsastus eli entinen vuosikatsastus, joka tulee suorittaa valtioneuvoston asetuksen liikenteessä käytettävien ajoneuvojen liikennekelpoisuuden valvonnasta 19.12.2002 mukaan autoille, niiden perävaunuille ja kevyille nelipyörille (mopoautot ja mönkijät) määräajoittain seuraavan taulukon 2.1 mukaisesti:

Ajoneuvoluokka	**Määräaikaiskatsastus on suoritettava**
a) Linja- ja kuorma-autot (M_2-, M_3-, N_2- ja N_3-luokka), erikoisautot, joiden kokonaismassa on suurempi kuin 3,5 tonnia, luvanvaraiseen liikenteeseen käytettävät henkilöautot (M_1-luokka) sekä sairasautot	Ensimmäisen kerran viimeistään vuoden kuluttua ajoneuvon käyttöönottopäivästä ja sen jälkeen vuosittain viimeistään käyttöönottopäivää vastaavana päivänä
b) Perävaunut, joiden kokonaismassa on suurempi kuin 3,5 tonnia (O_3- ja O_4-luokka)	Ensimmäisen kerran viimeistään vuoden kuluttua ajoneuvon käyttöönottopäivästä ja sen jälkeen vuosittain viimeistään käyttöönottopäivää vastaavana päivänä; kytkentäkatsastuksessa tiettyyn vetoautoon kytketty perävaunu saadaan kuitenkin tuoda määräaikaiskatsastukseen yhtä aikaa vetoauton kanssa
c) Pakettiautot (N_1-luokka) ja sairasautoja lukuun ottamatta erikoisautot, joiden kokonaismassa on enintään 3,5 tonnia	Ensimmäisen kerran kolmen vuoden kuluttua ajoneuvon käyttöönottopäivästä ja sen jälkeen vuosittain viimeistään käyttöönottopäivää vastaavana päivänä
d) Yksityiseen liikenteeseen käytettävät henkilöautot ja muut M1-luokan ajoneuvot kuin sairasautot, kevyet nelipyörät (L_{6e}-luokka) sekä nelipyörät (L_{7e}-luokka)	Ensimmäisen kerran kolmen vuoden kuluttua ajoneuvon käyttöönottopäivästä, toisen kerran viiden vuoden kuluttua ajoneuvon käyttöönottopäivästä ja sen jälkeen vuosittain viimeistään käyttöönottopäivää vastaavana päivänä
e) Perävaunut, joiden kokonaismassa on suurempi kuin 0,75 tonnia mutta enintään 3,5 tonnia (O_2-luokka)	Ensimmäisen kerran kalenterivuoden loppuun mennessä sinä vuonna, jolloin käyttöönottopäivästä on kulunut kaksi vuotta, ja sen jälkeen kahden vuoden välein kalenterivuoden loppuun mennessä
f) 1 päivänä tammikuuta 1960 tai sen jälkeen käyttöön otetut katsastusvelvolliseen ajoneuvoluokkaan kuuluvat museoajoneuvot	Kahden vuoden välein kesäkuun loppuun mennessä
g) Ennen 1 päivää tammikuuta 1960 käyttöön otetut katsastusvelvolliseen ajoneuvoluokkaan kuuluvat museoajoneuvot	Neljän vuoden välein kesäkuun loppuun mennessä

Taulukko 2.1 Määräaikaiskatsastusvelvolliset ajoneuvot ja niiden määräaikaiskatsastusvälit. (Valtioneuvoston asetus liikenteessä käytettävien ajoneuvojen liikennekelpoisuuden valvonnasta 19.12.2002)

Lain ajoneuvojen katsastusluvista 23.12.1998 mukaisesti katsastajilla on erilaisia katsastusoikeuksia ja katsastajaksi pääsylle asetetaan tiettyjä edellytyksiä, muun muassa peruskoulutusvaatimuksena on auto- tai konetekniikan teknikon tai insinöörin tutkinto sekä katsastajan perustutkinnon hyväksyttyä suorittamista. Lisäksi perusoikeuksien säilyttäminen edellyttää osallistumista vuosittain täydennyskoulutukseen.

Lisäksi Liikenneministeriön päätöksen ajoneuvojen katsastushenkilöstön lisäkoulutuksesta 19.21999 nojalla katsastajalla voi olla muitakin katsastusoikeuksia, joiden saamisen edellytyksenä on koulutus kyseiseen katsastuslajiin ja sen loppukokeen hyväksytty suorittaminen. Erikoiskatsastusoikeuksien säilyttäminen edellyttää kyseisen katsastuslajin kokeen hyväksyttyä suorittamista kolmen vuoden välein. Erikoiskatsastusoikeuksia ovat

- kevyen kaluston rekisteröinti- ja muutoskatsastus
- raskaan kaluston rekisteröinti- ja muutoskatsastus
- paineilmajarrut (raskas kalusto).

Näiden lisäksi katsastajalla voi olla oikeudet suorittaa ns. autotohtoritarkastuksia, jonka edellytyksenä ovat kurssin suorittaminen sekä TIR-kuntoisuusasiantuntijan ja hyväksytyn asiantuntijan oikeudet, joihin edellytetään kurssin hyväksyttyä suorittamista.

2.4 Työvuorojen jakamisen ongelmat ja tavoitteet

Seuraavaksi käsitellään muutamia perusasioita työaikalaista. A-Katsastuksen työehtosopimuksessa noudatetaan työaikalaista poiketen 37,5 tunnin viikoittaista työaikaa.

Työaikalaissa 9.8.1996 säädetään, niin työajaksi luetaan työhön käytetty aika sekä aika, jonka työntekijä on velvollinen olemaan työpaikalla työnantajan käytettävissä. Matkaan käytettyä aikaa ei lueta työaikaan, ellei sitä samalla ole pidettävä työsuorituksena. Säännöllinen työaika on enintään kahdeksan tuntia vuorokaudessa ja 40 tuntia viikossa. Viikoittainen säännöllinen työaika voidaan järjestää myös keskimäärin 40 tunniksi enintään 52 viikon ajanjakson aikana. Säännöllinen työaika saadaan 6 §:ssä säädetystä poiketen järjestää niin, että se on kolmen viikon pituisena ajanjaksona enintään 120 tuntia tai kahden viikon pituisena ajanjaksona enintään 80 tuntia (Työaikalaki 9.8.1996/605.)

Työn tarkoituksenmukaiseksi järjestämiseksi tai työntekijöille epätarkoituksenmukaisten työvuorojen välttämiseksi voidaan säännöllinen työaika 1 momentissa säädetystä poiketen järjestää niin, että se on kahden toisiaan seuraavan kolmen viikon ajanjakson aikana tai kolmen toisiaan seuraavan kahden viikon ajanjakson aikana enintään 240 tuntia. Säännöllinen työaika ei saa kummankaan kolmen viikon ajanjakson aikana ylittää 128 tuntia eikä yhdenkään kahden viikon ajanjakson aikana 88 tuntia. (Työaikalaki 9.8.1996/605.)

Työnantaja ja työntekijä voivat 6 §:n 1 momentista sekä työehtosopimuksen säännöllisen työajan pituutta ja sijoittamista koskevista määräyksistä poiketen sopia liukuvasta työajasta niin, että työntekijä voi sovituissa rajoissa määrätä työnsä päivittäisen alkamis- ja päättymisajankohdan. Sovittaessa liukuvasta työajasta on sovittava ainakin kiinteästä työajasta, työajan vuorokautisesta liukumarajasta, lepoaikojen sijoittamisesta sekä säännöllisen työajan ylitysten ja alitusten enimmäiskertymästä (Työaikalaki 9.8.1996/605.)

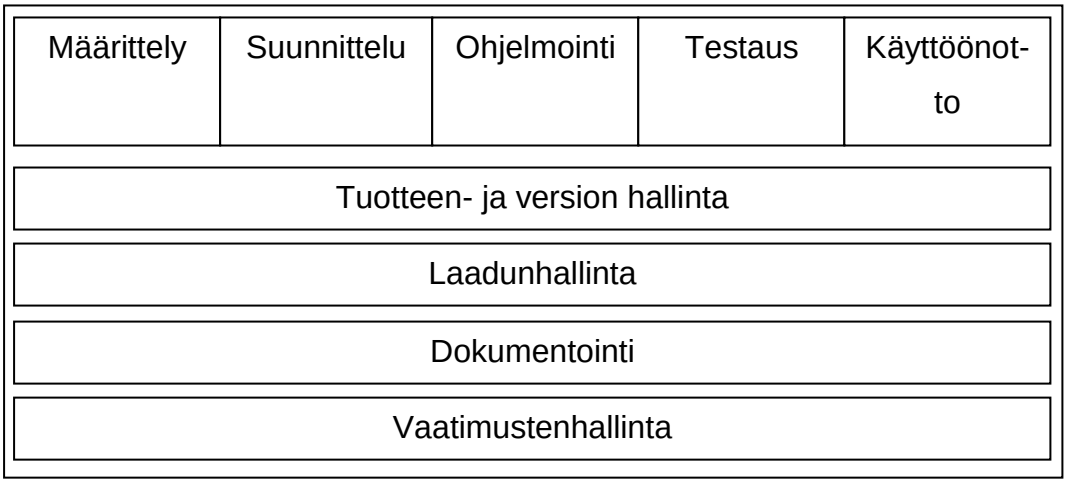
Liukuvassa työajassa säännöllistä vuorokautista työaikaa lyhentää tai pidentää liukuma-aika, joka voi olla enintään kolme tuntia. Viikoittainen säännöllinen työaika on keskimäärin enintään 40 tuntia. Edellä 1 momentissa mainittu enimmäiskertymä saa olla enintään 40 tuntia. Työnantaja ja työntekijä voivat sopia, että työajan ylitysten kertymää vähennetään työntekijälle annettavalla vapaa-ajalla (Työaikalaki 9.8.1996/605.)

Työvuorojen jakamisessa tulisi käyttää tasapuolisuutta ja lisäksi pitäisi huomioida työntekijöiden erilaiset katsastusoikeudet sekä mahdolliset osa-aikaisuudet. Nykyisessä järjestelmässä esimiehet syöttävät työvuorot Excel-taulukkoon, joka tulostetaan työvuorolistaksi. Tasapuolisuus on täysin esimiehen vallassa lisäksi työvuorojen suunnittelu ja syöttäminen on aikaa vievää.

KATASO-ohjelman tavoitteena on automatisoida työvuorojen suunnittelu niin pitkälle kuin mahdollista tasapuolisuus huomioiden, kuitenkin antaen akuuteissa tilanteissa esimiehille mahdollisuuden muokata työvuoroja.

3 Ohjelmistotuotanto

Ohjelmistotuotantoprosessin perustoimintoja ovat: määrittely, suunnittelu, ohjelmointi, testaus, käyttöönotto ja näiden lisäksi tuotteen- ja versionhallinta, laadunvarmistus, dokumentointi ja vaatimustenhallinta (kaavio 3.1).

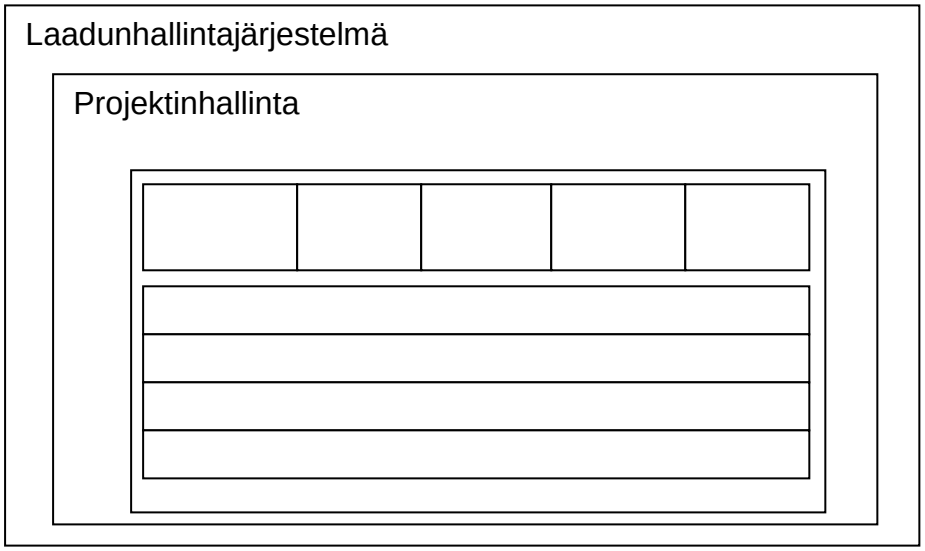


Kuva 3.1 Ohjelmistotuotannon vaiheet (Haikala & Mikkonen 2011, 29)

Määrittelyksi kutsutaan yleensä asiakasvaatimusten ja niistä johdettujen toimintojen dokumentointia asiakkaan kannalta. Suunnittelulla tarkoitetaan edellä mainitun määrittelyn mukaista järjestelmän teknistä suunnittelua. Ohjelmoinnin tarkoitus on antaa tietokoneelle, laitteelle tai ohjelmalle ohjeita, kuinka sen tulee toimia, ja yleensä se tehdään jonkin ohjelmointikielen avulla. Testauksella on tarkoitus etsiä järjestelmästä mahdollisia virheitä, ja samalla se mittaa ohjelmiston laatua. Ohjelmiston valmistuttua se käyttöönotetaan. Tuotteen- ja versionhallinnalla tarkkaillaan ohjelmistotuotteen tilaa, muun muassa mitä ja millaisia versioita komponenteista (ohjelman osista) on olemassa. Laadunhallinnaksi kutsutaan kaikkia niitä menetelmiä, joilla ohjelman virheet saadaan minimoitua eli myös testaus ainakin osittain kuuluu laadunhallintaan. Dokumentointi kattaa koko ohjelmistotuotannon osat, ja sen tehtävänä on kuvata ohjelmiston toimintaa ja rakennetta mahdollisimman tarkasti ja yksiselitteisesti ohjelman ehdoilla. Dokumentointi on jatkuva prosessi ja tärkeä osa ohjelmistotuotantoa. Vaatimustenhallinta on dokumentaatio ohjelman vaatimuksista kaikilla tasoilla. Lisäksi vaatimustenhallinta selvittää kuinka ohjelman vaatimusmuutokset käsitellään. (Haikala & Mikkonen 2011, 29-31; Dokumentointi 2013.)

3.1 Laadunhallinta

Kuvassa 3.1 esitetty ohjelmistotuotannon vaiheet on yksi projekti. Projekteja on pystyttävä hallitsemaan. Tätä kutsutaan projektinhallinnaksi. Useampi projekti voi muodostaa hankkeen. Tällöin voidaan ajatella, että näitä hallitaan laadunhallintajärjestelmällä, kuten kuvassa 3.2 ilmenee.



Kuva 3.2 Laadunhallintajärjestelmä (Haikala & Mikkola 2011, 29)

Laadunhallintaa varten yrityksillä on yleensä laatujärjestelmä, joka ei välttämättä tarkoita sitä, että yritys tekee hyvää laatua vaan ennemminkin sitä, että yritys kontrolloi hallitusti omaa laatuaan. Laadunhallinta on prosessi. Prosessi koostuu ihmisistä, tehtävistä, järjestelmistä, työkaluista, menetelmistä ja toisista prosesseista. Prosessien määreitä ovat tavoitteellisuus, ohjeistus, näkyvyys, suunnitelmallinen toiminnan kehittäminen ja omistajat. Prosesseita kuvataan yleensä kaavioilla. Tähän soveltuu hyvin muun muassa UML-kielen aktiviteettikaaviot. (Haikala & Mikkonen 2011, 137-151.)

Ohjelmistotuotantoprosessissa mittareina voidaan käyttää seuraavia kysymyksiä: ”Missä ollaan?”, ”Onko asetetut tavoitteet saavutettu?” ja ”Mitkä ovat tärkeitä kehityskohteita?” Yleensä mitataan projekteja, tuotteita ja prosesseja. Ohjelmistotuotannon käyttöön ei ole kehitetty kovinkaan luotettavia mittareita, niiden käyttö voi olla vaikeaa ja asenne niitä kohtaan negatiivinen. Mitä mittareilta tulee sitten vaatia? Ainakin seuraavia asioita: niitä ei voida käyttää väärin, niiden tulee mitata prosessia ja ryhmää, ne ovat luotettavia, tasapuolisia, riippumattomia, puolueettomia, ymmärrettäviä (myös asiakkaan kannalta), mahdollisimman

automatisoituja ja lisäksi niiden on pystyttävä osoittamaan taloudellinen hyöty. Mittareina voidaan käyttää esimerkiksi aikataulujen pitävyyttä, vähäistä virheiden määrää, takuutöiden määrä ja asiakastyytyväisyyttä.

Laadunhallintajärjestelmälle tehdään tarkastuksia eli auditointeja, joilla tarkastetaan kuinka hyvin laadunhallintajärjestelmässä kuvattuja toimintatapoja noudatetaan. Auditointeja on sekä sisäisiä että ulkoisia, ja ne tehdään määräajoittain esimerkiksi vuoden välein. Laadunhallintajärjestelmät ovat yleensä standardoituja ja tutuin niistä lienee ISO 9001-stardardi.

3.2 Vaatimustenhallinta

Vaatimustenhallinta ja vaatimusmäärittely ovat osa laajempaa käsitettä vaatimustenkäsittely. Vaatimustenhallinnan tärkein osa on muutostenhallinta. Vaatimustenkäsittely ja tuotteenhallinta ovat toistensa tukitoimintoja sekä hyvin lähellä toisiaan. Vaatimustenhallinnan tehtäväksi voidaan katsoa esimerkiksi dokumentointi vaatimuksen tilasta, vaatimuksien välisten riippuvuuksien seuranta ja raportointi. Vaatimusmäärittelyn dokumentaationa taas syntyy toiminnallinen määrittely (Haikala & Mikkonen 2011, 61-68.)

Mauri Tikka on Machina Ludens -verkkosivullaan – Vaatimustenhallinta (Requirements engineering) – tiivistänyt vaatimustenhallinnan ominaisuuden hyvin yhteen virkkeeseen ”*On helpompi löytää perille, jos tietää mihin on pyrkimässä”* (M. Tikka 2013). Vaatimustenhallinta on siis tavallaan selkokielinen kartta, jonka mukaan projektia ohjataan ja jonka projektin molemmat osapuolet ovat hyväksyneet. Vaatimustenhallinta on pääasiassa ihmisten välistä vuorovaikutusta ja sen vuoksi myös tärkeä osa ohjelman suunnittelua. Suunnitteluvaiheessa ideat ja havainnot olisi hyvä kirjata vaatimuksina dokumentaatioon. Vaatimustenhallinnan on otettava huomioon myös kuinka pitkäksi tuotteen ikä eli elinkaari suunnitellaan, koska vaatimukset muuttuvat ajan myötä. Ohjelmaan halutaan esimerkiksi lisäominaisuuksia, sen ulkonäköä halutaan päivittää, siinä havaitaan virheitä ja niin edelleen.

Mitä ovat vaatimukset? Vaatimukset määrittävät sen, että mitä ja miten toteutetaan ja kuinka vaatimusten toteutuminen todennetaan eli vaatimustenhallinta on todistus siitä, mitä on sovittu ja onko tavoitteet saavutettu. Vaatimuksia on toi-

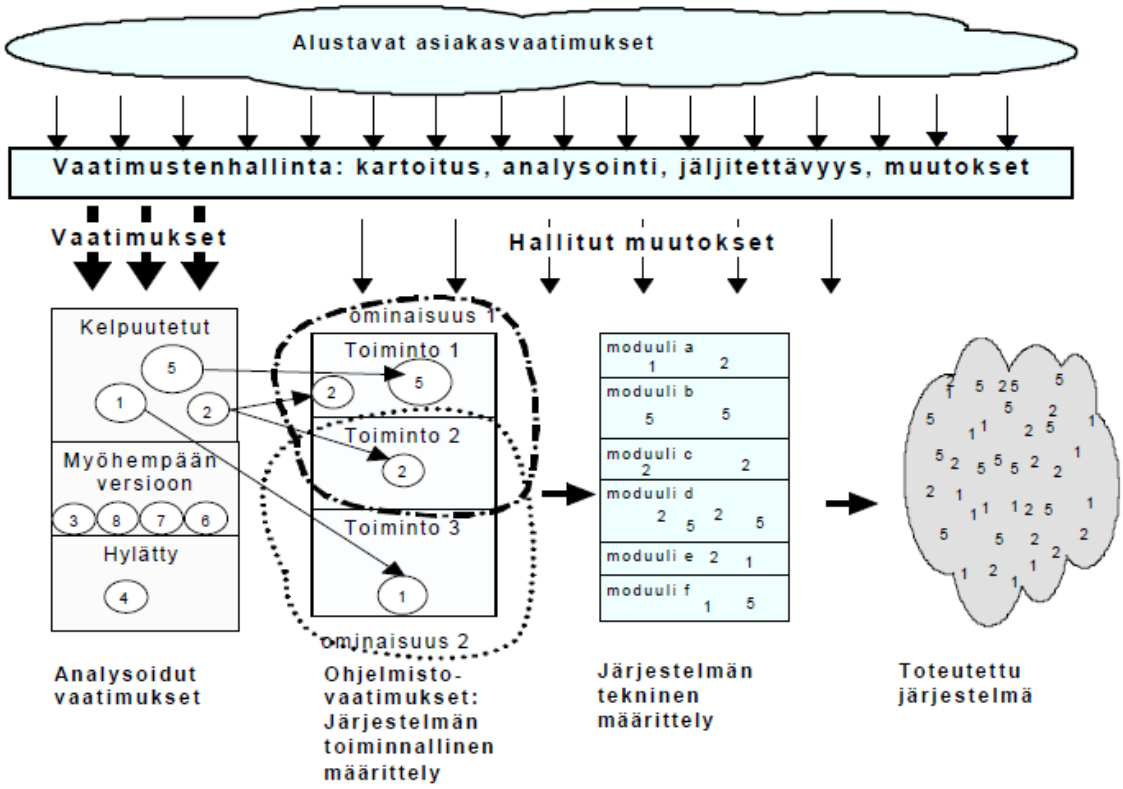
minnallisia ja ei-toiminnallisia, lisäksi vaatimuksiin luetaan myös reunaehdot. Asiakasvaatimukset ovat yleensä ohjelman toiminnallisuuteen liittyviä. Asiakasvaatimukset toteutetaan ohjelmistovaatimuksilla. Ohjelmistovaatimuksia esimerkiksi toiminnot, jotka määrittävät, miten tietty asiakasvaatimus toteutetaan käyttäjälle. Lopulta ohjelmistovaatimukset ovat joukko teknisiä vaatimuksia. Pekka Mäkinen on luentosarjassaan ”Laadukas vaatimustenhallinta” esittänyt millainen on hyvä vaatimus. Hyvä vaatimus on toteutettavissa ja testattavissa sekä se on selkeä ja ristiriidaton muiden vaatimusten kanssa. Hyvää olemassa olevaa vaatimustenhallintaa voidaan hyödyntää ohjelmaa päivitettäessä, jos dokumentaatio on ajan tasalla ja siinä olevat vaatimukset ovat niin sanotusti hyvätasoisia (P. Mäkinen.)

