VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Joni Hyyppä

TUOTANNONOHJAUS

Tuotantolinjan hienokuormitusohjelma

Liiketalous ja matkailu
2015

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

TIIVISTELMÄ

Tekijä	Joni Hyyppä
Opinnäytetyön nimi	Tuotannonohjaus, tuotantolinjan hienokuormitusohjelma
Vuosi	2015
Kieli	suomi
Sivumäärä	70 + 4 liitettä
Ohjaaja	Sirkka Hellman

Opinnäytetyön tavoitteena oli luoda protoversio tuotantolinjan hienokuormitusohjelmasta, jonka päätavoite oli tutkia, minkälainen hienokuormitusohjelma Vacon Oyj:n tuotantoon olisi paras mahdollinen.

Opinnäytetyö koostuu neljästä osasta. Ensimmäisessä osassa perehdytään tarkemmin teolliseen tuotantoon ja siihen, minkälainen merkitys tuotannonohjauksella on saavuttaa yrityksen asetetut tavoitteet. Tuotannonohjauksessa perehdytään myös tuotannon kuormitukseen ja siihen, miten kuormitus olisi parasta toteuttaa.

Toisessa osassa avataan opinnäytetyössä käytetyt menetelmät ja tekniikat sekä käytettävyyden merkitystä ohjelmistoprojektissa. Opinnäytetyön prosessi toteutettiin yhdistämällä ketterän menetelmän, protoilun ja Lean-ajattelutavan ominaisuudet, jotka katsottiin sopivan parhaiten tähän ohjelmistoprojektiin. Käytettävyyttä tutkimalla pyrittiin luomaan yksinkertainen ja selkeä käyttöliittymä.

Kolmannessa osassa määritellään työn vaatimusmäärittely ja avataan työn suunnittelua ja toteutusta. Vaatimusmäärittelyt toteutettiin käyttäjätarinoina ja UML-kaavioiden avulla, selkiyttämään hienokuormitusohjelman toimintoja. Vaatimusmäärittelyt olikin tässä projektissa haastavin osuus, koska aihe oli jokaiselle projektin jäsenelle melko uusi ja tämän vuoksi vaatimukset muuttuivat testivetoisen kehityksen ja jatkuvan testaamisen vuoksi, työn suunnittelun ja toteutuksen aikana.

Neljännessä osassa kuvaillaan työn tulokset ja arvioidaan projektin toteutus käytetyillä menetelmillä. Työn päätavoitteet saatiin toteutettua projektille määritellyssä aikataulussa. Lopuksi avataan hienokuormitusohjelman testauksessa käytetyt tekniikat. Pilottitestauksessa tulleita uusia toiminnallisuuksia, joita ei keritty toteuttamaan projektin aikataulun puitteissa, kuvaillaan työn jatkokehitysosiossa.

Avainsanat tuotannonohjaus, hienokuormitus, ohjelma

UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittelyn koulutusohjelma

ABSTRACT

Author	Joni Hyyppä
Title	Production Management; Production Line for Advanced Planning and Scheduling program
Year	2015
Language	Finnish
Pages	70 + 4 Appendices
Name of Supervisor	Sirkka Hellman

The aim of the thesis was to create a prototype version of a state advanced planning and Scheduling program what main objective was to investigate what kind of advanced planning and scheduling program would be the best as possible for production at Vacon.

The thesis consists of four sections. The section part deals with industrial production and the way the set targets can be in achieved the set goal in the production control. Production management focuses on the production load and how it should best be done.

The second section of the thesis describes the methodology and techniques as well as the importance of the availability in a software project. The thesis process was carried out by combining element of the agile method and Prototyping and Lean-thinking features that were considered most appropriate for this software project. By usability examining the aim was to create a simple and clear user interface.

The third section describes the work requirement specifications and explores work planning and implementation. Requirement specifications were implemented by using user stories and UML diagrams to clarify the finite capacity of the program functions. Requirements specifications were the most challenging part of this project, because the topic was relatively new and for each member of the project therefore the requirements changed test-driven development and continuous testing during design and the execution.

The fourth section describes the results of the work and estimates the used methods in the project implementation. The main objectives of the work were completed for the specified project on schedule. The techniques that were used for the advanced planning and scheduling program are described at the end. New functionalities which came out in pilot testing but could not be implemented in the project within the set time schedule are described in further development section of the thesis.

Keywords production management, advanced planning & scheduling, program

SISÄLLYS

TIIVISTELMÄ
ABSTRACT
KUVIO- JA TAULUKKOLUETTELO
MÄÄRITELMÄT JA LYHENTEET
1 JOHDANTO 8
1.1 Työn ongelma-alue 8
1.2 Opinnäytetyön tavoitteet 8
1.3 Opinnäytetyön rajaukset 9
1.4 Tutkimusmenetelmät 10
1.5 Rakenne 10
1.6 Kohdeyritys 11
2 TUOTANTO JA TUOTANNONOHJAUS 13
2.1 Tuotanto 13
2.2 Tuotannonohjaus 14
2.2.1 Vaikuttavat tekijät 15
2.2.2 Toiminnot tuotannonohjauksessa 17
3 TYÖKALUT JA TEKNIIKAT 21
3.1 Microsoft Access 21
3.2 SQL 22
3.3 Makro 23
3.4 VBA 23
3.5 Ketterä ohjelmistoprojekti 24
3.6 Protoilu 29
3.7 Extreme Programming (XP) 29
3.8 Ohjelmistokehitys ja Lean 33
4 SUUNNITTELU OHJELMISTOPROJEKTISSA 36
4.1 Tietokannan rakenne 36
4.2 Vaatimuksien määrittely UML:ää käyttäen 37
4.3 Käytettävyys 41
5 VAATIMUKSIEN MÄÄRITTELY 44
5.1 Nykytila 44

5.2 Tuleva tila 45
5.3 Vaatimusmäärittely 45
5.3.1 Käyttäjätarinat 47
5.3.2 Käyttötapaukset 48
6 KÄYTTÖLIITTYMÄN SUUNNITTELU 52
7 TIETOKANTA 56
8 OHJELMAN TOTEUTUS 57
8.1 Tietokanta 57
8.2 Käyttöliittymä 57
8.2.1 Päivänäkymä 57
8.2.2 Viikkonäkymä 62
8.2.3 Tuntiseuranta 62
8.2.4 Osaamismatriisi 63
8.2.5 Testivetoinen kehitys: Uudet vaatimukset 63
9 TESTAUS 65
10 YHTEENVETO 66
11 JATKOKEHITYS 68
LÄHTELUETTELO 69
LIITTEET

KUVIO- JA TAULUKKOLUETTELO

Kuva 1.	Tuotantomuodot	17
Kuva 2.	Karkeakuormituksen tasaus	18
Kuva 3.	SQL-kysely	22
Kuva 4.	Makro, lomakkeen sulkeminen	23
Kuva 5.	Hakukenttä	23
Kuva 6.	VBA-koodi	24
Kuva 7.	XP prosessimalli	32
Kuva 8.	Käyttötapauskaavio, Tiiminvetäjä	38
Kuva 9.	Luokkakaavio	39
Kuva 10.	Aktiviteettikaavio	40
Kuva 11.	Sekvenssikaavio, työvuorolistan tulostaminen	41
Kuva 12.	Käyttäjätarinat	47
Kuva 13.	Henkilöiden oikea määrä	58
Kuva 14.	Tavoitteiden hahmottaminen	63
Taulukko 1.	Tuotanto-ohjelma	19
Taulukko 2.	Lean, 7 hukkaa	34

MÄÄRITELMÄT JA LYHENTEET

Lean	johtamisfilosofia. Leanin avulla poistetaan tuottamattomat toiminnot tuotannosta.
Team Leader (TL)	tuotantolinjan tiiminvetäjä
Päällekkäisyys	tarkoitetaan tietoa, joka on tallennettu moneen eri paikkaan
Proto	ensimmäinen versio tuotteesta
Data Warehouse	tietovarasto

1 JOHDANTO

1.1 Työn ongelma-alue

Opinnäytetyön ongelma-alue kuvaillaan seuraavasti:

- kuormitus toteutettu ainoastaan karkealla tasolla
- tiedon kartoitus tietokantaan
- käyttöliittymän toiminnalliset ja ei-toiminnalliset vaatimukset
- projektin prosessi.

Vaconilla ei ole ollut aiempaa ohjelmaa, joka olisi kuormittanut tuotantolinjat hienokuormitustasolla, vaan kuormitus on toteutettu ainoastaan karkealla tasolla.

Seuraavat kysymykset ovat, mitä tietoja ohjelmaan tarvitaan ja mitä toiminnallisuuksia käyttöliittymältä halutaan. Koska aiempaa ohjelmaa ei ole olemassa, varmasti suurin kysymys ohjelman luontiin on, onko kaikki tarvitsemani tieto jo jossain tuotannonjärjestelmissä vai täytyykö työtä varten luoda joitakin uusia tietoja, jotta ohjelman saisi toiminaan haluamalla tavalla.

Kun ollaan uuden aiheen äärellä, projektissa käytettävät tekniikat ja menetelmät, täytyy pyrkiä valitsemaan sopiviksi tämänkaltaiseen projektiin.

1.2 Opinnäytetyön tavoitteet

Opinnäytetyön tavoitteena on luoda hienokuormitusohjelma tuotantolinjojen kuormittamiseen, jolla mahdollistetaan tiiminvetäjien tehokkaampi työskentely. Tuotantolinjan hienokuormitusohjelma on ensimmäinen vaihe kohti tehokkaampaa tuotannonohjausta. Tämä hienokuormitusohjelma ei ole lopullinen versio, vaan ohjelman tarkoitus on näyttää, miten tuotanto haluaa tuotantolinjojen hienokuormituksen toteutuvan. Vaasassa Vaconilla on 14 erilasta tuotantolinjaa ja noin 400 tuotannontyöntekijää, työn tarkoitus on tehostaa juuri tuotantolinjojen hienokuormittamisen osa-aluetta tuotannonohjauksesta.

Työn tavoitteet ovat seuraavat:

- tutkia, mitä tuotanto haluaa hienokuormitusohjelmalta
- luoda proto
- tehostaa tiiminvetäjän työskentelyä ohjelman avulla
- toteuttaa projekti, käyttäen ketterää projektinhallintaa ja testivetoista kehitystä
- parantaa tuotannonohjausta
- kerätä tieto, mitä tarvitaan Data Warehousen luomiseen.

Ohjelman tavoite on kuormittaa tuotantolinja hienokuormitustasolla. Tiiminvetäjän on nähtävä ohjelmasta tulevat tilaukset ja henkilötarpeen kullekin päivälle. Ohjelmaan on tarkoitus luoda neljä näkymää, päivänäkymä, viikkonäkymä, tuntiseuranta ja osaamismatriisi. Päivänäkymässä tiiminvetäjä kuormittaa tuotantolinjan tämän hetkiselle päivälle, eli kuka henkilö menee tekemään mitäkin laitetta ja mille tuotantoputkelle. Viikkonäkymässä tiiminvetäjää näkee +7 päivän tilanteen tulevista tilauksista ja pystyy tämän ansiosta näkemään, onko tulevina päivinä tulossa isoja piikkejä tilauksista. Tuntiseuranta näyttää tiiminvetäjälle tuntitasolla valmistuneet laitteet ja tavoitteet. Osaamismatriisin tarkoitus on näyttää ne laitteet, joita henkilöt ovat valmistaneet aiemmin.

Opinnäytetyö myös edesauttaa Data Warehousen luomista. Hienokuormitusohjelmaan käytetyn datan avulla nähdään se, mitä kaikkea tietoa tarvitaan Data Warehousen toteuttamiseen.

1.3 Opinnäytetyön rajaukset

Hienokuormitusohjelma tulee pilottikäyttöön ainoastaan yhdelle linjalle ja tietokanta tullaan rajoittamaan ainoastaan pilottilinjan tuotteilla. Käyttäjänä tulee toimimaan pilottilinjan tiiminvetäjä.

Opinnäytetyön rajaus tapahtuu seuraavasti:

- tuotannonohjauksen sisäistäminen
- tutustuminen projektihallinnan työkaluihin
- tuotantolinjan hienokuormitus

- tiedon määrittäminen ja kerääminen
- tietokannan suunnittelu ja toteutus
- käyttöliittymän suunnittelu ja toteutus.

1.4 Tutkimusmenetelmät

Työn prosessi eteni ketterän menetelmän (Agile) XP:een mukaisesti, jonka pääpaino keskittyy ohjelmointiin. Työn prosessiin otettiin myös mukaan menetelmiä Leanin ajatusmallista ja protoilusta. Tarkoitukseni oli saada jokainen käyttötapaus toimimaan yksitellen, joka mahdollisti näyttää käyttäjälle jatkuvasti sen, missä vaiheessa edetään. Sain myös välittömästi palautetta, jos toiminnallisuus ei vastannut odotuksia. Tämän tyylinen ohjelmistoprojekti toimi mielestäni hyvin, koska protoilun ansiosta huomattiin myös sellaiset toiminnot, joille ei ollutkaan käyttöä.

Työn aiheen saatuani, aloin tutkimaan ohjelman vaatimuksia ja määrittelyä mikä on tämän hetkisen kuormitusohjelman tilanne, mitä ongelmakohtia työhön liittyy, mitä tietoja ohjelma tarvitsee ja kenelle ohjelma tulisi käyttöön.

Ohjelman suunnittelussa ja toteutuksessa on otettu huomioon työn tärkein tavoite, eli tiiminvetäjän työn tehostamisen. Ohjelman käytettävyys täytyy olla yksinkertainen ja selkeä, jotta tiiminvetäjän olisi helppo tulkita ja käyttää ohjelmaa.

1.5 Rakenne

Aluksi käydään läpi tuotannon ja tuotannonohjauksen määritykset ja kuormittamisen peruskäsitteet. Aihe oli itselleni vieras, joten tämän vuoksi niiden ymmärtäminen on isossa roolissa ohjelmaa suunnitellessa. Työkalut ja tekniikat -kappaleessa kerrotaan työssä käytetyistä työkalusta ja tekniikoista tarkemmin ja avataan aihetta siitä, miten tietokannan ja käyttäjäystävällisen sovelluksen toteuttamiseen vaaditaan. Kappaleessa kuvataan myös työn prosessissa käytettyjä menetelmiä tarkemmin.

1.6 Kohdeyritys

Vaconin toiminta perustuu taajuusmuuttajien tuotekehitykseen, valmistukseen ja markkinointiin. Vacon perustettiin vuonna 1993 kolmentoista ABB:n henkilön toimesta, Vaconin toiminimi oli tällöin Vaasa Control ja ensimmäiset taajuusmuuttajat Vacon lanseerasi vuonna 1995. Vacon on kasvanut suuresti vuosien varrella, vuonna 2009 Vaconin liikevaihto oli 272,04 miljoonaa euroa ja vuonna 2013 403 miljoonaa euroa. (Kauppalehti 2015; Vacon Oyj, yhtiö.)

Vaconilla on useita tuotantotehtaita ja tuotekehitystä ympäri maailmaa, kuten Pohjois-Amerikassa, Aasiassa ja Euroopassa. Vuonna 2013 yhtiö työllisti yhteensä 1600 henkilöä. Tanskalainen Danfoss teki (9/14) ostotarjouksen Vaconista 1,038 miljardilla eurolla ja (12/14) Vacon ja Danfoss yhdistyivät. (Kauppalehti 2014; Vacon Oyj, Yhtiö.)

Vaconin taajuusmuuttajat perustustuvat asioihin, joilla voidaan pienentää hiilidioksidipäästöjä ja säästää energiaa, täten Vaconin taajuusmuuttajilla mahdollistetaan tuote, joka lisää energiatehokkuutta ja hyödyntää käyttöönsä uusiutuvaa energialähdettä. Vuonna 2013 pelkästään Vaconin taajuusmuuttajia käyttämällä säästettiin noin 55 TWh sähköenergiaa. Vaconin tuotteiden lisäksi, myös Vaconin tuotantoprosessit ovat matalapäästöisiä, koska tuotantoprosessi pitää sisällään ainoastaan loppukokoonpanon ja testauksen, jotka toteutetaan Vaconin tehtaalla. (Vacon Oyj, Tärkeimmät teemat.)

Vaconin tehdas pitää sisällään kaiken kaikkiaan seitsemän tuotantohallia. Vaconin tuotteet luokitellaan tuoteperheisiin, joita valmistetaan usealla tuotantolinjalla. Tuotantolinjat toimivat tiimeittäin, johon kuuluu pääsääntöisesti tiiminvetäjä ja 1–2 perehdyttäjää vuoroa kohden. Tuotantolinjoilla kuitenkin tuotetaan pääsääntöisesti ainoastaan yhtä tuoteperhettä. Vaconin tuotantoprosessi perustuu valmistamaan tuotteita tilauskohtaisesti, jossa tuotteista tehdään aluksi moduli, joka räätälöidään asiakkaalle vasta suunniteltuna valmistuspäivänä, jonka ajankohta on päivää ennen sovittua toimituspäivää.

Työntekijät työskentelevät yleensä kahdessa vuorossa, mutta muutamalla tuotantolinjalla on käytössä myös yövuoro. Pienten teholuokkien testaaminen toteutetaan kahdessa vuorossa ja suurten teholuokkien kolmessa vuorossa.

Tuotantolinjat jaetaan prosessi- ja kokoonpanolinjoihin. Prosessilinjoissa henkilöstön tehtävä on ainoastaan valvoa valmistusprosessia. Valmistuksessa käytettävät raaka-aineet ovat nestepitoisia ja tämän vuoksi prosessilinjat rakennetaan kiinteiksi, jotta materiaalien kulku tapahtuisi suoraviivaisesti. (Miettinen 1993, 32.)

Kokoonpanolinjan materiaalit voivat vaihdella ja kokoonpanolinjalla valmistetaan, joko tuotteita valmiiksi tai komponentteja. Henkilöstömäärät ovat yleensä suuria ja tuotteen valmistus etenee työpisteeltä toiselle. (Miettinen 1993, 32–33.)

2 TUOTANTO JA TUOTANNONOHJAUS

Tässä kappaleessa käydään läpi tuotantoon liittyvät käsitteet ja niiden merkitys. Aihe painottuu suurimmaksi osaksi tuotannonohjaukseen ja tuotannon kuormitukseen.

2.1 Tuotanto

”Tuotanto on asiakkaan tarpeiden tyydyttämistä edistävää toimintaa, jonka tuloksena tuotannontekijöiden avulla luodaan aineellisia ja aineettomia hyödykkeitä.” (Miettinen 1993, 11.)

Tuotantoa ja valmistusta on käsitelty samana asiana, koska eihän tuotteita syntyisi ilman valmistusta ja siitähän tuotannossa on kyse, eli valmistaa tuotteita. (Haverila, Uusi-Rauva, Kouri & Miettinen 2005, 350–351.)

Nykyään tuotannolla tarkoitetaan asioita, jotka liittyvät yrityksen kaikkiin toimintoihin tuotannossa, joilla mahdollistetaan tuote ja palvelu. (Haverila ym. 2005, 351). Yritysten toimintaympäristö on kehittynyt radikaalisti ja yritykset joutuvat entistä voimakkaammin kilpailemaan keskenään. Tämä aiheuttaa suuria ja uudenlaisia haasteita tuotannonohjaukselle. (Miettinen 1993, 7.)

Yritykset haluavat nykyään panostaa joustavuuteen, joka vaatisi monitaitoisia työntekijöitä. Osien standardointi ja vaihdettavuus ovat olleet suuressa osassa tuotantojen kehittymisessä. Enää ei ollut väliä mihin lopputuotteeseen kukin osa tultaisiin sijoittamaan, jonka ansiosta mahdollistettiin kasvattamaan tuotannon valmistusmääriä ja helpotettiin tuotannon suunnittelun toteuttamista. (Miettinen 1993, 10.)

