LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

HUOLTOLASKENTA

Huoltolaskentaohjelmisto

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikka
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2013
Sami Rivalli

Lahden ammattikorkeakoulu
Tietotekniikka

RIVALLI, SAMI: Huoltolaskentaohjelmisto

Ohjelmistotekniikan opinnäytetyö, 46 sivua, 3 liitesivua

Kevät 2013

TIIVISTELMÄ

Joukkojen huoltotarpeen ennakointi, huoltotilanteen seuranta ja huoltoresurssien kohdentaminen ovat oleellinen osa huolto-/logistiikkaupseerien tehtäviä. Ennakoinnin laskentaperusteet on tarkistettava aina silloin kun taistelutapa tai kokoonpanot (kalusto, materiaali ja käyttöperiaatteet) uudistuvat. Huoltotilannetietojen kerääminen ja yhdistäminen on nykymuotoisena työlästä ja virhealtista.

Työn tavoitteena oli automatisoida huoltotilanneilmoitusten yhdistäminen. Lisäksi kerättiin ja kehitetiin huoltolaskennassa käytettäviä laskentakaavoja ja algoritmeja.

Työssä luotiin huoltolaskennan malli, jonka pohjalta totetutettiin huoltolaskentaohjelmisto. Ohjelmistoon toteutettiin huoltotilanneilmoitusten tiedot yhdistävä huoltotilannelaskenta. Huoltotarpeen ennakointiin määriteltiin tappio- ja kulutuslaskenta, joka on liitettävissä huoltotilannelaskentaan.

Työn tuloksena on huoltotilannelaskenta, joka nopeuttaa merkittävästi hidasta ja virhealtista huoltotilanneilmoitusten yhdistämistä. Huoltotarvelaskennan kehittäminen vapauttaa toimijat rutiiniluontaisista ja virhealttiista tehtävien tekemisestä laskennan tulosten analysointiin.

Asiasanat: huoltolaskenta, pandas, puolustusvoimat, pyqt, python.

Lahti University of Applied Sciences
Information Technology

RIVALLI, SAMI: Calculations Module for Military Logistics

Bachelor's Thesis in Information Technology, 46 pages, 3 pages of appendices

Spring 2013

ABSTRACT

Anticipating, monitoring and targeting logistical needs are the main tasks of logistics officers in the defence forces. Anticipation methods should always be updated when usage of forces change or unit capabilities (equipment, material, personnel) develop. In the current form, collecting and aggregating support data requires manual work and is subject to human errors.

The goal of this study was to better integrate and automate the current collection and aggregation of operational logistics data. A second target was to collect and develop functions and algorithms used in logistics calculations.

A model was created for current operational support calculations. The model formed the basis for the development of logistics software. The software includes a module that combines situation reports used for calculations. A separate loss and consumption calculation was created, which can be attached to the main calculations when necessary.

The result of this study is a logistics service calculation model and related software. These tools significantly speed up the current slow and error-prone manual data integration and automation.

When using the developed tool, critical personnel is no more tied up performing various manual and time-consuming tasks. Instead, the personnel can use their time analysing the result of the developed calculations.

Key words: logistics calculations, pandas, finnish defence forces, pyqt, python

SISÄLTÖ

Käsitteet

Huoltotarve	Tappiot ja kulutus aiheuttavat huoltotarvetta.
Kulutusfunktio	Funktio, joka laskee varusteen kulutuksen.
Kulutuslaskenta	Kulutuslaskennalla tuotetaan kulutusennusteita. Kulutus on suoraan verrannollisia toiminnan tai taistelujen intensiteettiin.
Kuormalavakerroin	Käytetään tappio- ja kulutusennustelaskennassa ja huoltotarvelaskennassa, kun muutettetaan varusteen kappalemäärä kuormalavoiksi.
Latex	LaTeX on **ladontajärjestelmä**, joka rakentuu TeX-järjestelmän päälle. Se huolehtii automaattisesti esimerkiksi rivin- ja sivunjaosta, kuvien ja taulukoiden asettelusta, dokumentin sisäisistä viittauksista ja sisällysluettelosta.
MATI	Lyhenne sanoista Maavoimien tietojärjestelmä.
Massakerroin	Käytetään tappio- ja kulutusennustelaskennassa ja huoltotarvelaskennassa.
Raporttifunktio	Raporttifunktiot tuottavat raporttien latex-muotoisen sisällön pythonin pandas-kirjaston DataFrame-olioiden sisällöstä.
Skenaariotiedosto	Skenaariotiedosto sisältää arviotietoa siitä perusyksiköiden seuraavan vuorokauden tai pidemmän aikavälin toiminan tai taistelujen asteesta. Matala intensiteetti vastaa arvo 1 ja korkeaa arvo 6.
Tappiolaskenta	Tappiolaskennalla tuotetaan tappioennusteita. Tappiot ovat verrannollisia toiminnan tai taistelujen intensiteettiin.
Varuste	Varuste on yleisnimitys, jolla viitataan joko kalustoon tai materiaaliin.

1 JOHDANTO

Huoltokoulu on Lahdessa sijaitsevan Hämeen rykmentin joukkoyksikkö, jossa annetaan puolustusvoimien henkilöstölle huollon ja logistiikan perus- ja täydennyskoulutusta. Työn tilaaja on Huoltokoulun tutkimus- ja kehittämisosasto.

Joukkojen huoltotarpeen ennakointi, huoltotilanteen seuranta ja huoltoresurssien kohdentaminen ovat oleellinen osa huoltoupseerien tehtäviä. Ennakoinnin laskentaperusteet ovat muuttumassa taistelutavan ja kokoonpanojen uudistamisen myötä. Joukkojen kalusto, materiaali ja käyttöperiaatteet muuttuvat. Nykymuotoinen huoltotilannetietojen kerääminen ja yhdistäminen on työlästä ja virhealtista.

Työn tarkoituksena on pilotoida huoltotilannetietojen automatisointia. Työn tavoitteena on toteuttaa huoltolaskentaohjelmisto (ensimmäinen versio 1.0), joka automatisoi huoltotilanneilmoitusten yhdistämisen sekä tuottaa suorituskyky- ja huoltotarveraportteja huoltotilanneilmoitusten tiedoista. Oheistavoitteena on kerätä ja kehittää huoltolaskennassa käytettäviä laskentakaavoja ja algoritmeja. Huoltotilanneilmoitusten yhdistämisen automatisoinnin tarkoituksena on poistaa työläs ja virhealtis työvaihe. Tämä mahdollistaa sen, että huoltohenkilöstöllä on enemmän aikaa tulosten analysointiin. Raporttien tarkoituksena on havainnollistaa loppukäyttäjille huoltotarvelaskennan automatisoinnin ja visualisoinnin mahdollisuudet.

Laskennasta dokumentoidaan huoltolaskentaan liittyvä tiedonkeruu (data), datan esiprosessointi, varsinainen laskenta ja laskentatulosten visualisointi. Laskennan dokumentoinnin tarkoituksena on koota yhteen ja jäsentää huollon laskentaa siten, että dokumentaatiota voidaan hyödyntää huollon osuuksien kehittämistyössä osana maavoimien tietojärjestelmän (MATI) kehitystä.

Huoltolaskennan kehittäminen toteutetaan kolmessa vaiheessa, joista ensimmäinen toteutetaan tässä opinnäytetyössä. Ensimmäisessä vaiheessa suunnitellaan kokonaisuus ja rakennetaan huoltolaskentaohjelmiston ensimmäinen versio. Toisessa vaiheessa testataan ohjelmisto ja siihen liittyvä laskenta. Kolmannessa vaiheessa tehdään tarvittavat korjaukset ja muutokset sekä toteutetaan muut osakokonaisuudet ensimmäisessä vaiheessa laaditun kokonaisuunnitelman mukaisesti.