Ilkka Haikala ja Tommi Mikkonen ovat taas listanneet hyvän vaatimuksen ominaisuudet seuraavalla tavalla:

- tarkka ja ymmärrettävä
- testattava
- jäljitettävyys eteenpäin ja taaksepäin (Haikala & Mikkonen 2011, 61-68.)

Dokumentaation määrä riippuu suuresti sovelluksen luonteesta, mutta tyypillisimpiä dokumentoitavia asioita ovat luontipäivämäärä, tekijä, asiakas, vaatimuksen tyyppi, vaatimuksen kuvaus, suhde muihin vaatimuksiin, tarpeellisuus, pysyvyys (muutosherkkyys), testattavuus ja aika-arvio (Haikala & Mikkonen 2011, 61-68.)

Haikalan ja Märijärven mukaan vaatimustenhallinta etenee seuraavan kuva 3.3. mukaisesti.



Kuva 3.3 Vaatimustenhallinta ohjelmistoprojektissa (Haikala & Märijärvi 2004)

Vaatimustenhallintaan kuuluvat osana verifiointi, validointi, jäljitettävyys ja muutostenhallinta. Verifioinnilla tarkoitetaan vaatimusten toteutumista vaiheen tulosdokumentteihin. Validoinnilla eli kelpoistamisella pyritään osoittamaan järjestelmän vastaavuus asiakkaan tarpeisiin. Jäljitettävyydellä voidaan tarvittaessa todentaa, että tuote täyttää eteenpäin ja taaksepäin kaikki asiakasvaatimukset. Jäljitettävyyden helpottamiseksi on kehitetty niin sanottu jäljitettävyysmatriisi. Muutostenhallinnan tehtävänä on hallita määrittelyvaiheen jälkeisiä muutoksia. Kaikkia muutoksia on tarkasteltava kriittisesti. Lisäksi on selvitettävä mihin ja miten se vaikuttaa projektiin sen eri vaiheissa. Muutostenhallinnan kannalta jäljitettävyys on tärkeä ominaisuus. (Ovaska 2002.)

Vaatimustenhallinta lähtee liiketoiminnan tarpeista. Liiketoiminnan vaatimuksia ei yleensä kuitenkaan kirjata vaatimustenhallintadokumentteihin. Vaatimustenhallinnan tehtävänä on muuttaa korkealla abstraktiotasolla oleva kuvaus ohjel-

masta täsmälliseksi ja yksityiskohtaiseksi ohjelmointikielellä kuvatuksi ohjelmaksi (Haikala & Mikkonen 2011, 61-68.)

3.3 Versiohallinta

Versionhallinta mahdollistaa ohjelmiston kehityksen seurannan ja hallitun kehityksen. Versionhallinnan tehtävä on hallita ohjelmistoprojektin komponentteja ja konfiguraatioita eli hallinta-alkioita. (Ohjelmiston versionhallinta 2013.)

Versioksi kutsutaan jäädytettyä hallinta-alkiota eli toisin sanoen, johon ei enää tehdä muutoksia. Muutostarpeen ilmetessä muutokset toteutetaan tekemällä hallinta-alkiosta uusi versio. Hallinta-alkioiden ja niiden versioiden on oltava yksiselitteisesti identifioituja. Kehitysprojektin hallinta-alkion viimeisintä jäädytettyä versiota kutsutaan vaihetasoksi. Hyväksymisen jälkeen vaihetuote tuodaan tuotteenhallinnan piiriin, jonka jälkeen muutokset vaativat muodollisen hyväksymisen ja niistä on tiedotettava. (Haikala & Mikkonen 2011, 170.)

Hallinta-alkioista kirjattavia tietoja ovat ainakin itse hallinta-alkion lisäksi sen versionumero, vastuuhenkilö (tekijä), tila ja tilan muuttumispäivämäärät. Tilalla tarkoitetaan sitä onko esimerkiksi muutoksen teko valmis, aloitettu vai ei. Suositeltavaa on myös kirjata testiympäristöt sekä –tapaukset versionumeroineen ja laiteympäristöineen. (Haikala & Mikkonen 2011, 172.)

Versioiden numerointiin ei ole vakiintunutta käytäntöä, mutta versiopuu kasvaa yleensä lineaarisesti. Esimerkiksi versiota 1.0 seuraa versio 1.1 ja niin edelleen. Vanhat versiot on kuitenkin syytä säilyttää uudempien versioiden mahdollisten virheiden varalta, jolloin vanha toimiva versio voidaan palauttaa esimerkiksi virheiden korjauksen ajaksi. Tuotteenhallinnassa on kuitenkin koko ajan tiedettävä, mistä hallinta-alkioiden versioista ohjelman tietty versio koostuu. (Haikala & Mikkola 2011, 173.)

3.4 Dokumentointi

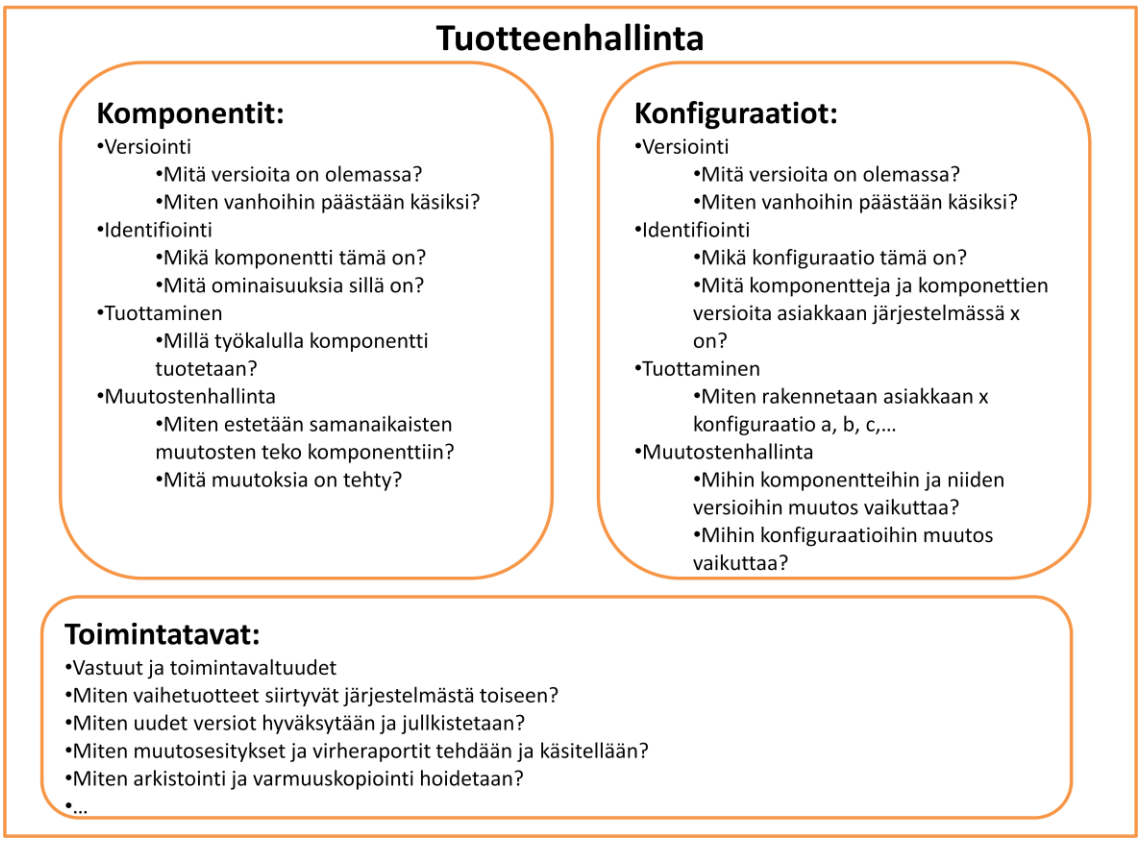
Ohjelmistoprojektin dokumentaation tulee olla ajantasaista muutoin siitä ei ole hyötyä. Kaikkea ei ole kuitenkaan tarvetta dokumentoida, vaan dokumentaation tulee tiivis. Tiiviydellä vältetään myös mahdolliset ristiriitaisuudet ylläpidon aikana. Dokumentaatiosta tulee jättää kaikki turhat yksityiskohdat pois ja kirjata vain

se johon ei saa tulla muutoksia. Näin dokumentaatiota on helpompi ylläpitää ja se ei vanhene muutosten myötä niin helposti (Haikala & Mikkonen 2011, 194.)

Keskeisin ohjelmistotyössä syntyvistä dokumenteista on arkkitehtuurikuvaus. Arkkitehtuurikuvaus on suunnittelijoiden näkemys järjestelmästä ja sen perusrakenteesta ja -filosofiasta ja sen tehtävänä on toimia ylläpito-ohjeena ja perehdytysmateriaalina järjestelmään. Toteutusvaiheessa dokumentaatio ohjelmistosta kannattaa mahdollisuuksien mukaan tuottaa automaattisesti ohjelmakoodista, sillä jos dokumentoinnin luominen on ylimääräinen toimenpide se jää usein tekemättä. Ohjelmakoodiin kannattaakin upottaa toimintalogiikkaan liittyvä yleistasoinen dokumentaatio. Dokumentaation päivittäminen kannattaa suorittaa viimeistään projektin päättyessä, mieluiten kuitenkin jo muutosten myötä. (Haikala & Mikkonen 2011, 194-195.)

3.5 Tuotteenhallinta

Tuotteenhallinta on kiinteä osa tuotetta ja organisaation tapaa toimia. Yleisiä ohjeita tuotteenhallinnan hoitamiseen on vaikea antaa. Tuotteenhallinnan tehtävä on huolehtia komponenttien kuten ohjelmamoduulien, toiminnallisen ja teknisen määrittelyn, testaussuunnitelmien ja käyttöohjeiden päivittämisestä ja vanhojen versioiden säilyttämisestä eli niiden hallinnasta (Haikala & Mikkonen 2011, s.169-177.) Kuvassa 3.4 on esitetty tuotteenhallinnan osat.



Kuva 3.4 Tuotteenhallinnan osa-alueet. (Haikala & Mikkonen 2011, s. 171)

Tuotteenhallinta jaetaan kahteen osaan: versiohallintaan ja muutostenhallintaan (Joensuun yliopisto, Tietojenkäsittelytieteenlaitos 2007). Versiohallinta on käsitelty tarkemmin luvussa 3.3.

Ohjelmistot koostuvat komponenteista, komponenteista muodostuu konfiguraatiot. Yhteinen nimitys komponenteille ja konfiguraatioille on hallinta-alkio. Hallinta-alkiot kehittyvät ohjelmiston elinkaaren aikana. Teknisten osa-alueiden kannalta tuotteenhallinta on hallinta-alkioiden hallintaa, josta voidaan erottaa seuraavat osa-alueet: saman komponentin eri versioiden hallinta, konfiguraatioversioiden hallinta ja versioiden sekä konfiguraatioiden luonnissa ja muutoksissa käytetyt toimintatavat. Hyvin järjestetty tuotteenhallinta helpottaa suunnittelijoiden yhteistyötä parantamalla kehityksen ja ylläpidon koordinointia. (Haikala & Mikkonen 2011.)

Markkinoilta löytyy tuotteenhallinnan ja sen osien tarpeisiin kehitettyjä erilaisia apuvälineitä, joita on kaupallisia sekä avoimen lähdekoodin ohjelmistoja.

4 Työssä käytetyt menetelmät

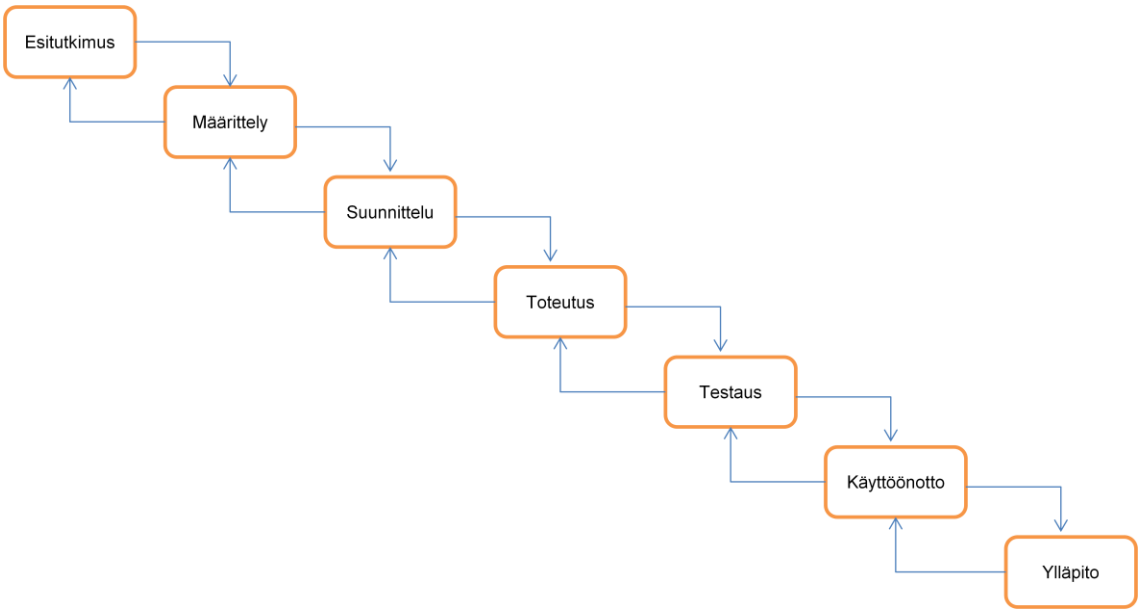
Ohjelmistotuotanto noudatti alkuaikoina suoraviivaisesti eteneviä vaihejakomalleja, joista tunnetuin on vesiputousmalli. Käytännössä ohjelmistotyö ei koskaan etene teoreettisen vesiputousmallin mukaisesti vaiheesta toiseen, vaan todellinen kehitystyön noudattama malli on yhdistelmä erilaisista teoreettisista malleista (asteittainen kehitys, protoilu, evoluutio …).

Nykyiset ohjelmistotuotannossa käytetyt vaihejakomallit ovat iteratiivisia. Iteratiivisessa toimintatavassa ei pyritä etenemään suoraviivaisesti koko järjestelmän laajuudella vaiheesta toiseen, vaan kehitystyötä tehdään pienin askelin eli inkrementein. Jokainen inkrementti on lisäys kehitteillä olevaan järjestelmään. Iteratiivisella toimintatavalla lopputuotteesta saadaan asiakkaan kannalta laadukkaampia ja toisaalta hankkeen riskit pystytään hallitsemaan paremmin. Eräs uraauurtava iteratiivinen malli on Rationalin kehittämä RUP-malli.

Uusin virtaus ohjelmistotuotannon työmenetelmissä ovat ketterät menetelmät, joissa perusajatuksena on toteuttaa järjestelmä ilman dokumentteja tai minimaalisin dokumentein. Tällainen toimintatapa edellyttää kehitystyön nopeaa etenemistä, asiakkaan päivittäistä yhteyttä kehitystiimiin sekä jokaiselta tiimin jäseneltä erittäin hyvää tietoteknistä ammattitaitoa.

4.1 Vesiputousmalli

Vaihejakomalleista kenties tunnetuin on Winston W. Roycen 1970-luvulla luoma niin sanottu vesiputousmalli, joka on esitetty kuvassa 4.1.



Kuva 4.1. Vesiputousmalli (Haikala & Mikkonen 2011, 37)

Royce ei itse käyttänyt mallistaan nimitystä vesiputousmalli, vaan hän käytti sitä esimerkkinä toimimattomasta ja virheellisestä ohjelmistokehitysmallista. Vesiputousmallia käytetään vaiheellisessa ohjelmistotuotantoprosessissa ja siitä on useita erilaisia variaatioita. Vesiputousmalli nimitys tulee siitä, että suunnittelu- ja toteutusprosessi etenevät kuten vesiputouksessa vaihe vaiheelta alaspäin. Alkuperäinen malli sisälsi seuraavat vaiheet: järjestelmävaatimukset, ohjelmistovaatimukset, analyysi, suunnittelu, ohjelmointi, testaus ja käyttöönotto. Vesiputousmalli edellyttää, että edellinen vaihe on saatettu valmiiksi ennen kuin siirrytään seuraavaan vaiheeseen. Teollisessa valmistuksessa vesiputousmalli monesti yksinkertaistuu siten, että vaiheiden väliset iteroinnit jäävät pois. Iteroinnin mukaan ottaminen sallii sen, että seuraava vaihe voidaan käynnistää ennen edellisen loppua. Tarkastellessa uudempia projektimalleja niiden sisältä voi löytää alkuperäisen vesiputousmallin idean hieman mukailtuna. Vesiputousmallin yhtenä etuna katsotaan olevan sen painotus kattavaan dokumentointiin. Lisäksi sen vaiheistus helpottaa projektin etenemisen seuraamista. (Haikala & Mikkonen 2011, 36-37.)

Esitutkimus

Esitutkimuksen tavoitteena on kartoittaa eri vaihtoehtoja esimerkiksi tehdäänkö itse vai ostetaanko ohjelmisto valmiina? Ohjelmiston alustava määrittely, tarve, kannattavuus ja riskien kartoitus kuuluvat myös esitutkimuksen piiriin. Lisäksi esitutkimuksessa päätetäänkö ohjelmistoprojekti käynnistää. (Haikala & Mikkonen 2011, 20-21.)

Alustavien asiakasvaatimusten analysointi tapahtuu esitutkimusvaiheessa, mutta niihin palataan yleensä myöhemmissäkin vaiheissa. Asiakasvaatimukset ovat yleensä aluksi hyvin puutteellisia ja ristiriitaisia. (Haikala & Märijärvi 2004,s 95.)

Toiminnallinen määrittely

Ohjelmien tuottaminen on teknisestä näkökulmasta dokumenttien tuottamista. Jos myös ohjelmat mielletään dokumenteiksi, niin ohjelmistotyö ei juuri muuta olekaan. Näin ajateltuna tuotantoprosessi on vaiheittainen kuvaus vaatimuksista niiden toteutukseksi. Jokaisella tasolla tuloksena on seuraavan tason syötteeksi spesifikaatiodokumentti. (Haikala & Märijärvi 2004, 61.) Vaatimusmäärittelyssä järjestelmälle asetetaan toiminnallisia ja ei-toiminnallisia vaatimuksia. Osa vaatimuksista on rajoitteita. Vaatimukset kerätään sidosryhmiltä ja ne on kuvattava järjestelmän ominaisuuksiksi. Vaatimusmäärittely jaetaan pääsääntöisesti kahteen osaan toiminnalliseen määrittelyyn ja tekniseen määrittelyyn. Toiminnallisen määrittelyn tarkoituksena on antaa järjestelmästä yleiskuvaus kuten ympäristö, toiminnot, käyttäjät, toteutusmenetelmien rajoitukset ja määrittelyhetkellä voimassa olevat oletukset riippuvuudet. Lisäksi toiminnalliseen määrittelyyn kuuluvat toimintojen ja tietokantojen suunnittelu, kuvaukset ulkoisista liittymistä sekä muiden ominaisuuksien kuten esimerkiksi suorituskyvyn, ylläpidettävyyden, siirrettävyyden, käytettävyyden ja tietoturvallisuuden suunnittelu. Toiminnallisessa määrittelyssä otetaan myös kantaa suunnitterajoitteisiin esimerkiksi standardeihin, laitteisto- ja ohjelmistorajoitteisiin. Teknisen määrittelyn tarkoitus on määritellä kuinka nämä ohjelmointi teknisesti toteutetaan (Paakki 2013; Vaasan ammattikorkeakoulu.) Toiminnallisia ominaisuuksia havainnollistetaan kaavioiden, kuten käyttötapauskaavioiden avulla. Kaavioihin tulee sisällyttää vain tarpeellinen tieto, jotta niissä oleva informaatio pystytään kirjoittamaan yksiselit-

teisesti auki. Toiminnallisen määrittelyn tarkoituksena on antaa vastaus kysymykseen: ”Miten sovellus tehdään?” (Itä-Suomen yliopisto 2009.) Hyvä määrittely on edellytys ohjelmistoprojektin onnistumiselle (Haikala & Märijärvi 2004, 65.) Dokumentaationa syntyy ohjelman toiminnallinen määrittely.

Suunnittelu

Perusongelmana ohjelmistotekniikassa on järjestelmävaatimusten ja toteutusteknologian yhteensovittaminen. Apuväline tähän ongelmaan on ohjelmistosuunnittelu, jonka keskeisenä osana on arkkitehtuurisuunnittelu. Arkkitehtuurisuunnittelun tehtävä on muuntaa järjestelmän kuvaus ongelmasta ratkaisuun ja lisäksi se määrittelee järjestelmän terminologian sekä työnjaon eri komponenttien välillä. Ongelman määrittelemiseen käytetään vaatimusmäärittelyä ja toiminnallista määrittelyä. Ratkaisuna ongelmiin on tekninen määrittely. Arkkitehtuurisuunnittelun jälkeen ohjelmisto ositetaan niin pieniin komponentteihin, että ne voidaan toteuttaa ohjelmointikielen avulla (Haikala & Mikkonen 2011,177-178.)