Teollisen tuotannon ensimmäiset askeleet otettiin Euroopassa 1800-luvulla, jolloin massatuotantoa alettiin toteuttaa. Siihen aikoihin tunnettu käsityö syrjäytettiin sarjatuotannolla, jonka edellytykset vaativat yritykseltä tuotesuunnittelun ja valmistustekniikan. (Lehtonen 2004, 60.) Kun tuotantoa toteutetaan tiloissa, jossa tuodaan markkinoille tuotteita, kutsutaan sitä teolliseksi tuotannoksi. Tuotteita joko tuotetaan tilauspohjaisesti tai kerätään varastoon. Yritykset valmistavat tuot-

teita joko oman suunnitelman mukaan tai räätälöivät niitä asiakkaan tarpeiden mukaan. (Miettinen 1993, 9; Uusi-Rauva, Haverila, Kouri 1994, 324.)

Teollisen tuotannon massatuotannolla mahdollistetaan edulliset hyödykkeet, jotka ovat hyvän elintasomme perusta. Ilman edullisia hyödykkeitä hyvinvointimme ei olisi läheskään näin hyvällä tasolla. (Uusi-Rauva ym. 1994, 324.) Henry Fordin keksintö vuonna 1913 on ehkä yksi historian merkittävimmistä teolliseen massatuotantoon liittyvistä yksittäisistä keksinnöistä. Ford yksinkertaisti työn ja määritteli auton osat standardeiksi. Ford loi samalla uuden tavan organisointiprosessille, jossa sarjatuotanto toteutettiin liukuhihnalla. Tämä mahdollisti halvempien autojen hinnan ja laajensi suuresti automyynnin asiakaskuntaa, koska autoja pystyivät ostamaan myös tavalliset kansalaiset. (Lehtonen 2004, 60.)

2.2 Tuotannonohjaus

Tuotannonohjauksen määrittely ja käsite alkoi syntyä 1950-lvulla, nimeä tuotannonohjausta ei vielä ollut, joten sitä kutsuttiin työnsuunniteluksi. Myöhemmin 1960-luvulla käsite työnsuunnittelu meni hieman eteenpäin, yritykset kohdistivat näkökulmansa varastoihin ja 1970-luvalla varastoiden ennakointiin, tällöin työnsuunnittelua alettiin kutsua nimellä tuotannonsuunnittelu. (Hokkanen 2011, 209.)

Hokkanen jakaa tuotannonohjauksen tavoitteet neljään kategoriaan:

- hyvä toimituskyky
- hyvä kapasiteetin käyttöaste
- pieni vaihto-omaisuuteen sidottu pääoma
- lyhyt kokonaisläpäisyaika (Hokkanen 2011, 209.).

Tuotannonohjauksella pyritään saamaan erilaiset tuotantojärjestelmien osat toimimaan yhdessä. Tuotantojärjestelmiä ovat markkinointi, myynti, tuotanto ja logistiikka. (Miettinen 1993, 23.)

Tuotannossa on yleensä monta erilaista ohjausjärjestelmää, kuten tuotannon-, talouden-, materiaalin-, laadun- ja markkinoinninohjausjärjestelmät. Jokaista ohjausjärjestelmää ei tulisi pitää erillään toisista, vaan ohjausjärjestelmien täytyisi

toimia yhdessä, osasana koko yrityksen systeemiä. Tuotannonohjaukset tärkeimmät tehtävät ovat suunnittelu, toteutus, informointi ja valvonta. (Miettinen 1993, 23.)

”Tuotannon ohjattavuus on tuotantojärjestelmän kyky saavuttaa sille asetetut operatiiviset ohjaustavoitteet.” (Miettinen 1993, 23.)

Ohjattavuudella on pyrkimys yhdistää tuotantoprosessi ja siihen kuuluvat resurssit. Resursseja ovat työntekijät, laitteet ja tilat. Miettinen jakaa ohjattavuuteen vaikuttavat tekijät kahteen luokkaan: ulkoiset ja sisäiset tekijät. On asioita kuten asiakkaiden toiveet ja kausivaihtelut, joihin yritys ei pysty itse vaikuttamaan ohjauksella, näitä asioita kutsutaan ulkoisiksi tekijöiksi. Sisäisiin tekijöihin luokitellaan kaikki tuotannon sisällä olevat asiat, joita yritys pystyy ohjaamaan, kuten tuotteen läpimenoaika ja varastojen koko. (Miettinen 1993, 24.)

2.2.1 Vaikuttavat tekijät

Tuotannonohjaukseen vaikuttavat tekijät Miettinen jakaa kuuteen ryhmään:

- asiakas
- aikajänne
- toiminta-ajatus
- liikeidea
- tuotantomuoto
- valmistusjärjestelmä (Miettinen 1993, 27.)

Yritysten välisen kilpailun koveneminen on nostanut **asiakkaiden** merkityksen vielä korkeammalle. Asiakkaiden toiveita on alettu kuuntelemaan varsinkin jo tuotteiden suunnittelun alkuvaiheessa, jotta asiakkaalle saataisiin valmistettua juuri sellainen tuote, kuin asiakas on toivonut. (Miettinen 1993, 27.) Tilaustuotteissa, kuten Vaconin taajuusmuuttajissa, asiakas pystyy hyvin vahvasti vaikuttaan tuotteen loppukokoonpanoon ja ominaisuuksiin.

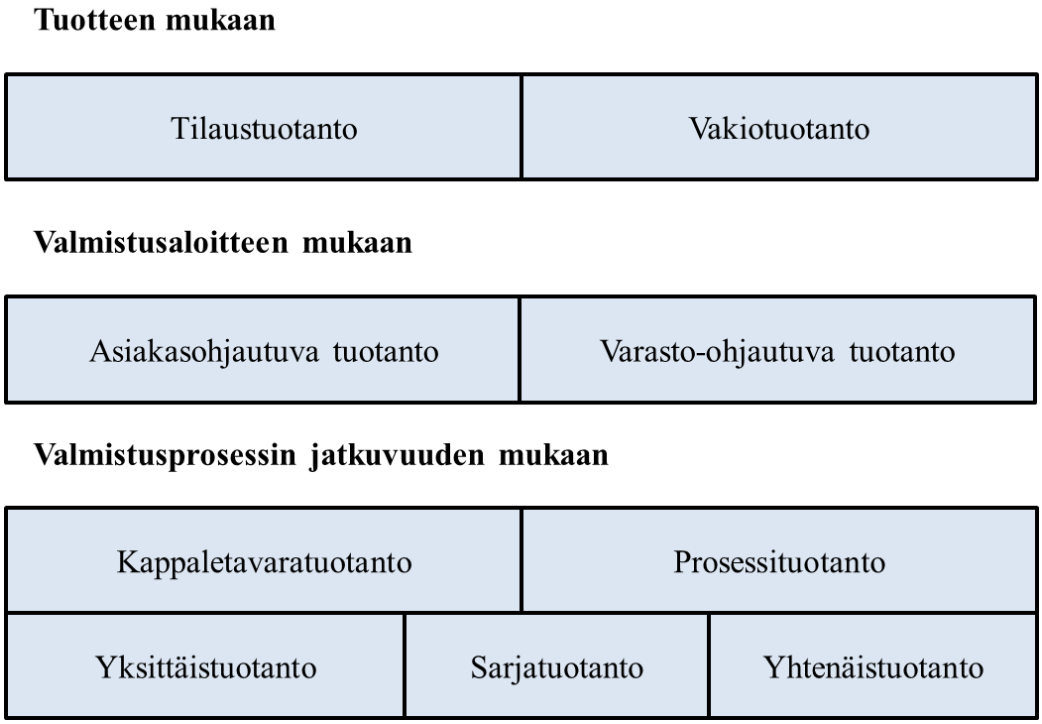
Yritykset pyrkivätkin käyttämään tuotteissaan modulointia, jolla helpotetaan eri asiakasvariaatioiden hallintaa. Tuotteen valmistuksessa otetaan asiakkaan toiveet huomioon vasta valmistusprosessin loppupäässä. (Miettinen 1993, 27.)

Aikajänteellä pyritään vaikuttamaan strategisen, taktisen ja operatiivisen suunnittelun tarkkuuteen. Tuotteiden elinkaarien lyhentyessä ja yrityksien toimintaympäristöjen muuttuminen dynaamisemmaksi, on aiheuttanut sen, että suunnittelujaksot ovat lyhentyneet merkittävästi. (Miettinen 1993, 28.)

Yrityksen **toiminta-ajatuksella** kerrotaan yrityksen suuntautuminen, eli mihin tarpeeseen tuotteita toteutetaan ja millä teknologialla tarpeet saadaan tyydytettyä. Toiminta-ajatus olisi parasta toteuttaa yleisellä tasolla, jolla kestettäisiin parhaiten uuden teknologian tuomat haasteet. (Miettinen 1993, 28–29.)

Yrityksen **liikeidea** muodostuu yleensä yrityksen kehittymisen johdolla, jossa yritys määrittelee tuotteet, asiakkaat ja palvelut. Yrityksen pyrkimys on kehittää liikeideaa jatkuvasti, jotta se vastaisi yrityksen toimintaympäristöä. Esimerkiksi, jos toimintaympäristö on matkan varrella kehittynyt suuntaan tai toiseen. (Miettinen 1993, 29.)

Tuotantomuodolla tuotanto luokitellaan jonkun luokittelun mukaan, kuinka yritys aikoo tuotantonsa toteuttaa. Tuotantomuodot toimivat perustana yrityksen tuotannolle, kuten toiminnan johtamiselle ja ohjaamiselle. Tuotantomuotojen jaon Haverila ym. määrittelevät tuotteen, valmistusaloitteen ja tuotantoerän koon mukaan (kuva 1). (Miettinen 1993, 29; Haverila ym. 2005, 353.)



Kuva 1. *Tuotantomuodot. (Haverila ym. 2005, 354.)*

2.2.2 Toiminnot tuotannonohjauksessa

Tässä luvussa perehdytään tuotannonohjauksen toimintoihin ja käydään läpi toimintoihin kuuluvia termejä.

Miettinen jakaa tuotannonohjauksen toiminnot seuraaviin ryhmiin:

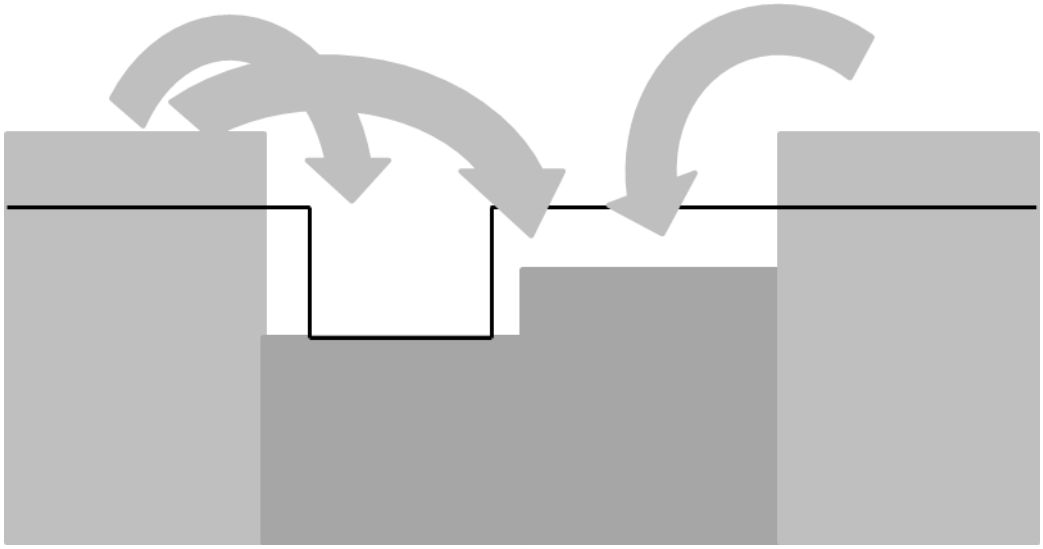
- tuotantosuunnittelu
- tuotantotekninen suunnittelu
- työnjärjestys
- tehdaspalvelu (Miettinen 1993, 36.)

Tuotantosuunnitelmalla on tavoitteena arvioida yrityksen kysynnän ennustaminen, kapasiteetin tarkka määrittely ja kuormitussuunnitelma. Näiden tuloksena syntyy yrityksen tuotanto-ohjelma. Tuotantosuunnittelu pyrkii kuormittamaan tuotantokapasiteetin mahdollisimman tasaiseksi, jotta toimitusajoissa kyetään pysymään. Yrityksellä täytyy myös olla tarkat tiedot käytettävissä olevasta kapasiteetista ja tulevasta kysynnästä. (Miettinen 1993, 36–37.)

”Kapasiteetilla tarkoitetaan tuotantokykyä, joka muodostuu tuotantokoneista, välineistä, tehdastilasta, työvoimasta ja energiasta.” (Miettinen 1993, 36.)

Kuormituksen suunnitellussa pääperiaate on kuormittaa määritelty kapasiteetti töille. Suunnittelu toteutetaan tehtaalla ja kuormitusryhmissä. Kuormitusryhmillä tarkoitetaan resurssia, jotka ovat samanlaisia ja toisiaan korvaavia. Kuormituksen suunnittelulla pyritään myös hankkimaan lisäkapasiteettia, jos tuotantotavoitteiden saavuttamiseksi tarvitaan ylitöitä. (Miettinen 1993, 38.)

Kuva 2 kuvaa kuormituksen periaatetta raaka-kuormituksen näkökulmasta. Kuormitussuunnitelman on vastattava kapasiteettia, eli jos tälle päivälle on liikaa töitä ja nettokapasiteetti ei riitä, ovat tämän päivän työt kuormitettava huomiselle, tällä tarkoitetaan kuormituksen tasaamista. Kuvassa 2 oleva viiva kuvaa yrityksen käytössä olevaa nettokapasiteettia ja pykälät tulevia töitä. Kuormituksessa on otettava huomioon luvatut toimitusajat, jotta niissä pysytään kiinni, jos kuitenkin töitä on liikaa ja kuormituksen tasauksella ei voida päästä toteuttamaan määrättyä toimitusaikaa, tällöin yrityksen täytyy hankkia lisäkapasiteettia töiden tekemiseen. (Miettinen 1993, 38–39.)



Kuva 2. *Karkeakuormituksen tasaus. (Miettinen 1993, 39.)*

Tuotanto-ohjelma (taulukko 1) perustuu myyntisuunnitelmaan, kun toteutetaan tuotanto-ohjelmaa varastotuotantoon ja tilaustuotannossa toteuttamiseen käytetään

tilauskirjaa. (Miettinen 1993, 39.) Tuotanto-ohjelman toteuttaminen tuo haasteita, jos tuotanto toteutetaan sarjatuotannossa, koska sarjatuotannossa tuotteita valmistetaan samanaikaisesti ja tuotteiden välillä siirrytään useammin. (Miettinen 1993, 41.)

	Tammi	Helmi	Maalis	Huhti	Touko	Kesä
Tuotelinja A						
Tuote 1	10	20	10	5	10	20
Tuote 2	30	25	20	20	30	30
Tuote 3	1	2	1	2	1	2
Yhteensä	41	47	31	27	41	52
Tuotelinja B						
Tuote 4	200	270	290	200	290	300
Tuote 5	300	230	220	300	215	180
Yhteensä	500	500	510	500	505	480

Taulukko 1. *Tuotanto-ohjelma. (Miettinen 1993, 40.)*

Tuotantoteknisessä suunnittelussa otetaan huomioon tuotteiden osien valmistuksen työnkulku, menetelmäsuunnittelu, työvälinesuunnittelu ja työvaiheajat, joilla suunnitellaan tuotannon ja kapasiteetin käyttö. (Miettinen 1993, 40.)

Työnjärjestelyn tavoitteena on valmistaa tuote. Tällä varmistetaan, että kaikki tuotteen valmistuksessa käytettävät asiat ovat kunnossa, kuten työvälineet ja materiaalit. (Miettinen 1993, 41.)

Tehdaspalvelu vastaa yrityksen resurssien kunnossapidosta. Tehdaspalvelun tarkoitus on pitää resurssit hyvässä kunnossa, jotta yritys saavuttaa hyvän toimintavarmuuden. Tehdaspalvelun työnkuvaan kuuluu ehkäisevä kunnossa pito, vikojen korjaaminen niiden tapahduttua, rutiinikunnossapito ja suunnitelmallinen kunnossapito. (Miettinen 1993, 43.)

Nykyään tärkeään rooliin on noussut ehkäisevä kunnossapito, jolla ennaltaehkäistään vikojen syntyminen ja täten pienennetään kunnossapidon kustannuksia ja parannetaan toiminnan käytettävyyttä ja laatua. (Miettinen 1993, 43–44.)

3 TYÖKALUT JA TEKNIIKAT

Hienokuormitustyökalu toteutetaan Accessilla, jolla luodaan tietokanta ja käyttöliittymä. Tiedot määritellään SQL-kyselyillä ja käyttöliittymän toiminnallisuus toteutetaan Microsoftin Accessin Makroilla ja Visual Basicilla. Visual Basicissa ohjelmointi toteutetaan Microsoftin Accessin VBA-kielellä. Tässä kappaleessa käydään myös läpi työssä käytettyä projektinhallintaa ja UML-kaaviot.

3.1 Microsoft Access

Microsoft Access on työkalu, jolla voidaan toteuttaa tietokantoja (Database), kyselyitä (Query) ja käyttöliittymiä, eli lomakkeita (Form). Käyttäjällä on mahdollista luoda tietokantasovelluksia omalle koneelle tai julkaista ne verkossa. Verkkoon julkaistu tietokantasovellus mahdollistaa muille käyttäjille selata tietokantaa selaimella.

Yleensä Accessiin siirrytään siinä vaiheessa kun tavalliset taulukkolaskentaohjelmat eivät enää riitä ja huomataan, olevan ongelmissa päällekkäisyyksien kanssa. Accessissa päällekkäisyydet vähenevät relaatiotietokannan avulla. Relaatiotietokannalla tarkoitetaan tietokantaa, jossa tiedot ovat jaettu tauluihin ja taulut ovat yhdistetty keskenään tauluissa sisältävien tieto-osien eli kenttien avulla. Käyttäjällä on myös mahdollista tuoda Accessiin valmiita Excel-laskentataulukoita ja Access osaa siirtää tiedot omiin tauluihin ohjatulla taulujen analysoinnilla, jos käyttäjä näin haluaa toimia. (Access 2010:n perustoiminnot 2010).

Accessilla voi luoda tietokannan ja käyttöliittymän samassa työkalussa. Kun sovellusta on tarpeeksi hiottu ja toiminnallisuus alkaa olla kunnossa, seuraavaksi siitä on helppo siirtyä Visual Studioon jatkokehittään sovellusta. Jos sovellus ei ole kovin iso tai käyttäjiä ei ole monta, silloin ei välttämättä tarvitse siirtyä ollenkaan Visual Studioon tai johonkin muuhun työkaluun. Jossain tilanteissa tietokanta alkaa olla liian suuri, jolloin Access hidastuu ja silloin voidaan miettiä siirtymistä toiseen työkaluun.

3.2 SQL

SQL on standardikieli, jolla käsitellään tietoja relaatiotietokantaohjelmissa. SQL:llä määritellään tietojoukko, jota halutaan näyttää tai etsiä. SQL-kielelle on määritelty syntaksi eli tietojoukkojen hakeminen vaatii SQL-lauseeseen oikeanlaisen järjestyksen toimiakseen. (Johdanto Accessin SQL-toimintoihin 2010).