Työssä rakennettava huoltolaskentaohjelmisto on suunniteltu käytettäväksi nykyisessä tuotantoympäristössä (MATI1) sekä tukemaan korvaavan järjestelmän (MATI2) kehitystyön määrittelytyötä. Esitetyt tekniset ratkaisut eivät sellaisenaan sovellu siirrettäväksi MATI2:een, mutta laskennan toimintamalli on siirrettävissä. Kuvattava toimintamalli on huoltolaskennan kehittämisen välivaiheen tuotos, jonka tarkoitus on palvella MATI2:n käyttäjävaatimusten määrittelytyötä.

Opinnäytetyössä käytettävät huoltolaskentaohjelmiston aineistot, kuten laskennan lähtötietoina käytettävät huoltotilanneilmoitukset, ovat kuvitteellisia. Lähtöaineiston, kuten kokoonpanojen henkilöstön, kaluston ja materiaalin, määrä ja laatu eivät vaikuta laskentaan. Huoltotilanneilmoituksissa ja muissa lähtötiedoissa käytetään tekijän generoimia joukkorakenteita, joiden henkilöstö, kalusto ja materiaalien määrä ja laatu ovat tähän työhön luotuja. Huoltolaskentaohjelmiston toiminta ei ole riippuvainen käytetyn lähtöaineiston oikeellisuudesta. Opinnäytetyössä käytettävät kokoonpanot ovat kuvitteellisia. Henkilöstön, kaluston ja materiaalin määrä ja laatu eivät vastaa loppukäyttäjän käytössä olevia kokoonpanoja. Huoltotilanneilmoitusten oletetaan olevan oikein täytettyjä, eikä huoltolaskentaohjelmistoon tehdä huoltotilanneilmoitusten sisällön tarkistuksia.

Tässä opinnäytetyössä rakennetaan huoltolaskentaohjelmiston raportointitoiminto. Työstä rajataan pois raporttien sisällön tuottavat funktiot. Varsinaisen sisällön tuottavat funktiot ovat helposti lisättävissä, ja ne lisätään ohjelmistoon huoltolaskennan kehittämisen kolmannessa vaiheessa.

Tässä opinnäytetyössä toteutettava käyttöliittymä rakennetaan mahdollisimman kevyeksi ja sen käytettävyydestä tingitään työn tavoitteesta ja vaiheistuksesta johtuen. Huoltolaskentaohjelmiston ohjelmakoodissa toteutetaan poikkeusten käsittely vain kriittisimmiltä osin ja niiltä osin kuin ohjelmiston pilotointi ja testaus niitä edellyttävät.

Opinnäytetyön toisessa luvussa kuvataan huoltolaskennan toimintaympäristöä ja ohjelmistotuotannon teoriaa. Kolmas luku esittelee huoltolaskentaohjemiston käyttötilanteita ja ohjelmiston tietorakenteita. Huoltolaskennan eri laskennat esitellään neljännessä luvussa. Viidennessä luvussa tarkastellaan työn tuloksia asetettuihin tavoitteisiin nähden sekä pohditaan esiin tulleita haasteita. Lopuksi mietitään työn tulosten jatkokehittämistä ja hyödyntämismahdollisuuksia.

2 HUOLTOLASKENNAN MÄÄRITTELY

2.1 Toimintaympäristön kuvaus

Huollon toimialoja ovat täydennykset, kunnossapito, kuljetukset, lääkintähuolto ja huoltopalvelut. Täydennyksillä tarkoitetaan joukkojen materiaalin täydentämistä niiden kulutusta vastaavalla määrällä tai joukolle määritettyjen täydennysoikeuksien mukaan. Kunnossapidolla tarkoitetaan toimia, joilla ylläpidetään ja korjataan kalustoa. Kuljetuksilla tarkoitetaan henkilöstön, kaluston tai materiaalin kuljettamista. Lääkintähuoltoon kuuluvat henkilöstön terveyden ylläpito, sairauden ja haavoittuneiden hoito. Huoltopalvelut sisältää henkilöstön henkisen ja fyysisen hyvinvoinnin ylläpitävät toimet, kuten vaatteidenvaihto, sotilaskotitoiminta, kenttäposti ja kaatuneidenhuolto. (HKäsik2001.)

Joukon toiminnan laatu, kuten taistelujen kiivaus, vaikuttaa siihen kuinka suuriksi joukon tappiot ja kulutus muodostuvat. Tappioita kohdistuu joukon henkilöstöön, materiaaliin ja kalustoon sekä kulutusta joukon hallussa olevaan materiaaliin. Joukon huoltotarpeella tarkoitetaan joukon toiminnasta johtuvaa huoltotoimenpiteiden tarvetta, joilla kompensoidaan kulutusta ja tappioita. Huoltotarpeen laskentaan käytetään erilaisia taulukkolaskentaohjelmilla toteutettuja tappio- ja kulutuslaskentatyökaluja. Laskentataulukoissa on joiltakin osin toteutuksia huollon toimintojen mitoittamisen näkökulmasta.

Tappiolaskennan tarkoituksena on arvioida toiminnasta aiheutuvia tappioita niin henkilöstön, materiaalin kuin kalustonkin suhteen. Joukon tappioita kompensoidaan huollon toimenpitein, jolloin tappiolaskennan tuloksia käytetään huoltojoukkojen suorituskykyjen mitoittamiseen. Kulutuslaskennan tarkoituksena on auttaa arvioimaan joukon toiminnasta aiheutuvaa materiaalin, kuten ampumatarvikkeiden, kulutusta. Tappio- ja kulutuslaskentataulukoiden tuloksia käytetään joiltakin osin huoltojoukkojen toiminnan mitoittamiseen tai käytön ohjaamiseen. Suorituskykylaskentaa ei varsinaisesti ole.

Joukkojen toiminnan pääprosesseja ovat tilanteen seuranta, suunnittelu ja toimeenpano. Seuraavaksi tarkastellaan pääprosesseja huollon näkökulmasta. Tilanteen seurannalla tarkoitetaan toimia, joiden tarkoitus on muodostaa, ylläpitää ja jakaa tilannekuvaa ja näin mahdollistaa tilanneymmärräyksen muodostumisen ja ylläpitämisen. Tällaisia toimia ovat tilannetietojen kerääminen, yhdistely, analysointi, varmistaminen, ylläpitäminen, tunnuslukujen

laskenta jne. Joukot seuraavat hallussaan olevan henkilöstön, kaluston ja materiaalin määrää ja laatua sekä ilmoittavat nämä ylemmälle johtoportaalleen. Tilanteen seurantaan liittyen päivitetään jatkuvasti alajohtoportaiden ja johtoportaan huoltotilannetta, suorituskykyä ja huoltotarvetta. Tilanteen seurannan tuotoksia käytetään sekä suunnittelussa että toimeenpanossa.

Suunnitteluprosessi on monivaiheinen ja monitahoinen kokonaisuus, jonka lopputuotteena on suunnitelma tehtävän toteutuksesta. Suunnittelun aikana tuotetaan toimeenpanossa käytettäviä asiakirjoja, kuten sotapelin tuloksista tappio- ja kulutusarviota, joita päivitetään toimeenpanon aikana. Lähtötietona ovat joukkojen resurssit sekä arvio joukon toiminnan määrästä ja laadusta.