Arkkitehtuurisuunnittelu-termi käsitetään yleensä kahdella tavalla, joista ensimmäinen kattaa ohjelmiston komponentit ja niiden väliset suhteet sekä suuntaviivat jatkokehitykselle ja lyhyen kuvauksen suunnitteluperiaatteista, toisen katsotaan sisältävän sen osan ohjelmistoa, joka ei muutu normaalin ylläpidon aikana. Arkkitehtuuri siis tarjoaa puitteet ohjelmistokehitykselle, testaukselle ja ylläpidolle (Haikala & Mikkonen 2011, 178.)

Arkkitehtuurisuunnittelun tärkeimpiä asioita on selkeys ja suoraviivaisuus sekä riittävä suorituskyky. Näistä muodostuu ohjelman tärkeimmät laatuominaisuudet. Muita laatuominaisuuksia voivat olla esimerkiksi luotettavuus, muunneltavuus, ylläpidettävyys, testattavuus ja siirrettävyys. Edellä mainittujen laatuominaisuuksien saavuttamiseksi ohjelma pitää jakaa mahdollisimman pitkälle toisistaan riippumattomiin komponentteihin (Haikala & Mikkonen 2011, 179.)

Arkkitehtuurisuunnitteluun on olemassa erilaisia malleja, joista yksi käytetyimmistä on kuvassa 4.2 esitetty Kruchtenin 4+1 –malli.

Kuva 4.2 Kruchtenin 4+1 -malli. (Kruchten 1995).

Toteutus

Toteutusvaiheessa tehdään ohjelman tekninen toteutus. Toteutusvaiheessa abstraktiot toteutetaan komponenttien avulla. (Haikala & Mikkonen 2011.) Abstraktiolla tarkoitetaan käsitteitä ja niiden muodostamista toimintoina. Toteutusvaihe seuraa suunnittelua. Tässä vaiheessa ohjelmisto tai sen osa toteutetaan, jollain sovelluskehittimellä tai ohjelmointikielellä. Mikäli projektin aiemmat vaiheet on toteutettu asiallisesti eli kaikki ohjelmiston rakennetta ja toiminnallisuutta koskevat ratkaisut on tehty, toteutusvaihe on yleensä melko suoraviivainen toimenpide. Toteutusteknologian tai -kielen valinta ei pelkästään takaa onnistunutta toteutusta. Onnistumisen kannalta tärkeintä on se, että toteutus vastaa vaatimusmäärittelyä. Onnistunut toteutus on siis kiinni siitä kuinka hyvin edelliset vaiheet on toteutettu ja kuinka toteutuksen aikana ilmeneviin uusiin vaatimuksiin pystytään vastaamaan. Ylläpidettävyys ja siirrettävyys on toteutusvaiheen suurimpia haasteita. Näiden haasteiden kohtaaminen on helpompaa mikäli suunnittelu ja toteutus suoritetaan kurinalaisesti. Erilaisista laitteisto-, suorituskyky- ja järjestelmävaatimuksista johtuen ohjelmisto kannattaa osittaa mahdollisimman pieniin moduuleihin, jolloin siirrettävyys erilaisuus ympäristöihin

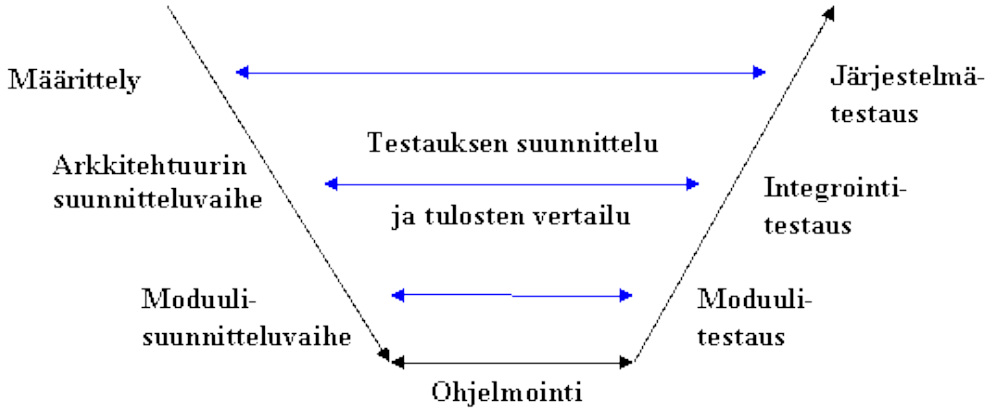
helpottuu ja testausvaiheessa virheiden löytyminen sekä korjaaminen ovat helpompia suorittaa. Kurinalaisuudessa ovat tärkeimpiä seikkoja ohjelmointityyli, vakiintuneet nimeämiskäytännöt ja dokumentointi. Tällöin muutkin kuin itse ohjelmoija pystyvät tarpeen vaatiessa tulkitsemaan ohjelmakoodia. (Pohjonen 2002, 34-35.)

Testaus

Ohjelmistojen testauksen yhteydessä testauksella tarkoitetaan järjestelmällistä virheiden etsintää. Testauksen työvaiheita ovat testauksen suunnittelu, testiympäristön luonti, testauksen suoritus ja tulosten analysointi. Lisäksi edellä mainittuihin läheisesti liittyy myös virheiden jäljitys ja korjaus, joihin kuluu yleensä yli puolet ohjelmistoprojektin resursseista. Testaus on siis olennainen osa ohjelmistoprojektia ja sen suorittamiseen on syytä kiinnittää huomiota. Testauksella on mahdollista osoittaa ohjelman virheiden olemassa olo, mutta virheettömyyttä sillä ei voida osoittaa. (Haikala & Mikkonen 2011, 205.)

Virhe on poikkeama spesifikaatiosta, joten johdonmukainen testaus ilman riittävää spesifikaatiota ei ole mahdollista. Tavallisimmat testauksessa käytetyt spesifikaatiot ovat toiminnallinen ja tekninen määrittely. Ongelmana on yleensä kuitenkin spesifikaatioiden tulkinta ja sen puutteellisuus. (Haikala & Mikkonen 2011, s. 205-206.)

Kehitystyö ja testauksen välistä suhdetta havainnollistetaan yleensä niin sanotulla V-mallilla, joka on esitetty kuvassa 4.3.



Kuva 4.3 Testauksen V-malli (Kautto 1996)

V-mallissa testaus suunnitellaan tapahtuvaksi testaustasoa vastaavalla suunnittelutasolla, ja testitulokset todetaan oikeiksi vertaamalla niitä vastaaviin dokumentteihin. (Haikala & Mikkonen 2011.)

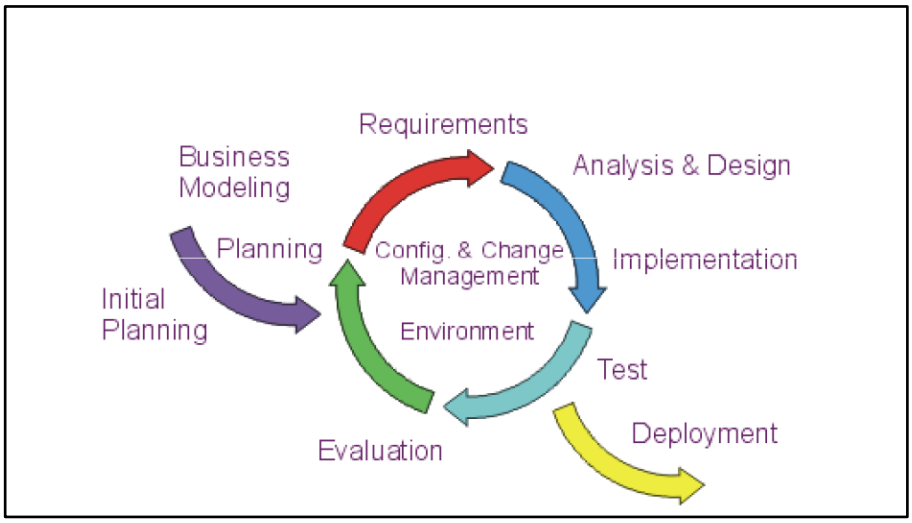
V-malli jakaa testauksen eri tasoihin, jotka ovat moduuli- eli yksikkötestaus, integrointitestaus ja järjestelmätestaus. Yksikkötestauksessa on testattavana yksittäinen moduuli tai luokka. Integrointitestauksessa testataan osajärjestelmiä, jotka on koostettu useammasta moduulista. Pääpaino integrointitestauksessa on rajapintojen toimivuus ja vertailukohteena on tekninen määrittely. Järjestelmätestaus tarkastelee koko järjestelmää ja tuloksia verrataan yleensä vaatimusmäärittelyyn, toiminnalliseen määrittelyyn ja käyttöohjeeseen. Testauksen määrään vaikuttavat yleensä käytettävissä olevien resurssien määrä. Testaukselle tulee aina antaa hyväksymisehdot, joilla testaus lopetetaan. Testauksesta tulee tehdä suunnitelma, josta selviää mitä, milloin ja miten on testattu sekä millaisia tuloksia odotettaan. Testauksen aikana esille tulleet virheet tulee raportoida ja analysoida. Virheraportteihin tulee kirjata ainakin seuraavat tiedot: virheen kuvaus, vakavuus, milloin ja miten se löytyi, olisiko sen voinut löytää jo aiemmin, milloin virhe on tehty ja olisiko sen syntyminen voitu estää. (Haikala & Mikkonen 2011.)

Käyttöönotto

Testauksen jälkeen ohjelmisto otetaan käyttöön. Käyttöönottoon liittyy myös useita asioita, jotka tulee ottaa ajoissa huomioon. Esimerkiksi tällaisia ovat tietojen, tiedostojen ja tietokantojen siirto uuteen ympäristöön. Lisäksi mahdolliset aiemmat ja rinnakkaiset järjestelmät tulee huomioida. Käyttäjien ja ylläpitäjien koulutuksesta tulee huolehtia riittävän ajoissa, minimi dokumentaationa tulee olla selkeä käyttöohjeistus. Koulutusta suunniteltaessa tulee miettiä kenelle, miten ja milloin se järjestetään? Tarvitaanko jollekin käyttäjäryhmälle erityiskoulutusta? Käyttöönottovaiheessa uusi järjestelmä saatetaan sijoittaa täysin uuteen laitteistoympäristöön ja uusiin tiloihin. Onko tämä ennakoitu riittävän ajoissa. Käyttöönottovaiheeseen liittyy useita kysymyksiä, joihin tulee löytää vastaus ajoissa. (Pohjonen 2002.)

4.2 Iteratiivisuus

Iteratiivinen ja inkrementaalinen ohjelmistokehitysprosessi etenee kuvan 4.4 mukaisesti.



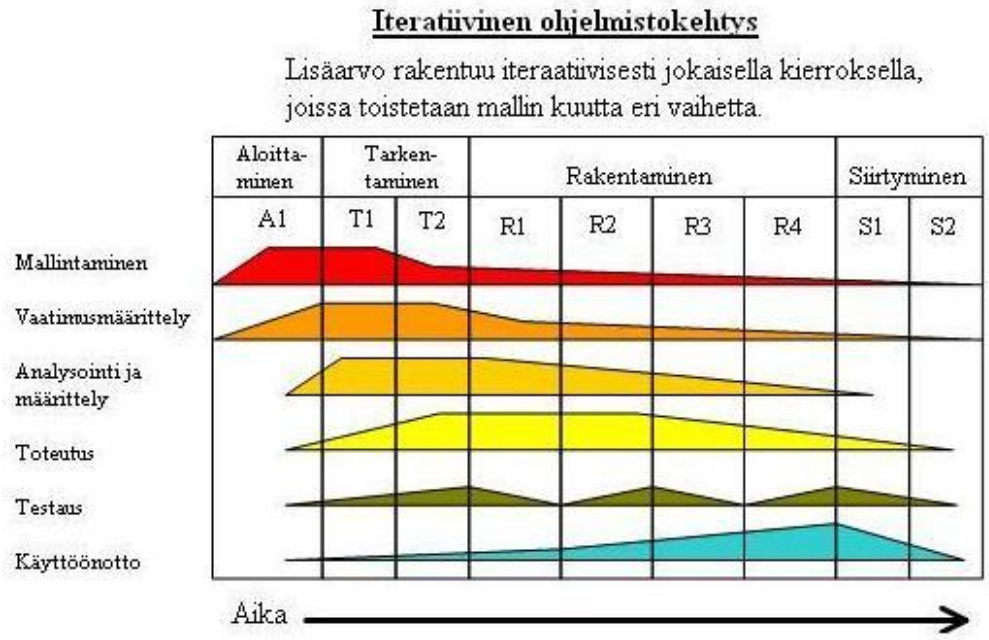
Kuva 4.4 Iteratiivinen ohjelmistokehitysprosessi (Kruchten. P. 1998)

Iteratiivisuus on projektimalliperhe, missä ohjelmiston arvioitu elinkaari on jaoteltu useisiin perättäisiin iteraatioihin (Haikala & Mikkonen 2011.)

Vesiputousmallin mukaisessa tuotekehityksessä tavallisesti kehityssuunnitelma määrittelee tulevaisuudessa tuotteeseen lisättävät ominaisuudet. Tähän kuitenkin liittyy muutamia ongelmia, kuten oppimisen tehokas hyödyntäminen, riskien hallinnan ennalta arvaamattomuus, riskien monimutkaisuus, vaatimusten tarkentuminen, myöhäinen asiakaspalaute, testauksen myöhäinen aloittaminen, itsepetos ja kärsimättömät asiakkaat. Ohjelmistoprosessi on tekninen suoritus, mutta myös oppimis- ja kommunikointiprosessi. Aiemmista versioista opitut asiat ovat unohtuneet. Riskit realisoituvat yleensä vasta testausvaiheessa, jolloin niille on myöhäistä tehdä mitään tai se tulee kalliiksi. Versioiden julkaisun välissä on vaatimuksissa tapahtunut tarkentumista tai on ilmennyt täysin uusia vaatimuksia. Iteratiivisessa kehitysmallissa edellä mainittuihin muutoksiin on vaikea reagoida ja versio voi olla jo vanhentunut käyttöönottovaiheessa, mikä selviää vasta asiakaspalautteesta. Testaus aloitetaan yleensä vasta myöhäisessä vaiheessa, jolloin löydettyjen virheiden korjaamiseen kuluu huomattavasti arvioitua pidempään. Projektisuunnitelmaa ei päivitetä. Määrittelyn ja suunnittelun val-

mistumispäivämäärät on lyöty lukkoon eikä huomioida onko esimerkiksi määrittely tehty riittävän hyvin suunnittelua silmällä pitäen, vaan todetaan sen olevan tarpeeksi hyvä. Jos sama itsepetos toistetaan suunnittelun päätteeksi, niin totuus realisoituu vasta testaus- ja toteutusvaiheessa. Projektin myöhästyessä asiakkaat alkavat olla kärsimättömiä tällöin myös projektihenkilökunnan usko projektin onnistumiseen saattaa horjua. (Haikala & Mikkonen 2011, 37-41.)

Edellisessä kappaleessa mainitut ongelmat ilmenevät jo hyvinkin yksinkertaisissa vesiputousmallin mukaisissa kehitysprosesseissa ja kertautuvat toistuvissa. Ratkaisuna tähän on esitetty tuotekehityssyklin lyhentämistä, jossa käytetään iteratiivista kehitystä. Syklin aikana toteutettavat vaatimukset voidaan jakaa tällöin iteraatioihin. (Haikala & Mikkonen 2011, 42.) Kuvassa 4.5 on esitetty iteratiivisen ohjelmakehitysmallin vaiheet.



Kuva 4.5 Iteratiivinen ohjelmakehitys (Partanen J. 2009)

4.3 Rational Unified Process eli RUP

RUP (Rational Unified Process) on iteratiivinen Rational Software Inc:n kehittämä räätälöitävissä oleva vesiputousmallia muistuttava prosessikehys, joka perustuu perättäisiin iteraatioihin. Sen rinnalla käytetään tekijänoikeudellisista syistä usein termiä UP eli Unified Process. (Wikipedia 2013.)

RUP-projektin iteraatiot muodostuvat neljästä vaiheesta: määrittely, suunnittelu, toteutus ja testaus. Näiden neljän vaiheen avulla prosessi voidaan esittää kuten vesiputousmallissa, vaikka pääpaino onkin jokaisen vaihekehityksen iteraatiossa. Jokaisella vaiheella on päämäärä, jonka avulla pystytään testaamaan onko tavoitteisiin päästy. Toteutusvaiheen tarkoitus on tutkia järjestelmää, tarkastaa kustannusarvio ja budjetti. Lisäksi luodaan käyttötapausmalli, projektisuunnitelma, riskiarviointi ja projektikuvaus. Tarkentumisvaiheen tarkoituksena on vähentää havaittuja riskejä. Lisäksi tarkentumisvaiheessa tehdään ongelma-alueanalyysi. Projektin perusarkkitehtuuri saa myös muotonsa tässä vaiheessa. Rakennusvaiheessa rakennetaan ohjelmisto. Suunnitellaan ja ohjelmoidaan komponentit ja toiminnot. Rakennusvaiheen seurauksena syntyy ensimmäinen ohjelman julkaistava versio. Siirtymisvaiheessa ohjelma muunnetaan tuotantokelpoiseksi. Lisäksi tämä vaihe sisältää käyttäjien koulutuksen ja ohjelman testaamisen. (IBM 2011.)

Mika Romppasen mukaan RUP on iteratiivinen ohjelmiston kehitysprosessi, jonka alkuperäinen idea on ollut sitoa UML-kuvaukset ja käyttötapaukset osaksi ohjelmistokehitysprosessia. Lisäksi sen tarkoituksena on luona prosessimalli, joka perustuu niin sanotusti parhaisiin käytäntöihin. RUP-prosessi korostaa seuraavia asioita: visuaalista mallintamista, iteratiivista kehittämistä, vaatimustenhallintaa, komponenttipohjaista arkkitehtuuria, laadunvarmistusta ja muutosten hallintaa. (Romppanen.)

Rational Unified Process esittää *Kuusi parasta tapaa* ohjelmistotuotantomallin, joka listaa vikojen minimoimiseksi ja tuottavuuden parantamiseksi seuraavat asiat joita ohjelmistokehityksessä tulee seurata:

- Kehitä iteratiivisesti eli selvitä vaatimukset mahdollisimman tarkkaan etukäteen
- Hallinnoi vaatimuksia eli pidä käyttäjien vaatimukset aina mielessä
- Käytä komponentteja eli osita projekti testauksen helpottamiseksi
- Suunnittele visuaalisemmin eli käytä kaavioita ohjelman hahmottamiseen
- Valvo laatua eli tee testauksesta tärkeä osa projektista
- Valvo muutoksia eli käytä versionhallintaa apuna (IBM 2011.)

4.4 Ketterät menetelmät

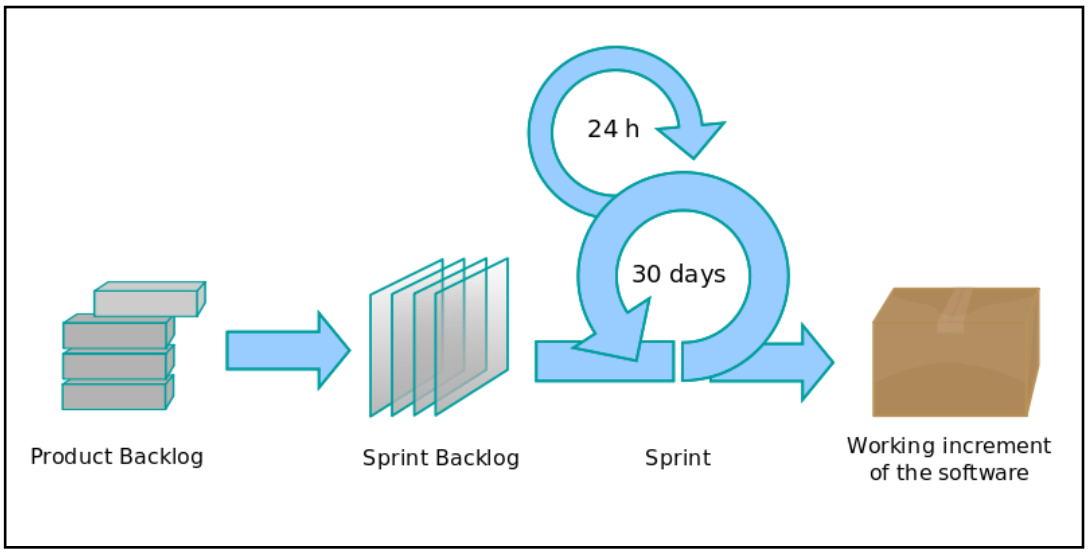
Ketteryydellä tarkoitetaan kevyttä iteratiivista ohjelmointikehitysmenetelmää, joista eniten huomiota on saanut Kent Beckin kehittämä XP (xXtreme Programming). Ketterät menetelmät on kehitetty vastapainoksi raskaille suunnitelmaohjatuille ohjelmistokehitysmenetelmille. Ketterien menetelmien edistämistä ajava Agile Alliance-järjestö on maininnut, että periaatteessa prosessi, työkalut, dokumentaatio ja niin edelleen ovat turhia. Järjestön mielestä tärkeintä on tyytyväinen asiakas ja toimiva ohjelmisto. (Haikala & Mikkonen 2011.) Järjestön 12 perusperiaatetta on lueteltu seuraavaksi:

1. Tärkeintä on täyttää asiakkaan vaatimukset julkaisemalla jatkuvasti ja aikaisin uusia hyödyllisiä versioita ohjelmistosta.
2. Hyväksytään ja otetaan vastaan muuttuvat vaatimukset, jopa kehityksen loppuvaiheessa. Ketterät menetelmät valjastavat muutoksen asiakkaan kilpailueduksi.
3. Luovutetaan toimivia versioita kehitettävästä ohjelmistosta säännöllisesti, mielellään lyhyin väliajoin muutamasta viikosta muutamaan kuukauteen.
4. Liiketoiminnan ammattilaisten ja kehittäjien täytyy työskennellä päivittäin yhdessä koko projektin ajan.
5. Rakennetaan projektit motivoituneiden yksilöiden ympärille ja annetaan heille ympäristö ja tuki jota he tarvitsevat, sekä luotetaan, että he saavat työn tehtyä.
6. Kaikkein tehokkain tapa välittää tietoa kehitystiimille ja kehitystiimissä, on kasvokkain tapahtuva keskustelu.
7. Toimiva ohjelmisto on ensisijainen edistymisen mitta.
8. Ketterät menetelmät suosivat kestävää kehitystä. Rahoittajien, kehittäjien ja käyttäjien tulisi kyetä pitämään jatkuvasti yllä tasainen työtahti.
9. Jatkuva huomion kiinnittäminen tekniseen laatuun, sekä hyvään rakenteeseen ja suunnitteluun, lisää ketteryyttä.
10. Yksinkertaisuus - taito maksimoida työn määrä, jota ei tarvitse tehdä - on olennaista.
11. Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat nousevat itseorganisoituvista tiimeistä.