SQL-kielen yleisimmät lauseet:

- SELECT (mitä halutaan hakea)
- FROM (taulu, mistä haetaan)
- WHERE (ehdot haulle)
- ORDER BY (lajittelee tiedon)
- GROUP BY (Koostefunktion määrittely, kentät)
- HAVING (Koostefunktion määrittely, ehdot)

Esimerkkilauseessa (kuva 3) haetaan tietyn runkokoon tilauksia päivämäärän mukaan. SELECT lause sisältää yhden koostefunktion, joka on SUM, eli kyseinen toiminto laskee summan sen päivän kappalemäärille, jos niitä sattuu olemaan useampia.

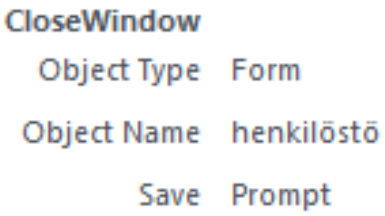
Koostefunktion vuoksi GROUP BY lause on pakollinen, koska SELECT lause sisältää muitakin tietoja kuin pelkästään SUM-koostefunktion. INNER JOIN lauseella mahdollistetaan tiedon hakeminen kahdesta eri taulusta. Näin täytyy toimia, koska ”tbl_tilaus” taulussa ei ole tietoa runkokoosta.

```
SELECT Sum(kappalemäärä) AS Laitteita_yhteensä,
vahvistettu_toimitusPvm, runkotunnus
FROM tbl_tilaus
INNER JOIN tbl_tilausrivi ON tbl_tilaus.tilausnro =
tbl_tilausrivi.tilausnro
GROUP BY vahvistettu_toimitusPvm, runkotunnus;
```

Kuva 3. *SQL-kysely*

3.3 Makro

Makro on Microsoft Accessin oma työkalu, joka sisältyy Accessiin. Makroilla toteutetaan toiminnallisuutta lomakkeisiin. Esimerkiksi voidaan sulkea tietty lomake painikkeen avulla, jonka toiminnallisuutta on muokattu makrolla (kuva 4).



Kuva 4. *Makro, lomakkeen sulkeminen*

3.4 VBA

VBA (Visual Basic for Applications) on ohjelmointikieli, joka toimii Accessissa. VBA-koodia käytetään samanlaiseen tarkoitukseen, kuin edellä mainittua Makroa, eli sovelluksen toiminnallisuuden lisäämiseen. Jos käyttäjä tietää jo tietokannan suunnitteluvaiheessa, että tietokantaa tullaan käyttämään verkkoselaimessa, silloin toiminnallisuus on parempi toteuttaa Makroilla, koska VBA-koodi ei toimi selaimessa. Turvallisuusriski on myös olemassa, joten suositeltavaa on käyttää VBA-koodia ainoastaan silloin, kun tietokanta on sijoitettu omalle koneelle. (Johdanto Accessin SQL-toimintoihin 2010).

Esimerkeissä (kuva 5 & kuva 6) ohjelmoidaan painikkeelle (btnHae) toiminnallisuus, joka hakee tekstikenttään (txtHae) syötetyn tekstin perusteella tietokannasta tietoa. Tuloksena sovellus näyttää ainoastaan ne tiedot, jotka vastaavat syötettyä hakukriteeriä. Ohjelmoijan täytyy tietenkin määrittää hakukriteerit koodin sisälle, jotta sovellus tietää mistä kentistä tietoa täytyy hakea.

Kuva 5. *Hakukenttä*

```
Private Sub btnHae_Click()
Dim strsearch As String
Dim Task As String

'tarkistetaan onko hakukenttään kirjoitettu mitään
If IsNull(Me.txtHae) Or Me.txtHae = "" Then
  MsgBox "Kirjoita hakusana.", vbOKOnly, "Hakusana puuttuu"
  Me.txtHae.SetFocus
Else
'määritetään hakukriteetit
strsearch = Me.txtHae.Value
   Task = "SELECT * FROM putken_ohjaus_päivä WHERE ((runkotunnus Like ""*" & strsearch & "*"") OR
   (kappalemäärä Like ""*" & strsearch & "*"")OR
   (putkitunnus Like ""*" & strsearch & "*"")OR
   (Suunniteltu_valmistumisPvm Like ""*" & strsearch & "*""))"
Me.RecordSource = Task

End If
End Sub
```

Kuva 6. *VBA-koodi*

3.5 Ketterä ohjelmistoprojekti

Tässä kappaleessa kuvataan protoilun ja Extreme Programming (XP) ominaisuudet ja toimintamallit ohjelmistoprojektia toteuttaessa. Toteutin työni prosessin käyttäen molemmista mielestäni tähän työhön sopivimpia ominaisuuksia.

Vuonna 2001 Agile Alliance -niminen ketterä menetelmiä kehittävä järjestö julkaisi kirjan nimeltään Agile Manifesto. Haikalan ja Mikkosen (2011) mukaan kirjassa kuvaillaan turhiksi prosessit, työkalut ja dokumentaatio, mutta ei kuitenkaan täysin sivuuteta ohjelmistonkehitystyön tärkeyttä. Kaikista tärkeimmäksi seikaksi kirjassa kuvailtiin tyytyväinen käyttäjä ja käyttökelpoinen ohjelmisto. (Haikala, Mikkonen 2011, 43–44.)

Manifestin pääperiaatteita ovat Haikalan ja Mikkosen mukaan seuraavat:

- asiakkaalle näytettävä toimiva ohjelmisto alusta lähtien
- hyväksyttävä vaatimuksien muuttuminen projektin aikana
- ohjelmistoversioiden julkaisuväli toteutettava viikoittain
- motivoituneet yksilöt projektiin
- kasvokkain keskustelu

- luottamus työntekijöihin
- tärkein mittari projektille on toimiva ohjelmisto
- ei ylitöitä
- huomion kiinnittäminen tekniseen erinomaisuuteen ja hyvään suunnitteluun
- asioiden tekeminen yksinkertaisilla tavoilla (minimoidaan vähemmän tärkeämmät tehtävät)
- parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseohjautuvasti
- palaverit väliajoin, kuinka tehostaa ja hienosäätää käytettävää työtapaa ja prosessia (Haikala, Mikkonen 2011, 45.)

Ketterien menetelmien yleisimpiin käytöntöihin sisältyvät päiväpalaverit, jatkuva integraatio, jälkitarkastelu, refaktorointi, testivetoinen kehitys, yksikkötestit, yhteisomistus, tasainen tahti, pariohjelmointi ja käyttäjätarinat. (Tolvanen 2015.) Seuraavaksi avataan jokaista käytäntöä hieman tarkemmin.

Päiväpalaverilla tarkoitetaan lyhyitä palavereita, jotka toteutetaan istuen tai seisten. Päiväpalaverin kesto vaihtelee 5-15 minuutin välillä. Palaverissa on koolla koko tiimi ja palaverin tarkoitus on käydä läpi mitä kukin jäsen teki eilen, mitä aiotaan tehdä tänään ja mitkä ongelmat haittaavat työntekoa. Palavereissa jokainen projektin jäsen tietää, mitä kukin jäsen tekee ja kuinka projekti etenee. Jos päiväpalaverissa esiintyy suuria ongelmia, varataan päiväpalaverin jälkeen uusi palaveri samalla päivälle ongelmien ratkomiseen. Palaverin tarkoitus ei ole ”kytätä” miten kukin jäsen hoitaa työnsä vaan palaverin tarkoitus on tuoda läpinäkyvyyttä projektiin ja hoitaa ongelmat pois, jotka haittaavat työskentelyä. Palaveriin saavat osallistua myös projektin ulkopuoliset henkilöt, mutta he eivät voi osallistua keskusteluun, vain ainoastaan kuuntelemiseen. (Tolvanen 2015.)

Jatkuvalla integraatiolla pyritään yhdistämään ohjelmiston kaikki komponentit kertarypistyksenä projektin loppuvaiheessa. Tämä edellyttää jatkuvaa integraatiota ja koko ohjelmiston koostamista alusta loppuun, jotta loppurypistys saataisiin toteutettua mahdollisimman nopeasti. (Tolvanen 2015.)

Jälkitarkastelulla pyritään tehostamaan työskentelyä ja parantamaan työtapoja. Termistä jälkitarkastelu voi nopeasti päätellä, sen pitämistä vasta palaverin lopussa, mutta näin ei kuitenkaan toimita, koska loppupalaveri olisi liian laaja ja tällöin jälkitarkastelussa tuodut hyvät kehitysideat eivät enää liittyisi kyseiseen projektiin. Järkitarkastelu on tarkoitus pitää jokaisen iteraation lopuksi ja tämän ansiosta jälkitarkastelujen pituus lyhenee huomattavasti verrattuna siihen, jos jälkitarkastelut pidettäisiin vasta projektin päätyttyä. (Tolvanen 2015.)

Refaktorointilla pidetään ohjelmointikoodi ymmärrettävänä, selkeänä ja helposti ylläpidettävä. Tämä mahdollistaa muutoksien tullessa koodin helpon ymmärtämisen ja täten muutoksien toteuttaminen on paljon helpompaa. Jokainen projektin jäsen pystyy tekemään muutoksia juuri tämän takia, koska koodi on toteutettu selkeästi ja koodi on helposti ymmärrettävää. (Tolvanen 2015.)

Testivetoinen kehitys pyrkii testaamaan ohjelmistoa jatkuvasti projektin edetessä. Tämä mahdollistaa virheiden pitämisen vähäisenä ja luo virheiden korjaamisesta miellyttävämmän prosessin, kuin projektin loppupäässä, jossa pitäisi korjata iso määrä virheitä yhdellä kerralla. Yleensä testit on toteutettu vasta projektin loppupäässä, mutta tämä saattaa aiheuttaa sen, ettei virheitä enää keritä korjaamaan. (Tolvanen 2015.)

Tolvanen jakaa projektin loppupäässä toteutetun virheenkorjaamisen ongelmat seuraavasti:

- Projektin loppupuolelle muodostuu epämiellyttävä testaa-korjaa -vaihe.
- Projektin aikana kumuloituvat virheet hidastavat ohjelmiston kehittämistä.
- Ohjelmistossa olevat virheet voivat johtaa toimintoihin, jossa virheitä pyritään kiertämään, ja kun lopulta varsinainen virhe korjataan, sen kiertämiseksi tehty purkkakorjaus menee rikki. (Tolvanen 2015.)

Vaatimusmäärittely voidaan joissakin kohdissa korvata, testeillä tehtyjen tulosten perusteella. (Tolvanen 2015.)

Yksikkötesteillä varmistetaan ohjelmiston koodia. Yksikkötestin tarkoitus on testata toimiiko koodi halutulla tavalla ja määrittää miten ohjelman täytyisi toimia.

Testivetoisessa kehityksessä prosessin idea on kirjoittaa testit ennen ohjelmointia ja testit toteutetaan yleensä jatkuvassa integraatiossa, josta kerroin aiemmin. Jatkuvassa integraatiossa testit auttavat löytämään virheet, jotka ovat voineet aiheuttaa esimerkiksi toisen virheen korjaaminen. (Tolvanen 2015.)

Yhteisomistuksessa ohjelmiston omistus kuuluu kaikille projektin jäsenille eli kaikki jäsenet voivat muokata ja lisätä koodia kuhunkin luokkaan ohjelmistossa. Yhteisomistuksessa voi myös olla luokkakohtaisia omistajuuksia, jossa yhdelle jäsenelle kuuluu ainoastaan yksi luokka. Yhteisomistus edellyttää hyvää tiimihenkeä ja jokaisen jäsenen täytyy ottaa vastuu tuotteesta. Virheiden sijainnista huolimatta, virheiden vastuu on aina kaikkien jäsenien vastuulla, eikä ainoastaan sen, jonka toteuttamassa koodissa virhe sijaitsee. (Tolvanen 2015.)

Tasainen tahti kuvastaa projektin työtahtia. Ylitöitä ei toteuteta ja työtahti tulee olla sellainen, jossa jokainen projektin jäsen jaksaisi tehdä sitä loputtomiin. (Tolvanen 2015.)

”Ihminen jaksaa tehdä pitempää päivää tuottavasti muutaman päivän tai viikon, mutta sen jälkeen alkaa väsymys heikentää suoritusta.” (Tolvanen 2015.)

Tolvanen pitää tärkeänä, ettei aivoja rasiteta liikaa ja työmäärät pidetään kohtuullisena, koska ihminen kestää pitkiä päiviä ainoastaan hetken ja tämän jälkeen pitkät päivät alkavat näkyvä työn suorituksessa. Tasaisella työtahdilla pidetään projektin jäsenien työmoraali ja työteho korkealla. (Tolvanen 2015.)

Pariohjelmoinnissa kaksi projektin jäsentä toimii saman tietokoneen äärellä, toinen kirjoittaa koodia ja toinen katsoo vierestä. Aluksi voi tulla sellainen kuva, tämähän on pelkkää ajanhukkausta, mutta pariohjelmoinnin hyödyt ovat kuitenkin merkittävät. Kun toinen jäsenistä ohjelmoi ja vastaan tulee ongelma, vieressä olevalla jäsenellä saattaa olla jo vastaus mietittynä kyseiseen virheeseen. Vieressä oleva jäsen myös huomaa paremmin virheet, jotka koodia kirjoittaneelta jäseneltä saattaa jäädä huomaamatta. Parit myös kehittävät toinen toistaan, toisella jäsenellä voi olla hyvää nippelitietoa esimerkiksi pikanäppäimistä tai muista ohjelmointiin

liittyvistä vinkeistä. Parit voivat myös vaihtaa paikkojaan, ihan kuinka heistä itsestään tuntuu. (Tolvanen 2015.)

Tolvanen kuvailee vielä pariohjelmoinnista kertyviä sosiaalisia vaikutuksia seuraavasti:

- Keskittyminen työntekoon on parempaa, kun joku toinen on mukana.
- Ohjelmoija ei käy yhtä helposti lukemassa sähköpostia, mikä yleensä aiheuttaa katkoksen työn tekemiseen.
- Ulkopuoliset häiriötekijät vähenevät - jos kaksi ihmistä näyttävät keskittyvän tiivisti työn tekemiseen, heitä ei häiritä läheskään yhtä helposti kuin henkilö joka työskentelee yksin. (Tolvanen 2015.)

Käyttäjätarinat toimivat määrittelemään ohjelmistolle vaatimukset. Ketterissä menetelmissä ja kehityksessä ei ole tapana toteuttaa täydellistä vaatimusmäärittelyä. Jostain on kuitenkin lähdettävä ja käyttäjätarinat ovat siihen nopea ja helppo tapa. Muuta ei tarvitse tietää kuin KUKA tekee MITÄ ja MIKSI. Käyttäjätarinoita kirjoitetaan mahdollisimman paljon miettimättä sitä, mitkä kaikki niistä toteutetaan. Myöhemmässä vaiheessa käyttäjätarinoita voidaan hieman tiivistää. (Tolvanen 2015.)

Tolvanen jakaa hyvien käyttäjätarinoiden piirteet seuraavasti:

- **riippumattomuus** (priorisointi ja mitkä ovat toisistaan riippuvaisia.)
- **neuvoteltavuus** (tarinoiden joustavuus.)
- **arvokkuus** (käyttäjälle lisäarvoa, ei ohjelmoijalle.)
- **arvioitavuus** (koska suunnittelu tehdään tarinoiden pohjalta.)
- **pienuus** (suuren monimutkaisia ja riskialttiita, pitkät tarinat jaetaan osatarinoihin.)
- **testattavuus** (tarinat täytyy olla testattavissa.) (Tolvanen 2015.)

Käyttäjätarinoita ei saa pitää täydellisenä määrittelynä, vaan ne ovat asioita joista täytyy keskustella. Hyvin suurissa projekteissa käyttäjätarinat saattavat olla vääränlainen ratkaisu, mutta tarinat voidaan tällöin jakaa ”korkean tason” tarinoihin,

jolloin jokaiselle tarinalle luodaan oma teemansa, mikä edes auttaa tarinoiden ryhmittelyä. (Tolvanen 2015.)

3.6 Protoilu

Kun toteutetaan jokin prototyyppi ominaisuudesta tai toiminnosta ohjelmistoon, kutsutaan sitä protoiluksi. Kohteena voi olla esimerkiksi uuden käyttöliittymän toteuttaminen. Protoilu jaetaan kahteen päävaihtoehtoon, evoluutioprototyyppi (evolutionary prototype) ja poisheitettävä (throw-away prototype). Evoluutioprototyypin tarkoitus on koota kaikki toiminnallisuudet yksi kerrallaan, joista syntyy loppujen lopuksi valmis ohjelmisto. Evoluutioprototyyppi vastaa melkein samaa asiaa kuin iteratiivinen kehitys. (Haikala & Mikkonen 2011, 38.)

Evoluutioprototyypissä on omat ongelmansa. Huonot prototyypit voivat jäädä valmiiseen ohjelmistoon, mikä aiheuttaa tilanteen, jossa prototyyppi toimii täysin oikein, mutta se hidastaa muiden osien toimintaa ja projektin loppusuoralla tällaiset on helppo jättää korjaamatta. Testausvaiheessa tällaiset prototyypit saattavat mennä testausvaiheessa läpi ja täten ongelmat nousevat esiin vasta ohjelmiston valmistuttua. (Haikala, Mikkonen 2011, 39.)

Poisheitettävässä prototyypissä edetään tiettyyn järjestelmän malliin ja aloitetaan tämän jälkeen alusta. Toteutuksen uudestaan aloittamisessa voidaan käyttää hyväksi käytettyjä toimintoja tai korvata ne uusilla. Käyttöliittymän suunnittelussa käytetään yleensä poisheitettäviä prototyyppejä mallintamaan tietynlaista toimintoa käyttöliittymässä. Aluksi toiminnot voidaan kuvailla asiakkaalle tai käyttäjälle ihan vain paperilla ja myöhemmin toteuttaa toiminnot ohjelmoinnilla käyttöliittymään. Käyttöliittymän näkymää ei kannata protovaiheessa toteuttaa valmiin näköiseksi, vaan ohjelmiston on tarkoitus näyttää keskeneräiseltä, ettei asiakas luule ohjelmistoa jo valmiiksi. (Haikala, Mikkonen 2011, 38–39.)

3.7 Extreme Programming (XP)

XP:n menetelmä perustuu ohjelmointilähtöiseen lähestymistapaan, jossa käyttäjän kanssa kommunikointi tapahtuu yleensä koodin perusteella, kuten myös ohjelmiston uusien osien määrittely ja ymmärtäminen. Ohjelmointi onkin yksi XP:n tär-

keimmistä osa-alueista. XP on iteratiivinen ja inkrementaallinen, mikä edellyttää vahvaa läsnäoloa käyttäjältä (on-site), koska XP:ssä ohjelmiston osat rakennetaan pala palalta ja täten käyttäjän rooli on tärkeä ohjelmiston jatkuvan kehityksen seurannassa ja palautteen antamisessa jo aikaisessa vaiheessa. Käyttäjältä saatu palaute mahdollistaa myös jatkuvien muutoksien teon ohjelmoinnissa. (Beck 2001, xvii-xviii.)

XP:ssä ei myöskään toteuteta niin sanottuja päähän pistoksia. Tällä tarkoitetaan sitä, jos käyttäjällä tulee mieleen jokin ominaisuus, jota saatettaisiin tarvita tulevaisuudessa, tätä ei lähdetä toteuttamaan. XP:ssä panostetaan siihen, että toteutetaan ainoastaan ne ominaisuudet, joita nyt tarvitaan. Tätä prinsiippiä kutsutaan nimellä YAGNI. (Lindberg 2003, 44.)

Beck jakaa XP: ydinarvot neljään ja avainkäytännöt 12 eri kategoriaan. XP: ydinarvot jakautuvat seuraavasti:

- viestintä
- yksinkertaisuus
- palaute
- rohkeus (Beck 2000, 29.)

Näillä neljällä ydinarvolla mahdollistetaan työryhmän jäsenten välinen tehokas kommunikointi, yksinkertainen ohjelmisto, tehokas ja jatkuva palaute ja testien kirjoittaminen ennen ohjelmointia. (Beck 2000, 29–34.)