Toimeenpanoprosessilla tarkoitetaan tehtävän toimeenpanoa suunnitteluprosessin lopputuotteena syntyneen suunnitelman pohjalta. Seurannan kautta kerättyä tilannetietoa verrataan suunnitteluprosessin aikana tuotettuihin arvioihin ja ennusteisiin. Toimeenpanon aikana analysoidaan päivitettyjä huoltotilannetietoja, laaditaan lyhyen aikavälin tappio- ja kulutusarvioita sekä arvioidaan joukkojen suorituskykyä ja huoltotarvetta. Johtoportaassa analysoidaan alajohtoportaiden ja koko johtoportaan suorituskykyä käytettävissä olevan tilannetiedon perusteella ja arvioidaan suorituskykyä vasten niitä vaatimuksia, joita johtoportaalla tai sen alajohtoportailla on. Analyysin tuloksena tehdään päätöksiä resurssien kohdentamisesta. Analysoimalla edellisiä osana kokonaistilannetta saadaan perusteet ohjata johtoportaan ja sen alajohtoportaiden toimintaa. Tämän ohjauksen pohjalta luodaan johtoportaan huoltojoukon tilaukset sekä muiden alajohtoportaiden tukeutumisen järjestelyt.

Joukot tekevät tilanneilmoituksia määrävälein. Varsinainen tilanneilmoitus laaditaan iltaisin ja muutosilmoitus aamuisin. Tilanneilmoituksen osa, joka käsittelee huoltoa on huoltotilanneilmoitus. Huoltotilanneilmoitus on tilanneilmoituksen osa, jossa käsitellään huoltoa. Huoltotilanneilmoituksessa ei ilmoiteta tarkkoja lukemia kalustosta ja materiaalista, vaan arvioita joukon suorituskyvystä toimialoittain ilmoitushetkellä, kuluvan vaiheen lopussa ja seuraavan vaiheen lopussa. (HKäsiK2001, s.151.)

Perusyksiköt seuraavat hallussaan olevien varusteiden määrää ja laatua. Joukkoyksikkö määrittää perusyksiköille ajankohdat, joista perusyksiköt lähettävät joukkoyksikön esikuntaan huoltotilanneilmoituksen. Joukkoyksikön esikunnassa yhdistetään perusyksiköiden huoltotilanneilmoitukset. Yhdistämisen tuotoksena syntyy joukkoyksikön oma huoltotilanneilmoitus, jonka se edelleen

lähettää ylemmälle johtoportaalle. Joukkoyksikön esikunnassa analysoidaan perusyksiköiden tilannetta huoltotilanneilmoitusten ja kokonaistilanteen kautta. Analyysin tuloksena ohjataan joukkoyksikön omien huoltoresurssien käyttöä ja tehdään tarvittavat tukitarve-esitykset ylemmälle johtoportaalle.

2.2 Vaatimustenkäsittely

Huoltotilanneilmoitukselle ja sen käytölle tilaaja asetti seuraavat *asiakasvaatimukset*:

1. Perusyksiköt ilmoittavat resurssitilanteensa (henkilöstö, kalusto ja materiaali) huoltotilanneilmoituksella ml. vajeet.
2. Huoltotilanne1ilmoitus toimii tilauksena (joukot eivät tee erillisiä tilauksia).
3. Ilmoitusajakohtien välissä huoltotilanneilmoitusta käytetään joukoissa resurssitilanteen ylläpitoon.
4. Johtoportaan huoltotilanneilmoitus luodaan alajohtoportaiden huoltotilanneilmoitusten pohjalta.
5. Huoltotilanneilmoituksen muotona excel-taulukko.
6. Huoltotilanneilmoituksia on voitava käsitellä yksittäisissä tietokoneissa ilman verkkoyhteyksiä (Standalone ohjelma, ei client-server).
7. Testaus (kokeiluharjoituksessa) toukokuussa 2013.

Asiakasvaatimuksista (AV) ja toimintaympäristön kuvauksen pohjalta johdettiin seuraavat *ohjelmistovaatimukset* (OV):

1. Laske huoltotilanneilmoitus. (AV-1, AV-4)
2. Tallenna huoltotilanneilmoitus excel-työkirjana. (AV-1, AV-3, AV-5)
3. Laske ja esitä suorituskyky. (AV-1)
4. Tallenna suorituskyky asiakirjana. (AV-)
5. Laske ja esitä huoltotarve. (AV-2)
6. Tallenna huoltotarveraportti asiakirjana. ()
7. Laske ja esitä tukitarve. (AV-2)
8. Tallenna tukitarve excel-työkirjana. (AV-2, AV-5)
9. Muodosta tilaukset. (AV-1, AV-2)
10. Tallenna tilaukset excel-työkirjana. ()
11. Tallenna tilaukset johtamissanomana. ()
12. Tallenna tilaukset SAH 751 asiakirjana. ()

Asiakasvaatimus kuusi (AV-6) vaikutti tekniseen toteutukseen rajaamalla pois erilaiset selaimella käytettävät ulkoiset palvelinratkaisut. Asiakasvaatimus (AV-7) vaikuttaa taulukossa 1 esitettyyn toteutuksen vaiheistukseen siten, että ensimmäisessä vaiheessa suunnitellaan kokonaisuus. Lisäksi toteutetaan huoltolaskentaohjelmiston versio 1.0. Painopiste on huoltotilannelaskennassa,

koska huoltolaskennan testaus on vaiheessa kaksi (taulukko 1). Suorituskykylaskenta, huoltotarvelaskenta ja tilausten generointi toteutetaan siten, että niiden konsepti on testattavissa. Testauksen jälkeen tehdään korjaukset ja osittain toteutettujen toimintojen varsinainen toteutus.

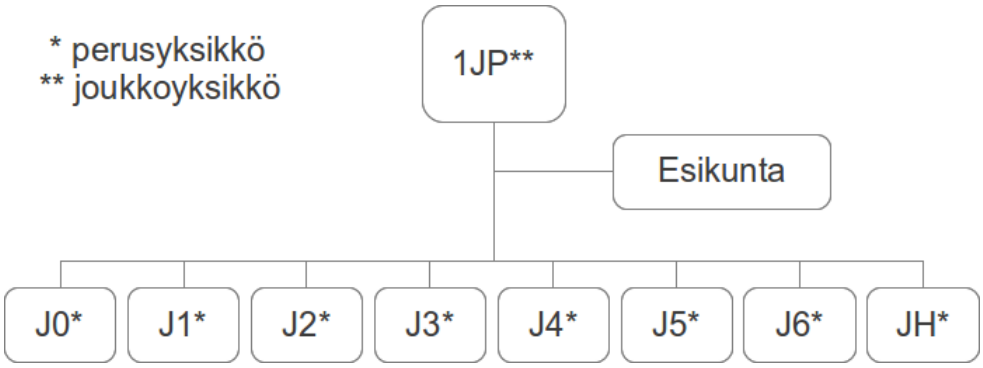
TAULUKKO 1 Huoltolaskentaohjelmiston toteutuksen päävaiheet

VAIHE	KUVAUS
1	Suunnittelu, huoltotilannelaskennan toteutus ja muiden toimintojen osittainen toteutus
2	Testaus (toukokuu 2013)
3	Korjaukset ja muiden toimintojen varsinainen toteutus (suorituskykylaskenta, huoltotarvelaskenta, tilausten generointi)

Suunnittelussa käytettyjä periaatteita olivat skaalautuvuus, tulosten käytettävyys, siirrettävyys (prosesseihin sitomattomuus) ja tulosten havainnollistaminen. Skaalautuvuudella tarkoitetaan toimintamallin soveltuvuutta käytettäväksi niin joukkuetasolla, perusyksikkötasolla ja joukkoyksikkötasollakin. Tulosten käytettävyydellä tarkoitetaan sitä, että laskennan syötteet ja tuotokset ovat käytettävissä ilman erillisiä ohjelmistoja. Siirrettävyydellä tarkoitetaan sitä, että toimintamalli toimintoineen ei ole riipuvainen muista prosesseista tai työnkuluista, ja on siten siirrettävissä tietojärjestelmästä toiseen. Tulosten havainnollistamisella tarkoitetaan excel-työkirjojen numeerisen tiedon esittäminstä siten, että tulokset ovat helpommin ymmärrettävässä muodossa. Lisäksi tulokset ovat ryhmiteltynä siten, että eri käyttäjien erilaiset tietotarpeet on huomioitu.