12. Tasaisin väliajoin tiimi miettii miten voisi tulla entistä tuottavammaksi, ja sitten säätää ja muokkaa toimintaansa sen mukaisesti. (Kosonen 2005.)

4.4.1 Scrum

Viime vuosien käytetyin ketterä menetelmä on Scrum. Ensimmäisen kerran Scrum mainittiin Harward Business Review-lehdessä vuonna 1986 ja jo silloin esitettiin Scrum-menetelmän keskeisimmät periaatteet. Yksi Scrumin merkittävimmistä eduista on yksinkertaisuus. Scrum ei ole kuitenkaan projektinhallintamenetelmä, vaan projektin toteutusvaiheeseen tarkoitettu menetelmä toteuttaa sen iteraatiot. (Haikala & Mikkonen 2011, s. 47.) Kuvassa 4.6 on esitetty Scrum-prosessin periaate.



Kuva 4.6 Scrum –prosessikuvaus (Lakeworks 2009).

Scrum-prosessissa on kolme roolia: tuotteen omistaja, Scrum-mestari ja tiimi. Tuotteen omistajan rooli muistuttaa perinteisessä ohjelmistokehitysprosessissa tuotepäällikön roolia, Scrum-mestaria voi kuvailla tietynlaisena projektipäällikkönä ja tiimiä projektiryhmänä. Tuotteen omistajan tehtävä on vastata taloudellisesta tuloksesta ja hän toimii rajapintana sidosryhmien välillä. Lisäksi hän kerää ohjelmiston vaatimukset työlistaksi, jota hän myös ylläpitää. Työlista voi sisältää lähes mitä tahansa, mutta yleisimmin siihen listataan: tuotteen ominaisuuksia, käyttötapauksia, käyttäjätarinoita, vaatimuksia, virheraportteja, dokumentaation kehittämisideoita ja arkkitehtuurin parannusehdotuksia. Scrum-mestarilla ei ole päätösvaltaa, kuten projektipäälliköllä. Hänen tehtävänään on

vastata, että prosessia noudatetaan. Lisäksi hänen tehtävänään on valmentaa tuotteen omistajaa ja tiimiä. Pyrähdysten (Sprint) tulokset ovat myös hänen vastuullaan. Hän myös vastaa siitä, että tehtävää ei merkitä valmiiksi ennen kuin kaikki valmistumisen ehdot on täytetty. Tiimi on itseorganisoituva ja sen tulee mielellään koostua taustoiltaan erilaisista ohjelmistoalan ammattilaisista. Tiimin ideaali koko on noin seitsemän kokopäiväistä työntekijää. Tiimi vastaa pyrähdyksen osioinnista työntekijöiden kesken sopivan kokoisiin paloihin. (Haikala & Mikkola 2011, 47-49.)

Scrum-menetelmässä projekti etenee pyrähdyksissä (sprint), joiden pituus on 30 kalenteripäivää. Pyrähdys aloitetaan tyypillisesti suunnittelukokouksella, johon osallistuvat tuotteen omistaja, Scrum-mestari ja tiimi. Kokouksen kesto on päivä. Aluksi tuotteen omistaja esittää työlistan ja sen jälkeen sovitaan, mitkä työlistan alkiot otetaan mukaan seuraavaan pyrähdykseen. Työlista ositetaan 4-16 tunnin tehtäviin. Pyrähdykseen sovitut tehtävät tehdään ja työlistaa ei saa pyrähdyksen aikana muuttaa. Pyrähdyksen aikana pidetään päivittäin noin 15 minuutin Scrum-kokous, jossa jokainen tiimin jäsen vastaa seuraaviin kysymyksiin: Mitä olet tehnyt edellisen kokouksen jälkeen? Mitä teet seuraavaksi? Mitkä esteet hidastavat mahdollisesti työskentelyäsi? Pyrähdyksen katselmointikokous pidetään pyrähdyksen jälkeen, johon osallistuvat tiimi, Scrum-mestari, tuotteen omistaja ja mahdollisesti myös sidosryhmien edustajia. Tiimi informoi sidosryhmien edustajia pyrähdyksen tuloksista ja kerää heiltä palautetta. Pyrähdyksen arviointipalaverin, joka on kestoltaan muutaman tunnin mittainen, tarkoituksena on arvioida pyrähdyksen onnistuminen tiimin, Scrum-mestarin ja tuotteen omistajan kesken. Lisäksi voidaan miettiä toimintamallien muuttamista ja kehittämistä seuraavaan pyrähdykseen. (Haikala & Mikkonen 2011, 50-51)

4.4.2 Lean

Lean on pikemminkin ajattelumalli, jota käytetään apuna prosessin kehittämisessä, kuin itse menetelmä. Ajattelumallin ovat kehittäneet Mary ja Tom Poppendieck. He ovat kirjoituksissaan esittäneet useita eri versioita ajatusmallinsa perusperiaatteina, mutta kaksi ovat keskeisimpiä. Ensinnäkin pyri eliminoimaan hukka. Hukka on kaikki, mikä ei tuota asiakkaalle lisäarvoa. Myös varastointi on

kärjistettynä hukkaa. Toinen keskeisistä periaatteista on ihmiskeskeisyys. Eli keskity niihin työntekijöihin, jotka tuottavat lisäarvon eli ”duunareihin”. Edellä mainittuun liittyy myös työntekijöiden arvostus, tehokas kommunikointi, hyvä työilmapiiri sekä koulutus ja toiminnan tehostaminen (Haikala & Mikkonen 2011, 54-55.)

Ilkka Kouri on esitelmässään vuonna 2010 Lean management maininnut, että Lean on johtamistapa ja -kulttuuri, joka vaatii sitoutumista ja pitkäjänteisyyttä. Tulokset Lean-mallissa ovat seurausta siitä, että asiat tehdään oikein kaikilla yrityksen tasoilla. Hänen mukaansa myös nopea läpäisyaika ja tarve ovat Lean-mallissa tuotannon lähtökohtina. Kouri myös listaa edellisessä kappaleessa mainittuja hukkatekijöitä, jotka hänen mielestään ovat: ylituotanto, turhat varastot, tarpeeton kuljettelu ja liikkuminen, odottelu ja joutoaika, yli- tai virheellinen käsittely, virheelliset tuotteet ja inhimillisten voimavarojen tuhlaus. (Kouri 2010.)

4.4.3 Kanban

Kanban on työnhallintatapa, jossa on vain kolme perusperiaatetta. Mallin on kehittänyt David Anderson. Työn kulkua visualisoidaan alla kuvassa 4.7 kaltaisella taululla. Työt ovat kukin omalla kortillaan ja ne siirtyvät taululla sarakkeesta toiseen riippuen missä vaiheessa ne ovat. Kussakin vaiheessa töiden määrä on rajoitettu. Tätä kutsutaan kaistanrajoittimeksi. Läpimenoajan mittaamien eli kehitysprosessia voidaan parantaa ja optimoida, kun tiedetään kunkin työn eri vaiheissa viettämä aika. Peruserona Kanbanin ja Scrumin välillä on pyrähdysten poistaminen. Seuraava työ voidaan aloittaa kun kaistalla on tilaa. Toimintatapa sopii esimerkiksi ylläpitotehtäviin. Scrum ja Kanban-malleja on myös yhdistelty, jolloin on syntynyt niin sanottu Scrumban. (Haikala & Mikkonen 2011, 55-56.)

Mitä tehtävää vielä on	Työn alla	Valmis
tehtävät c, d, e	tehtävä b	tehtävä a

Kuva 4.7 Yksinkertainen Kanban-prosessitaulu.

5 Työssä käytetyt tekniikat

Tässä luvussa käsitellään perusteet opinnäytetyössä käytetyistä tekniikoista, joita ovat relaatiotietokannat, SQL-kieli ja UML-kieli.

5.1 Relaatiotietokanta

Tietokanta on tietovarasto tai tietopankki. Tietokannan ei tarvitse välttämättä olla sähköisessä muodossa vaikka näin yleisesti luullaankin. Tietokanta voi olla manuaalinen kortisto tai jopa kalenteri. Puhuttaessa tietokannasta se yleensä kuitenkin käsitetään sähköiseksi tietovarastoksi, jossa on toisiinsa yhteydessä olevaa merkityksellistä tietoa. Tietokanta on rakennettu jotain tiettyä tarkoitusta ja käyttäjäkuntaa varten. Tietokantoja voi olla kooltaan ja rakenteeltaan hyvinkin erilaisia. Tietokannan perusrakenteen tulee olla sellainen, että kukin tieto tallennetaan vain yhteen paikkaan, tietoa voidaan hakea joustavasti, rakenteen muuttaminen on joustavaa ja se on tietoriippumaton. (Ekonoja 2004.)

Relaatiomallin esitti vuonna 1970 E.F.Codd. Relaatiomalli on matemaattinen teoria ja se ei ota kantaa tietokantojen fyysiseen toteutukseen. Tärkeimpänä relaatiotietokantojen etuna pidetään tietojen helppoa saatavuutta. Tiedot relaatiotietokannassa ilmaistaan tauluina ja niiden välistä yhteyttä kutsutaan relaatioksi. Tietoa relaatiotietokannoista haetaan vain tiedon nimien tai arvojen perusteella. Jokaisella taulun rivillä on yhtä monta kenttää ja jokaisessa kentässä tulee olla yksiselitteinen perusavain. Perusavain vastaa jotain reaalimaailman kohdetta. Jokaiseen kohteeseen sisällytetään vain välittömästi liittyvät ominaisuudet. Yksittäinen tieto voidaan relaatiotietokannasta hakea taulun nimellä, perusavaimella, avaimen arvolla, kentän nimellä ja lisäksi monilla muilla tavoilla. Luotaessa kenttiä niille yleensä määritellään seuraavia ominaisuuksia: kentän nimi, tietotyyppi, maksimipituus, tiedon pakollisuus, oletusarvo ja syöttömaski. Lisäksi on päätettävä käytetäänkö kiinteä vai vaihtuvamittaista kenttää. Tietueet koostuvat yhdestä tai useammasta kentästä. Taulun yksi rivi on tietue. Jokaisella taululla tulee olla määriteltynä perusavain, joka yksilöi taulun sisältämät tietueet. Perusavain muodostetaan vähintään yhdestä taulun kentästä. Perusavain ei saa puuttua, koska tietokannan hallintajärjestelmä pitää taulut järjestyksessä sen avulla. Perusavaimien lisäksi relaatiotietokannassa voi olla niin sanottuja

toissijaisia avaimia, niiden tarkoitus on nopeuttaa tiedon hakemista tietokannasta. Relaatiotietokantoihin liittyy myös käsite viite-eheys jolla tarkoitetaan sitä, että jonkin taulun kentästä viitataan jonkin toisen taulun perus- tai toissijaiseen avaimeen. Tässä yhteydessä puhutaan äiti- ja lapsi-tauluista. Lapsi-tauluksi kutsutaan taulua, josta viitataan avaimen avulla äiti-tauluun. Tietokannan hallintajärjestelmiä ovat muun muassa MySql, Oracle ja Access. Tietokannan hallintajärjestelmien perusvaatimuksia ovat muun muassa: skaalautuvuus, tehokkuus, yhteensopivuus, tietoriippumattomuus, turvallisuus, tiedon eheys, perusominaisuudet (haku, tallennus, päivitys) ja yhteiskäytön mahdollisuus. (Ekonoja 2004; Sarja 2006.)

5.2 SQL

SQL-kieli (Structured Query Language) kehitettiin IBM:n Database 2-tietokannan hallintajärjestelmään varten 1970-luvulla lopulla. Ensimmäinen kaupallinen versio kielestä esiteltiin 1979. ANSI-standardi kielestä julkistettiin 1986. SQL-kieli on muodostunut standardiksi relaatiotietokantojen kyselykielenä ja lähes kaikki keskeiset tietokantojen hallintajärjestelmät hyödyntävät sitä. SQL-kielen avulla relaatiotietokannoissa voidaan suorittaa seuraavia toimintoja: tietokannan rakenteen muokkaus, turva-asetusten ja käyttäjäoikeuksien muokkaus, kyselyjen suoritus ja tietokannan päivitys. (Huotari 2012; 2Kmedia. 2013.) SQL-kieli on joukko-orientoitunut ja se ei ole menettelytapoihin sidottu. Se ei ole myöskään riippuvainen järjestelmästä tai laitteistosta. (Savon ammatti- ja aikuisopisto 2013.) SQL-kieli voidaan upottaa myös sovellukseen ja siitä on eri valmistajilla omia versioitaan.

5.3 UML

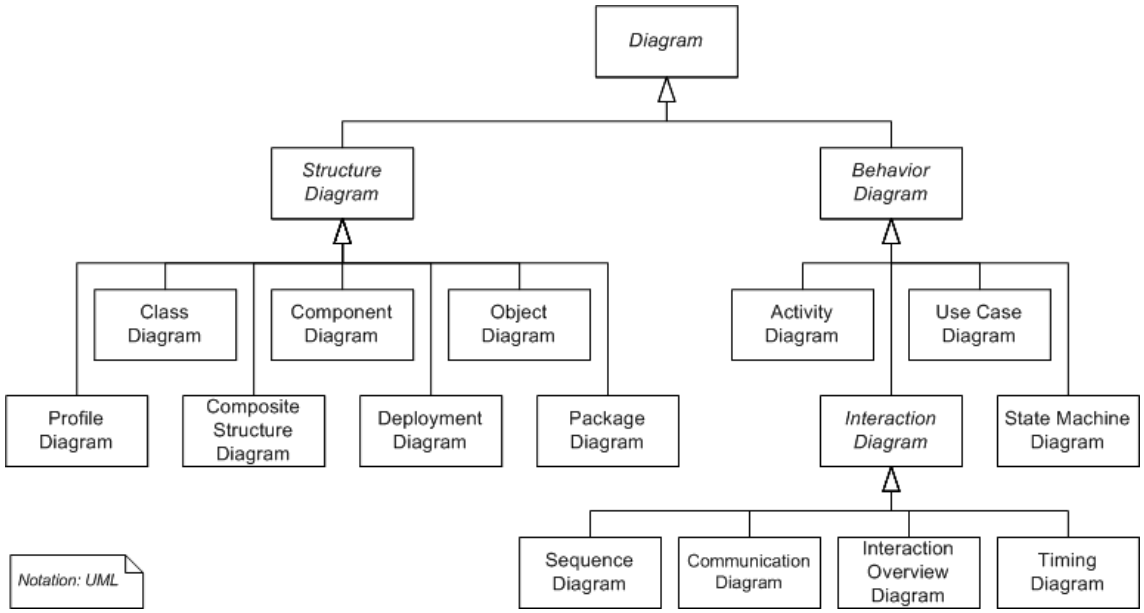
UML-kieli eli Unified Modeling Language on kehitetty oliopohjaisen ohjelmistosuunnittelun käyttöön. Sen tarkoitus visualisoida ohjelmistojen toimintaa, sisältöä ja järjestelmää, jossa ohjelmistoa suoritetaan. UML-kieli on laitteisto ja järjestelmä riippumaton. (Heikkinen 2008.)

UML-kielen historia ulottuu vuoteen 1994, jolloin Grady Booch ja James Rumbaugh alkoivat yhdistää kehittelemiään oliomallinnustekniikoitaan. OMG (Object Management Group) esitti vuonna 1996 ensimmäisen version kielestä. Nykyisin

kielestä on muodostunut standardi ja se on päivittynyt versioon 2. UML-kielen kehittäjät ovat sisällyttäneet kieleen aiemmin kuvassa 4.4 esitetyn Kruchtenin 4+1-mallin, joka helpottaa UML-mallin tulkintaa. (Raisamo 2007; Heikkinen 2008.)

UML-kielen kaaviotyypit

UML-kielen versiossa 2 on määritelty erilaisía kaaviotyyppejä, joita on kaikkiaan 13 ja ne on jaettu kolmeen ryhmään: rakennetta kuvaaviin, käyttäytymistä kuvaaviin ja vuorovaikutusta kuvaaviin. Kaavioita ei tarvita tiettyyn ohjelmiston määrittelyn tai suunnittelun vaiheeseen vaan niiden käytön ratkaisee millaisesta ohjelmistosta on kyse. (Heikkinen 2008.) Kuvassa 5.1 on esitetty UML-kaaviohierarkia.

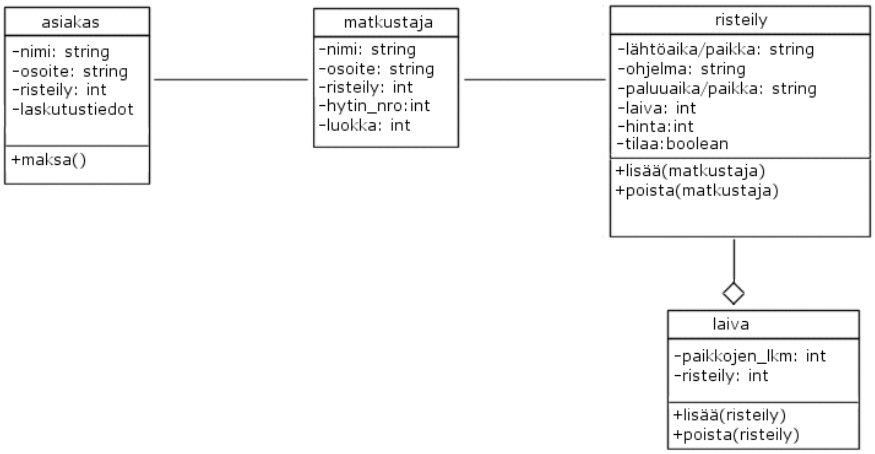


Kuva 5.1 UML-kielen kaaviohierarkia (Merson 2011.)

Seuraavaksi käsitellään hieman tarkemmin seuraavat kaaviotyypit: luokkakaavio (class diagram), käyttötapauskaavio (use case diagram), sekvenssikaavio (sequence diagram), yhteistyökaavio (interaction diagram, collaboration diagram), tilakaavio (state machine diagram), ja aktiviteettikaavio (activity diagram).

Luokkakaavio

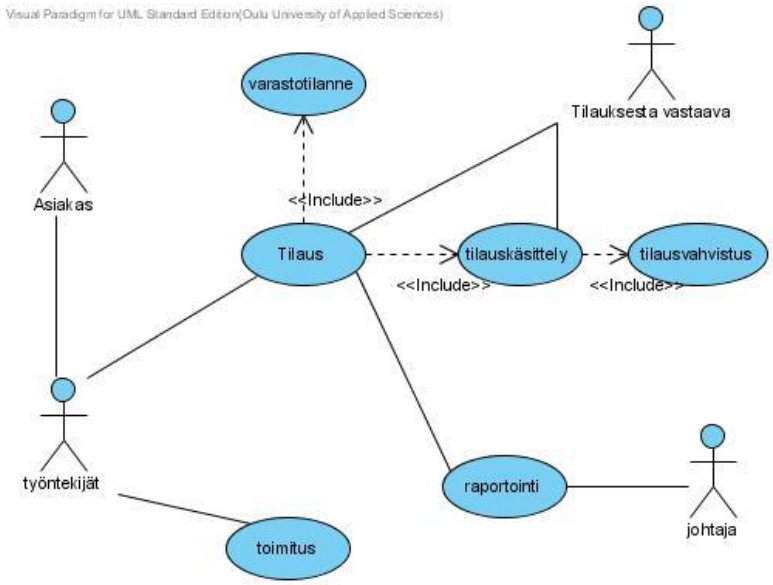
Luokkakaavio on työkalu järjestelmän staattiselle kuvaamiselle. Luokkakaavio kuvaa järjestelmästä loogiset kokonaisuudet, mahdolliset operaatiot, eri kokonaisuuksien tietosisällöt ja eri kokonaisuuksien väliset suhteet toisiinsa. Luokkakaavion peruskomponentti on luokka, joka sisältää operaatiot ja ominaisuudet. (Lappeenrannan teknillinen yliopisto 2003.) Kuvassa 5.2 on esitetty esimerkki yksinkertaisesta luokkakaaviosta.



Kuva 5.2 Esimerkki yksinkertaisesta luokkakaaviosta (Heimonen 2002).