XP:n käytännöt Beck jakaa seuraavasti:

- The Planning Game (Projektin suunnittelu)
- Small releases (pieniä julkaisuja)
- Metaphor (mefora)
- Simple design (yksinkertainen suunnittelu)
- Testing (testaus)
- Refactoring (refaktorointi)
- Pair programming (pariohjelmointi)

- Collective ownership (yhteisomistajuus)
- Continuous integration (jatkuva integraatio)
- 40-hour week (40 tunnin työviikot)
- On-site customer (läsnä oleva asiakas)
- Coding standards (koodistandardit) (Beck 2000, 54.)

Kuvailen tässä ne käytännöt, joita ei kuvailta vielä aiemmin luvussa ketterä ohjelmistoprojekti (luku 3.5).

Projektin suunnittelu toteutetaan yhdessä asiakkaan ja ohjelmoijien kanssa. Asiakas luo käyttäjätarinat ja ohjelmoijat arvioivat tarpeet ja aikataulun tarinoiden toteuttamiseen. (Beck 2000, 55–56.) Jokainen **julkaisu toteutetaan niin lyhyinä** ja pieninä kuin mahdollista. Julkaisu sisältää ainoastaan vaatimusmäärittelyn tärkeimmät vaatimukset. Tällä mahdollistetaan yksinkertaisen järjestelmän nopea toteuttaminen ja uusia julkaisua voidaan näyttää päivittäin tai ainakin kuukausittain asiakkaalle. (Beck 2000, 56.) **Metaforien** avulla helpotetaan ymmärtämään järjestelmän termejä ihmisen näkökulmasta. Metaforat määritellään asiakkaan ja ohjelmoijan toimesta. (Beck 2000, 56–57.)

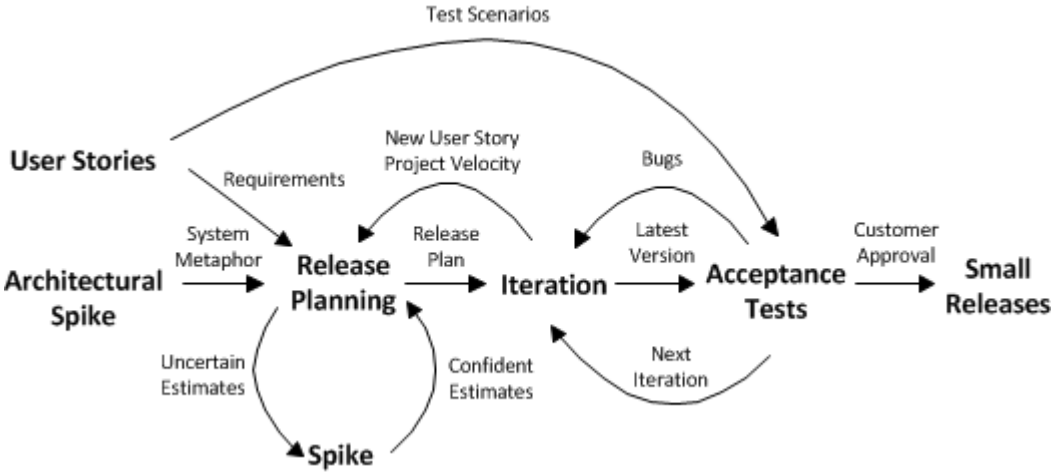
Yksinkertainen suunnittelu pyrkii poistamaan tarpeettoman kompleksisuuden ja ylimääräiset koodit ohjelmasta. Suunnitellaan toiminnallisuudet ainoastaan niihin määrittelyihin, mitä on määritelty. (Beck 2000, 57.) Jatkuva **testaus** toteutetaan yksikkö- ja funktionaalisella testauksella. Asiakas yleensä toteuttaa funktionaaliset testit. (Beck 2000, 57–58.) Maksimi **työmäärä viikossa** on 40 tuntia, mutta yhden ylityöviikon on mahdollista työskennellä. Toista ylityöviikkoa ei suvaita. (Beck 2000, 60.)

Läsnä oleva asiakas on tärkeässä roolissa ongelmiston onnistumisessa, joten käyttäjän on istuttava palavereissa yhdessä tiimin kanssa. Keskustelut kehittäjän ja asiakkaan välillä auttavat toteuttamaan sellaisen ohjelmiston, kuin asiakas itse haluaa. Jos kehittäjälle on kertynyt kysymyksiä, koskien ohjelmaa on vastauksien saaminen kasvokkain paljon tehokkaampaa kuin sähköpostiviestien vaihtelu. (Beck 2001, 60–61.)

Koodistandardit auttavat pitämään ohjelmointikielen helposti tulkittavana. Samaa koodia ei ole olemassa monessa eripaikassa ja ohjelmointityyli on samanlaista koko ryhmällä. Koodia ei kirjoiteta yhteen, vaan jokainen metodi erotellaan omalla lauseella ja nimetään mahdollisimman realistisella nimellä vastaamaan juuri sitä koodia, mitä on kirjoitettu. (Beck 2001, 61.)

Seuraavassa kuvassa (kuva 7) kuvaillaan XP:n prosessimalli, joka jakautuu viiteen ryhmään:

- arkkitehtuurinen erityistehtävä ja käyttäjätarinat (Architectural Spike, User Stories)
- julkaisun suunnittelu (Release Planning)
- iteraatio (Iteration)
- hyväksymistestaus (Acceptance Tests)
- pienet julkaisut (Small Releases)



Kuva 7. *XP prosessimalli (Extreme Programming 2013.)*

Xp:ssä käyttäjätarinat ja arkkitehtuurin erityistehtävien tulos, toimii isossa roolissa projektien suunnittelussa. Kun kehittäjä joutuu muokkaamaan arkkitehtuurista pohjaa, ohjelmointikoodia tai kehittämään jotain uutta, jotta saisi toteutettua kaikki käyttäjän vaatimukset, kutsutaan tätä arkkitehtuurin erityistehtäviksi. Jokaiseen käyttäjätarinan toteutukseen käytetään aikaa 1-3 viikkoa, riippuen käyttäjätarinan laajuudesta. (Extreme Programming 2013.)

Ohjelmiston julkaisua toteutetaan XP:ssä iteraatioiden avulla. Jokainen iteraatio käy läpi hyväksymistestausvaiheen ja iteraatioiden pituus vaihtelee yhdestä viikosta kolmeen viikkoon. Iteraatiot kuitenkin suositellaan pitämään lyhyinä, noin viikon pituisina. Julkaisun suunnitteluvaiheessa luodut prioriteetit luodaan ensin ja jos aikaa jää, tällöin loputkin käyttäjätarinat. Hyväksymistestausvaihe pitää sisällään käyttäjätarina vaatimukset. Tarinat testataan testeillä, jotka ovat kirjoitettu ennen ohjelmointia ja jokaisen hyväksymistestauksen jälkeen testataan uusi käyttäjätarina, niin kauan kuin kaikki käyttäjän vaatimukset ovat toteutettu. Käyttäjälle ohjelmisto siirtyy vasta sitten, kun lopullinenkin ohjelmisto on testattu ja tällöin käyttäjä päättää hyväksyykö ohjelmiston. (Extreme Programming 2013.)

3.8 Ohjelmistokehitys ja Lean

Tässä projektissa on pyritty käyttämään seuraavia Leanin perusperiaatteita vähentämään hukkatyötä ja täten toimimaan mahdollisimman tehokkaasti. Ketterämenetelmä mahdollistaa tämän myös, mutta projektissa halutaan nähdä myös, miten Lean-ajattelutapa sopii ohjelmistokehitykseen.

Leaniä on sovellettu lähinnä ainoastaan autotuotantoon, mutta nykypäivänä yhä useampi yritys, kuten ohjelmistoyritykset ovat pyrkineet kopioimaan Leanin työkaluja omaan organisaatioon, kuitenkaan onnistumatta siinä kovinkaan hyvin. Leanin kopioiminen ainoastaan yhteen osa-alueeseen organisaatiossa, ei juuri vaikuta yritykseen ja tämän vuoksi se ei ole vaikuttanut positiivisesti tuloksiin. Leanin ajattelutapa on tärkeää sisäistää ja ymmärtää, ennen kuin Leaniä voidaan käyttää omassa ohjelmistoyrityksessä tai ohjelmistokehityksessä. Lehtinen kertoo työssään seitsemän Leanin perusperiaatetta, joita voidaan soveltaa ohjelmistokehyksessä:

- poista hukka
- rakenna laatu ohjelman sisään
- luo tietoa
- lykkää sitoutumista
- toimita nopeasti
- kunnioita ihmisiä

- optimoi kokonaisuus (Lehtinen 2011, 33.)

Lean-ajattelutapa vaikuttaa melko läheltä aiemmin kertomaani ketterämenetelmää, mutta näitä kahta ei kuitenkaan sovi tulkita samana asiana, koska ketterät menetelmät ovat paljon käytännönläheisempiä ja Lean pikemminkin filosofia. Paljon samaa näissä kahdessa menetelmässä kuitenkin on, koska molemmat pyrkivät tuottamaan käyttäjälle mahdollisimman nopeasti toimivan version tuotteesta ja eliminoimaan turhan työn ja ylimääräisen dokumentoinnin. (Lehtinen 2011, 34.)

Miten Lean-ajattelutavalla saataisiin lisää tehokkuutta ohjelmistokehitykseen?

Lehtinen kuvailee Leanin seitsemän hukkaa taulukossa 2, jossa vasemmalla kuvaillaan tuotannon ja oikealla, samaa asiaa vastaavaa ohjelmistokehityksen hukkaa.

Seitsemän tuotannon hukkaa	**Seitsemän ohjelmistokehityksen hukkaa**
Ylituotanto	Ylimääräiset ominaisuudet
Odottaminen	Viiveet
Kuljetus	Tuotosten siirrot
Yliprosessointi	Uudelleenoppiminen
Varastointi	Osittain tehty työ
Liike	Tehtävien vaihdot
Virheet	Virheet

Taulukko 2. *Seitsemän hukkaa. (Lehtinen 2011, 39.)*

On olemassa paljon ohjelmistoja, jotka sisältävät aivan liikaa ominaisuuksia. Onhan se varmasti mukavaa kokeilla uusia kikkoja ja toiminnallisuuksia, mutta nämä valitettavasti kasvattavat liikaa ohjelmoijan työtä, ohjelman elinkaaren aikana. Viiveellä tarkoitetaan vähän samaa asiaa kuin ketterämenetelmän ”läsnä olevaa asiakas” viittausta. Viiveellä pyritään pitämään käyttäjän ja kehittäjän välinen kommunikointi viiveettömänä, täten kehittäjä saa käyttäjän milloin tahansa kiinni, kun kysyttävää syntyy. (Lehtinen 2011, 39–40.)

Tuotosten siirroilla tarkoitetaan dokumenttien ja kaiken muun tiedon siirtämistä kehittäjältä kehittäjälle. Näitä täytyy välttää, koska kaikkea tietoa ja taitoa ei vaan pystytä dokumentoimaan. Mieluiten voitaisiin käyttää kasvotusten keskustelua ja keskeneräisten töiden julkaisua tiimeissä. Uudelleen oppimisella tarkoitetaan ”ahaa” elämystä, kun huomataan jotain, mikä aiemmin olikin jo tiedossa, mutta oli päässyt kokonaan unohtumaan. Osittain tehty työ vastaa tuotannossa olevaa ylijäämää, joita voivat olla keskeneräiset laitteet jne. Tuotekehityksessä tällä tarkoitetaan koodia, joka on aloitettu, mutta ei ole saatettu loppuun. Nämä saattavat vaikuttaa negatiivisesti ohjelman valmistumiseen ja käyttöönottoon. (Lehtinen 2011, 40–41.)

Tehtävien vaihdoilla tarkoitetaan kehittäjän lähettämistä toiseen projektiin tai tehtäviin. Tämän voidaan verrata siihen, kun kehittäjän puhelin soi jatkuvasti 10 minuutin välein ja tämän vuoksi kehittäjä ei millään saavuta omaa keskittymistasoaan, koska keskittymisen saavuttaminen vaatii ainakin 15 minuuttia. Virheiden korjaaminen jo aikaisessa vaiheessa, on paljon parempi vaihtoehto, kuin korjattaisiin virhe vasta projektin loppupuolella. (Lehtinen 2011, 41–42.) Tähän voidaan hyvin käyttää aikaisemmin mainittua ketterän menetelmän testivetoista kehitystä.

4 SUUNNITTELU OHJELMISTOPROJEKTISSA

”Mitä monimutkaisemmasta ja laajemmasta kokonaisuudesta on kyse, sitä tärkeämpää on hyvä suunnittelu”. (Hovi ym. 2005, 20.)

4.1 Tietokannan rakenne

Tietokannan rakenne on hyvä miettiä jo ennen suunnittelua. Hovi ym. jakavat tietokannan rakenteen ominaisuudet seuraavasti:

- kattavuus
- selkeä ja ymmärrettävyys
- muutosjoustavuus
- yleiskäyttöisyys
- eheys
- ohjelmointi mukavuus
- suorituskyky eli tehokkuus. (Hovi ym. 2005, 21.)

Kattavuudella pyritään toteuttamaan tietokannan yhteydet ja kyselyissä tarvitsemat tiedot. Tietokannan rakenne on oltava yksinkertainen ja kyselyjen toteuttaminen täytyisi tehdä mahdollisimman helpoksi, jotta vältyttäisiin monimutkaisilta kyselyratkaisuilta. Muutosjoustavuudella tarkoitetaan sitä, jos tietokantaa halutaan laajentaa, on sen toteuttaminen onnistuttava ilman pahimpia ohjelmien muutoksia. (Hovi ym. 2005, 21.)

Tietokannan rakenne on hyvä rakentaa toimimaan erilaisissa ympäristöissä, ilman rakenteeseen kohdistuvia muutoksia. Eheydellä pyritään pitämään yllä tietokannan oikeellisuutta ja välttämään toistuvia tietoja. Tietokannassa olevien sarakkeiden selkeä nimeäminen ja selkeät tietorakenteet edesauttavat ohjelmointia. Tehokkuudella varmistetaan tapahtumien kulku kivuttomasti ja luontevasti. (Hovi ym. 2005, 21.)

4.2 Vaatimuksien määrittely UML:ää käyttäen

Oliomallinnuksen kuvausmenetelmä (UML) luotiin vuonna 1995 Boochin, Jacobsonin ja Rumbaughin toimesta, jossa yhdistettiin aiemmin käytetyt menetelmät Booch, OMT ja OOSE yhdeksi kokonaisuudeksi. Lisäpiirteitä otettiin myös muista menetelmistä vielä lisäksi, jotta saatiin kaikki tarvittava samaan menetelmään. (Hovi ym. 2005, 120.)

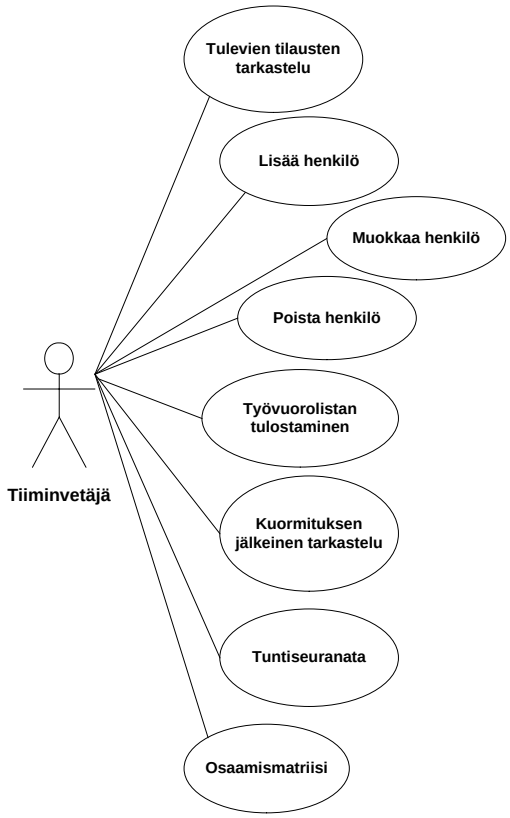
UML mahdollistaa lähestyä toteuttavaa järjestelmää tai ohjelmaa erilaisista näkökulmista. (Hovi ym. 2005, 120.) Käytän myös työssäni vaatimuksien määrittelyyn käyttötapaus-, luokka-, aktiviteetti-, ja sekvenssikaaviota. Kaavioita on myös muita vaihtoehtoja, mutta jätän niiden kuvaukset pois tässä luvussa. Tässä luvussa kuvaillaan yksi esimerkki jokaisesta kaaviosta ja loput kaaviot ovat sijoitettu liitteeseen 1.

Käyttötapauskaavio (use case diagram) kuvaillaan ohjelman käyttäjät, mitä toiminnallisuuksia ohjelmalta vaaditaan ja mitä tietoja käyttäjät tarvitsevat. (Hovi ym. 2005, 120; Haikala & Märijärvi 2004, 158.) Käyttötapauskaavion lisäksi käyttötapaukset kuvaillaan vielä teksteillä, jotta kuvauksen saisi mahdollisimman tarkasti kuvailtua. (Hovi ym. 2005, 120.) UML-standardin mukaiseen kuvaukseen kuuluvat vielä lisäksi käyttötapauksien välisiä suhteita, jotka kuvataan käyttötapauskaaviossa nimellä include ja extends. Include määrittelee käyttösuhteen tapauksien välille ja extends laajennussuhteen. Laajennussuhdetta ei suositella käytettäväksi, jos halutaan pitää käyttötapauskaavio yksinkertaisena ja selkeänä. (Haikala & Märijärvi 2004, 158–159.)

Haikala ja Märijärvi kuvailevat hyvän käyttötapauksen ominaisuudet seuraavasti:

- ymmärrettävyys
- kuvaa asiakasvaatimuksia
- testattavuus
- koko
- sopiva tarkkuus (Haikala & Märijärvi 2004, 159.)

Ymmärrettävyydellä osataan toteuttaa käyttötapaukset siten, kuten muutkin ymmärtävät tulkita niitä. Asiakasvaatimuksien kuvailu on myös tärkeää pitää selkeänä ja lyhyenä, esimerkiksi käyttötapaus kuvaus ”Käyttäjä luo tilauksen” on paljon parempi kuin ”Käyttäjän täytyy saada luotua ohjelmassa tilaus”. Käyttötapauksista on tärkeää muodostua yksi kokonaisuus, joita pystytään testaamaan käyttöliittymässä yksitellen tai peräkkäisinä testitapauksina. Hyvä käyttötapaus ei saa olla isompi kuin A4-arkki, joten kokoa kannattaa miettiä käyttötapausta luodessa. ”Sopivalla tarkkuudella” tarkoitetaan kaikkia yksityiskohtia käyttötapauksista, joita suositella luomaan. Käyttötapauksilla kuvataan ainoastaan ohjelman toteuttamisen tärkeimmät osat. (Haikala & Märijärvi 2004, 159.)