2.3 Käyttötapaukset

Suunnitelun taktisena kehyksenä käytettiin joukkoyksikkö, jonka periaatteellinen organisaatio esitellään kuviossa 1. Joukkoyksiöitä ovat mm. pataljoona ja patteristo. Joukkoyksikkö koostuu perusyksiköistä, joita ovat mm. komppania ja patteri. Huoltolaskentaohjelmiston kehitystä varten luotiin organisaatio, joka koostui yhdestä joukkoyksiköstä (1 jääkäripataljoonasta) ja kahdeksasta sen johtamasta perusyksiköstä. Perusyksiköt nimettiin J0-J6 ja huoltojoukko nimettiin JH.



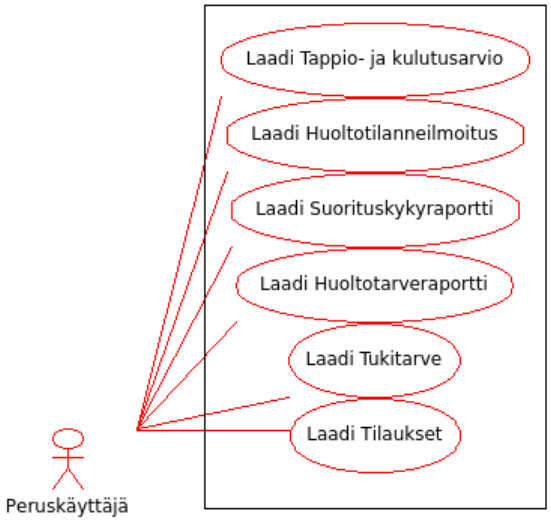
KUVIO 1 Opinnäytetyössä käytettyn joukkoyksikön organisaatio

Opinnäytetyössä tuotettavan huoltolaskentaohjelmiston (versio 1.0) käyttöympäristönä on joukkoyksikön esikunta, jossa työasemat ovat samassa lähiverkkossa. Perusyksiköt lähettävät huoltotilanneilmoituksen johtamissanoman liitetiedostona joko käytössä olevilla viestijärjestelmillä. Huoltotilanneilmoituksien lähettämisen varajärjestelmänä käytetään lähettiä, jolla on huoltotilanneilmoitus tallennettuna USB-muititikkulle.

Huoltolaskennan kehittämisen toisessa vaiheessa (taulukko 1) testataan huoltolaskentaohjelmiston ensimmäinen versio ja huoltotilanneilmoitusten rakenne. Testausharjoituksessa huoltolaskentaohjelmisto asennetaan erilliselle kannettavalle tietokoneelle, joka ei ole liitettynä joukkoyksikön lähiverkkoon. Huoltolaskentaohjelmiston ja joukkoyksikön lähiverkossa olevien työasemien väliseen tiedonsiirtoon käytetään USB-muistitikkua.

Huoltotilanneilmoitukset siirretään USB-muistitikulla työasemaan, johon huoltolaskentaohjelmisto on asennettuna. Huoltolaskentaohjelmisto lukee huoltotilanneilmoitukset USB-muistitikulta annetusta hakemistosta ja tallentaa muodostamansa huoltotilanneilmoitukset (vaje ja rivivahvuus) samaan hakemistoon.

Perusyksiköt seuraavat hallussaan olevia resursseja päivittämällä huoltotilanneilmoitukseen (Excel-työkirja) eri resurssien vajeet. Tiedot lähetetään joukkoyksikön esikuntaa käskettyinä ajankohtina tai käsketyn aikavälin mukaisesti. Esikunta muodostaa joukkoyksikön huoltotilanneilmoituksensa (vaje) rungon koostamalla ohjelmallisesti yhteen alajohtoportaiden huoltotilanneilmoitukset (vaje). Ohjelmallisesti koostetun joukkoyksikön huoltotilanneilmoituksen rungon tietosisältönä on perusyksiköiden yhteen lasketut määrävahvuudet, tavoitevahvuudet, rivivahvuudet, kumulatiiviset vajeet ja vajeprosentit sekä perusyksiköiden vajeet. Joukkoyksikön huoltotilanneilmoituksen (vaje) numeerisen tiedon analysoinnin jälkeen siihen lisätään sanallisia arvioita numeerisen tiedon rinnalle. Perusyksiköiden huoltotilanneilmoituksista muodostetaan joukkoyksikössä toinen huoltotilanneilmoitus, jossa perusyksiköiden osalta esitetään vajeen asemesta rivivahvuus. Tätä tiedostoa käytetään suorituskykyn laskennan syötteenä.



KUVIO 2 Huoltotarvelaskennan käyttötapaukset

Käyttötapauksen KT-1 (Laadi tappio- ja kulutusennuste) tavoitteena on tuottaa perusyksiköiden huoltotilanneilmoituksista uudet huoltotilanneilmoitukset, joista on vähennetty tappio- ja kulutuslaskennan mukaiset tappiot ja kulutukset. Tappio- ja kulutusennustelaskentaan osallistuu huoltopäällikkö. Toiminnon tuloehtona on, että sotapelin tulokset ovat käytettävissä. Käyttäjä avaa huoltolaskentaohjelmiston, valitsee laskentahakemiston ja käynnistää tappio- ja kulutusennustelaskennan. (Mikkonen 2011, 79-80.)

Käyttötapauksen KT-2 (Laadi huoltotilanneilmoitus) tavoitteena on tuottaa koostettu joukkoyksikön huoltotilanneilmoitus perusyksiköiden huoltotilanneilmoituksista. Koosteen laskennassa yhdistetään kaikkien perusyksiköiden huoltotilanneilmoitukset ja luodaan koko johtoportaan huoltotilanneilmoitus. Käyttötapaukseen osallistuu joukkoyksikön huoltopäällikkö. Toiminnon tuloehtona on, että perusyksiköiden huoltotilanneilmoitukset ovat saatavilla. Käyttäjä avaa ohjelmiston, valitsee laskentahakemiston ja käynnistää huoltotilannelaskennan. (Mikkonen 2011, 79-80.)

Käyttötapauksen KT-3 (Laadi suorituskykyraportti) tavoitteena on tuottaa suortituskykyraportti joukkoyksikön huoltotilanneilmoituksesta. Suorityskykylaskentaan osallistuu joukkoyksikön huoltopäällikkö. Toiminnon tuloehtona on, että huoltotilanneilmoituksen rv-versio (KT-2:n tuotos) on käytettävissä. Käyttäjä avaa ohjelmiston, valitsee laskentahakemiston ja käynnistää suorituskykylaskennan. (Mikkonen 2011, 79-80.)

Käyttötapauksen KT-4 (Laadi huoltotarveraportti) tavoitteena on tuottaa huoltotarveraportti joukkoyksikön huoltotilanneilmoituksesta (tarve). Huoltotarvelaskentaan osallistuu joukkoyksikön huoltopäällikkö ja laskennan tuloehtona on, että joukkoyksikön huoltotilanneilmoituksen tarve-versio on käytettävissä. Käyttäjä avaa ohjelmiston, valitsee laskentahakemiston ja käynnistää huoltotarvelaskennan. (Mikkonen 2011, 79-80.)

Käyttötapauksen KT-5 (Laadi tukitarve) tavoitteena on tuottaa joukoyksikön huoltotilanneilmoituksen tarve-versio. Tukitarpeen laadintaan osallistuu joukkoyksikön huoltopäällikkö. Toiminnon tuloehtona on, että huoltojoukon huoltotilanneilmoitukset ovat käytettävissä. Käyttäjä avaa huoltolaskentaohjelmiston, valitsee laskentahakemiston ja käynnistää tukitarvelaskennan. (Mikkonen 2011, 79-80.)