Käyttötapauskaavio

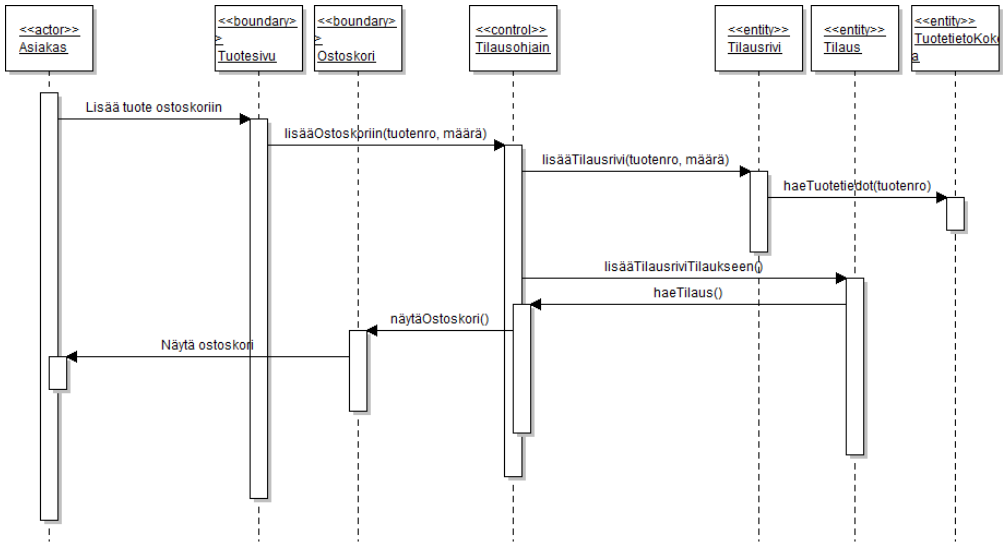
Käyttötapauskaavioita käytetään havainnollistamaan käyttäjille ohjelmiston tarjoamia toimintoja sekä palveluita. Käyttötapauskaavioista ja niihin liitetyistä selitteistä muodostuu käyttötapausmalli. Käyttötapauskaavioita käytetään yleensä ohjelmiston kehitystyön alkuvaiheessa määritettäessä toiminnallisia vaatimuksia, mutta niitä voidaan käyttää vielä testausvaiheessakin todentamaan ohjelman vaatimusten mukainen toiminta. Käyttötapauskaaviot eivät vielä ota kantaa ohjelmiston rakenteeseen tai toimintaan. Käyttötapauskaavio on käyttäytymiskaavio, jonka peruskomponentteja ovat käyttäjät ja käyttötapaukset. Käyttötapauksella tulee olla selkeä lähtökohta, päämäärä ja merkitys käyttäjälle. (Korpimies 2011.) Kuvassa 5.3 on esitetty yksinkertainen käyttötapauskaavio.



Kuva 5.3 Esimerkki käyttötapauskaaviosta (Ukonaho & Remes 2010).

Sekvenssikaavio

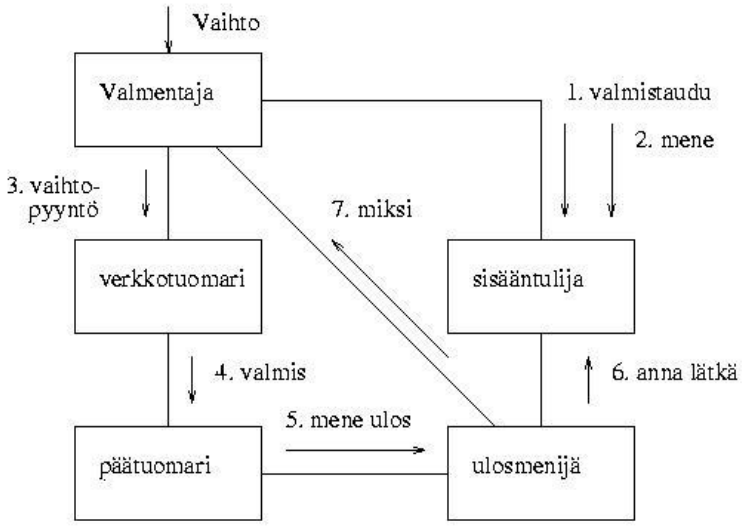
Sekvenssikaavion tehtävänä on kuvata oliojoukon välistä vuorovaikutusta tietyssä tilanteessa. Kaaviossa aika on pystyakselilla ja sen kulkusuunta on ylhäältä alas, vuorovaikutukset ovat vaakasuuntaisia nuolia ja osallistujat ovat pystysuuntaisia viivoja. Sekvenssikaavion etuna on, että se pystyy hyvin osoittamaan olioiden välisen yhteistyön. Haittapuolena on, etteivät kuvaa käyttäytymisen täsmällistä rajausta. Sekvenssikaavioita kannattaa hyödyntää vain tärkeimmissä suoritusoluissa mahdollisine poikkeuksineen. Lisäksi sekvenssikaavioiden avulla voi tunnistaa kokonaan uusia luokkia ja päivittää ne luokkakaavioon. Sekvenssikaavioiden dokumentointi koostuu luokkien ja käyttötapausten dokumenteista. (Raisamo 2007; Kokkoniemi 2009.) Kuvassa 5.4 on esitetty esimerkkinä yksi verkkokaupan käyttötapaus sekvenssikaaviona.



Kuva 5.4 Esimerkki sekvenssikaaviosta (Harju 2011).

Yhteistyökaavio

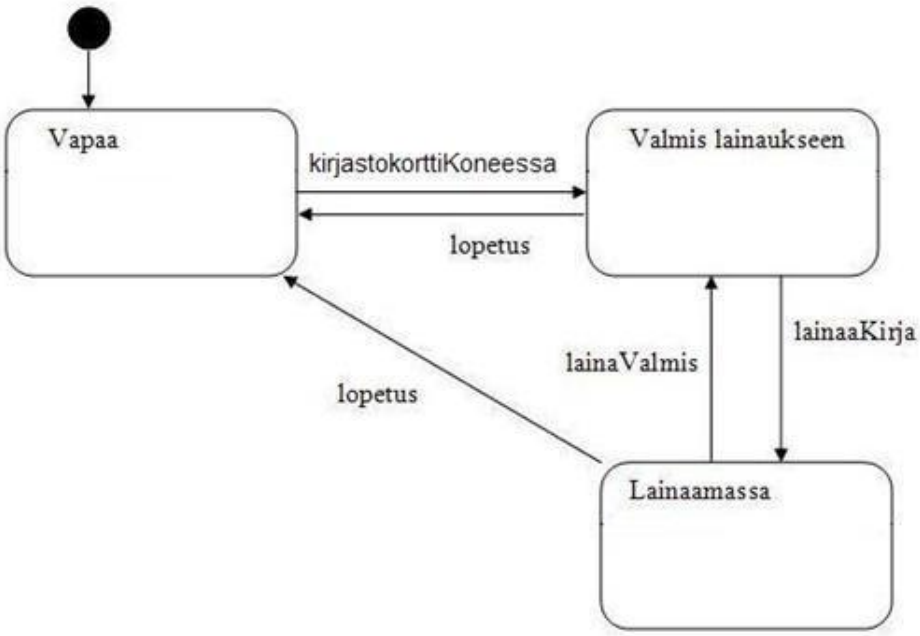
Yhteistyökaavion tehtävä on kuvata olioiden välistä vuorovaikutusta. Olioiden yhteydet kuvataan symbolien välisinä yhtenäisinä viivoina. Yhteistyökaavio on vaihtoehto sekvenssikaaviolle. Yhteistyökaavion kuvaustekniikka muistuttaa selkeästi ohjelmointia ja aliohjelmakutsuja. (Raisamo 2007; Nikula 2009.) Esimerkki yhteistyökaaviosta on kuvassa 5.5.



Kuva 5.5 Esimerkki yhteistyökaaviosta (Taina 1999).

Tilakaavio

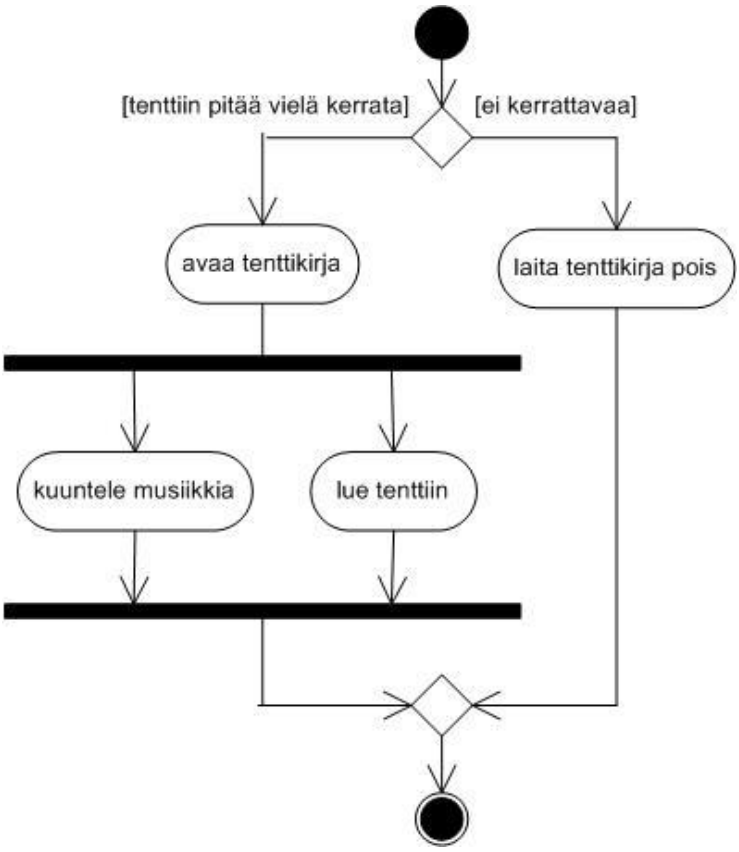
Tilakaavioilla kuvataan yleensä järjestelmän tai sen osan siirtymistä tilasta toiseen. Tilakaavioilla voidaan antaa parempi yleiskuva järjestelmän toiminnasta kuin esimerkiksi sekvenssikaavioilla. Tilakaavion keskeisimmät komponentit ovat tilat ja siirtymä. Tila kuvataan kaaviossa pyöristettynä suorakaiteena ja siirtymät nuolilla. Kaavio voi sisältää myös alku- ja lopputilan. Tilakaavioita voidaan käyttää tarvittaessa myös muille olioille, joiden käytös voidaan ymmärtää tilojen vaihtumisena. (Raisamo 2007; Korpimies 2011.) Esimerkki tilakaaviosta on kuvassa 5.6.



Kuva 5.6 Esimerkki Tilakaaviosta (Korpimies 2011)

Aktiviteettikaavio

Aktiviteettikaavio eli toimintokaavio on muunnelma tilakaaviosta, jolla voidaan kuvata olion, olioiden tai koko ohjelman toimintaa. Aktiviteettikaavioon voidaan liittää useampi käyttötapaus ja lisäksi kaavio voidaan jakaa sarakkeisiin, jotka vastaavat kutakin suorittajaa (Raisamo 2007; Niinimäki 2011.) Kuvassa 5.7 on esimerkki aktiviteettikaaviosta.



Kuva 5.7 Esimerkki aktiviteettikaaviosta (Korpimies 2011).

6 Opinnäytetyöprojektin kulku

Opinnäytetyössä on noudatettu Saimaan ammattikorkeakoulun opinnäytetyöprosessia, joka sisältää seuraavat vaiheet: aiheen valinta, suunnittelu, toteuttaminen, raportointi ja työn viimeistely.

Omassa opinnäytetyössäni olen noudattanut opinnäytetyöprosessin vaihejakoa, mutta olen nimennyt vaiheet ohjelmistoprojektin vaiheiden mukaisesti seuraavasti:

- Aiheen valinta	Esitutkimus
- Suunnittelu	Projektin suunnittelu
- Toteuttaminen	Toiminnallinen määrittely
- Raportointi	Raportointi
- Työn viimeistely	Työn viimeistely

6.1 Esitutkimus

Esitutkimuksen tavoitteena on selvittää alustavasti ohjelman tarpeellisuus ja sen tuoma hyöty, toiminnalliset vaatimukset pääpiirteittäin ja toteutukseen liittyvät riskit.

Ohjelma katsottiin asiakkaan kannalta tarpeelliseksi ja sen tuoma ajallinen hyöty riittäväksi toteutusta ajatellen. Lisäksi asiaa edesauttoi se, että teen toiminnallisen määrittelyn oman katsastajan työni ohessa sekä omalla ajalla ilman erityistä korvausta.

Ohjelman esitutkimuksessa tarpeelliseksi ominaisuuksiksi listattiin seuraavat asiat:

- ohjelman käyttö tulee rajata erilaisilla käyttöoikeuksilla (esimiehet muokkaus ja työntekijät vain luku)
- henkilöstön eri katsastusoikeudet tietokannassa
- työvuorot tietokannassa, lisäys/muokkaus mahdollisuus
- tulostusmahdollisuus (työvuorolista)
- päiväkohtainen muokkausmahdollisuus: lomat, vapaapäivät, sairauslomat, koulutus ja osa-aikaisuus

- työvuorolistan postitusmahdollisuus

Toivottavia lisäominaisuuksia ovat:

- osittainen toimintojen automatisointi
- varmistus eri katsastusoikeuksien omaavien katsastajien riittävyydestä eri aikoina: lomat, vapaapäivät, sairauslomat, kurssit, eri työvuorot ja toimipisteet

Toimintolistauksen määrittämistä varten keskustelin A-Katsastuksen Etelä-Karjalan aluepäällikön Jari Joenperän kanssa sekä oman työni ohessa paikallisten esimiesten Harri Hietalan ja Riitta Kuosan kanssa.

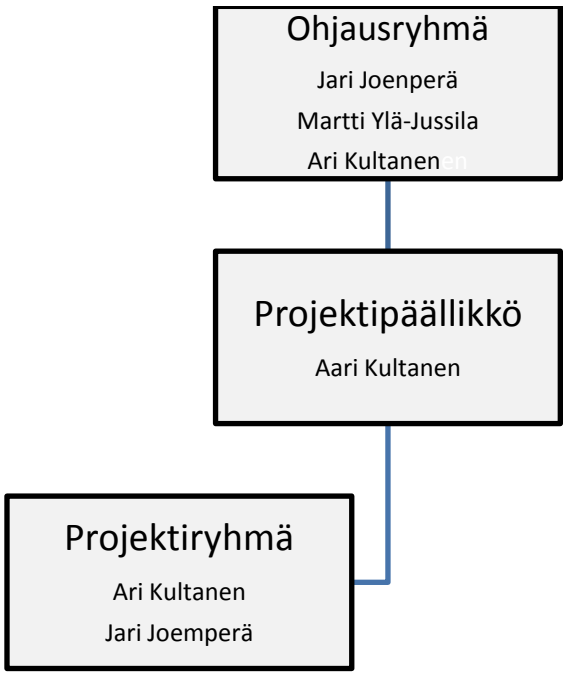
6.2 Projektin suunnittelu

Projektin lähtökohtana oli helpottaa ja tehostaa esimiesten ajankäyttöä työvuorojen suunnittelussa. Projektin tarkoituksena on luoda toiminnallinen määrittely ja käyttötapauskuvaukset sekä niihin liittyvä dokumentaatio KATASO-ohjelmaa varten. Ajatus opinnäytetyön aiheeksi syntyi seuratessani esimiesten työskentelyä työvuorojen suunnittelun parissa. Tällä hetkellä esimiehet syöttävät työvuorot manuaalisesti sitä varten luotuun Excel-taulukkoon, joka vie aikaa. KATASO-projekti suunniteltiin toteutettavaksi opinnäytetyönäni.

Dokumentaationa syntyi esitutkimusraportin oheen projektisuunnitelma, jossa käydään läpi ohjelman tarkoitus ja kattavuus, tuote ympäristöineen, määritelmät, termit sekä lyhenteet. Lisäksi projektisuunnitelmassa käydään läpi nykyinen järjestelmä sekä muut mahdolliset järjestelmät, projektiorganisaation, tavoitteet, keskeyttämiskriteerit ja päättäminen. Projektin ositus, työmäärän arviointi, käytetyt menetelmät ja tekniikat ovat osa projektisuunnitelmaa kuten myös palaverikäytännöistä sopiminen. Se kuinka riskejä hallitaan ja mitä riskejä tiedetään tai oletetaan jo olevan, kuuluu myös projektisuunnitelmaan. Projektin kustannusarvio, hylätyt ideat ja jatkokehitysajatukset ovat projektisuunnitelman osia. Suunnitelmat ohjelmistoon liittyvästä koulutuksesta, asennuksesta ja käyttöönotosta mainitaan myös projektisuunnitelmassa. Lopuksi arvioidaan projektin onnistuminen.

Organisaatio

KATASO-projektissa projektiorganisaatio (kuva 6.1) koostui kolmesta henkilöstä. Projektipäällikkönä ja tiiminä toimin itse (Ari Kultanen), opinnäytetyönohjaaja lehtori Martti Ylä-Jussila ja asiakkaan edustajana aluepäällikkö Jari Joenperä Lappeenrannan A-Katsastuksesta.



Kaavio 6.1 KATASO-projektiorganisaatio

Projektiorganisaation sidosryhmälle ja henkilöille on jaettu vastuualueet. Projektipäällikkönä vastuullani on projekti kokonaisuudessaan. Muutamia tärkeimpiä vastuita ovat esimerkiksi projektisuunnitelman laatiminen, projektin käynnistäminen, resursseista vastaaminen, vastaaminen aikatauluista ja työn edistymisestä sekä vastaa dokumentoinnista ja laatii loppuraportin.

Martti Ylä-Jussilan vastuulla on Saimaan ammattikorkeakoulun opinnäytetyöprosessin ohjeiden noudattamisen valvonta, oppimiseni edistäminen ja palautteen sekä ideoiden antaminen.

Asiakkaan, A-Katsastus Oy, vastuulla tarjota riittävät resurssit projektin läpi vientiin sekä määritellä projektin tavoitteet ja tarpeet projektipäällikön kanssa.

Vaiheet aikataulu, työmääräarviot

Projektin kokonaistyömääräksi on arvioitu 400 tuntia. Projektin vaiheistusta, aikataulutusta sekä työmääräarvioita on havainnollistettu taulukossa 6.1.

Tehtävä	Arvio alkup.	Arvio nyt	Suunniteltu	tammi.12	helmi.12	maalis.12	huhti.12	touko.12	kesä.12	heinä.12	elo.12	syys.12	loka.12	marras.12	joulu.12	tammi.13	helmi.13	maalis.13	huhti.13	touko.13	toteuma
	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	
Teorian opiskelua 90h	**140**	**160**	**140**	17	17	13	5	0	0	0	0	15	10	10	0	10	18	20	5	0	**140**
- Kirjallisuus	25	30	25	12	12	3											3				30
- Toiminnanohjaus	3	3	3			3															3
- Toiminnallinen määrittely	80	100	80									10	10	10		10	15	20	5		80
- UML	10	5	10			5	5														10
- StarUML	2	2	2			2															2
- Tietokannat	10	15	10	5	5																10
- Muut	10	5	10									5									5
Määrittely 100h	**131**	**151**	**131**	8	8	16	6	0	0	0	0	17	17	17	0	12	17	2	10	0	**130**
- Katsastusaseman toimintakuvaus	5	5	5			5															5
- Käyttötapauskuvaukset	40	45	40			4							10	10		5	10				39
- Käsitteet, tiedot, tietokanta	40	40	40			1	4					10	5	5		5	5				35
- Ei toiminnalliset vaatimukset	5	5	5									5									5
- Arkkitehtuurin suunnittelu	6	6	6			6															6
- Suunnittelu, pähkäilyä	20	30	20	8	8		2					2	2	2		2	2	2			30
- Katselmointi	10	10	10																		0
- Dokumentin siistiminen	5	10	5																10		10
Tekniikoiden valinta/opiskelu 100h	**40**	**40**	**40**	15	0	15	0	0	0	0	0	0	0	0	0	10	0	0	0	0	**40**
- Vaihtoehtoihin tutustuminen 10h	10	10	10			10															10
- Tietokannat (Access 2007) 10h	10	10	10	5		5															10
- Palvelimet 10h	5	2	5																		0
- Työkalut 20h	20	15	20	10												10					20
Projektin hallinta 30h	**28**	**32**	**30**	1	0	14	5	4	0	0	0	1	0	1	0	1	0	1	0	6	**34**
- Ohjauspalaverit ja muistiot	8	12	10			4	4	4													12
- Projektisuunnittelma	10	10	10			9	1													5	15
-Aikataulu, tuntiseuranta	10	10	10	1		1						1		1		1		1		1	7
Opinnäytetyön raportti ja rutiinit 80h	**81**	**81**	**81**	14	0	0	0	0	0	0	0	0	0	0	0	0	0	20	20	32	**86**
- Esiselvitys	15	15	15	10																	10
- Aiheanomus	2	2	2	2																	2
- Kypsyysnäyte	3	3	3																	3	3
- Seminaariesitys 1 ja 2	4	4	4	2																2	4
- Raportin kirjoitus	50	50	50															20	20	20	60
- Kieliasun tarkistus	5	5	5																	5	5
- Raportin julkaisu	2	2	2																	2	2
Yhteensä h/kk	**420**	**428**	**422**	55	25	58	16	4	0	0	0	33	27	28	0	33	35	43	35	38	**430**

Taulukko 6.1 Opinnäytetyön aikataulutus

Työskentelytapa

Ensimmäisenä kokoonnuttiin projektiryhmän kanssa miettimään projektin tarpeellisuutta ja sovittiin, että laadin aiheesta esitutkimusraportin. Raportin valmistuttua keväällä 2012 päätettiin projekti aloittaa.

Projektin aluksi tarkoituksenani oli perehtyä aiheeseen kirjallisuuden avulla. Aiempi kokemukseni projektityöskentelystä on erittäin vähäinen. Vaatimusmäärittely käsitteenä oli tuttu, mutta en ollut aiemmin tutustunut tarkemmin aiheeseen.

Työn edistymistä seurattiin pitämällä noin kerran kuukaudessa ohjaustyöryhmän palaveri, jossa käytiin lävitse mitä olen saanut aikaan ja mitä aioin seuraavaksi tehdä. Ohjaustyöryhmän palavereista laadittiin muistiot sekä projektisuunnitelmaa päivitettiin palaverien jälkeen. Tämän lisäksi pidimme opinnäytetyöni ohjaajan Martti Ylä-Jussilan kanssa seurantapalavereja, joissa kävimme lävitse ideoita ja ajatuksia projektin etenemistä ajatellen.

Oman työni edistymisestä pidin päiväkirjaa Excel-taulukon muodossa, jonka päivittäminen usein unohtui.