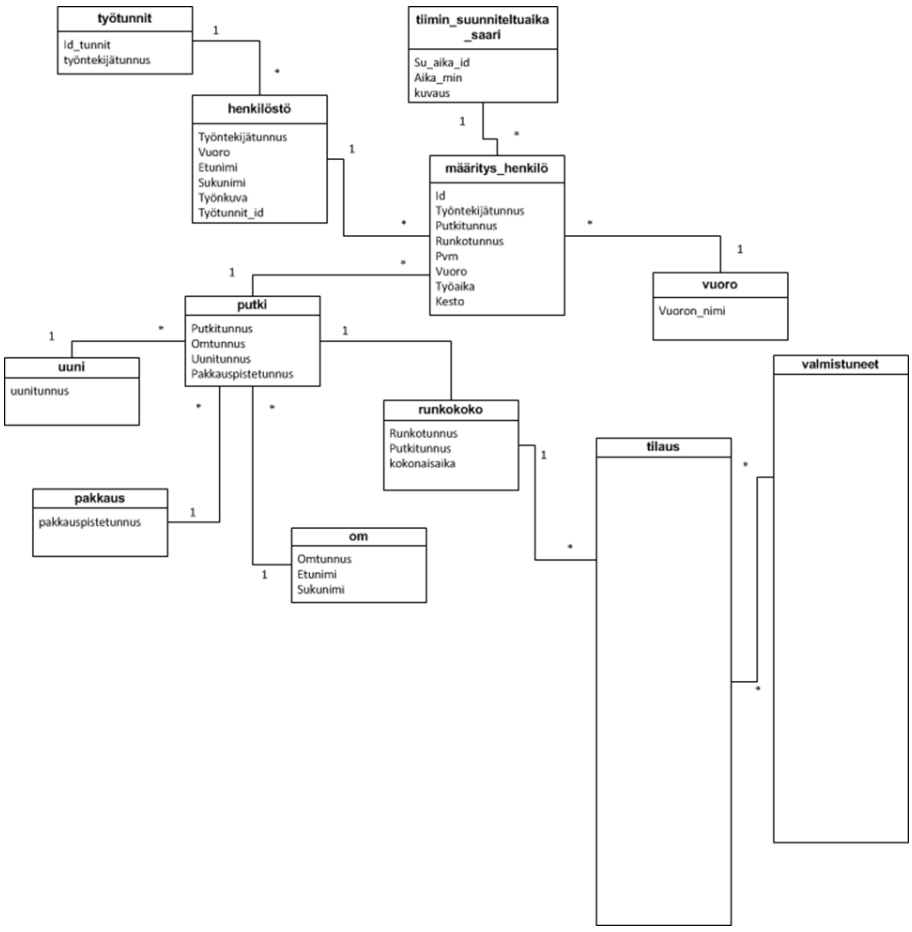
Kuva 8. *Käyttötapauskaavio, Tiiminvetäjä.*

Esimerkkikuvassa kuvataan hienokuormitusohjelman käyttötapauskaavio (kuva 8). Käyttötapauskaaviossa tiiminvetäjä toimii toimijana. Tiiminvetäjä pystyy tarkastelemaan tulevia tilauksia ja hienokuormittaan tuotantolinjansa. Hienokuormi-

tuksen jälkeen tiiminvetäjä pystyy tulostamaan työvuorolistan, tuntiseurannan ja osaamismatriisin. Hienokuormituksessa tiiminvetäjällä on mahdollista lisätä, poistaa ja muokata kuormitettuja henkilöitä. Käyttötapaukset on kuvailtu tarkemmin luvussa käyttötapaukset.

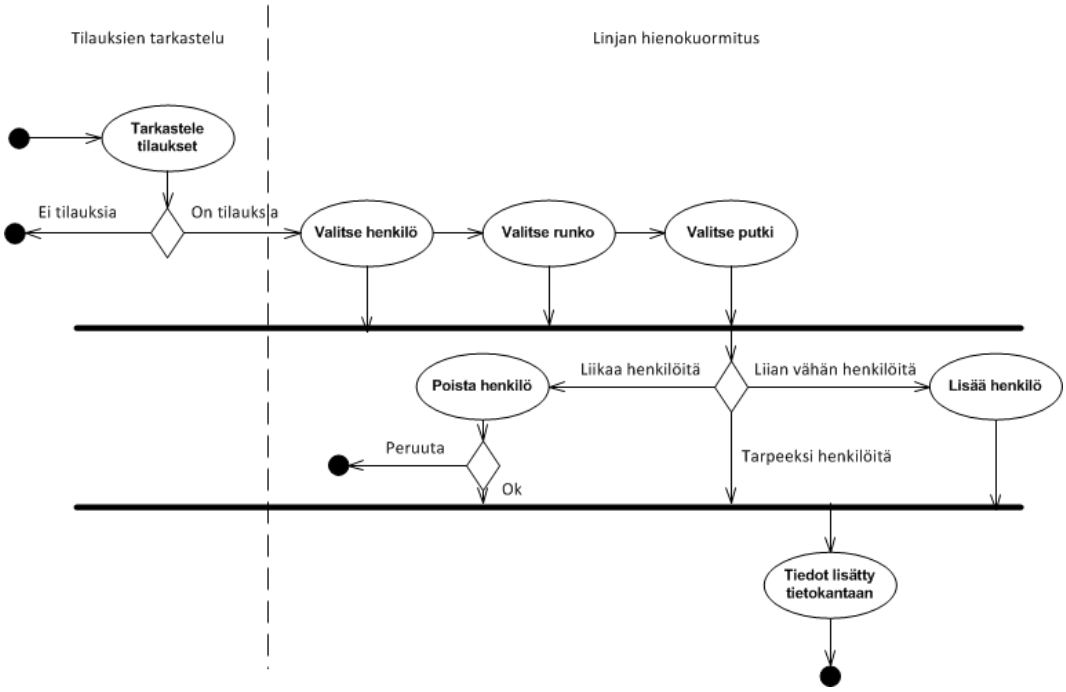
Luokkakaavio (class diagram) kuvaillaan luokka ja sen attribuutit, operaattorit ja luokassa olevien tietojen väliset yhteydet. Luokkakaavioita on mahdollista toteuttaa perustuen käsitemalliin, määrityksiin tai luokkien toteutukseen, viimeisin näistä on kaikista käytetyin tapa. Luokka kuvailee käsiteanalyysissä luotua oliota, joka voi olla asiakas, työntekijä jne. (Hovi ym. 2005, 121.)

Esimerkki mallissa (kuva 9) kuvaillaan hienokuormitusohjelman luokkakaavio, jossa on kaiken kaikkiaan 12 taulua. Kaksi taulua, joiden tiedot on peitetty, haetaan toisesta järjestelmästä ja loput taulut luotiin itse.



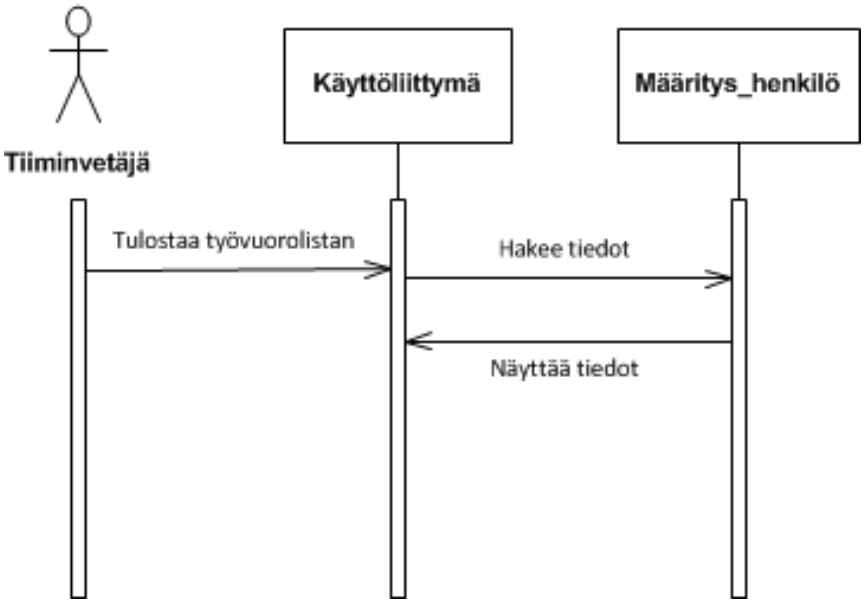
Kuva 9. *Luokkakaavio*

Kun halutaan kuvailla ohjelman toimintoja ja tehtäviä, käytetään niiden toteuttamiseen **aktiviteettikaaviota** (activity diagram). Toimintoja voivat olla esimerkiksi ”Lähetä tilaus” ja ”Tarkista tilausvahvistus”. Aktiviteettikaaviolla tarkastellaan käsitteiden välisiä yhteyksiä ja määritellään minkälaisia kenttiä, tietokantaan täytyisi luoda. (Hovi ym. 2005, 121.) Kuvassa 10 kuvaillaan hienokuormitusohjelman aktiviteettikaavio tilausten tarkastelusta ja linjan hienokuormituksesta.



Kuva 10. *Aktiviteettikaavio. Tilausten tarkastelu ja linjan hienokuormitus*

Sekvenssikaaviolla kuvataan tapahtumaketjujen eteneminen, johon voidaan myös lisätä toimintoja. (Haikala & Märijärvi 2004, 154–155.) Kun sekvenssikaaviossa on yhdistetty kaksi luokkaa kommunikoimaan, kertoo se luokkien välillä olevan assosiaatio ja vastaan ottavalla luokalla vastaava metodi. Nuolet, jotka suuntaavat oikealle sekvenssikaaviossa, luokitellaan ”kutsuiksi” ja paluunuolet palauttavat kutsutun arvon. (Haikala & Märijärvi 2004, 155.) Kuva 11 kuvailee hienokuormitusohjelman sekvenssikaavion, työvuorolistan tulostamisesta.



Kuva 11. *Sekvenssikaavio, työvuorolistan tulostaminen.*

4.3 Käytettävyys

Käyttäjäystävällisellä ohjelmalla tarkoitetaan ohjelmistoa, joka perustuu seuraaviin asioihin:

- ymmärrettävä
- vaivaton
- kattava
- esteettisesti miellyttävä (Wiio 2004, 29.)

Ymmärrettävän ohjelmiston käyttäjä itse osaa päätellä, kuinka ohjelmisto toimii ja täten pystyy helposti käyttämään ohjelmiston ominaisuuksia. Kun käyttäjä osaa itse esittää kysymyksiä, mitä ohjelmistolla voidaan tehdä, kertoo se jo hyvin ymmärrettävästä ohjelmistosta, koska käyttäjä yleensä löytää itse vastauksen. (Wiio 2004, 29.)

Vaivattomalla ohjelmistolla tarkoitetaan käyttäjän kannalta katsottuna yksinkertaista ohjelmistoa. Ohjelmiston ominaisuus ei saisi viedä liikaa aikaa käyttäjältä, jotta käyttäjä säästyisi taloudellisilta tappioilta. Hyvin ymmärrettävä ohjelmisto, voi myös olla vaivalloinen. (Wiio 2004, 30.)

Ohjelmisto, joka kattaa kaikki käyttäjän tarpeet ja toimii siten, miten käyttäjä on ohjelmiston toivonutkin toimivan, niin tätä voidaan kutsua **kattavaksi** ohjelmistoksi. Esim. ohjelmien tallennettaessa videonauhuriin, jos halutaan nauhoittaa monta ohjelmaa peräkkäin, videonauhuri ei osaa ilmoittaa tietoa mahtuvatko kaikki ohjelmat kyseiseen videokasettiin. Tämän vuoksi täytyy laskea kaikki ohjelmat yhteen ja päättää mahtuvatko kaikki ohjelmat samalle videokasetille. (Wiio 2004, 31.)

Esteettisesti suunnitellut ohjelmistot, kuten yrityksen verkkosivut viestittävät käyttäjälle yrityksen yhtenäistä imagoa. Huonosti suunnittelut verkkosivut taas kuvastavat käyttäjälle laadun uupumista ja täten huonosti suunnitellut verkkosivut eivät houkuttele käyttäjää jäämään sivustolle pitemmäksi aikaa. (Wiio 2004, 31.)

Nykypäivänä voidaan jo kuvitella, käyttäjän osaavan toimia uuden ohjelmiston kanssa melko hyvin eli käyttäjällä on tietokoneen käytön yleistaidot. Yleistaitoihin luokitellaan selainikkunoiden, komentopainikkeiden ja valikoiden ymmärtäminen ja käyttö. Näiden taitojen avulla käyttäjän pitäisi osata käyttää täysin uutta ohjelmistoa ensimmäisen kerran, joten käyttöliittymän tehtävä on auttaa käyttäjää mahdollisimman hyvin löytämään haluamansa lopputuloksen vaivattomasti ja yksinkertaisesti. (Wiio 2004, 131.)

Ajatellaan, että käyttäjällä on olemassa jonkinlaiset navigointitaidot eli hän osaa käyttää valikoita ja työkalupalkkeja. Valikot, painikkeet jne. ovat tavallaan tienviittoja käyttäjälle, jotta käyttäjä löytäisi haluamansa tiedon ohjelmistosta. Käyttöliittymän suunnittelussa on erittäin tärkeää ottaa huomioon käyttäjän näkökulma. (Wiio 2004, 153–154.). Ohjelmisto jossa käyttäjän täytyisi jatkuvasti muistella miten sinne haluttuun tietoon nyt päästiinkään, en usko käyttäjän pitävän ohjelmistoa lainkaan miellyttävänä.

Mitä paremmin käyttäjät ymmärtävät ohjelmiston logiikan, eli painikkeet ja valikot, sitä vähemmän heidän tarvitsee käyttää muistiaan, jonkun toiminnon toteuttamiseen. Käyttäjälle ymmärrettävän ohjelmiston logiikkaa voi hyödyntää käytettävyystesteillä. Käytettävyystesteillä pyritään testaamaan miten käyttäjä ymmärtää ja tulkitsee ohjelmiston logiikkaa. . (Wiio 2004, 153–154.)

On tärkeää nimetä valikot ja painikkeet käyttäjän näkökulmasta ymmärrettävällä tavalla, jotta käyttäjä tietää mitä painikkeen alta löytyy. (Wiio 2004, 156.)

Käyttöliittymän testauksessa kootaan yksi osa tai osia kerrallaan. Tällä mahdollistetaan testivetoinen kehitys ja nähdään miten suunnitellut toiminnot toimivat käytännössä. Testaukseen on hyvä ottaa muutama ulkopuolinen henkilö mukaan tai ainakin käyttäjä, jolle ohjelmisto on tulossa käyttöön. Tällä mahdollistetaan jatkuvan palautteen saaminen ja pystytään kehittämään käyttöliittymän toimintoja jo matkan varrella. *(Wiio 2004, 217–218.)*

5 VAATIMUKSIEN MÄÄRITTELY

Tässä kappaleessa kuvaillaan aluksi työn nykytila ja tuleva tila. Seuraavassa luvussa avataan tietokannan suunnittelua, joka pitää sisällään vaatimusmäärittelyn, käyttäjätarinat ja käyttötapaukset. UML-kaaviot kuvailtiin jo tarkemmin luvussa 4.2. Työn toteutus on pääsääntöisesti edennyt teoriaosuudessa mainitun ketterän ohjelmistoprojektin ja Lean-ajattelun tavalla, vaikka työn suunniteluun otettiinkin mukaan käyttötapaukset ja UML-kaaviot. Jokainen käyttötapaus on toteutettu ja testattu yksi kerrallaan, jotta käyttäjälle on voitu alusta asti näyttää toimivaa ohjelmistoa ja tämän ansiosta on saatu jatkuvaa palautetta käyttäjältä. Käyttötapauksien toteuttamisesta yksi kerrallaan kuvaillaan tarkemmin luvussa käyttöliittymän toteutus. Työnhukkaa on pyritty vähentämään toteuttamalla ainoastaan ne tärkeät ja juuri nyt tarvittavat ominaisuudet, joita tiiminvetäjä on vaatinut. Kaikki ylimääräiset ominaisuudet on jätetty toteuttamatta.

Ohjelman rakenteeseen kuuluvat tietokanta ja käyttöliittymä. Ohjelma myös ottaa yhteyden toiseen järjestelmään, josta haetaan osa tiedoista.

5.1 Nykytila

Vaconilla ei ole ollut aiempaa ohjelmaa, jolla saataisiin aidon hienokuormituksen positiivista dataa. Kuormitus on toteutettu tällä hetkellä ainoastaan karkealla tasolla.

Tiiminvetäjä on joutunut etsimään monesta eri järjestelmästä tietoja, jotta tiiminvetäjä tietäisi kuinka kuormittaa tuotantolinjansa. Tämän hetkiset työkalut kuormittamiseen ovat olleet pääsääntöisesti Excel ja Post-it-laput. Tahtotilana on ohjata hienokuormitusta tarkemmin ja täten nostaa tuotannon tehokkuutta. Hienokuormitusta täytyisi myös pystyä seuraamaan reaaliajassa. Tällä tarkoitetaan ohjelman näyttävän tiiminvetäjälle sen, paljonko kuormitetulla henkilöstömäärällä täytyisi tulla laitteita ja paljonko valmistuneita laitteita tuli. Tämä auttaisi ongelmien seuraamisessa ja havainnollistamisessa.

5.2 Tuleva tila

Opinnäytetyön tavoitteena on luoda tuotannonohjausohjelma tuotantolinjojen kuormittamiseen, jolla mahdollistetaan tiiminvetäjien tehokkaampi työskentely ja helpotetaan tiiminvetäjien työtä. Ohjelman pääsivulle (päivänäkymä) on luotava kuormitusnäkymä ja reaalinäkymä. Kuormitusnäkymään on saatava näkymään yhdelle riville laitetunnus, laitemäärä, henkilötarve, määritelty henkilöitä yhteensä, aamuvuoro ja iltavuoro. Määritellyt henkilöt on saatava toimimaan siten, kun tiiminvetäjä kuormittaa henkilöitä aamuvuoroon ja iltavuoroon, ohjelman täytyisi laskea summa siihen kenttään ja näyttää punasta, jos henkilöiden määrä ei vastaa henkilötarpeeseen ja vihreätä, jos määrä vastaa.

Kuormituksen jälkeen täytyisi saada näkymä, joka näyttäisi kuormitustilanteen, eli kuinka paljon laitteita ja henkilötarvetta olisi vielä jäljellä. Tämän avulla tiiminvetäjä näkisi tarvitseeko hänen vielä kuormittaa lisää henkilöitä tai onko henkilöitä kuormitettu liikaa. Kun henkilöt ovat kuormitettu, täytyisi myös luoda näkymä tuntiseurannasta ja osaamismatriisista. Tuntiseurantanäkymässä nähtäisiin, paljonko kuormitetulla henkilöstömäärällä täytyisi valmistua laitteita ja paljonko laitteita on valmistunut. Osaamismatriisi näyttäisi tiiminvetäjälle ne henkilöt, jotka ovat valmistaneet mitäkin runkokokoa ja täten tiiminvetäjän on helppoa nähdä, mihin putkelle henkilön kuormittaa.

5.3 Vaatimusmäärittely

Tässä luvussa kuvaillaan työn vaatimukset ja tärkeimmät pääperiaatteet ja toiminnot, joita ohjelmalta vaaditaan.

Vaatimusmäärittelyt jaoteltiin seuraavanlaisiin ryhmiin:

- yleiskuvaus ohjelmasta
- toiminnalliset vaatimukset
- ei-toiminnalliset vaatimukset
- tietokanta
- käyttöliittymä.

Yleiskuvaus: Ohjelman pääkäyttäjänä toimii tuotantolinjan tiiminvetäjä, joka ohjaa linjan henkilöitä ja koko linjan toimintaa. Ohjelman käyttötarkoitus on mahdollistaa tiiminvetäjälle tehokkaampi ohjelma, jolla tiiminvetäjä pystyy hienokuormittamaan tuotantolinjansa, jonka tuloksena syntyy tuotantolinjan työvuorolista. Tiiminvetäjän on myös nähtävä tuntiseuranta ja osaamismatriisi. Käyttäjäympäristöön kuuluu ainoastaan tiiminvetäjä tässä ohjelmaversiossa ja toteutus toteutetaan Accessilla. Ohjelma käyttää kahden muun Vaconin järjestelmän tietoja hyväkseen, joita ovat tulevat tilaukset ja valmistuneet laitteet. Muut tarvittavat tiedot luodaan itse tietokantaan.