Käyttötapauksen KT-6 (Laadi tilaukset) tavoitteena on tuottaa tilaukset joukkoyksikön huoltotilanneilmoituksesta. Tilausten generiointiin osallistuu joukkoyksikön huoltopäällikkö. Toiminnon tuloehtona on, että huoltotarvelaskennan tuottaman huoltotilannilmoituksen tarve-version lukemat on päivitetty suorituskykyraporttien, huoltotarveraporttien sekä tappio- ja kulutusennusteiden analyysin pohjalta. (Mikkonen 2011, 79-80.)

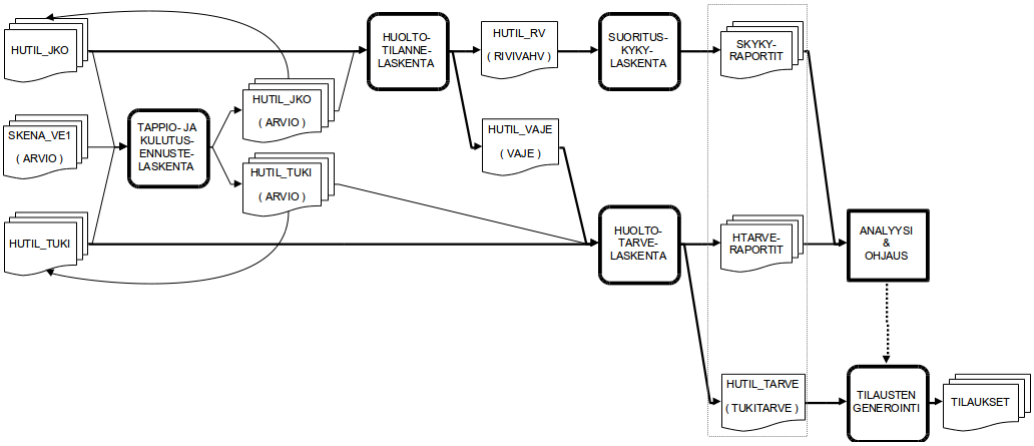
Huoltolaskentaohjelmiston käyttötapaukset ovat liitteessä yksi.

3 LASKENTAMALLIN SUUNNITTELU

3.1 Huoltolaskennan käyttötilanteet

Huoltolaskennan näkökulmasta tilanteen seuranta tukee sekä suunnittelua että toimeenpanoa. Huoltolaskennalle tunnistettiin kaksi käyttötilannetta, joissa molemmissa tilanteen seuranta on lähtötietoja tuottavana osana. Käyttötilanteita ovat suunnittelun huoltolaskenta ja toimeenpanon huoltolaskenta. Suunnittelussa tuotetaan ennusteraportteja, kun taas toimeenpanossa tuotetaan ennusteraporttejen lisäksi tilanneraportteja. Suunnitteluvaiheessa ennusteiden aikajänne on muutamasta vuorokaudesta viikkoihin, kun toimeenpanossa se on 1-2 vuorokautta. Tilanneraportit perustuvat ilmoitusajankohdan toteumaan.

Huoltolaskenta koostuu tappio- ja kulutusennuste-, huoltotilanne-, suorituskyky- ja huoltotarvelaskennoista sekä tilausten generoinnista. Laskentojen syötteet ja tuotokset on esitetty kuviossa 3. Kaikki ’HUTIL_’ alkuiset elementit (jko, tuki, vaje, rv ja tarve) ovat huoltotilanneilmoituksia. Ne eroavat toisistaan hieman syntytapansa, tiedon omistajansa, tietosisältönsä ja käyttötarkoituksensa suhteen. Huoltotilanneilmoituksen yksityiskohtaisempi rakenne esitellään luvussa 3.2.1.



KUVIO 3 Huoltolaskennan vaiheet, syötteet ja tuotokset

Huoltolaskennanohjelmiston tuotoksista huoltotilanneilmoitus eri versioineen ja tilaukset sisältävät numeerista dataa. Numeerisista huoltotilanneilmoituksista tuotetaan suorituskyky- ja huoltotarveraportit eri tarvitsijoille huomioiden kunkin erilaiset tietotarpeet. Raporteissa numeerinen aineisto esitetään ryhmiteltynä ja havainollistettuna.

Tappio- ja kulutusennustelaskennalla skenaariotiedostosta, perusyksiköiden huoltotilanneilmoituksista ja huoltojoukon huoltotilanneilmoituksesta (jko, tuki) tuotetaan uusia huoltotilanneilmoituksia, joista on vähennetty skenaarioarvion mukaan lasketut tappiot ja kulutukset (taulukossa 2). Huoltotilannelaskenta tuottaa perusyksiköiden huoltotilanneilmoituksista joukkoyksikön oman huoltotilanneilmoituksen (vaje) sekä suorituskykylaskennan syötteenä käytettävän huoltotilanneilmoituksen (rv). Suorituskykylaskenta tuottaa suorituskyvyn arviointiin tunnuslukuja ja niiden havainnollistamiseksi erilaisia graafeja, kuten pylväs- ja viivakuvaajia. Huoltotarvelaskenta tuottaa huoltotilanneilmoituksista (tuki ja vaje) huoltotarveraportin ja huoltotilanneilmoituksen (tarve). Huoltotarveraportti on samalla tavalla havainnollistettu kuin suorituskykyraportti sisältäen koosteen perusyksiköiden huoltotarpeesta suhteessa huoltojoukon kykyyn tukea. Tilaukset generoidaan huoltotilanneilmoituksesta (tarve), jonka sisältöä muokataan tilannerapottien ja ennusteraporttein analyysien pohjalta. Käyttäjä valitsee tuotettavien tilausten rakenteen, joka voi olla johtamissanoma, Excel-työkirja tai SAH751-lomake.

TAULUKKO 2 Laskentojen syötteet ja tuotokset

Syötteet	Laskenta	Tuotokset
HUTIL_JKO HUTIL_TUKI	Tappio- ja kulutusennuste	HUTIL_JKO HUTIL_TUKI
HUTIL_JKO	Huoltotilanne	HUTIL_VAJE HUTIL_RV
HUTIL_JKO HUTIL_TUKI	Huoltotarve	HUTIL_TARVE Huoltotarveraportti
HUTIL_JKO	Suorituskykyraportti	Suorituskykyraportti
HUTIL_TARVE	Tilausten generointi	JOSA, Excel tai SAH751

Huoltolaskennan lähtöaineisto, tavoitteet ja käyttö eroavat hieman käyttötilanteiden kesken. Suunnittelussa tuotettavat ennusteraportit perustuvat tappio- ja kulutusennustelaskennalla tuotettuihin huoltotilanneilmoituksiin. Tappio- ja kulutusennustelaskennan yhden laskentakierroksen aikajänne on 24 tuntia. Lisäksi edellisen laskentakierroksen tuottamia huoltotilanneilmoituksia käytetään seuraavan laskentakierroksen syötteinä (kuvio 3). Suunnittelussa ennusteraportit voidaan tuottaa jokaisella tappio- ja kulutusennustelaskennan kierroksella tai harvemmin. Toimeenpanossa tilanneraportit lasketaan

perusyksiköiden (kuviossa 3 HUTIL_JKO merkinnällä) ja huoltojoukon (tuki) todellisista, toteuman mukaisista huoltotilanneilmoituksista. Toimeenpanovaiheen ennusteraportit lasketaan samoista lähtötiedoista kuin tilanneraportitkin, mutta ne ajetaan ensin tappio- ja kulutusennustelaskennan kautta.