6.3 Määrittely

Määrittely aloitettiin esitutkimuksessa kerättyjen vaatimusten analysoinnilla.

Toiminnalliseen määrittelyyn laadittiin ensimmäiseksi käyttötapausluettelo ja jokaiselle käyttötapaukselle lyhyt kuvaus. Ohjelman käyttötapausten ja toimijoiden välistä yhteyttä havainnollistettiin UML:n käyttötapauskaaviolla

Kolmantena tehtävänä oli määritellä mitä tietoja ohjelma tarvitsee toimiakseen ja kuinka ne tallennetaan. Tallennusratkaisuksi valitsin relaatiotietokantarakenteen. Tämän jälkeen kuvasin tietokannan rakenteen domain model -kaaviolla ja laadin jokaiselle yksilötyypille oman taulukon, jossa kuvattiin yksilötyypin attribuutit.

Seuraavaksi suunnitteluvuorossa oli ohjelman käyttöliittymä ja sen lomakkeet. Lomakkeiden havainnollistamiseen käytin Microsoft C#:n lomakkeen suunnittelutyökalua. Tietyn toiminnon lomakkeen hahmotellun jälkeen aloin miettiä kuinka lomakkeen tulee toimia ja kirjoitin sen käyttötapauskuvauksen taulukkomuotoon. Esimerkkejä on esitelty luvussa 7. Lisäksi tiettyjä toimintoja kuten päävalikon ja työvuorojen generoinnin havainnollistamiseksi käytin vuokaavioita. Vuokaavioiden avulla esimerkiksi työvuorojen generointi on helpompi ymmärtää kuin lukemalla tekstiä. Toiminnallisen määritelmän lopuksi olen määritellyt käytettävän laitteiston ja käyttöjärjestelmän asiakkaan vaatimusten pohjalta. Lisäksi

olen myös listannut toiminnallisen määritelmän loppuun järjestelmän muita ominaisuuksia, suunnittelurajoitteita sekä muutamia jatkokehitysideoita.

6.4 Opinnäytetyöraportin kirjoittaminen

Opinnäytetyöraportin kirjoittamiseen kuluva aika on arvioitu noin 80 tunniksi, tämä pitää omalla kohdallani suhteellisen hyvin paikkaansa. Opinnäytetyöraportin kirjoittamisen aloitin jäsentelemällä sen osa-alueiksi otsikkojen avulla. Teoreettiseen osuuteen materiaalin löytäminen osoittautui tietyiltä osin melko hankalaksi. Toisaalta uhkaavasti lähestyvä ”deadline” ei antanut mahdollisuutta paneutua tarkemmin alan kirjallisuuteen enää opinnäytetyön loppuvaiheessa. Joten turvauduin Internetiin, josta tietoa suhteellisen helposti tiettyihin asioihin löytyi, mutta mahdollisesti luotettavuuden kustannuksella. Opinnäytetyöraportissa olen esitellyt ohjelmistotuotannon perusteita ja käyttämiäni menetelmiä kirjallisuuden avulla sekä kuvannut projektin kulkua koko sen 2,5-vuotisen historian ajalta.

7 KATASO-ohjelman esittely

Tässä luvussa esitellään muutama keskeisin KATASO-ohjelman ominaisuus pääpiirteittäin. Tarkemmat kuvaukset löytyvät KATASO-ohjelman toiminnallisesta määrittelystä.

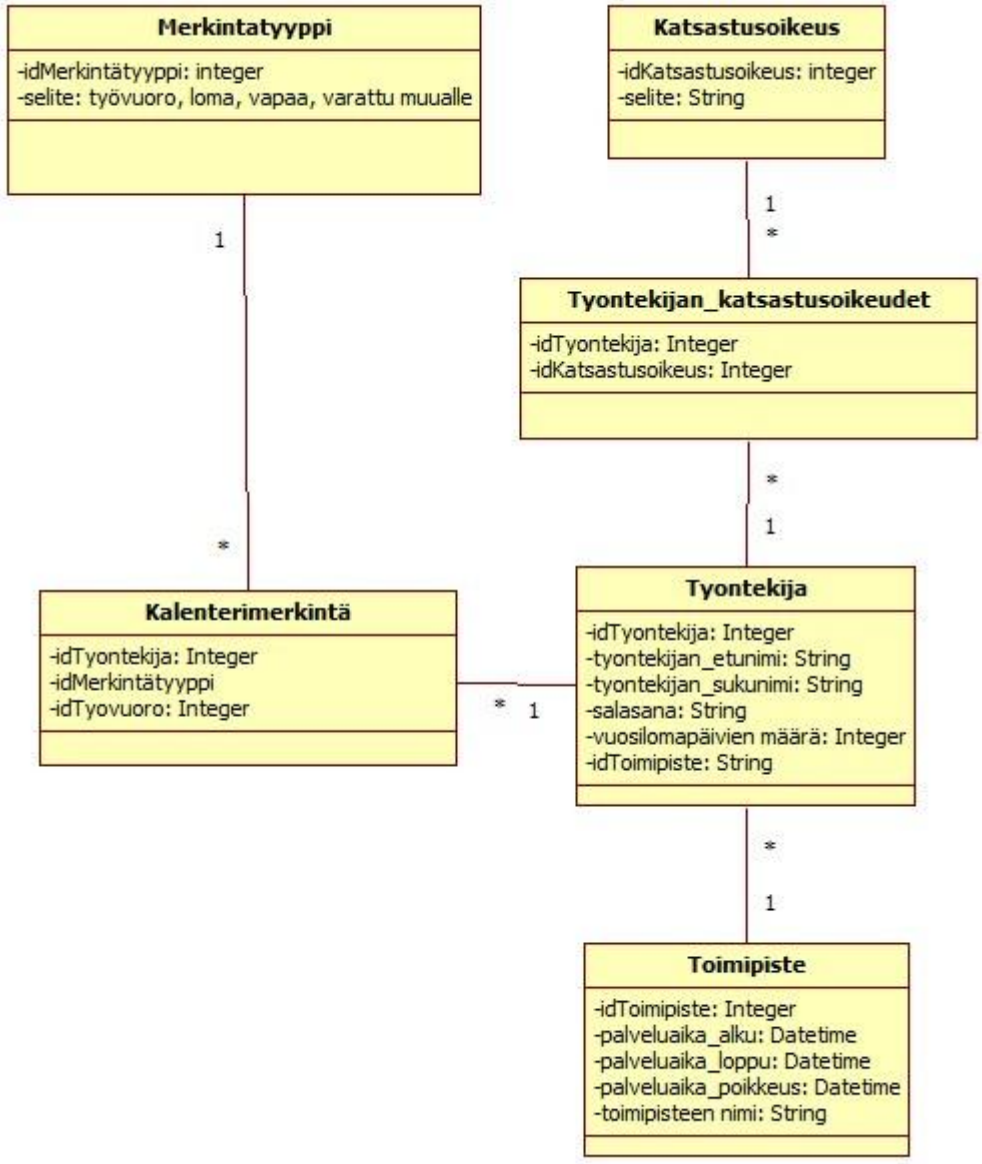
7.1 Toiminnallisen määrittelyn sisältö

Sisällysluettelo

Kuva 7.1 Katsastusaseman työvuorojen suunnitteluohjelman (KATASO) toiminnallisen määrittelyn sisältöluettelo

7.2 Tiedot ja tietokantakaavio

Tässä luvussa käsitellään KATASO-ohjelman tietokantakaavio tietosisältöineen, joka esitetty kuvassa 7.2.

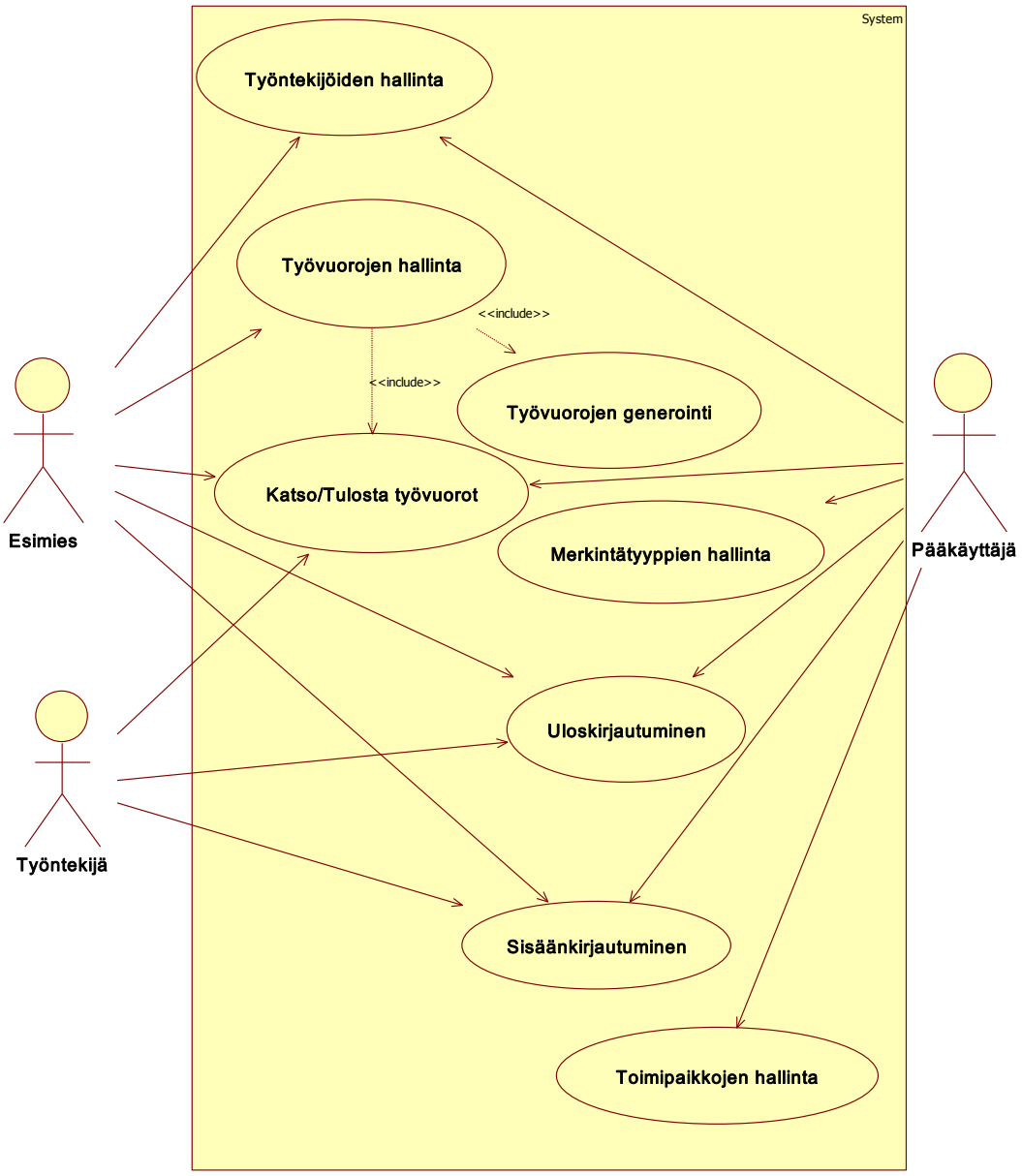


Kuva 7.2 KATASO-ohjelman tietokantakaavio

Tietokantakaavion tehtävä on kuvata ohjelmalle tarpeelliset tiedot ja niiden väliset yhteydet eli assosiaatiot sekä tietotyypit.

7.3 Käyttötapauskaavio

Tässä luvussa esitellään KATASO-ohjelman käyttötapauskaavio, joka on kuvassa 7.3. Käyttötapauskaavion on tarkoitus antaa yleiskuva ohjelman toiminnoista.



Kuva 7.3 KATASO-ohjelman käyttötapauskaavio eli Domain Model.

7.4 Esimerkkejä käyttötapauskuvauksista

Seuraavaksi esittelen muutaman keskeisimmän käyttötapauskuvauksen lomakkeineen KATASO-ohjelmasta.

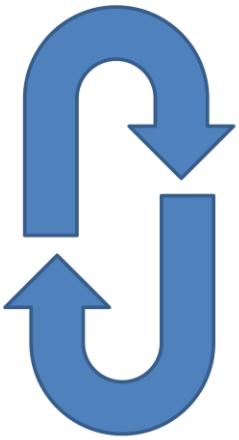
7.4.1 Työvuorojen generointi

Työvuorojen generointi on tärkein KATASO-ohjelman käyttötapauksista, koska ohjelman tarkoituksena on nimenomaan automatisoida työvuorojen suunnittelu mahdollisimman pitkälle. Työvuorojen generointiin olen kehittänyt kaksi erilaista menetelmään: aritmeettinen ja poissulkeva. Työvuorojen generointiin on asetettu useita rajoitteita ja ehtoja, jotka esitellen myöhemmin kappaleessa 7.5, kuten myös työvuorojen generointialgoritmit vuokaavioineen. Ensimmäisenä esittelen aritmeettisen menetelmän käyttötapauskuvauksen ja seuraavaksi poissulkevan menetelmän käyttötapauskuvauksen.

Aritmeettinen menetelmä työvuorojen generointiin

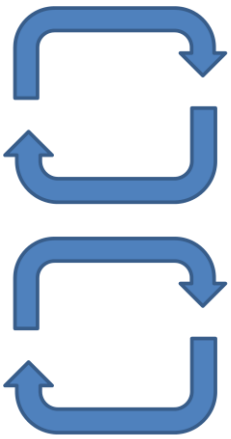
Nimi	Työvuorojen generointi (aritmeettinen menetelmä)
Kuvaus	Järjestelmä generoi alustavat työvuorot työntekijöiden kalenterimerkintöjen ja katsastusoikeuksien perusteella. Pääkäyttäjällä tai esimiehellä on mahdollista manuaalisesti vielä muokata työvuoroja työvuorojen hallinta lomakkeella.
Toimijat	Järjestelmän pääkäyttäjä, esimiehet
Alkutila ja alkuehdot	Alkutilana on se, että työtekijän hallinta-lomakkeella on syötetty järjestelmään työntekijän kalenterimerkinnät (työvuorot, lomat, ylityövapaat jne.) sekä kyseisen työntekijän katsastusoikeudet.
Tyypillinen käyttötapauksen kulku	1. Järjestelmä tarkastaa edellisen työntekijän edellisen viikon työvuoron ja sulkee tämän vaihtoehdon pois generoidessaan seuraavan viikon työntekijän työvuoroa sekä huomioi kyseisen työntekijän katsastusoikeudet.. 2. Järjestelmä generoi työntekijälle työvuorot 7 viikon jaksoissa vuoden alusta lähtien huomioiden luvun 7.5 työvuorojen generointi alussa mainituin rajoittein ja ehdoin.
Virhe 1 Tarvittavia tietoja ei löydy	1. Näytölle ilmestyy virheilmoitus -lomake ” Haluttuja tietoja ei löydy tietokannasta. Tietokantavirhe!” 2. Virheilmoitus kuitataan OK-painikkeella ja ohjelma yrittää generoida työvuorot uudelleen. 3. Mikäli ongelma toistuu kolme kertaa näytölle ilmestyy virheilmoitus -lomake ”Ei yhteyttä tietokantaan. Ota yhteys pääkäyttäjään tai tietohallintoon.”
Lopputila ja jälkiehdot	Kaikille työntekijöille on generoitu onnistuneesti työvuorot 7 viikkoa eteenpäin ja kaikkiin tehtäviin on varattu työntekijä.

Kuvissa 7.3 ja 7.4 on esitetty aritmeettisen generointimenetelmän katsastajien kierrätys ensin normaalisti ja sen jälkeen kesälomakautena. Kesälomakauden ulkopuolella kaikkia katsastajien kierrätetään eri tehtävissä heidän katsastusoikeuksiensa mukaisesti. Raskaan kaluston linjalla tarvitaan kaksi katsastajaa ympäri vuoden, joten kaksi raskaan kaluston katsastajaa voidaan hyödyntää kevyen kaluston katsastustehtäviin. Kesälomakaudelle on sovittu rajoitteeksi, että vain yksi raskaan ja yksi kevyen kaluston katsastaja voi olla saman aikaisesti lomalla, jolloin on järkevää taata riittävä palvelukyky kierrättämällä raskaan kaluston katsastajia omassa syklissään ja kevyen kaluston katsastajia omassaan.



raskas				
raskas				
raskas				
raskas				
kevyt				
kevyt				
kevyt				
kevyt				

Kuva 7.4 Katsastajien kierrättäminen kesälomakauden ulkopuolella.



raskas				
raskas				
raskas				
raskas				
kevyt				
kevyt				
kevyt				
kevyt				

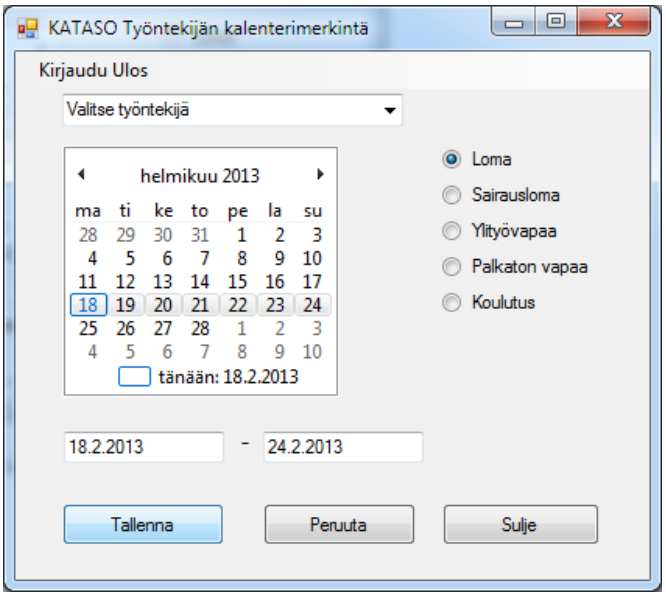
Kuva 7.5 Katsastajien kierrättäminen kesälomakaudella.

Poissulkeva menetelmä työvuorojen generointiin

Nimi	Työvuorojen generointi (poissulkeva menetelmä)
Kuvaus	Järjestelmä generoi alustavat työvuorot työntekijöiden kalenterimerkintöjen ja katsastusoikeuksien perusteella lajitellen katsastajat oikeuksien perusteella nousevaan järjestykseen ja työvuorot vaatimusten perusteella laskevaan järjestykseen. Pääkäyttäjällä tai esimiehellä on mahdollista manuaalisesti vielä muokata työvuoroja työvuorojen hallinta lomakkeella.
Toimijat	Järjestelmän pääkäyttäjä, esimiehet
Alkutila ja alkuehdot	Alkutilana on se, että työtekijän hallinta-lomakkeella on syötetty järjestelmään työntekijän kalenterimerkinnät (työvuorot, lomat, ylityövapaat jne.) sekä kyseisen työntekijän katsastusoikeudet.
Tyypillinen käyttötapauksen kulku	1. Järjestelmä tarkastaa edellisen työntekijän edellisen viikon työvuoron ja sulkee tämän vaihtoehdon pois generoidessaan seuraavan viikon työntekijän työvuoroa sekä huomioi kyseisen työntekijän katsastusoikeudet. 2. Järjestelmä generoi työntekijälle työvuorot 7 viikon jaksoissa vuoden alusta lähtien huomioiden luvun 7.5 työvuorojen generointi alussa mainituin rajoittein ja ehdoin.
Virhe 1 Tarvittavia tietoja ei löydy	1. Näytölle ilmestyy virheilmoitus –lomake ” Haluttuja tietoja ei löydy tietokannasta. Tietokantavirhe!” 2. Virheilmoitus kuitataan OK-painikkeella ja ohjelma yrittää generoida työvuorot uudelleen. 3. Mikäli ongelma toistuu kolme kertaa näytölle ilmestyy virheilmoitus -lomake ”Ei yhteyttä tietokantaan. Ota yhteys pääkäyttäjään tai tietohallintoon.”
Lopputila ja jälkiehdot	Kaikille työntekijöille on generoitu onnistuneesti työvuorot 7 viikkoa eteenpäin ja kaikkiin tehtäviin on varattu työntekijä.

7.4.2 Työntekijän kalenteri

Työntekijän kalenteriin merkitään esimiesten ja/tai pääkäyttäjän toimesta työntekijän poissaolot, kuten vuosi- ja sairauslomat, ylityövapaat, palkattomat vapaat ja koulutukset eli syyt joiden vuoksi työntekijä ei ole käytettävissä työvuorojen generointiin. Työntekijän kalenteri on siis tärkeä osa ohjelmaa työvuorojen generointia varten. Työntekijän kalenterimerkintä-lomake on esitetty kuvassa 7.6.

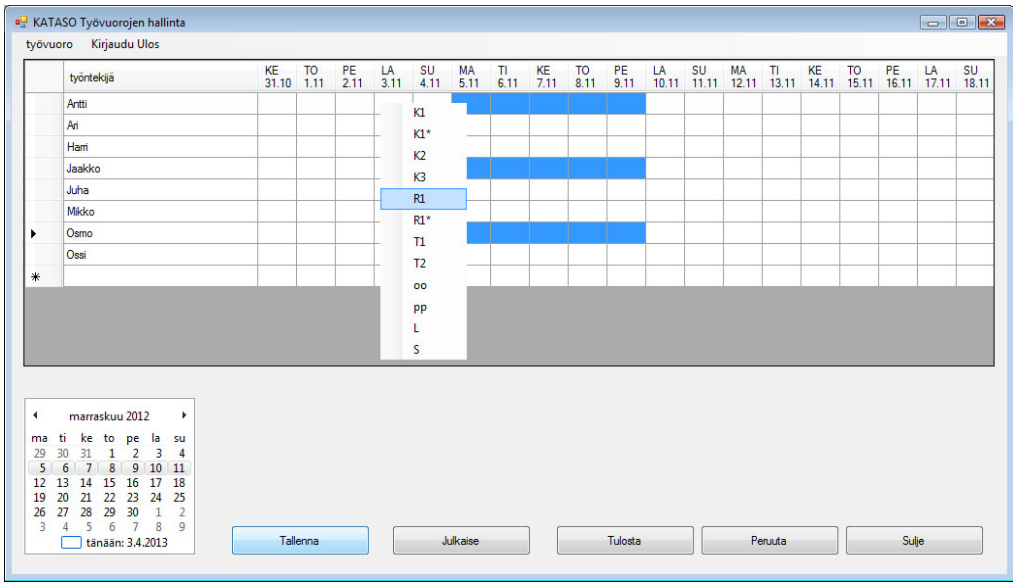


Kuva 7.6 Työntekijän kalenterimerkintä-lomake.