Toiminnallisia vaatimuksia ovat henkilön lisääminen, poisto ja muokkaus tuotantolinjaan. Tiiminvetäjän täytyy päästä valitsemaan mihin putkelle henkilö lisätään ja mitä laitetta henkilö laitetaan valmistamaan. Näkymät pitäisivät sisällään tulevat tilaukset ja henkilötarpeen sekä reaaliaikaisen näkymän hienokuormituksesta. Tuntiseurantanäkymä näyttäisi valmistuneet laitteet päivän aikana ja osaamismatriisinäkymä henkilöiden osaamisen laitekohtaisesti.

Ongelmakohtia nykyisessä toiminnassa on ollut, ettei tiiminvetäjällä ole ollut lainkaan selkeää ohjelmaa tuotantolinjansa hienokuormittamiseen. Tiiminvetäjä on käyttänyt Exceliä työvuorolistojen tekemiseen ja yrittänyt hakea tietoa Vaconin monista järjestelmistä, jotta saisi tiedon, paljonko henkilöitä täytyisi suunnilleen kuormittaa. Keskustelut tiiminvetäjien kanssa antoi sellaisen tuloksen, että tiiminvetäjällä kuluu liian kauan tiedon etsimiseen ja hakemiseen.

Ohjelman aloitusvaiheessa rajoituksia ei luotu, vaan pyritään toteuttamaan kaikki edellä mainitut toiminnot, jos vain aika riittää. Jos aika ei tule riittämään tämän työn puitteissa, toteutetaan jäljelle jääneet toiminnot jatkokehityksessä. Käyttötapaukset tulevat esiin myöhemmin, käyttötapaukset luvussa.

Ei-toiminnallisissa vaatimuksista tärkein on ohjelman käytettävyys. Tiiminvetäjä painotti ohjelman käyttöliittymää, joka olisi hyvä olla helppokäyttöinen, selkeästi luettavissa ja jos vain mahdollista, samantapaiset toiminnot samassa näkymässä, kuitenkin siten, ettei selkeys kärsi. Tiiminvetäjä myös piti tärkeänä selkeää navigointia, eri näkymien välillä.

Kun ohjelma etenee pilottikäyttöön tiiminvetäjälle, tietokanta ja ohjelmointikoodit salataan salasanalla, jottei käyttäjä vahingossa pääsee muuttamaan mitään, mikä vahingoittaisi tai sotkisi ohjelman toimivuutta.

Tietokannan ja käyttöliittymän suunniteluun ja kuvaukseen perehdytään tarkemmin kappaleissa kuusi ja seitsemän.

5.3.1 Käyttäjätarinat

Kuvassa 12 näkyy kuusi erilaista käyttäjätarinaa tiiminvetäjälle. Nämä kaikki käyttäjätarinat priorisoitiin tärkeiksi, koska ne kaikki ovat tärkeässä roolissa tulevassa ohjelmassa ja tiiminvetäjä kuvaili ne erittäin oleelliseksi osaksi ohjelmaa. Ohjelmaan tulee varmasti ohjelmiston edetessä muitakin toimintoja, mutta nämä kuusi ovat ne tärkeimmät, joita lähdetään toteuttamaan. Testauksessa jokainen käyttäjätarina testataan yksitellen ja toteutettu osa näytetään käyttäjälle, joka antaa palautetta välittömästi. Palautteen perusteella käyttäjätarinoita voi tulla lisää tai ne muuttuvat kokonaan.

Tiiminvetäjä — 1 = tärkeä, 2 = ei niin tärkeä, 3 = toteutetaan, jos ehditään

ID	Käyttäjätarina	Prioriteetti
K01	Tiiminvetäjänä haluan tarkastella tulevia tilauksia, jotta tiedän henkilömäärän, kuinka paljon minun täytyy kuormittaa henkilöitä tuotantolinjalle.	1
K02	Minun täytyy saada kuormitettua tietty henkilö, tietylle putkelle ja tietylle rungolle ja pystyä muokkaamaan ja poistamaan tarvittaessa henkilöitä. Jotta saan kuormitettua juuri oikean määrän henkilöitä.	1
K03	Kuormituksen jälkeen minun täytyy saada tulostettua työvuorolista, jonka voin näyttää henkilöille, kun he saapuva töihin.	1
K04	Minun täytyy myös nähdä kuormitus reaaliajassa, jotta näen paljonko laitteita kuormitetulla henkilömäärällä täytyisi tulla.	1
K05	Tiiminvetäjänä minun täytyy nähdä tuntiseuranta, jotta tiedän kuinka paljon laitteita tulee tunnissa ja sen paljonko laitteita olisi pitänyt tulla. Tämä auttaa huomaamaan ongelmat, jos tavoitteisiin ei ole päästy.	1
K06	Tiiminvetäjänä en voi muistaa kuka henkilö on tehdyt mitäkin laitetta, joten haluan tarkastella osaamismatriisia. Osaamismatriisi kertoo tiedon, kenet henkilön voin laittaa valmistamaan mitäkin runkoa.	1

Kuva 12. *Käyttäjätarinat.*

5.3.2 Käyttötapaukset

Käyttötapaus 1.

Nimi: Tulevien tilausten tarkastelu

Osallistujat: Tiiminvetäjä

Esiehdot: Tiiminvetäjä on joko, päivänäkymässä tai viikkonäkymässä

Kuvaus: Tiiminvetäjä valitsee päivänäkymän tai viikkonäkymän. Päivänäkymässä näkyviin tulevat tilaukset, jotka on tehtävä tänään ja viikkonäkymässä tilaukset (+7 vrk). Päivänäkymässä käyttöliittymä näyttää myös henkilötarpeen, jotta tiiminvetäjä tietää, paljonko henkilöitä täytyy kuormittaa kyseisiä tilauksia kohden.

Poikkeukset: Jos tilauksia ei ole kummallakaan näkymällä, ovat kentät tyhjiä.

Lopputulos: Tiiminvetäjällä on tieto tulevista tilauksista ja henkilötarpeesta. Tiiminvetäjä on täten valmis toteuttamaan tuotantolinjan kuormituksen.

Käyttötapaus 2.

Nimi: Linjan hienokuormitus: Henkilön lisäys

Osallistujat: Tiiminvetäjä

Esiehdot: Tiiminvetäjän on oltava päivänäkymässä.

Kuvaus: Tiiminvetäjällä on mahdollista kuormittaa henkilöt aamuvuoroon tai iltavuoroon.

Poikkeukset: Jos tilauksia ei ole kyseiselle päivälle, henkilöitä ei täten pysty kuormittamaan.

Lopputulos: Tiiminvetäjän hienokuormitus rivi tallentuu tietokantaan ja tulee näkyviin samaan käyttöliittymään, missä hienokuormitus tapahtui.

Käyttötapaus 3.

Nimi: Linjan hienokuormitus: Henkilön muokkaus

Osallistujat: Tiiminvetäjä

Esiehdot: Tiiminvetäjän on oltava päivänäkymässä.

Kuvaus: Tiiminvetäjällä on mahdollista muokata kuormitettuja henkilöitä aamu-, ja iltavuorossa.

Poikkeukset: Jos henkilöitä ei ole lisätty kyseiselle päivälle, ei ole henkilöitä mitä muokata.

Lopputulos: Henkilö on muokattu ja tiedot päivittyvät tietokantaan.

Käyttötapaus 4.

Nimi: Linjan hienokuormitus: Henkilön poistaminen

Osallistujat: Tiiminvetäjä

Esiehdot: Tiiminvetäjän on oltava päivänäkymässä.

Kuvaus: Tiiminvetäjällä on mahdollista poistaa kuormitettuja henkilöitä aamu-, ja iltavuorossa.

Poikkeukset: Jos henkilöitä ei ole lisätty kyseiselle päivälle, ei ole henkilöitä mitä poistaa.

Lopputulos: Henkilö on poistettu ja tiedot päivittyvät tietokantaan.

Käyttötapaus 5.

Nimi: Työvuorolistan tulostaminen

Osallistujat: Tiiminvetäjä

Esiehdot: Tiiminvetäjän on oltava päivänäkymässä.

Kuvaus: Tiiminvetäjä tulostaa työvuorolistan henkilöistä, joita tiiminvetäjä on kuormittanut kyseiselle päivälle. Täten henkilöt tietävät mihin putkelle heidän täytyy mennä töihin ja mitä runkoa täytyy valmistaa.

Poikkeukset: Hienokuormitus on oltava toteutteluna.

Lopputulos: Tiiminvetäjä on tulostanut työvuorolistan.

Käyttötapaus 6.

Nimi: Kuormituksen jälkeinen tarkastelu

Osallistujat: Tiiminvetäjä

Esiehdot: Tiiminvetäjän on oltava päivänäkymässä ja hienokuormitus on täytynyt toteuttaa.

Kuvaus: Tiiminvetäjä pystyy seuraamaan riittääkö kyseinen henkilömäärä toteuttamaan kaikki tilaukset vai täytyykö henkilöitä lisätä lisää tai poistaa, jos henkilöitä on kuormitettu liikaa.

Poikkeukset: Jos hienokuormitusta ei ole toteutettu, silloin näkymässä ei näy mitään.

Lopputulos: Tiiminvetäjä on kuormittanut tuotantolinjansa oikein ja oikealla henkilömäärällä. Samalla tiiminvetäjä näkee, paljonko kyseisellä henkilömäärällä täytyy tulla laitteita, eli päivän tavoite.

Käyttötapaus 7.

Nimi: Tuntiseuranta

Osallistujat: Tiiminvetäjä

Esiehdot: Tiiminvetäjän on oltava prosessiohjausnäkymässä.

Kuvaus: Tiiminvetäjä näkee kuinka monta tilausta on valmistunut jokaisena tuntina ja tavoitteen, paljonko kuormitetulla henkilömäärällä olisi pitänyt tulla laitteita.

Laitteet ovat jaoteltu runkokoon mukaan ja näkyvät ainoastaan tämän päivän mukaan.

Poikkeukset: Jos valmistuneita laitteita ei ole, näkymä on tyhjä.

Lopputulos: Tiiminvetäjä näkee onko tavoitteisiin päästy.

Käyttötapaus 8.

Nimi: Osaamismatriisi

Osallistujat: Tiiminvetäjä

Esiehdot: Tiiminvetäjän on oltava osaamismatriisinäkymässä

Kuvaus: Tiiminvetäjä pystyy seuraamaan henkilöiden osaamista eli tiiminvetäjä näkee ketkä henkilöt osaavat tehdä mitäkin runkoa.

Poikkeukset: Jos henkilö ei ole tehnyt vielä yhtään laitetta, ei henkilöä tällöin näy listalla.

Lopputulos: Tiiminvetäjä tietää ketkä henkilöt tarvitsevat vaihtelua runkoihin ja sen, kuka osaa valmistaa mitäkin laitetta. Tämä helpottaa tiiminvetäjän työskentelyä siinä kohdassa, kun hänen täytyy valita henkilölle runkokoko.

6 KÄYTTÖLIITTYMÄN SUUNNITTELU

Käyttöliittymän suunnittelu toteutettiin Accessilla jokainen käyttötapaus kerrallaan. Tässä kappaleessa kuvaillaan käyttöliittymän eteneminen ketterää ohjelmistoprojektia ja protoilua käyttäen. Erillistä kirjautumista ei toteuteta vaan tiiminvetäjälle luodaan oikeudet kyseiseen kansioon, jossa ohjelma sijaitsee. Seuraavissa luvuissa kuvaillaan käyttöliittymän lomakkeet käyttötapauksittain.

Käyttöliittymän toiminnallisuudet voivat vaikuttaa seuraavissa versiossa jo valmiilta, mutta toiminnallisuutta ei ole vielä hiottu haluamalle tasolle. Jokaisesta lomakkeesta ei ole vielä tässä vaiheessa valmista versiota, vaan ainoastaan ensimmäiset protoversiot. Valmiit versiot kuvaillaan tarkemmin kappaleessa käyttöliittymän toteutus. Seuraavissa protoversioissa, ei vielä kaikissa lomakkeissa käytetä ”oikeaa” tietoa, vaan tiedot on luotu itse.

Suunnittelun eteneminen olisi varmasti helpompaa sisäistää kuvien avulla, mutta toimeksiantajan toimesta, kuvat hienokuormitustyökalusta eivät ole sallittuja tässä työssä, koska ohjelma kuvastaa Vaconin toimintamallia kuormittaa tuotantolinjoja.

Suunnittelun eteneminen toteutettiin seuraavanlaisessa järjestyksessä:

- päivänäkymä
- viikkonäkymä
- tuntiseuranta
- osaamismatriisi.

Päivänäkymästä toteutettiin useita versioita, koska se on ns. hienokuormitusohjelman pääsivu, jossa tiiminvetäjä toteuttaa kuormituksen. Päivänäkymä sisältää monta erilaista käyttötapausta, joita ovat tilaukset, henkilön lisääminen, poistaminen ja muokkaaminen aamu-, ja iltavuoroon, kuormituksen jälkeinen tarkastelu ja työvuorolistan tulostaminen. Päivänäkymää testattiin jokaisen version valmistuttua ja esiteltiin tiiminvetäjälle. Saatujen palautteiden perusteella, saatiin hiottua näkymä selkeämmäksi ja käyttäjäystävällisemmäksi. Palautteiden perusteella nä-

kymään tuli myös lisäyksiä, jotka kuvaillaan kappaleessa Testaus: Uudet vaatimukset.

Päivänäkymästä kuvaillaan kaksi versiota, jotta huomataan miten hienokuormitusohjelman jatkuva testaaminen vaikutti käyttöliittymän suunnitteluun. Versioita toteutettiin enemmänkin, mutta eiköhän oleellinen tule selväksi näissä kahdessa versiossa.

Päivänäkymän *ensimmäisessä versiossa* ei ole lainkaan vielä kuormituksen jälkeistä näkymää, vaan ainoastaan näkymät tilauksista, tarvittavasta henkilötarpeesta ja henkilöiden kuormittaminen aamu-, ja iltavuoroon. Näitä versioita testattiin ja testauksesta saatujen palautteiden perusteella päivänäkymästä luotiin toinen versio.

Hienokuormitusohjelmassa näkyy vasemmalla puolella kyseisen päivät tilaukset ja oikealla henkilötarve. Kummassakin näkymässä on kategorioitu määrät runkokoon mukaan. Alhaalla vasemmalla ovat henkilöiden määritysnäkymät, jotka ovat aamuvuoro ja iltavuoro. Tässä ensimmäisessä versiossa ei ole vielä lainkaan kuormituksen jälkeistä näkymää, vaan ainoastaan pyritty hakemaan tämän päivän tilaukset ja henkilötarve.

Päivänäkymän *toisessa versiossa*, alkaa vähitellen hahmottua kaikki päivänäkymään määritetyt käyttötapaukset ja näkymäkin on muuttunut selkeämmäksi, edelliseen versioon verrattuna. Kun tiiminvetäjä lisää henkilöitä aamu-, tai iltavuoroon, lisää ohjelma henkilöt kuormituksen jälkeiseen näkymään, josta näkyy kuinka paljon tiiminvetäjän täytyy vielä henkilöitä lisätä. Kuormituksen jälkeinen näkymä, myös näyttää tässä versiossa laitteiden jäljellä olevan määrä, kuinka paljon on vuoroa jäljellä (min) ja kuinka paljon laitteita tulee päivän aikana, kyseisellä määrällä henkilöitä.

Tilaukset-näkymä, joka on toisessa versiossa nimellä tuotantolinjan kuormitus, näyttää kyseisen päivän tilaukset, tilauksiin tarvittavan henkilömäärän, milloin tilaukset täytyisi olla tehtynä ja kuinka paljon henkilöitä on sillä hetkellä kuormitettu. Jos määritelty-kenttä näyttää vihreää, on henkilöitä määritelty tarpeeksi ja

jos punaista, henkilöitä täytyy vielä lisätä. Molemmista versioista kuitenkin puuttuu vielä kokonaan työvuorolistan tulostaminen. Työvuorolistasta oli kyllä luotu raportti, mutta painiketta ei ole vielä tässä vaiheessa luotu.

Viikkonäkymässä näytetään koko viikon tilaukset ja henkilötarve järjestettynä jokaiselle päivälle tuotteittain. Viikkonäkymästäkin versioita luotiin muutamaan otteeseen, jotta päästäisiin kohti haluttua lopputulosta. Aluksi kuvaillaan *ensimmäinen versio* viikkonäkymästä ja sen jälkeen viikkonäkymä pysyikin melko samanlaisena, aivan lopulliseen versioon asti.

Viikkonäkymä sisältää ensimmäisessä versiossa tilaukset, henkilötarpeen, ja henkilöiden kuormittamisen. Tähän näkymään on myös lisätty kuinka paljon viikossa on laitteita yhteensä ja ne vähentyivät aina sitä mukaan, kun henkilöitä kuormitettiin linjalle.

Viikkonäkymän *toisessa versiossa* poistettiin kokonaan henkilöiden kuormitus, koska testauksien jälkeen nähtiin parhaana toteuttaa henkilöiden kuormitus ainoastaan päivänäkymässä. Viikkonäkymän tarkoitus on pelkästään näyttää tiiminvetäjälle viikon tilanne tilauksista ja henkilötarpeesta.

Tuntiseurannan tarkoitus on näyttää tiiminvetäjälle tavoite laitemääristä ja kuinka paljon laitteita on tullut tuntitasolla. Tuntiseuranta näkymässä näkyy, kuinka paljon laitteita on tullut jokaisen tunnin kohdalla ja kuinka paljon aiemmin kuormitetulla henkilömäärällä olisi pitänyt tulla laitteita tunnissa. Tuntiseuranta näyttää tunnit aamuvuorosta iltavuoroon.

Tässä versiossa on vielä liikaa kenttiä näkyvillä, joiden ei lopullisessa versiossa enää pidä näkyä. Esimerkiksi tunti ja valmistunut sarake näkyvät tässä versiossa sen takia, jotta nähdään toimiiko ohjelma oikealla tavalla. Jos tunti näyttää 8 ja vasemmalla oleva aikajana näyttää samaa, niin voidaan päätellä ohjelman toimivan oikein. Tuntiseurannan on tarkoitus näyttää ainoastaan tämän päivän valmistuneet laitteet ja siksi ”valmistunutpvm” sarake on olemassa, jotta nähdään ilmestyykö tuntiseurantaan muiden päivän valmistuneita laitteita. Tämä auttaa havaitsemaan helposti viat ohjelmassa.

Tuntiseurannan ensimmäinen versio osui sen verran lähelle oikeaa, että siihen ei muutoksia toteutettu, ainakaan vielä suunnittelun osalta.

Osaamismatriisin tarkoitus on näyttää tiiminvetäjälle, mitä kukin työntekijä osaa tehdä eli mitä runkoja he ovat olleet tekemässä ja millä putkella. Osaamismatriisin avulla tiiminvetäjän on helppo kuormittaa tuotantolinjansa, koska hän näkee osaamismatriisista ne henkilöt, mitä laitetta kukin henkilö on valmistanut.

Osaamismatriisi laskee tässä ensimmäisessä versiossa määrän jokaiselle rungolle ja järjestää henkilöt määrän mukaan, suurimmasta pienimpään.

7 TIETOKANTA

Accessilla luotu käyttöliittymä hakee ja vie tiedot tietokantaan. Accessilla toteutettu tietokanta sisältää 12 taulua, joista kaksi taulua on haettu toisista järjestelmistä. Tietokannan relaatio varmasti kuvailisi yhteydet ja taulut tarkemmin, mutta niitä ei saatu julkistaa toimeksiantajan pyynnöstä.

Tilaukset ja valmistuneet taulut toteutetaan kyselyllä toisessa järjestelmässä. Projektin alussa haasteita toi tiedon rajaaminen, eli mitä tietoa tarvitaan toisista järjestelmistä, jotta ohjelman saisi toimimaan halutulla tavalla. Päätettiin ottaa kaikki tiedot tilauksista ja valmistuneista laitteista, jotta kaikki tieto olisi varmasti käytettävissä. Kun kyselyt olivat valmiita toisissa järjestelmissä, linkitettiin ne suoraan Accessiin. Accessissa hain tarvitsemat tiedot kyseisistä tauluista, kyselyjen avulla.