3.2 Huoltolaskennan tietojen kuvaus

Huoltolaskentaohjelmiston lähtötietoina käytettävien ja laskennan tuottamien huoltotilanneilmoitusten lukemiseen ja muokkaamiseen käytetään taulukkolaskentaohjemistoa. Laskennan tuottamien raporttien lukuun käytetään pdf-lukuohjelmia. Huoltolaskentaohjelmiston laskennassa käyttämä data on kokonaisuudessaan laskennan lähtötiedoissa eli skenaariotiedostoissa ja huoltotilanneilmoituksissa. Skenaaritiedostoja käytetään tappio- ja kulutusennustelaskennassa. Huoltolaskentaohjelmistossa ei ole taltioituna yhtään laskennassa käytettävää vakiota tai muuta numeerista tietoa.

3.2.1 Huoltotilanneilmoituksen tiedot ja rakenne

Huoltotilanneilmoituksia on useita eri tyyppejä. Ne ovat tietosisällöltään ja rakenteeltaan samanlaisia, mutta niillä on eri käyttötarkoitus. Koska huoltojoukko on perusyksikkö, niin se ylläpitää kahta huoltotilanneilmoitusta (jko ja tuki). Huoltotilanneilmoituksien eroja kuvataan taulukossa 3.

TAULUKKO 3 Huoltotilanneilmoituksen tyypit ja käyttötarkotukset

Lyhenne	Selite
_JKO	Perusyksiöiden laatima huoltotilanneilmoitus, jonka ne lähettävät joukkoyksikön (1JP) esikuntaan.
_TUKI	Huoltojoukon laatima huoltotilanneilmoitus, jonka se lähettää joukkoyksikön (1JP esikuntaan.
_VAJE	Joukkoyksikössä (1JP) perusyksiköiden huoltotilanneilmoituksista laskentu kooste, jossa perusyksiköiden osalta esitetään vajeet varusteittain. (Toinen nimivaihtoehto on HUTIL_JKO, koska sisältö ja käyttötarkoitus ovat samat.)
_RV	Joukkoyksikössä (1JP) perusyksiköiden huoltotilanneilmoituksista laskentu kooste, jossa perusyksiköiden osalta esitetään rivivahvuudet varusteittain.
_TARVE	Joukkoyksikön huoltojoukon täydennystarve, joka saadaan vähentämällä joukkoyksikön vaje (HUTIL_VAJE) huoltojoukon tuesta (HUTIL_TUKI).

Huoltotilanneilmoitus totetutettiin kahdeksan taulukkoa eli välilehteä sisältävänä Excel-työkirjana. Välilehdet ja niistä käytetyt lyhenteet ovat suorituskyky (skyky), henkilöstö (henk), aseet (ase), pimeätoimintavälineet (pita), viestivälineet (vval), ajoneuvot (ajon) ja kriittinen materiaali (kmat) sekä pakkausrekisteri (prek).

Henkilöstö-välilehdelle kirjataan henkilöstövaje henkilöstöryhmittäin ja henkilöstövajeen laadun mukaan. Kalustovaje kirjataan aseet, pimeätoimintavälineet, viestivälineet ja ajoneuvovälilehdille kaloustoittain. Materiaalivaje kirjataan kriittisen materiaalin välilehdelle materiaaleittain. Vajeen ilmoituksessa käytetään yleensä yksikkönä lukumäärää. Suurien kappalemäärien materiaaleissa käytetään paremmin hahmotettavia yksiköitä kuten tuliannos. Suorituskyky-välilehdelle kirjataan muiden välilehtien numeerisen tiedon sanalliset tarkennukset, näistä johdettu arvio joukon suorituskyvystä ja johtopäätöset suorituskyvyn vaikutuksesta joukon tehtävään.

Huoltotilanneilmoituksen kaikkien välilehtien ensimmäisen sarakkeen rivitunnisteet ovat samat riveillä 2-7. Riviltä 8 alkaen esitetään perusyksikön alajohtoportaat, jotka ovat erilaiset eri perusyksiköissä. Taulukossa 4 esitetyt rivitunnisteet poikkeavat rivistä kahdeksan alkaen suorituskykylaskennan syötteenä käytettävässä huoltotilanneilmoituksessa, jossa alajohtoportaiden osalta esitetään rivivahvuus vajeen asemesta.

Huoltotilanneilmoitukseen käsin syötettäviä tietoja ovat määrävahvuus, tavoitevahvuus ja perusyksiköiden joukkueiden vajetiedot. Muiden rivien arvot lasketaan käsin syötettyjen rivien tiedoista. Rivitunnisteiden välinen matemaattinen yhteys ja laskentakaavat ovat kuviossa 4.

TAULUKKO 4 Huoltotilanneilmoituksen rivitunnisteet

Lyhenne	Selite	Tietotyyppi	Laskentakaava
MV	Joukon määrävahvuuden mukainen vahvuus	int	-
TV	Tehtävä(/tavoite)vahvuus (Hnek-välilehdellä KV)	int	-
RV	Joukon hallussa ilmoitushetkellä oleva määrä	int	$RV = TV - VAJE$
VAJE	Alajohtoportaiden yhteenlaskettu vaje	int	$\sum_{i=1}^{n} vaje_i$
VAJE-%	Vaje suhteellinen osuus	double	$VAJE - \% = \frac{VAJEx100}{YHT}$
JO	Alajohtoportaan vaje	int	-

Huoltotilanneilmoituksen ase, pita, vval, ajon, muut ja kmat välilehtien rakenne on kuvattu kuviossa 5. Huoltotilanneilmoituksen kaksi ensimmäistä välilehteä poikkeavat hieman edellisistä. Suorituskykyvälilehdellä on vain käyttäjän kirjoittamaa tekstiä, joka sisältää johtopäätöksiä muiden välilehtien informaatiosta. Henkilöstövälilehden rivillä 3 on kirjavahvuus (KV), kun muissa on tavoitevahvuus (TV).

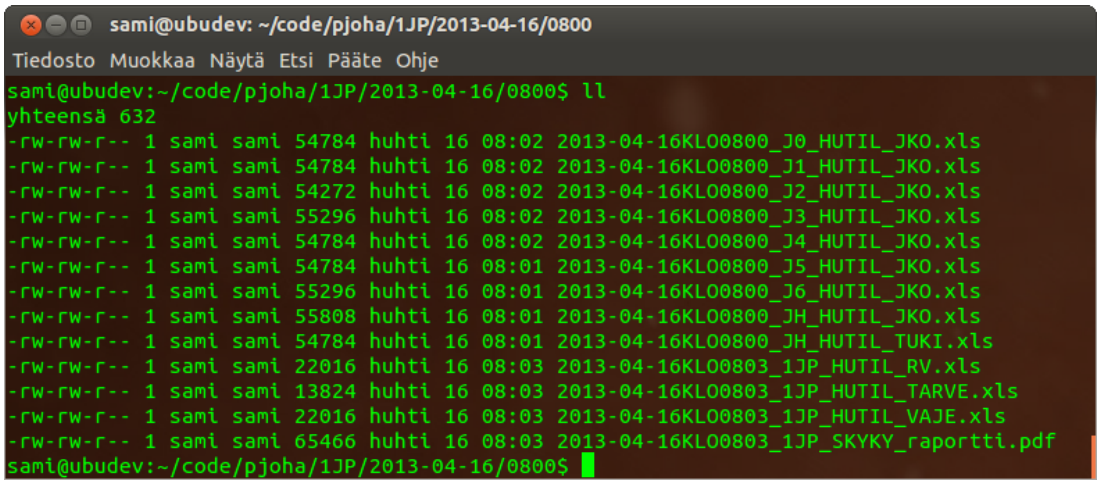
TAULUKKO 5 Huoltotilanneilmoituksen välilehtien ase, pita, vval, ajon, muut ja kmat rakenne