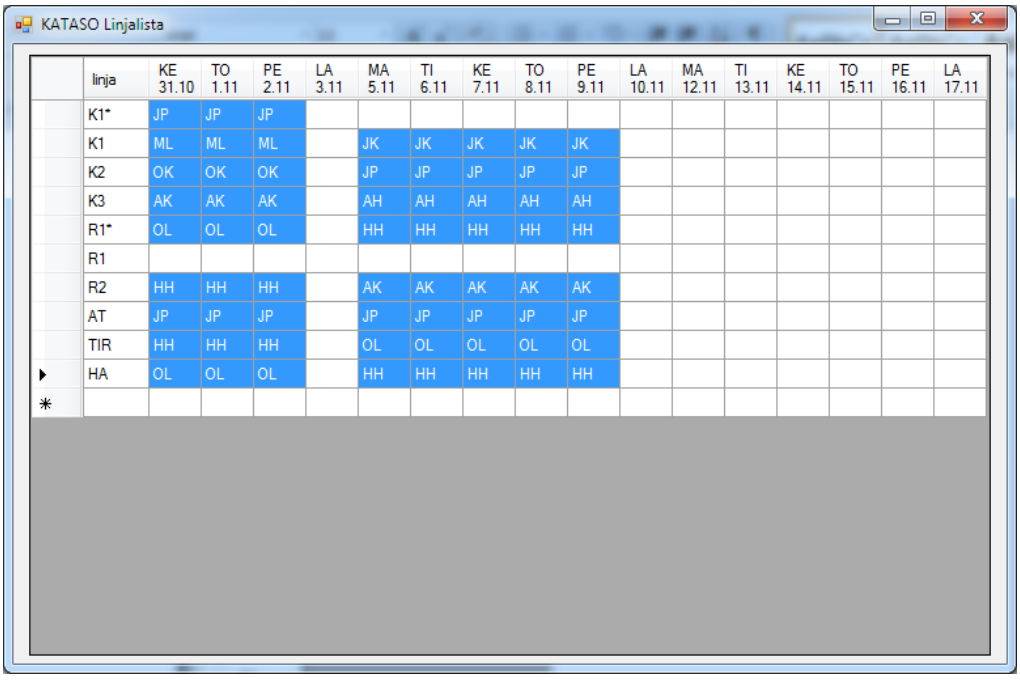
Nimi	Kalenterimerkintä
Kuvaus	Kalenterimerkintä on työntekijäkohtainen ja siihen merkitään esimiehen/pääkäyttäjän toimesta työntekijän poissaolot eli ajat jolloin työntekijä ei ole käytettävissä työvuorojen generointiin.
Toimijat	Pääkäyttäjä/esimies
Alkutila ja alkuehdot	Pääkäyttäjä/esimies on valinnut päävalikosta Kalenterimerkintä-painikkeen ja hänelle on avautunut kuvan 7.6 mukainen lomake.
Käyttötapaus 1 Pääkäyttäjä/esimies	1. Pääkäyttäjä/esimies valitsee työntekijän alasvetovalikosta, jonka kalenteria ollaan muokkaamassa. 2. Pääkäyttäjä/esimies valitsee valinta-painikkeilla poissaolon syyn: loma, sairausloma, ylityövapaa, palkaton vapaa tai koulutus. 3. Tämän jälkeen valitaan joko kalenterista maalaamalla tai kirjoittamalla tekstikenttiin poissaolon alkamis- ja loppumisajankohta. Kalenteri ja tekstikentät ovat yhteydessä toisiinsa eli merkittäessä kummalla tavalla tahansa poissaolo näkyy reaaliaikaisena toisessa vaihtoehdossa. 4. Painettaessa Tallenna-painiketta tiedot päivittyvät tietokantaan. Toimenpiteen voi peruuttaa Peruuta-painikkeella. 5. Ohjelma suljetaan Sulje-painikkeella.
Lopputila ja jälkiehdot	Työntekijälle on merkitty poissaolot Kalenterimerkintä-tauluun ja ne otetaan huomioon työvuoroja generoitaessa ja ne näkyvät Työvuorojen hallinta -lomakkeella ko. merkintänä.

7.4.3 Työvuorojen hallinta

Työvuorojen hallinnalla tarkoitetaan tässä tapauksessa esimiesten ja pääkäyttäjän mahdollisuutta muokata manuaalisesti generoituja työvuoroja. Kuvassa 7.6 on esitetty työvuorojen hallinta-lomake ja kuvassa 7.7 linjalista, joka avautuu työvuorojen hallinta-lomakkeen oheen, josta esimies tai pääkäyttäjä näkee onko kaikki linjat miehitetty ja löytyykö kaikille töille tekijä.



Kuva 7.7 Työvuorojen hallinta-lomake.



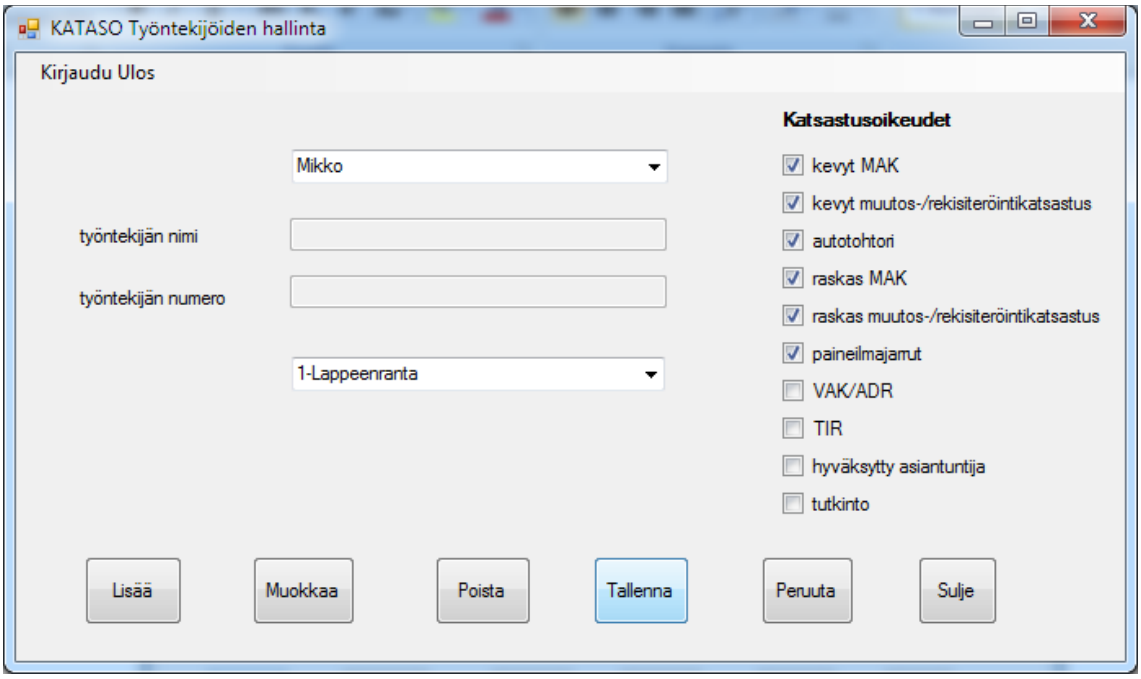
Kuva 7.8 Linjalista-lomake.

Nimi	Työvuorojen hallinta –lomake
Kuvaus	Pääkäyttäjä/esimies valitsee työntekijän/työntekijät, joiden työvuoroja hän haluaa manuaalisesti muokata sekä ajanjakson, jolta työvuoroja muokataan.
Toimijat	Pääkäyttäjä, esimiehet
Alkutila ja alkuehdot	Pääkäyttäjä/esimies on painanut KATASO Päävalikosta Työvuorojen hallinta –painiketta ja hänelle on avautunut kuvan 7.7 mukainen työvuorojen hallinta –lomake. Samanaikaisesti näytölle avautuu myös kuvan 7.8 mukainen linjalista, josta pääkäyttäjä/esimies näkee reaaliaikaisesti onko kaikille toimenpiteille varattu työntekijä kaikkiin työvuoroihin. Järjestelmä on generoinut alustavat työvuorot valmiiksi huomioiden työntekijän katsastusoikeudet ja työntekijän kalenterin -merkinnät tiettyjen alkutietojen ja parametrien perusteella. Työvuorojen hallintalomaketta voi muokata vain yksi esimies kerrallaan, jonka jälkeen hän julkaisee työvuorolistan nähtäväksi/muille muokattavaksi.
Tyypillinen käyttötapauksen kulku	1. Pääkäyttäjä/esimies voi tarvittaessa muokata kunkin työntekijän tai vaihtoehtoisesti useamman työntekijän työvuoroja manuaalisesti päivä, viikko tai kuukausi kerrallaan hiirellä ”maalaamalla” halutut päivät ja klikkaamalla hiiren oikeaa näppäintä, jolloin avautuu valikko, josta pääkäyttäjä/esimies voi vaihtaa aktivoidulle ajanjaksolle työntekijän työvuoron. 2. Ohjelma tarkistaa, että kaikkiin työvuoroihin ja tehtäviin on varattu työntekijä kullekin päivälle. Mikäli näin ei ole ohjelma kysyy mukaisella lomakkeella varmistusta tallennukselle. 3. Muokatut työvuorot tallennetaan tietokantaan painamalla Tallenna-painiketta. 4. Julkaise-painikkeella esimies julkaisee työvuorolistan muiden nähtäväksi/muiden esimiesten muokattavaksi. 5. Halutessaan esimies voi tulostaa työvuorot painamalla Tulosta –painiketta. 6. Työvuorolista tulostuu Windowsissa määriteltyyn oletus tulostimeen.
Virhe 1 Työntekijälle ei ole valittuna työvuoroa	1. Mikäli työntekijälle ei ole valittuna työvuoroa vaikka hän on työntekijän kalenterin mukaan käytettävissä ohjelma ilmoittaa virheestä mukaisella lomakkeella, jossa on mainittu työntekijä sekä aikaväli jolle ei ole valittuna työvuoroa. 2. Virhe voidaan hyväksyä painamalla Hyväksy -painiketta, jolloin ohjelma tallentaa työvuorolistan sellaisenaan tai korjata painamalla Korjaa -painiketta, jolloin ohjelma siirtyy työvuorojen hallinta -lomakkeelle.
Virhe 2 Linjalistassa ei ole täytetty kaikkia tehtäviä	1. Mikäli kaikille katsastusaseman palveluissa määritetyille tehtäville ei ole määritelty tekijää tietylle aikavälille palveluajan puitteissa. Ohjelma ilmoittaa siitä kaltaisella virheilmoituslomakkeella. 2. Virhe voidaan hyväksyä painamalla Hyväksy -painiketta (riittämätön miehitys esimerkiksi epidemian vuoksi), jolloin ohjelma tallentaa työvuorolistan sellaisenaan tai korjata painamalla Korjaa -painiketta, jolloin ohjelma siirtyy työvuorojen hallinta -lomakkeelle.

Lopputila ja jälkiehdot	Kaikille käytettävissä oleville katsastajille on generoitu ja tarvittaessa manuaalisesti muokattu työvuorot ja järjestelmästä onnistuneesti kirjauduttu ulos. Mikäli pääkäyttäjä/esimies on halunnut tulostaa työvuorot tulostimelle on tulostaminen onnistunut.

7.4.4 Työntekijöiden hallinta

Työntekijöiden hallinnalla tarkoitetaan esimiesten ja pääkäyttäjän mahdollisuutta muokata työntekijöiden katsastusoikeuksia, jotka uran aikana voivat muuttua sekä työntekijöiden ensisijaisia toimipisteitä. Pilottiversiossa toimipisteitä on ainoastaan yksi kappale. Työntekijääiden hallinta-lomake on esitetty kuvassa 7.8.



Kuva 7.9 Työntekijöiden hallinta-lomake.

Nimi	Työntekijöiden hallinta
Kuvaus	Lomakkeella lisätään uusi työntekijä tietoineen, muokataan tai poistetaan nykyisen työntekijän tietoja.
Toimijat	Järjestelmän pääkäyttäjä, esimiehet
Alkutila ja alkuehdot	Alkutilana on se, pääkäyttäjä/esimies on painanut päävalikkolomakkeella painikkeen Työntekijöiden hallinta ja hänelle on avautunut (kuva 7.9) KATASO Työntekijöiden hallinta –lomake.
Käyttötapaus 1 Lisää työntekijä	1. Pääkäyttäjän/esimiehen painaessa Lisää-painiketta aktivoituvat Työntekijän nimi sekä Työntekijän numero –kentät. 2. Näihin kenttiin syötetään näppäimistöllä uuden työntekijän nimi ja hänet yksilöivä työntekijän numero. 3. Tämän jälkeen hänelle määritetään alasvetovalikosta ensisijainen toimipiste. 4. Hänelle määritetään katsastusoikeudet merkkaamalla oikeat

	valintaruutu- näppäimet. 5. Hänelle kirjataan vuosilomapäivien lukumäärä aktivoimalla haluttu radionäppäin 6. Lopuksi hänen tietonsa tallennetaan tietokantaan painamalla Tallenna. 7. Ohjelma varmistaa vielä tallennuksen Kysely-lomakkeella ”Tahdotko varmasti tallentaa työntekijää koskevat muutokset?”, johon vastataan Kyllä- tai Ei-painikkeella . 8. Toimenpiteen voi peruuttaa painamalla Peruuta-painiketta. 9. Työntekijöiden hallinta -lomake suljetaan toimenpiteen päätteeksi painamalla Sulje-painiketta tai järjestelmästä kirjaudutaan suoraan ulos vasemman ylänurkan Menu-valikosta painamalla Kirjaudu ulos –valintaa. 10. Pääkäyttäjä luo ja luovuttaa työntekijöille kullekin oman henkilökohtaisen salasanan, salasanaa ei generoida tai salata mitenkään.
Käyttötapaus 2 Muokkaa työntekijä	1. Pääkäyttäjän/esimiehen painaessa Muokkaa-painiketta aktivoituu alasvetovalikko. 2. Alasvetovalikosta pääkäyttäjä/esimies valitsee muokattavan työntekijän. 3. Seuraavaksi hän voi muokata työntekijän tietoja: muuttaa katsastusoikeuksia ja/tai vaihtaa ensisijaisen toimipisteen. 4. Lopuksi muutetut tiedot tallennetaan tietokantaan painamalla Tallenna -painiketta 5. Ohjelma varmistaa vielä tallennuksen Kyselylomakkeella ”Tahdotko varmasti tallentaa työntekijää koskevat muutokset?”, johon vastataan Kyllä- tai Ei-painikkeella . 6. Toimenpiteen voi peruuttaa painamalla Peruuta-painiketta. 7. Työntekijöiden hallinta -lomake suljetaan toimenpiteen päätteeksi painamalla Sulje-painiketta tai järjestelmästä kirjaudutaan suoraan ulos vasemman ylänurkan Menu-valikosta painamalla Kirjaudu ulos –valintaa.
Käyttötapaus 3 Deaktivoi työntekijä	1. Pääkäyttäjän/esimiehen painaessa Poista-painiketta aktivoituu alasvetovalikko. 2. Alasvetovalikosta pääkäyttäjä/esimies valitsee deaktivoitavan työntekijän. 3. Lopuksi muutetut tiedot tallennetaan tietokantaan painamalla Tallenna -painiketta 4. Työntekijän tiedot eivät katoa tietokannasta, mutta työntekijälle ei ole enää mahdollista merkitä mitään kalenterimerkintöjä ja hänet on poistettu valikoista eli hän niin sanotusti deaktivoitu. 5. Ohjelma varmistaa vielä tallennuksen Kyselylomakkeella ”Tahdotko varmasti tallentaa työntekijää koskevat muutokset?”, johon vastataan Kyllä- tai Ei-painikkeella . 6. Toimenpiteen voi peruuttaa painamalla Peruuta-painiketta. 7. Työntekijöiden hallinta -lomake suljetaan toimenpiteen päätteeksi painamalla Sulje-painiketta tai järjestelmästä kirjaudutaan suoraan ulos vasemman ylänurkan Menu-valikosta painamalla Kirjaudu ulos –valintaa.
Virhe 1 Ttyöntekijä on jo olemassa	1. Näytölle ilmestyy virheilmoituslomake ” Samoilla identifikaatiotiedoilla löytyy jo tiedot tietokannasta.” 2. Virheilmoitus kuitataan OK-painikkeella ja ohjelma siirtyy takaisin Työntekijän hallinta –lomakkeelle.
Virhe 2 Työntekijän tieto-	1. Näytölle ilmestyy virheilmoituslomake ” Haluttuja tietoja ei löydy tietokannasta. Tietokantavirhe!”

ja ei löydy tietokannasta	2. Virheilmoitus kuitataan OK-painikkeella ja ohjelma siirtyy takaisin Työntekijän hallinta –lomakkeelle.
Virhe 3 Työntekijän tieto-ja ei löydy tietokannasta	1. Näytölle ilmestyy virheilmoituslomake ”Ei yhteyttä tietokantaan. Ota yhteys pääkäyttäjään tai tietohallintoon.” 2. Virheilmoitus kuitataan OK-painikkeella ja ohjelma siirtyy takaisin Työntekijän hallinta –lomakkeelle.
Lopputila ja jälkiehdot	Pääkäyttäjän/esimiehen valitsema toimenpide on onnistuneesti suoritettu ja lisätyt/muokatut/poistetut tiedot ovat onnistuneesti tallennettu tietokantoihin. Sivu on suljettu ja ohjelma on palautunut päävalikko-lomakkeelle tai vaihtoehtoisesti ohjelmasta on kirjauduttu onnistuneesti ulos.

7.5 Työvuorojen generointialgoritmit

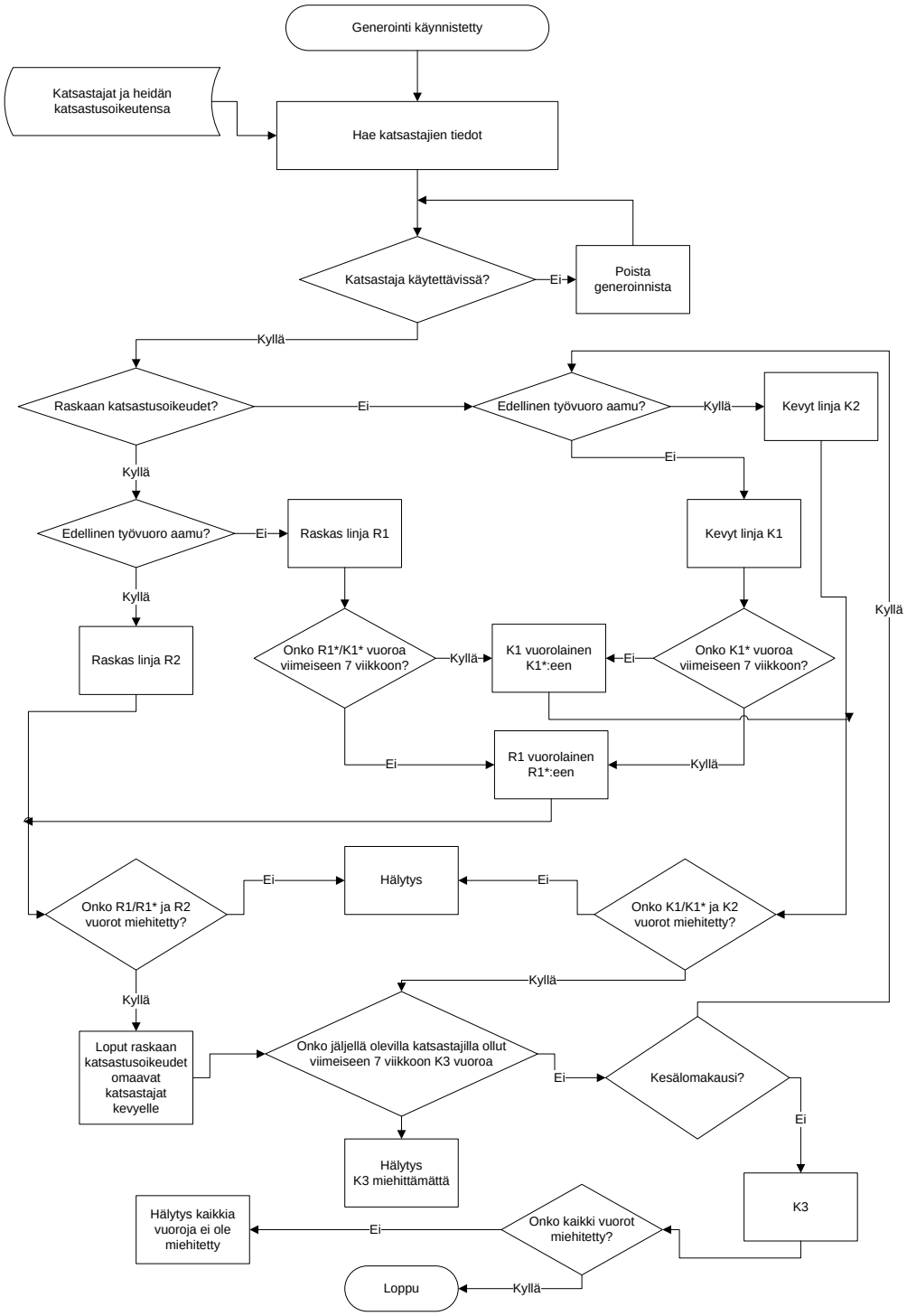
Työvuorojen generoinnille on asetettu seuraavia rajoitteita ja ehtoja, jotka on lueteltu seuraavassa taulukossa 7.1. Taulukossa luetelluilla ehdoilla taataan riittävä palvelu, työntekijöiden mahdollisimman tasapuolinen kohtelu työvuoroja suunniteltaessa sekä työehtosopimuksen mukainen 7,5 tunnin työaika.