8 OHJELMAN TOTEUTUS

8.1 Tietokanta

Tietokanta toteutettiin Accessilla ja toteutuksen kanssa ei tullut vastaan ongelmia. Taulujen luominen oli aluksi haastavaa, koska ei vielä tiedetty mitä kaikkea tietoa täytyisi luoda, jotta ohjelma saisi kaikki tiedot toteutuakseen. Tietokannasta luotiinkin projektin alussa monta erilaista versiota ja näissä versioista tieto luotiin itse, eikä haettu muista järjestelmistä.

Kun tietokanta saatiin luotua, tämän jälkeen Accessilla otettiin yhteys toiseen järjestelmään, josta saatiin lopulliset tarvittavat tiedot ohjelman toteuttamiseen.

8.2 Käyttöliittymä

Tässä luvussa kuvaillaan käyttöliittymän toteutus yleisellä tasolla, koska hienokuormitusohjelman toteutusta ei saanut julkistaa. Käyttöliittymän toteutus toteutettiin yksi näkymä kerrallaan, kuten suunnittelussa tulikin jo esille. Käyttöliittymää myös testattiin jo niillä tiedoilla, joita luotiin itse tietokantaan. Tietokannan ja käyttöliittymän versiot etenivät melko samalla viivalla projektin alussa, koska tietokannan versioita täytyi päästä testaamaan käytännössä.

Lean-ajattelutavan ja ketterämenetelmien yhdistäminen auttoi poistamaan käyttöliittymän suunnittelussa jo sellaiset ominaisuudet, joita ei oikeasti tarvittu vielä tässä ohjelmassa. Näillä kahdella menetelmällä mahdollistettiin hukkatyön minimointi ja täten ohjelman tehokkaampi toteutus.

Kappaleen alussa kuvaillaan käyttötapaukset, jotka olivat vaatimusmäärittelyn priorisointilistalla ja myöhemmin ne käyttötapaukset, jotka tulivat esille vasta testauksien jälkeen. Nämä kuvaillaan luvussa: Testivetoinen kehitys: Uudet vaatimukset.

8.2.1 Päivänäkymä

Päivänäkymään tuli muutama toiminnallinen lisäys, joita ei ollut otettu huomioon vaatimusmäärittelyssä. Kun henkilöä valitaan aamu- tai iltavuoroon, näyttää va-

likkopainike reaaliaikaisen henkilölistan, josta tiiminvetäjä näkee aktiiviset työntekijät. Täten tiiminvetäjä on helppo nähdä, ketkä työntekijät ovat käytettävissä.

Päivänäkymässä olevat toiminnot eivät muuttuneet toteutuksessa ja ovat tässäkin vaiheessa seuraavat:

- tilausten tarkastelu
- henkilön kuormitus aamu- ja iltavuoroon
- kuormituksen jälkeinen tarkastelu
- työvuorolistan tulostaminen.

Tilausten tarkastelun kysely on toteutettu erikseen jokaiselle runkokoolle ja täten laskettu runkokoon tilaukset. Kyselyssä täytyi ottaa huomioon toimituspäivä, joka muutettiin SELECT -lauseessa suunniteltu valmistuspäiväksi ja tämän vuoksi toimituspäivää täytyi miinustaa yhdellä päivällä. Kun halutaan saada henkilötarve tietoon, täytyy laskea paljonko yksi henkilö suunnilleen kerkeää tehdä työpäivän aikana.

Jos määrä vastaa henkilötarvetta, näyttää ohjelma vihreää ja jos alle, punaista (kuva 13). Tämä toteutettiin Accessin Conditional Formatting Rules Managerilla. Runkokoot käyttävät myös samaa toimintoa erottuakseen toisistaan.

Kuva 13. *Henkilöiden oikea määrä*

Kyselyssä otettiin myös huomioon viikonloput, joka on määritelty kyselyn HAVING -lauseessa. Jos viikonpäivä on perjantai, ohjelma ymmärtää hakea maanantain tilaukset, jotka täytyy valmistaa perjantaina. Pyhäpäiviä ei vielä otettu huomioon tässä hienokuormitusohjelman protoversiossa.

Henkilöiden kuormitus aamu- ja iltavuorossa sisältävät yhden ja saman taulun, jotka eroteltiin erikseen kyselyllä lomakkeen sisällä. Molemmissa näkymissä on

olemassa vuoroniminen kenttä, joka on piilotettu. Vuoron oletusarvoksi on määritelty aamunäkymässä ”Aamu” ja iltavuorossa ”Ilta”. Tällä mahdollistetaan molemmissa näkymissä olevien henkilöiden automaattinen sijainti, aamu- tai iltavuoroon riippuen näkymästä, mitä käyttäjä täyttää. Henkilöiden kuormituksessa valitaan työntekijä ja runkoko, muut kentät täyttyvät automaattisesti.

Kun henkilö on valittu, valitaan tämän jälkeen runkoko. Runkokoon valintapainike hakee kaikki kyseisen linjan runkokoot. Runkokoon valitsemisen jälkeen, putkitunnus tulee automaattisesti putkitunnuskenttään ja tämän vuoksi kyselyyn täytyy myös ottaa putkitunnus mukaan. Runkokoon valinnan jälkeen, ohjelma päivittyy ja kuormitettu henkilö tulee näkyviin kuormituksen jälkeiseen näkymään.

Runkokoon valintapainikkeen toteutus:

```
SELECT runkokoko.runkotunnus, runkokoko.putkitunnus
FROM runkokoko
ORDER BY runkokoko.runkotunnus;
```

Automaattinen täyttö putkitunnukselle toteutetaan seuraavalla VBA-koodilla:

```
Private Sub runkotunnus_Change()
Me.putkitunnus.Value = Me.runkotunnus.Column(1)
End Sub
```

Ja ohjelman päivitys:

```
Private Sub runkotunnus_AfterUpdate()
Forms!päävalikko!navi!subformm.Form.Refresh
End Sub
```

Henkilön kestolla määritellään, kuinka paljon päivästä kyseinen henkilö tekee valittua runkokokoa. Kesto on oletuksena ykkönen, mutta käyttäjä voi halutessa muuttaa sen esimerkiksi 0,5:ksi. Tämä mahdollistaa lisätä samannimisen henkilön uudestaan toiselle runkokoolle, eli täten sama henkilö voi valmistaa puolet päivästä yhtä ja puolet päivästä toista laitetta.

Kun kyseessä on päivänäkymä, henkilöiden määritysnäkymässä täytyy olla funktio, joka lisää henkilöt tälle päivälle. Näkymässä on piilotettu päivämääräsarake, johon on lisätty Date() -funktio. Date() hakee automaattisesti tämän päivän ja täten henkilö kuormittuu juuri sille päivälle ja käyttäjän ei tarvitse erikseen valita päivämäärää. Automaattisuudella pyritään helpottamaan käyttäjän työtä, jotta käyttäjällä olisi mahdollisimman vähän täytettävää. Kaikki mitä ohjelmassa on vain mahdollista automatisoida, on pyritty automatisoimaan.

Käyttäjällä on mahdollista hakea tiettyä kuormitettua henkilöä ja poistaa henkilöitä. Muokkaaminen onnistuu myös suoraan määritysnäkymässä. Henkilöiden hakeminen on toteutettu VBA-koodilla, kuten myös henkilön poistaminen.

Henkilöiden hakeminen onnistuu enteriä painamalla tai hae-painikkeella. Seuraava VBA-koodi on painikkeesta:

```
'Painikkeella
Private Sub btnHae_Click()

Dim strsearch As String
Dim Task As String
'Tarkistaa onko kenttään syötetty mitään
If IsNull(Me.txtHae) Or Me.txtHae = "" Then
MsgBox "Kirjoita hakusana.", vbOKOnly, "Hakusana puuttuu"
Me.txtHae.SetFocus
'Hakukriteetit
Else
strsearch = Me.txtHae.Value
Task = "SELECT * FROM qry_määritys_henkilö_päivä WHERE vuoro='aamu' AND
((työntekijätunnus Like ""*" & strsearch & "*"") OR (putkitunnus Like ""*" & strsearch
& "*"")OR (runkotunnus Like ""*" & strsearch & "*""))"
Me.RecordSource = Task
End If
End Sub
```

Haun jälkeen, kaikkien henkilöiden hakeminen:

```
‘Hae kaikki
Private Sub btnHaeKaikki_Click()
```

```
Dim Task As String
‘Kysely
Task = "SELECT * FROM qry_määritys_henkilö_päivä  WHERE vuoro='Aamu'"
Me.RecordSource = Task
End Sub
```

Henkilön poistaminen:

```
‘Henkilön poisto
Private Sub Command112_Click()
‘Varmistus
If MsgBox(Prompt:="Haluatko varmasti poistaa?", Buttons:=vbYesNo, Title:="Poista") =
vbYes
Then
‘Poistaminen
On Error Resume Next
DoCmd.RunCommand acCmdDeleteRecord
 If Err.Number = 0 Then
MsgBox Prompt:="Poistettu", Buttons:=vbOKOnly, Title:="Poistettu"
Else
‘Virheilmoitus
MsgBox Prompt:="Tietoja ei voitu poistaa, koska tietoja ei löytynyt tai käyttöoikeudet
puuttuvat", Buttons:=vbOKOnly, Title:="Virhe"
End If
Else
‘Peruurus
MsgBox Prompt:="Peruutettu", Buttons:=vbOKOnly, Title:="Peruutettu"
End If
End Sub
```

Kuormituksen jälkeinen näkymä tulee vasta näkyviin, kun kuormitukset on toteutettu tilauksille. Kuormituksen jälkeinen näkymä näyttää seuraavat: runkokoon, laitteita jäljellä, henkilötarve jäljellä ja laitteita tehty yhteensä. Suunnitteluosiossa kuormituksen jälkeisessä näkymässä oli näkyvillä, kuinka paljon vuoroa on vielä jäljellä toteuttamaan kaikki tilaukset, mutta sitä ei havaittu hyväksi, joten se poistettiin käytöstä. Tuntiseurannan avulla nähdään tarpeeksi hyvin tilauksien eteneminen ja täten saatiin yksi sarake pois kuormituksen jälkeisestä näkymästä.

Työvuorolistan tulostaminen toteutetaan Accessin raportilla. Työvuorolista tulostaa kuormitetut henkilöt raportille kyselyllä. Kysely täytyy rajoittaa ainoastaan sen hetkiseen päivään, jotta raportissa ei näy muiden päivien kuormitettuja henkilöitä. Tämä rajaus on toteutettu HAVING -lauseessa.

8.2.2 Viikkonäkymä

Viikkonäkymässä näkyy koko viikon tilanne tilauksista ja henkilötarpeesta. Molemmat ovat toteutettu Accessin PivotChart näkymällä. Viikkonäkymässä näkyy kaikki runkokoot ja henkilötarve runkokohtaisesti päiväjärjestyksessä.

8.2.3 Tuntiseuranta

Tuntiseurannan tarkoitus on näyttää tavoite ja valmistuneet laitteet tuntitasolla. Tavoite syntyy siitä, kuinka paljon kuormitetulla henkilöstöllä täytyisi tulla laitteita tunnissa. Valmistuneet laitteet, eli tulleet laitteet näyttävät kuinka paljon laitteita on valmistunut tunnissa. Aamuvuoron aikaväli on klo 05–14 ja iltavuoron klo 14–23. Tuntiseurannassa on omat välilehdet jokaiselle runkokoolle, jonka toteutus toi aluksi haasteita, mutta lopputulokseen saa olla täysin tyytyväinen, että kyseisen näkymän sai toteutettua Accessilla.

Kyselyllä täytyi rikkoa valmistuneiden laitteiden päivämäärä tuntitasolle, joka toteutettiin DatePart(”h”,[PARTITION_DATE]) määrityksellä SELECT -lauseessa. Tässä kyselyssä käytettiin myös reaalityöaikaa, mutta työaika oli tässä tapauksessa rikottu tunneiksi. Esimerkiksi, jos työntekijät olivat kahvilla klo. 10.00–10.15, tällöin tunnissa on reaalityöaikaa 45 minuuttia. Tällä luodaan reaaliaikainen tulos tavoitteista. HAVING -lauseessa on määritelty, mitä tuntia lasketaan ja mistä reaaliajan työtunnista työaika haetaan.

Tuntiseurannan hahmottamista on pyritty helpottamaan väreillä. Punainen kuvaa tavoitteesta jäämisen ja vihreä tavoitteessa pysymisen. Kuva 14 kuvaa kuinka värien käyttö on toteutettu.

Kuva 14. *Tavoitteiden hahmottaminen.*

8.2.4 Osaamismatriisi

Osaamismatriisi jäi hieman vaisuksi, koska aikataulu tuli vastaan. Saatiin kuitenkin toteutettua ne asiat, mitä vaatimusmäärittelyssä oli priorisoitu. Osaamismatriisi näyttää ne henkilöt, ketkä ovat valmistaneet jompaakumpaa runkokokoa.

8.2.5 Testivetoinen kehitys: Uudet vaatimukset

Ohjelman jatkuva testaaminen toi muutaman lisäyksen ohjelman toiminnallisuuksiin, joita ovat:

- päivänäkymä +1 päivä
- työntekijöiden hallintapaneeli
- näkymä käytettävistä henkilöistä

Päivänäkynä +1 päivä on täysin samannäköinen kuin normaali päivänäkymä. Päivänäkymä +1 lisää jokaiseen näkymään +1 päivän, jonka avulla käyttäjä pystyy kuormittamaan seuraavan päivän tilauksia.

Kysely on myös täysin samanlainen, kuin normaalissa päivänäkymässä, mutta HAVING -lauseen IF-funktiossa on määritelty DateValue(Now()+1, joka hakee ne tilaukset, joiden suunniteltu aloituspäivä olisi huomenna.

Työntekijöiden hallintapaneelin avulla käyttäjä voi määritellä käytettävät henkilöt. Hallintapaneelin näkymä näyttää henkilöiden tilanteen maanantaista perjantaihin ja käyttäjä voi merkata, onko työntekijä kierrossa jossain muualla linjalla, tai onko työntekijä lomalla. Ohjelma myös laskee näkymän alaosaan käytettävissä olevat resurssit päiväkohtaisesti. Hallintapaneelinäkymässä on olemassa molemmille tiiminvetäjille oma henkilölista, joita he päivittävät. Kun vuoro-kenttään muutetaan aamuvuoro, päivänäkymässä henkilöt näkyvät aamukuormituksen pai-

nikevalikossa. Tällä estetään, etteivät tiiminvetäjät vahingossa sijoita samaa henkilöä aamu-, ja iltavuoroon.

Työntekijöiden hallintapaneelin luotua, voidaan luoda näkymä **käytettävissä olevista henkilöistä**, kun käyttäjä kuormittaa henkilöitä. Toiminnot eivät paljoa muuttuneet, vaan ainoastaan henkilöiden valintapainike. Valintapainikkeesta voidaan nyt nähdä kyseisen viikon tilanne käytettävissä olevista henkilöistä päivänäkymässä, jotka ovat määritelty hallintapaneelissa aamuvuoroon. Täten käyttäjän ei tarvitse navigoida edestakaisin, kahden näkymän välillä.

Kun kaikki lomakkeet saatiin luotua, luotiin vasta navigointipainikkeet lomakkeiden välille. Navigointi tapahtuu päävalikossa, joka toimii ohjelman pääsivuna ja se aukeaa, kun ohjelman käynnistää. Päävalikon aloitusnäkymä on päivänäkymä.

9 TESTAUS

Testaaminen toteutettiin projektin aikana testivetoisella kehityksellä, joka mahdollisti näyttää tiiminvetäjälle, jokaisen toteutetun komponentin jälkeen toimiva ohjelmisto. Tiiminvetäjää miellytti tämänkaltainen testaaminen, koska tiiminvetäjä näki jatkuvasti tilanteen ohjelmasta ja pystyi hyvin vaikuttamaan ohjelman toiminnallisuuteen ja auttoi kehittämään hienokuormitusohjelmaa oikeaan suuntaan.

Kun toteutettiin ainoastaan yksi näkymä ja näkymän yksi komponentti kerrallaan, vaikutti se positiivisesti työn etenemiseen ja testaukseen. Kun ollaan uuden aiheen äärellä, niin ensimmäisellä kerralla on miltei mahdotonta luoda, juuri oikeanlainen käyttöliittymä. Jatkuva testaaminen toi uusia versioita näkymistä ja komponenteista, joilla saatiin toteutettua juuri sellainen näkymä, kuin käyttäjä oli toivonut.

Hienokuormitusohjelmaa testattiin lähes päivittäin ja ongelmien ilmaantuessa, niihin pyrittiin reagoimaan välittömästi. Käyttäjäläheinen ohjelmistokehitys varmistaa oikean tien testaukselle ja täten ohjelmasta tulee käyttäjän kaltainen, jonka käytettävyyden käyttäjä helposti ymmärtää.

Lopullinen testaus toteutettiin kahden viikon rypistyksenä, jonka aikana tiiminvetäjä testasi ohjelmaa päivittäin ja kirjoitti ylös huomioita ohjelman käytettävyydestä, toiminnallisuudesta ja mahdollisista lisätoiminnoista. Testauksesta tullut palaute kuvaillaan myöhemmin yhteenveto kappaleessa (10). Jatkuvan testaamisen ansiosta, projektin loppupäähän ei jäänyt isoa pinoa virheitä korjattavaksi, vaan ainoastaan hieman hienosäätöä, joista keskusteltiin tiiminvetäjän kanssa paikan päällä testauksen yhteydessä ja toteutettiin tarvittavat muutokset ohjelmaan saman tien.

10 YHTEENVETO

Projektin aloittaminen oli kaikista haastavinta, koska minulla ei ollut lainkaan tuntemusta, eikä ymmärrystä tuotannonohjauksesta ja siitä, mitä kaikkea täytyy ottaa huomioon tuotantolinjan hienokuormituksessa. Teoriaan perehtymisen jälkeen tuotannonohjaus alkoi hahmottua ja itse ohjelman toteutus pystyttiin aloittamaan. Olimme aluksi epävarmoja Accessista, koska emme tienneet pystyykö Accessilla toteuttamaan näinkin monimutkaisen ohjelman.

Tiedon kartoitus oli seuraavaksi haastavin vaihe, eli ne tiedot mitkä haluttaisiin Vaconin muista järjestelmistä. Tietokanta rakentui taulu taululta ja samalla myös käyttöliittymä, koska luotuja tietoja täytyi päästä testaamaan. Tiedot joita ajateltiin tarvitsevan, luotiin tietokantaan itse ja testattiin päästäänkö näillä tiedoilla haluttuun toiminnallisuuteen. Kun tietojen kartoitus oli saatu päätökseen ja ”oikea” data kerätty muista järjestelmistä, käynnistyi vasta ohjelman toteutus.

Vaatimuksien määrittely, kuten käyttäjätarinat ja UML-kaavioiden suunnittelu oli erittäin haastavaa tässä projektissa, koska aihe oli melko uusi kaikille. Tämän vuoksi vaatimukset muuttuivatkin melko radikaalisti, suuntaan tai toiseen ohjelmaa testattaessa. Ohjemaan toteutettiin ainoastaan ne toiminnot, jotka pidettiin tärkeänä juuri nyt ja täten kaikki ylimääräiset toiminnot jätettiin pois ja keskityttiin vain ja ainoastaan luomaan mahdollisimman yksinkertainen ohjelma, tuotantolinjojen hienokuormittamiseen.