	A	B	C	D	E	F	G	H	I	J
1	YKSIKKO	VARUSTE A	VARUSTE B	VARUSTE C	VARUSTE D	VARUSTE E	VARUSTE F	VARUSTE G	VARUSTE H	
2	MV	90	200	20	50	90	200	20	50	
3	TV	100	25		60		250		160	
4	RV	**95**	**25**	**18**	**40**	**88**	**160**	**1**	**132**	
5	VAJE	**5**	**0**	**2**	**20**	**2**	**90**	**19**	**28**	
6	VAJE-%	5	0	10	33	2	36	95	18	
7	JOHTO				2					
8	KNTO-JA_TUKIJ			1	5	1			8	
9	1JAAKJ	2			8		24	5	6	
10	2JAAKJ	1			2	1	22	2	4	
11	3JAAKJ	1					18	7	7	
12	HJ	1		1	3		26	5	3	

3.2.2 Tiedostojen nimeäminen

Huoltotilanneilmoituksen tiedostonimi koostuu neljästä osasta, joiden välisenä erottimena käytetään _ merkkiä (alaviiva). Ensimmäisessä osassa on ilmoitusajankohdan aikaleima muodossa vvvv-kk-ppKLOhhmm. Toisessa osassa on joukon nimi kuten 1JK. Kolmannessa osassa on vakioteksti "HUTIL". Neljännessä osassa on huoltitilanneilmoituksen tarkenne, joka voi olla JKO, TUKI, VAJE, RV ja TARVE. Esimerkiksi 1JK:n huoltotilanneilmoitus 27.3.2013 klo 17:00 nimetään seuraavasti: 2013-03-27KLO1700_1JK_HUTIL_JKO.xls.

Joukkoyksikössä huoltotilanneilmoitukset tallennetaan hakemistorakenteeseen, joka alkaa joukon nimellä kuten 1JP. Hakemistorakenteen toisella tasolla hakemistot nimetään päivämäärän perusteella muodossa vvvv-kk-pp. Kolmannen tason hakemistot nimetään ilmoitusajankohdan kellonajan mukaan muodossa hhmm. Neljännelle tasolle tallennetaan perusyksiköiden huoltotilanneilmoitukset ja niistä huoltolaskennan tuotoksina lasketut koosteet ja raportit.



KUVIO 4 Esimerkki hakemiston ja tiedostojen nimeämisestä

Huoltotilanneilmoitusten nimeämisessä on tarkoituksellisesti toistoa, koska tiedoton nimestä pitää ilmetä tiedoston tallennuspaikka hakemistorakenteessa. Hakemisto muodostaa huoltolaskennan näkökulmasta kokonaisuuden, joka sisältää kaikki laskennan lähtötiedot ja laskennan tuotokset. Esimerkki 1 jääkäripataljoonan perusyksikön huoltotilanneilmoituksen tallennuspolusta: ../1JP/2013-04-15/1800/2013-04-15KLO1800_1JK_HUTIL.xls

3.2.3 Muut tiedot

Varustus-tiedostossa yhden varusteen (kalusto, materiaali, vast.) tiedot ovat yhdellä rivillä. Varusteen tietoja käytetään tappio- ja kulutusennustelaskennassa sekä huoltotarvelaskennnassa (taulukko 6).

TAULUKKO 6 Varusteen tiedot

TIETO	SELITE
id	Yksilöivä tunnus
nimi	Varusteen nimi
LT	Laukausta per tuliannos, k_{lt}
LP	Laukausta per pakkaus, k_{lp}
PL	Pakkausta per kuormalava, k_{pl}
LM	Kuormalavan paino, k_{lm}
LK	Kuormalavakerroin, k_l
MK	Massakerroin, k_m
a	Tuliannos-yhtälön vakio
b	Tuliannos-yhtälön vakio
c	Tuliannos-yhtälön vakio
d	Tuliannos-yhtälön vakio
i	Tappio-% yhtälön vakio
j	Tappio-% yhtälön vakio
k	Tappio-% yhtälön vakio
l	Tappio-% yhtälön vakio

Pakkausrekisterissä on eri varusteiden (kalusto / materiaali) pakkauksiin liittyvää tietoa (taulukko 7). Pakkausrekisterin tietoja käytetään tappio- ja kulutusennustuslaskennassa sekä huoltotarvelaskennnassa. Pakkausrekisterin sarakenimien selitteet ovat samat kuin taulukossa 6. Sarakkeiden B - E lukuarvot ovat kappalemääriä.

TAULUKKO 7 Esimerkki pakkausrekisterin tietueista

	A	B	C	D	E	F	G
1	ASE	LT	LP	PL	LM	KLK	MK
2	ASE A	50	500	50	800	0,002	1,6
3	ASE B	100	1000	40	1000	0,0025	2,5
4	ASE C	200	1500	30	1000	0,004444444	4,444444444
5	ASE D	400	4000	20	800	0,005	4

Skenaariotiedostoon tallennetaan arvio perusyksiköiden toiminnan asteesta. Skenaariotiedosto on Excel-työkirja, jonka rakenne muistuttaa huoltotilanneilmoituksen rakennetta. Kuviossa 8 perusyksiköt ovat riveillä ja toiminnan tai taistelujen astetta kuvaava lukuarvo sarakkeessa, jonka ensimmäisellä rivillä otsakkeena on päivämäärä muodossa vvvv-kk-pp. Sarakkeiden lukumäärä määrittää laskentakierrosten määrän.

TAULUKKO 8 Esimerkki skenaariotiedoista

	A	B
1	YKSIKKO	VVV-KK-PP
2	J0	4
3	J1	5
4	J2	4
5	J3	6
6	J4	5
7	J5	4
8	J6	5
9	JH	6
10	VARALLA	
11	VARALLA	
12	VARALLA	
13	VARALLA	

4 HUOLTOLASKENTAOHJELMISTON TOTEUTUS

4.1 Toimintaperiaate ja kehitysympäristö

Huoltolaskentaohjelmiston toimintaperiaattena on, että ohjelmisto lukee tiedostoista laskennan lähtöaineiston, tekee tarvittavat laskennat ja tallentaa tiedostoihin laskennan tuotokset. Lähtöaineistot ovat Excel-työkirjoja ja tuotokset ovat sekä Excel-työkirjoja että pdf-dokumentteja. Ohjelmisto käyttää lähtöaineiston lukemiseen ja tuotosten tallentamiseen samaa hakemistoa, jonka käyttäjä on ennen laskentaa valinnut. Käyttäjän valitsemat laskenta ja laskentahakemisto määrittävät mitkä tiedostot luetaan lähtötiedoiksi.

Huoltolaskentaohjelmiston kehitysympäristön valintaan vaikuttivat opinäytetyön tarkoitus ja tavoitteet sekä työn kehityksellinen luonne. Huoltotilanneilmoitukset ovat Excel-työkirjoja, joten niiden käyttö on helposti omaksuttavissa. Excel-työkirjoja voidaa käyttää, vaikka niiden laskentaan käytettyä huoltolaskentaohjelmistoa ei olisi käytettävissä. Huoltolaskentaohjelmisto toteutetaan erillisellä ohjelmointikielellä Excel-makrojen sijasta, vaikka huoltotilanneilmoitusten laskenta niilläkin onnistuu. Perusteena ovat laajemmat jatkokehitysmahdollisuudet ja riippumattomuus. Huoltolaskentaohjelmiston toteutuksen kannalta oleellisimmat ohjelmistot ovat taulukossa 9.