Taso	Rajoite tai ehto
Päivätaso	1. Vain yksi katsastaja K1* tai R1* sekä K3 -vuoroon (päivän avaus sekä päätös) 2. Raskaalla linjalla oltava kaksi katsastajaa, joista R1/R1*- ja toinen R2-vuorossa. 3. Kevyellä puolella vähintään yksi mieluiten kaksi katsastajaa K1 tai K1* -vuorossa sekä K2 -vuorossa tai mikäli lomat, vapaat tai sairauslomat edellyttää K3-vuoron katsastaja muina kuukausina kuin kesä-heinä-elokuussa. 4. Autotohtorioikeudet omaava katsastaja oltava läsnä kello 9.00-16.00, joko kevyellä tai raskaalla linjalla. 5. Raskaan kaluston rekisteröinti- ja muutoskatsastus oikeudet omaava katsastaja läsnä vähintään kello 9.00-16.00. 6. Kevyen kaluston rekisteröinti- ja muutoskatsastus oikeudet omaava katsastaja läsnä vähintään kello 9.00-17.00. 7. Hyväksytty asiantuntija läsnä vähintään kello 9.00-16.00. 8. Kevyellä puolella myös vähintään raskaan määräaikaiskatsastusoikeudet omaava katsastaja läsnä kello 9.00-16.00 ellei lomat, vapaat tai sairauslomat muuta edellytä. 9. Palveluaika on 8.00-18.00 sekä lauantaisin 8.00-14.00. 10. Raskaan ja kevyen kaluston katsastuksille tulee löytyä katsastaja kulloisenkin palveluajan puitteissa mikäli ne kuuluvat katsastusaseman palvelujen piiriin.
Viikkotaso	1. Mustolan toimipisteelle mies pääsääntöisesti Lauritsalan toimipisteeltä /M ellei lomat, vapaat tai sairauslomat muuta edellytä. Toimipiste avoinna pääsääntöisesti viikoittain kello 13.00- vain raskaan kaluston katsastuksille eli katsastajalla tulee olla kaikki raskaan kaluston katsastusoikeudet. 2. Ei samaa työvuoroa perättäisillä viikoilla ellei lomat, vapaat tai sairaus-

	lomat muuta edellytä. 3. Viikoittain kello 8.00 ja kello 9.00 vuorot pääsääntöisesti vuorottelevat pois lukien K1*/R1* ja K3 -vuoroja. 4. K1*/R1* sekä K3 -vuorot toistuvat pääsääntöisesti kahdeksan viikon välein ellei lomat, vapaat ja sairauslomat muuta edellytä. 5. Leirin toimipisteen lauantaivuoro pois lukien kesäkuukaudet kiertää viikoittain katsastajalta seuraavalle taulukon 4.1 mukaisesti. 6. Syyskuun ensimmäisen lauantaivuoron tekee listalla seuraava katsastaja, johon toukokuun viimeinen lauantaivuoro ikään kuin päättyi. 7. Leirissä suoritetaan vain kevyen kaluston määräaikaiskatsastuksia.
Lomat	1. Poissaolot haetaan työntekijän kalenterimerkintä taulusta. 2. Kesälomakuukausia ovat Touko-Syyskuu, joille kunkin katsastajan kesälomat tulee pääsääntöisesti sijoittaa, loma toiveet pyritään ottamaan mahdollisuuksien mukaan huomioon. 3. Kesä-Heinä-Elokuussa yhtenäisen lomajakson pituus kolme viikkoa, muut lomat pidetään tämän ajanjakson ulkopuolella. 4. Kesä-Heinä-Elokuussa ei ole K3-vuoroa eikä lauantai vuoroa, lisäksi palveluaika on arkisin poikkeuksellisesti 8.00-17.00. 5. Vain yksi raskaan kaluston katsastusoikeudet omaava katsastaja voi olla kerrallaan vuosilomalla samanaikaisesti. 6. Kevyen katsastajia voi olla samanaikaisesti vuosilomalla tai vapaalla yhdestä kolmeen kappaletta edellyttäen, että kaikille erilaisille katsastustoimenpiteille löytyy katsastaja palveluajan puitteissa. 7. Aseman katsastuksista vastaava henkilö (aluepäällikkö) ja hänen sijaisensa eivät voi olla vuosilomalla samanaikaisesti. 8. Raskaan kaluston katsastajien R1* ja K1* vuorot vuorottelevat ellei lomat, vapaat ja sairauslomat muuta edellytä. Toistumisväli esimerkiksi R1*-vuorolla on 16 viikkoa. 9. Lomalla olevat R1*/K1* ja K3 -vuoroihin generoidut katsastajat korvataan R1/K1 ja K2 -vuoroihin generoiduilla katsastajilla, siten että mikäli mahdollista edellisellä neljän viikon jaksolla ei ole ollut R1*/K1* ja K3 -vuoroja. 10. Kesälomat ovat niin sanotusti kiertäviä. 11. Esimies hoitaa oman tehtävänsä ohella päällikön hallinnolliset tehtävät, ei muita eli on käytettävissä työvuoroihin normaalisti.
Vuositaso	1. Vuositasolla työvuorojen generoinnin tulee kohdella pääsääntöisesti kaikkia katsastajia tasapuolisesti.

Taulukko 7.1 Työvuorojen generoinnin rajoitteita ja ehtoja.

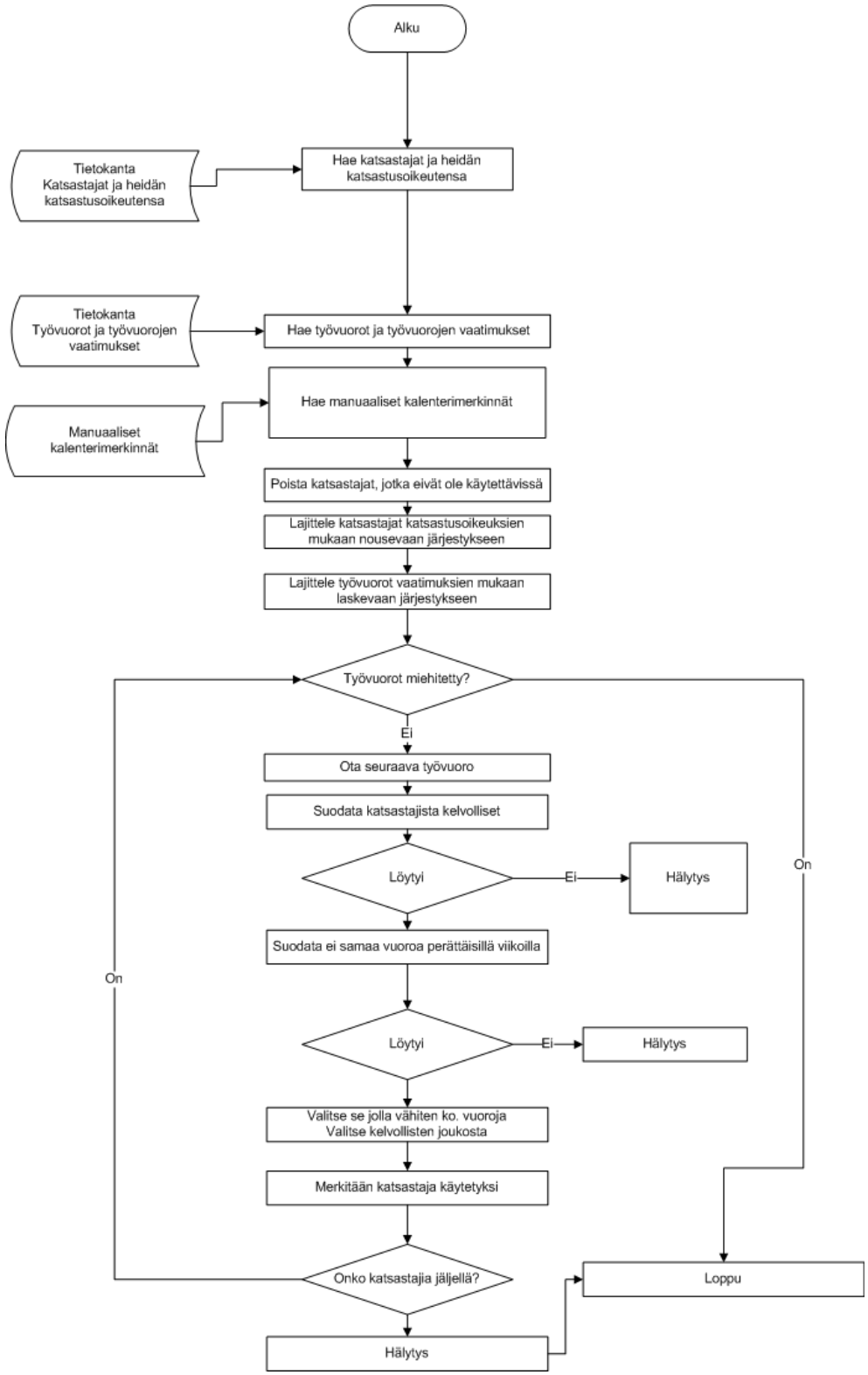
Aritmeettisessa generoinnissa kaikkia katsastajia kierrätetään eri tehtävissä katsastusoikeuksien mukaisesti eri tehtävissä viikko kerrallaan sykleissä pois lukien kesäkuukaudet. Kesäkuukausina, lomien vuoksi, raskaan kaluston ja kevyen kaluston katsastajat kiertävät omissa sykleissään. Yksittäisen katsastajan perustyövuorot kiertävät siten, että kello 8.00-16.15 vuoroa seuraa 9.00-17.15. Poikkeuksena on kesälomakauden ulkopuolella 10.00-18.15 työvuoro, joka samoin kuin ympärivuotinen 7.45-16.00 työvuoro eivät saa toistua kuin vähintään seitsemän viikon välein. Aritmeettisen generointialgoritmin vuokaavio on esitetty kuvassa 7.9 ja poissulkevan menetelmän kuvassa 7.10.



Kuva 7.10 Työvuorojengenerointialgoritmin 1 (aritmeettinen) vuokaavio.

Poissulkeva generointi jakaa katsastajat katsastusoikeuksien perusteella nousevaan ja työvuorot vaativuuden perusteella laskevaan järjestykseen. Ohjelma generoi ensin kevyen katsastusoikeuksien katsastajat kevyelle linjalle ja sen jälkeen raskaan kaluston oikeuksien katsastajat raskaalle linjalle, siten että ensin täytetään yksi katsastaja kullekin linjalle kuhunkin työvuoroon ja jos katsas-

tajia jää tämän jälkeen jäljelle niin heidät kevyen kaluston linjalle. Edellä mainitulla generointimenetelmällä pyritään siihen, ettei katsastajien katsastusoikeudet menisi tavallaan hukkaan.



Kuva 7.11 Työvuorojengenerointialgoritmin 2 (poissulkeva) vuokaavio.

8 Yhteenveto ja pohdinta

Asiakkaan tavoitteena oli saada valmis helppokäyttöinen, looginen ja helposti omaksuttava sekä yhtenäisen ulkonäön omaava ohjelma työvuorojen suunnitteluun. Toteutuessaan ohjelma olisi tehostanut esimiesten työajan käyttöä ja automatisoinut työvuorojen suunnittelun. Asiakkaan tavoitteiden täyttyminen jäi kuitenkin osittaiseksi, koska tekninen määrittely ja ohjelmointi jätettiin projektin edetessä suosiolla pois. Asiakkaalle luovutetaan tässä vaiheessa ohjelman toiminnallinen määrittely-dokumentaatio, joka sisältää kuvassa 7.1 esitetyt kohdat. Asiakkaan tavoitteiden kannalta vaikeinta määrittää työvuorojen generointialgoritmit. Helpoimpia seikkoja asiakkaan tavoitteiden kannalta oli määrittää tietokanta tietosisältöineen sekä käyttötapauskaavio. Ohjelman toteutus jonkun toisen opinnäytetyön aiheena on edelleen mahdollista. Tästä kuitenkin pitää sopia asiakkaan kanssa erikseen. Asiakas ei ole vielä tehnyt päästöstä järjestelmän totuttamisesta. Ohjelman yksityiskohtainen suunnittelu ja ohjelmointi ovat työmäärältään ja vaativuudeltaan sopiva ammattikorkeakoulun opinnäytetyöksi.

Oppimistavoitteenani oli perehtyä toiminnallisen määrittelyn tekoon ja sen pohjalta ohjelman tekniseen määrittelyyn eli ohjelmointiin. Teknisestä määrittelystä ja ohjelmoinnista luovuttiin jo opinnäytetyön varhaisessa vaiheessa, koska vaatimusmäärittelyn tekoon kului ennakoitua enemmän aikaa. Oma työni ei ole projektiluonteista vaan enemmänkin prosessityötä, tämän vuoksi jouduin tutustumaan myös projektityöhön. Toiminnallisen määrittelyn laatimiseksi minun tuli tutustua aiheeseen liittyvään kirjallisuuteen sekä Internetistä löytyvään materiaaliin. Osana toiminnalliseen määrittelyyn kuuluu muun muassa tietokannan suunnittelu sisältäen käsitekaavion ja tietosisällön. Lisäksi minun tuli suunnitella ohjelman käyttötapauskaavio ja yksittäiset käyttötapaukset mahdollisine lomakkeineen sekä niiden kulku. Suurin haaste oli suunnitella työvuorojen generointi, jonka tuli huomioida suuri määrä erilaisia ehtoja eri työvuoroineen ja katsastusoikeuksineen. Lisäksi generoinnin tuli kohdella työntekijöitä mahdollisimman tasapuolisesti, ottaa huomioon työaikalainsäädäntö ja voimassa olevat työehtosopimukset. Helpointa oli kenties lomakkeiden suunnittelu, joka onnistui mielestäni hyvin. KATASO-ohjelman toiminnallinen määrittely on pyritty pitämään

mahdollisimman yksinkertaisena, jotta asiakkaan yhden toimipisteen pilottikäyttöön suunnatun version ohjelmointi olisi helppoa.

Jatkokehittelyideoina ensimmäisenä mieleeni tulee esimerkiksi sähköinen lomatoivelista, jota voidaan tarvittaessa muokata esimiesten toimesta ja sen jälkeen liittää työvuorojen generoinnin yhdeksi ehdoksi. Jatkossa ohjelmaan on mahdollista ja varmasti tuleekin lisätä uusia toimintoja ja käyttötapauksia. Tietokannan tietosisältö tulee varmasti laajenemaan, jolloin alun perin tietokantaratkaisuksi suunniteltu Microsoft Access -tietokanta jäänee auttamatta ominaisuuksiltaan riittämättömäksi. Otettaessa KATASO-ohjelma laajempaan käyttöön tulee ottaa huomioon muutkin kasvavat tarpeet, kuten tietoturvallisuus ja tietosuoja seikat ainakin siinä tapauksessa mikäli ohjelma liitetään osaksi muita asiakkaan käytössä olevia järjestelmiä. Pilottiversiossa toiminnallinen määrittely ei esimerkiksi vaadi salasanoilta minkäänlaista salausalgoritmia.

Tätä opinnäytetyöraporttia kirjoitettaessa toiminnallinen määrittely-dokumentti on valmis ja se toimii dokumentaationa mahdollista toteutusta varten.

Kuvat

Taulukot

Lähteet

2Kmediat.com. 2013. http://www.2kmediat.com/sql/. Luettu 28.5.2013.

A-Katsastus Oy 2013, Konserni lyhyesti. http://www.a-katsastus.com/A-Katsastus-konserni/Sivut/Konsernilyhyesti.aspx. Luettu 7.5.2013

Ekonoja, A. 2004. Henkilökohtaisen tiedonhallinnan perusteet. Informaatioteknologia. Jyväskylän yliopiston IT-tiedekunta ja avoin yliopisto. http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index1.html. Luettu 28.5.2013

Haikala, I. & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. Helsinki. Talentum.

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

Heikkinen, H. 2008. Helsingin yliopisto. Tietojenkäsittely tieteenlaitos. UML-kuvauskielten käyttö ohjelmistojen vaatimusmäärittelyissä. http://www.cs.helsinki.fi/u/taina/sem/gradu/k-2008/hh-seminaariraportti.pdf. Luettu 28.5.2013

Huotari, J. 2012. Jyväskylän ammattikorkeakoulu. SQL-kielen perusteet – luentomateriaali. http://homes.jamk.fi/~huojo/opetus/IIZO3030/SQLopas.pdf. Luettu 28.5.2013.

IBM. 2011. 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 26.5.2013

Itä-Suomen yliopisto. Tietojenkäsittelytieteenlaitos. 2009. Vaatimusmäärittely. http://www.cs.uku.fi/kurssit/opr/vaatimusmaarittely.shtml. Luettu 29.5.2013

Joensuun yliopisto, Tietojenkäsittelytieteenlaitos. 2007. Tuotteenhallinta. http://cs.joensuu.fi/tSoft/tuothall.htm. Luettu 21.5.2013

Kokkoniemi, J. 2009. Ohjelmistotekniikka. Harjoitustyösivut. Oulun yliopisto. Tietojenkäsittelytieteenlaitos. http://www.tol.oulu.fi/kurssit/otekniikka/slides/Harjoituskalvot7.pdf. Luettu 30.5.2013

Korpimies, K. 2011. Ohjelmistotekniikan menetelmät –verkkokurssi. Helsingin avoin yliopisto. http://www.helsinki.fi/~korpimie/ohma/rakenne_ja_kayttaytyminen.html. Luettu 30.5.2013

Kosonen, S. Ohjelmoinnin opetus Extreme Programming -hengessä Pro Gradu-tutkielma. 2005. Jyväskylän Yliopisto. Tietotekniikan laitos. https://jyx.jyu.fi/dspace/bitstream/handle/123456789/12492/URN_NBN_fi_jyu-2005318.pdf?sequence=1. Luettu 26.5.2013

Kouri, I. 2010. Lean management – miten vähemmän voi olla enemmän? http://tredea-fi-bin.directo.fi/@Bin/711bb0eed770a8d9383efe93d8e1fbf2/1369763788/application/pdf/42650/Lean_Kouri.pdf. Luettu 28.5.2013

Kruchten , P. 1998. The Rational Unified Process an introduction. Toinen painos. Massachusetts. Addison Wesley Longman Inc.

Kultanen, A. 2012. Esitutkimusraportti. Saimaan ammattikorkeakoulu. Tekniikka. Tietotekniikka.

Laki ajoneuvojen katsastusluvista 23.12.1998/1099

Liikenneministeriön päätös ajoneuvojen katsastushenkilöstön lisäkoulutuksesta 19.2.1999/201.

Mäkinen, P. Laadukas vaatimustenhallinta, http://www.softqa.fi/pdf/nohau-200902.pdf. Luettu 21.5.2013

Niinimäki, H. 2011. Oliomallinnus-luentomateriaali. Vaasan yliopisto. http://lipas.uwasa.fi/~hkn/oliom/Om-luento-6.pdf. Luettu 30.5.2013

Nikula, U. 2009. Ohjelmistotuotanto-luentomateriaali. Lappeenrannan teknillinen yliopisto. http://www2.it.lut.fi/kurssit/08-09/CT20A4001/Luentomateriaali/Luento07.pdf. Luettu 30.5.2013

Oulun kauppaoppilaitos. Kehittämistyön vaiheet ja elinkaarimalli. http://www.okol.org/verkkokurssit/datanomi/tietojarjestelmien_kaytto_ja_kehittaminen/johdatus_tietojarjestelmiin/kehittamistyon_vaiheet_ja_elikaarimallit/kehittamistyon_vaiheet_ja_elinkaarimallit_asia.htm. Luettu 23.5.2013

Ovaska, P. Ohjelmistotekniikka Vaatimustenhallinta. 2002. http://www2.it.lut.fi/kurssit/01-02/010758000/vaatimustenhallinta.pdf. Luettu 25.5.2013

Paakki, J. 2011. Ohjelmistojen vaatimusmäärittely-luentomateriaali. Helsingin yliopisto. Tietojenkäsittelytieteenlaitos. http://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf. Luettu 29.5.2013

Pohjonen, R. 2002. Tietojärjestelmien kehittäminen. Toinen painos. Jyväskylä. Docendo.

Raisamo, R. 2007. Olioperustaisten ohjelmointikielten periaatteet. Tampereen yliopisto. http://www.cs.uta.fi/opok/2007/opok2007-11-kaaviot.pdf. Luettu 29.5.2013

Romppanen, M. Rational Unified Process (RUP). www.karvakuono.net/filut/rup.ppt. Luettu 26.5.2013

Sarja, J. 2006. Relaatiotietokanta. http://verkkopedagogi.net/vanhat/fi/sisalto/materiaalit/access2003/luku0375c6.html?C. Luettu 28.5.2013

Savon ammatti- ja aikuisopisto. 2013. Pieni SQL-opas. http://www.aedu.sakky.fi/opinnet/visbas/kappale16.htm. Luettu 28.5.201

Taina, J. Ohjelmistoprosessit ja ohjelmiston laatu -luentomateriaali. 2009. Helsingin yliopisto. Tietojen käsittelytieteenlaitos. http://www.cs.helsinki.fi/u/taina/opol/k-2009/pdf/luvut7-9_2.pdf. Luettu 23.5.2013

Tikka M. 2011, Vaatimustenhallinta (Requirements Engineering), http://machina.fi/vaatimustenhallinta-2.php. Luettu 21.5.2013

Työaikalaki 9.8.1996/605

Vaasan ammattikorkeakoulu. Ohjelmiston määrittely. Määrittelydokumentit. http://www.cc.puv.fi/~tka/kurssit/Ohjelmiston_maarittely/maarittelydokumentit.html. Luettu 29.5.2013

Valtioneuvoston asetus liikenteessä käytettävien ajoneuvojen liikennekelpoisuuden valvonnasta 19.12.2002/1245