Ohjelman toteutus eteni hyvää tahtia, käyttäen ketterän ohjelmistoprojektin menetelmiä. Osa toiminnoista saatiin toteutettua yllättävänkin nopeasti, juuri sellaiseksi kuin haluttiin ja osaa toiminnoista kehitettiin jatkuvasti testauksen yhteydessä. Oli erittäin mukavaa päästä kokeilemaan ketterän menetelmiä käytännössä ja lisäämällä siihen Leanin ajattelutapaa ja protoilua. Hyvä tapa mielestäni ohjelmistoprojektissa on valita näistä kolmesta menetelmistä ne menetelmät, jotka omasta mielestä sopisivat parhaiten projektiin.

Hienokuormitusohjelman pilottikäytöstä saatujen tiiminvetäjän palautteiden perusteella, ohjelmaan saa olla hyvinkin tyytyväinen. Tiiminvetäjä nostaa erityisesti

esille tuntiseurannan, jolla nähdään valmistuneet laitteet reaaliajassa ja hallintapaneelin, joka mahdollistaa päivittää henkilöiden tietoja siten, että ne henkilöt jotka ovat esim. lomalla, eivät enää näy päivänäkymän kuormitusnäkymässä. Tiiminvetäjä pitää myös hyvänä viikkonäkymää, josta pystytään tarkastelemaan koko viikon tilauksia ja täten reagoimaan isoihin tilauspiikkeihin.

Data Warehousen kehittäminen aloitettiin tämän projektin valmistuttua, koska työni määritti ne tiedot, joita tarvittiin Data Warehousen luomiseen.

Niin kuin aiemmin tuli jo esille, olimme epävarmoja Accessista, että kykeneekö sillä toteuttamaan tämän hienokuormitusohjelman, mutta projektin jälkeen näkökulma muuttui hieman erilaiseksi. Access sopii erittäin hyvin uusien ohjelmien protoiluun, varsinkin kun liikutaan täysin uudella osa-alueella. Accessissa saa nopeasti luotua tietokannan ja datan, myös datan hakeminen muista järjestelmistä toteutuu helposti. Käyttöliittymän luominen samassa työkalussa on Accessin tärkein ominaisuus, joka mahdollistaa tietokannan testaamisen vaivattomasti. Tuli ainakin yllätyksenä se, mihin kaikkeen Access soveltuu ja mitä kaikkea sillä pystyy tekemään.

Tämä projekti oli suuri oppimisretki, jossa kehittyivät ohjelmointitaidot ja se, mitä kaikkea täytyy ottaa huomioon ohjelmistoprojektia toteuttaessa. Oli myös opettavaista, kokeilla ensimmäistä kertaa kaikkia kolmea menetelmää ja yhdistää ne yhdeksi kokonaisuudeksi osana ohjelmistoprojektin prosessia, joita olivat protoilu, ketterä menetelmä ja Lean. Projektia toteuttaessa myös huomattiin, kuinka iso merkitys jatkuvalla testauksella ja käyttäjän läsnäololla on ohjelman lopputulokseen.

Projektiin osallistuneiden henkilöiden työpanos oli isossa roolissa, jotta projekti saatiin toteutettua aikatauluun mennessä. Toteuttamalla juuri ne toiminnot mitä oikeasti tarvitaan, minimoitiin mahdollinen hukkatyö, joka edesauttoi projektia pysymään aikataulussa.

11 JATKOKEHITYS

Jatkokehitettävää jäi viimeisimmän testauksen ja pilottikäytön jälkeen, joita ei keritty enää toteuttamaan työn aikataulun puitteissa. Päivänäkymälomakkeen kuormituksen jälkeiseen näkymään toivottiin ominaisuus, joka näyttäisi tiedot reaaliajassa, kuten tuntiseurannassakin, koska tilauksia saattaa tulla lisää kesken päivän ja niihin ei tässä versiossa vielä pystytä reagoimaan, koska tilaukset päivittyvät ainoastaan yöllä ohjelmaan.

Hallintapaneeliin toivottiin uutena ominaisuutena, valita henkilöitä päivätasolla toiseen vuoroon. Esimerkiksi henkilö on maanantai-keskiviikko välisen ajan aamuvuorossa ja loput päivät viikosta iltavuorossa. Osaamismatriisin haluttiin lisätä aikarajausominaisuus, jolla voitaisiin nähdä aikavälin avulla, mitä henkilöt ovat tehneet ja kuinka paljon.

Jatkokehityksessä otetaan myös mahdollisesti lisää tuotteita työkaluun ja toinen tiiminvetäjä käyttämään työkalua, täten saadaan uutta näkökulmaa hienokuormitusohjelman käytettävyydestä.

LÄHTELUETTELO

Haikala, I. & Mikkonen, T. (2011). *Ohjelmistotuotannon käytännöt*. 12. painos. Helsinki. Talentum.

Haikala, I. & Märijärvi, J. (2004). *Ohjelmistotuotanto*. 10. painos. Helsinki. Talentum.

Haverila, M., Uusi-Rauva, E., Kouri, I. & Miettinen, A. (2005). *Teollisuustalous*. 5. painos. Tampere. Infacs Johtamistekniikka Oy.

Hokkanen, S., Karhunen, J. & Luukkainen, M. (2011). *Johdatus logistiseen ajatteluun*. 6. painos. Kangasniemi. Sho Business Development.

Hovi, A., Huotari, M. & Lahdenmäki, T. (2005). *Tietokantojen suunnittelu & indeksointi*. 2. painos. Jyväskylä. Docendo.

Extreme Programming 2013. Viitattu 17.3.2015. http://www.extremeprogramming.org/

Kauppalehti 2015. Viitattu 7.1.2015. http://www.kauppalehti.fi/5/i/porssi/porssikurssit/osake/tulostiedot.jsp?klid=1064

Kauppalehti 2014. Viitattu 7.1.2015. http://www.kauppalehti.fi/uutiset/vacon-siirtyi-osaksi-danfossia/kr9xf9jd

Beck, K. (2000). *Extreme Programming Explained, Embrace Change*. Addison-Wesley.

Lehtinen, M. 2011. *Lean-ohjelmistokehityksen käyttöönoton jälkeisiä ohjelmistokehitysnopeutta rajoittavia tekijöitä*. Viitattu 18.3.2015. https://jyx.jyu.fi/dspace/bitstream/handle/123456789/39954/URN:NBN:fi:jyu-201210102653.pdf?sequence=1

Lehtonen, J-M. (2004). *Tuotantotalous* Helsinki. WSOY.

Lindberg, H. (2003). *Extreme Programming.* Viitattu 22.3.2015. https://tampub.uta.fi/bitstream/handle/10024/91382/Lindberg_Harri.pdf?sequence=1

Miettinen, P. (1993). *Tuotannonohjaus ja logistiikka.* Helsinki. Painatuskeskus.

Office 2010. *Access 2010:n perustoiminnot.* Viitattu 8.1.2015. https://support.office.com/fi-fi/article/Access-2010n-perustoiminnot-268acfed-2484-4822-acb3-c30e58045588?ui=fi-FI&rs=fi-FI&ad=FI#

Office 2010. *Johdanto Accessin SQL-toimintoihin*. Viitattu 8.1.2015. https://support.office.com/fi-fi/article/Johdanto-Accessin-SQL-toimintoihin-d5f21d10-cd73-4507-925e-bb26e377fe7e#bm1

Office 2010. *Johdanto ohjelmointiin.* Viitattu 10.1.2015. https://support.office.com/fi-fi/article/Johdanto-Access-ohjelmointiin-25edaefe-e917-4608-8ba0-dab7c75cbe0c

Tolvanen, P. *Ketteryys haltuun: Yleisimmät ketterät käytännöt.*. Viitattu 24.2.2015. http://www.meteoriitti.com/Artikkelisarjat/Ketteryys-haltuun/Ketteryys-haltuun-Yleisimmat-ketterat-kaytannot/

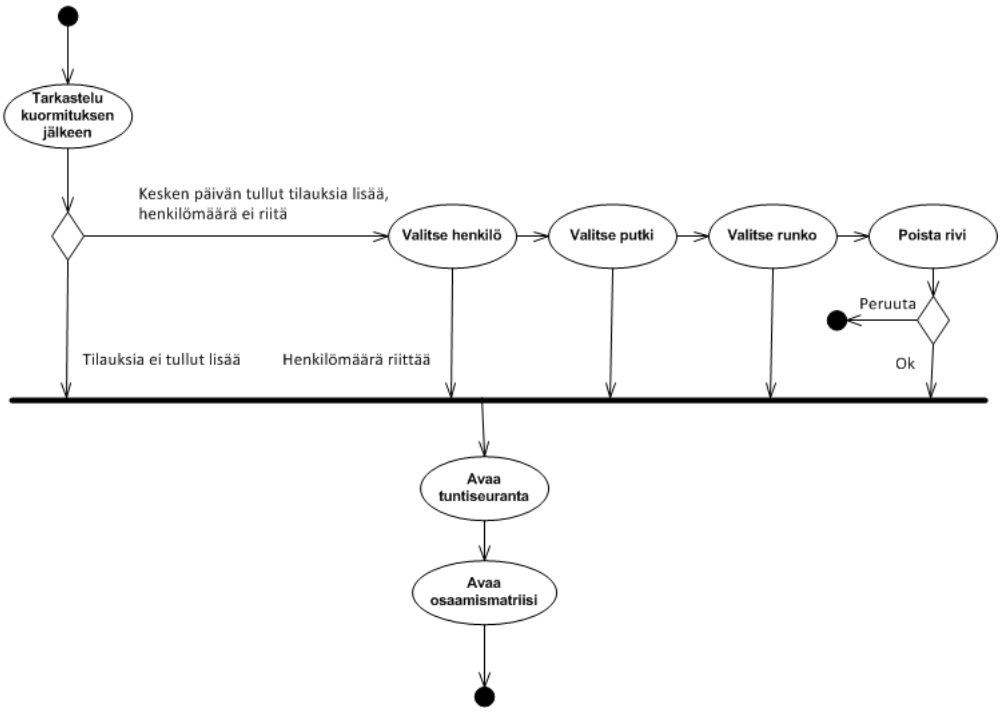
Uusi-Rauva, E., Haverila, M. & Kouri, I. (1994). *Teollisuustalous*. 2. painos. Ylöjärvi. Infacs Johtamistekniikka Oy.

Vacon. *Tärkeimmät teemat.* Viitattu 7.1.2015. http://www.vacon.com/fi-FI/Vacon/Yhteiskuntavastuu/Vaconin-tarkeimmat-teemat/

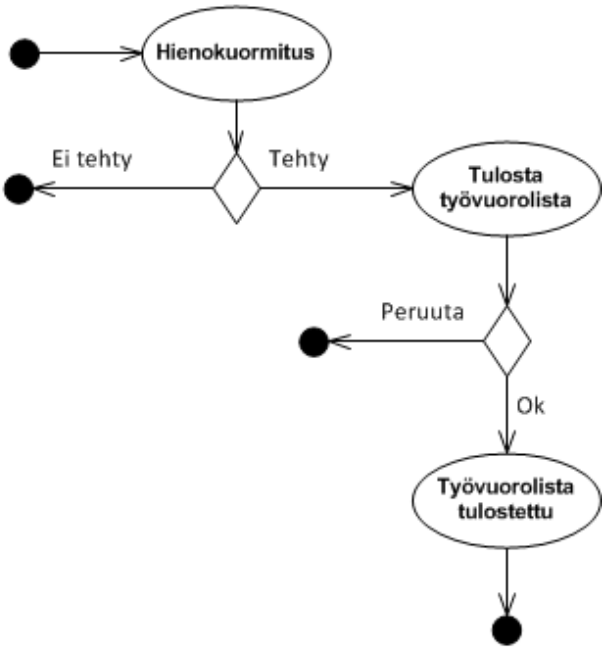
Vacon. *Yhtiö.* Viitattu 7.1.2015. http://www.vacon.com/fi-FI/Vacon/yritys/

Wiio, A. (2004). *Käyttäjäystävällisen sovelluksen suunnittelu.* Helsinki. Edita Prima Oy

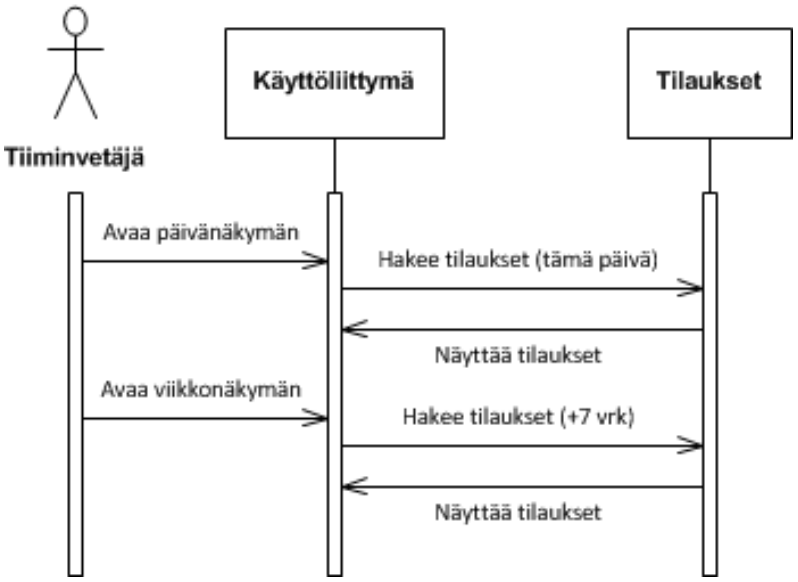
Aktiviteettikaavio. Kuormituksen jälkeinen tarkastelu, tuntiseuranta ja osaamismatriisi.



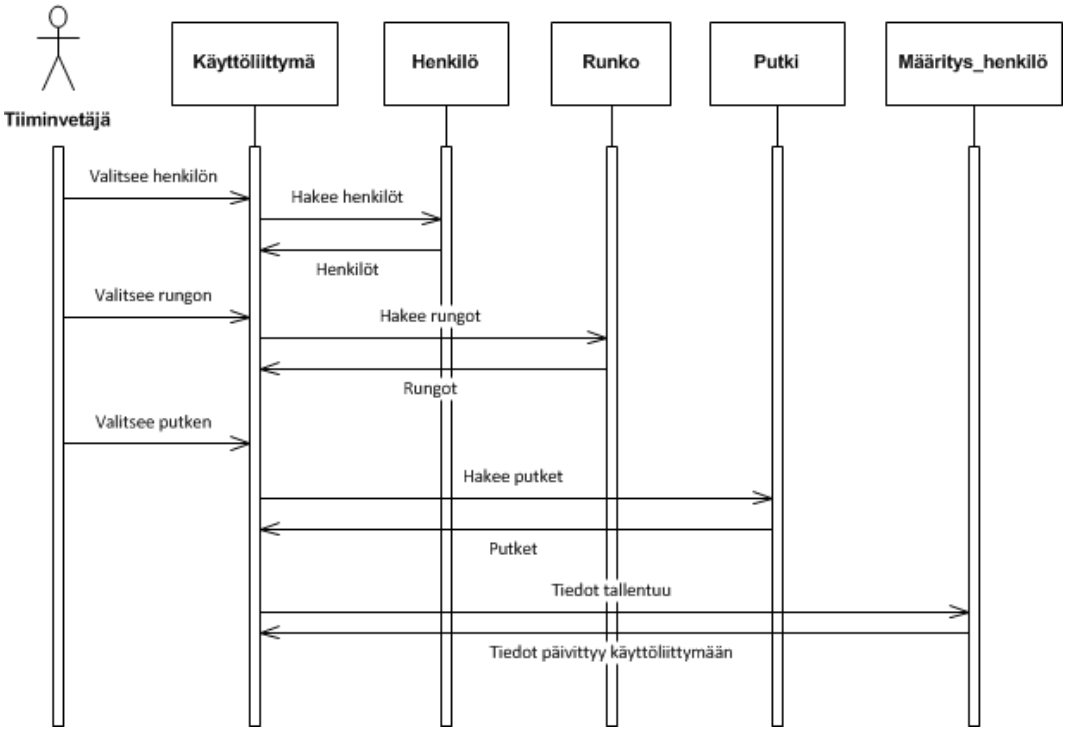
Aktiviteettikaavio, työvuorolistan tulostaminen.



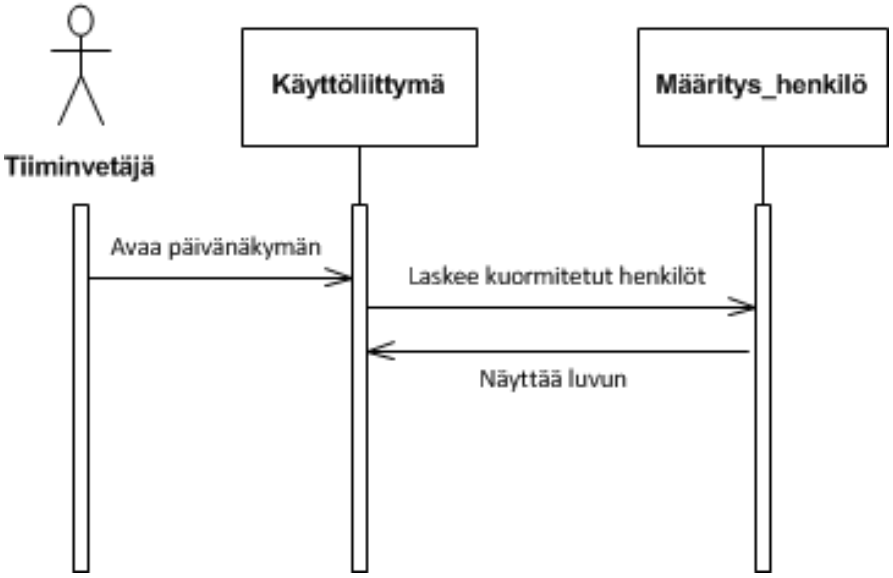
Sekvenssikaavio, tulevat tilaukset.



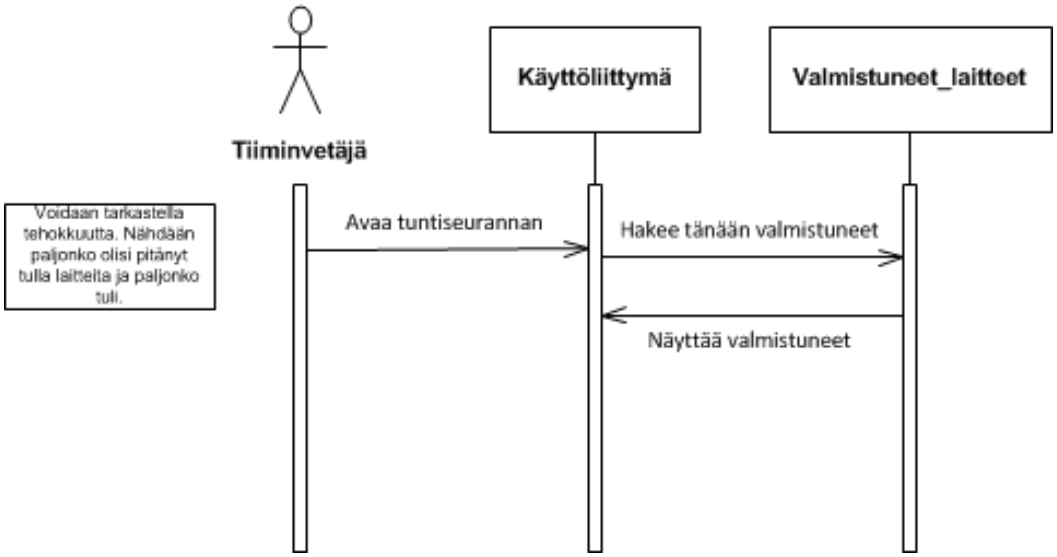
Sekvenssikaavio, hienokuormitus.



Sekvenssikaavio, kuormituksen jälkeinen tarkastelu.



Sekvenssikaavio, tuntiseuranta.



Sekvenssikaavio, osaamismatriisi.