TAULUKKO 9 Kehitysympäristön tärkeimmät ohjelmistopaketit

OHJELMA	VERSIO
Ubuntu	12.10
python	2.7.3-0ubuntu7
ipython	0.13.1 rc3-0ubuntu1
ipython-notebook	0.13.1 rc3-0ubuntu1
python-numpy	1:1.6.2-1ubuntu1
python-matplotlib	1.1.1-1
pandas	0.8.0-2
python-qt4	4.9.3-4
Qt Designer	4.8.3
texlive	2012.20120611-4

Python on yleiskäyttöinen korkean tason ohjelmointikieli, joka on ohjelmoinnin opetuksen alkuvaiheessa suosittu kieli. Pythonilla kirjoitetun ohjelmakoodin syntaksi on helposti luettavaa. Ohjelmointikielenä Python on hyvin ilmaisuvoimaista, mikä mahdollistaa sen, että muutamalla ohjelmarivillä voidaan toteuttaa laajoja toimintoja. Pythonin ominaisuuksia ovat täysin dynaaminen tyypitys ja automaattinen muistinhallinta. Python tukee useita ohjelmoinnin lähestymistapoja, kuten olio-ohjelmointia ja proseduraalista ohjelmointia. Kuten muutkin dynaamiset ohjelmointikielet, niin myös Pythonia voidaan käyttää tulkattavana ohjelmointikielenä. Kolmansien osapuolien työkalujen avulla Pythonilla voidaan luoda itsenäisesti ajaettavia ohjelmia. (Wikipedia 2012a.)

Excel-työkirjojen tietojen lukemiseen ja tallentamiseen käytettiin xlrd-kirjastoa. Kirjastoja Numpy ja Pandas käytettiin taulukoiden muokkaamisen. Numpy tarjoaa tuen moniuloitteisille taulukoille ja matriiseille. Lisäksi se tarjoaa matemaattisia funktiota taulukoiden käsittelyyn. Pandas-kirjasto tarjoa tehokkaat ja helppokäyttöiset työkalut datan käsittelyyn ja analysointiin. Pandas-kirjasto on opinnäytetyön tärkein yksittäinen työkalu. (McKinney 2012, 4.)

Pandas-kirjastoa käytettiin raporttien sisällön visualisointiin. Taulukoiden informaatio visualisoidaan eli havainnollisetetaan erilaisilla pylväs- ja viivakuvaajilla. Visualisoinnin tuottavat raporttifunktiot toteutetaan vasta huoltolaskennan kehittämisen kolmannessa vaiheessa. Kolmannessa vaiheessa kuvaajien piirtoon käytetään lisäksi Matplotlib-kirjastoa, joka on laaja piirtokirjasto (McKinney 2012, 5). Raportit tehtiin LaTeX ladontajärjestelmällä, joka rakentuu TeX-järjestelmän päälle (Wikipedia 2012b). Latex huolehtii makrojen avulla raportin rakenteesta, kuten rivin- ja sivunjaosta sekä kuvien ja taulukoiden asettelusta. Käyttäjäystävällisyyttä lisäävät makrot, jotka huolehtivat dokumentin sisäisistä viittauksista ja sisällysluettelosta. Latex soveltuu teknisten ja tieteellisten julkaiusjen tekoon (LaTeX project team 2010).

Qt on alustariippumaton ohjelmistojen kehistysympäristö, joka soveltuu niin tekstipohjaisten kuin graafistenkin ohjelmistojen tekoon. Esimerkki Google Earth-ohjelmisto on kehitetty Qt:n avulla. Digian omistama Qt Development Frameworks kehittää Qt-ympäristöä, joka koostuu C++ -luokkakirjastosta ja alustariippumattomasta ohjelmointiympäristöstä (Wikipedia 2012c) (Digia 2012). Qt-kirjastoa voi hyödyntää myös Python ohjelmointikielellä käyttämällä Qt:n Python sidontaa, jonka nimi on PyQt. PyQt sisältää yli 400 luokka ja 6000 funktiota ja metodia. (Riverbank 2012; Wikipedia 2012d.)

IPython on vuorovaikutteinen Python ohjelmointikielen tulkki, jota käytetään päätteessä (McKinney 2012, 5). Se tarjoaa syntaksin korotuksen ja osaa täydentää komennot tabulaattorilla (McKinney 2012, 5). IPython Notebook on edellisestä tehty verkkoselaimella käytettävä versio (McKinney 2012, 72).

Notebookin avulla opeteltiin python ohjelmointikieltä, tarvittavien kirjastojen käyttöä ja luotiin tarvittajat laskenta-algoritmit. Notebook soveltui hyvin huoltolaskentaohjelmiston kehitysvaiheeseen sen vuorovaikutteisen ominaisuuden takia. Kuviossa 5 esitellään kehitysvaiheen loppuvaihetta, jossa osa toiminnallisuuksista oli jo siirretty erillisten luokkien ja funktioiden tehtäviksi.

KUVIO 5 Esimerkki IPython Notebookin käytöstä kehitysvaiheessa

IPython Notebookin toimintaperiaate on yksinkertainen. Python-ohjelmakoodi kirjoitetaan Notebookin solun input osioon, joka on merkinnän ’In [27]:’ oikealla oleva tummennettu suorakaiteen muotoinen alue (kuvio 5).
Python-ohjelmakoodin aiheuttama lopputuotos sijoitetaan solun output osioon,

joka alkaa merkinnästä ’Out [27]:’. Solun input osion viimeisen rivin komento $'group.henk[:5].plot(kind='bar')'$ tuottaa solun output osion sisällön. Sisältö koostuu kuvaajasta ja sen yläpuolella olevasta viittauksesta matplotlib-olioon. Python-ohjelmakoodissa olevan print-komennon tuotos on input ja output -osioiden välissä oleva taulukko.

Notebookilla luotiin asiakirja, johon ohjelmakoodi lisättiin useissa pienissä soluissa. Notebook soveltuu hyvin iteratiivisesti etenevään ja luonteeltaan kokeilevaan kehitykseen, koska ohjelmakoodi voidaan jakaa useisiin soluihin ja suorittaa ohjelmakoodi solu kerrallaan. Suoritetujen solujen ohjelmakoodi jää python-tulkin muistiin ja on siten muiden myöhemmin suoritettavien solujen käytössä. Solussa oleva ohjelmakoodi ei toimi, jos sen sisältämän ohjelmakoodin edellyttämä ohjelmakoodi on suorittamatta. Tästä ominaisuudesta johtuen ohjelmakoodin kehitys eteni siten, että ohjelmakoodi pilkottiin useisiin soluihin, jotka suoritettiin niin pitkälle kuin uuden kehitettävän toiminnallisuuden testaaminen edellytti. Tämä mahdollisti uuden toiminnallisuuden nopean kehittämisen ja testaamisen, koska koko ohjelmakoodia ei tarvinnut ajaa joka kerta.

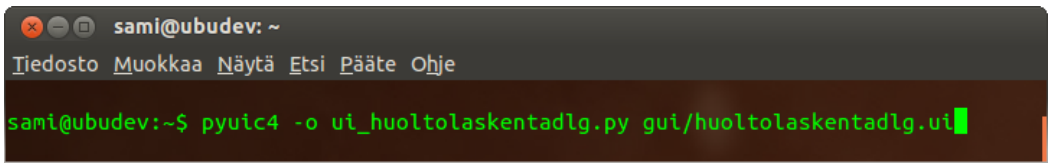
4.2 Huoltolaskentaohjelmiston käyttöliittymä

Huoltolaskentaohjelmistolle rakennettiin yksinkertainen ja helppokäyttöinen normaalin standalone-ohjelman käyttöliittymä, koska huoltolaskennan kehittämisen testausvaiheessa (5/2013) ohjelmiston testaajina ovat todelliset loppukäyttäjät. IPython Notebookilla tehty toteutus käyttö vaatii Python osaamista. Käyttöliittymän kehittämiseen ei panostettu, koska työn painopiste oli toimivien laskentafunktioiden ja -algoritmien kehittämisessä eikä loppukäyttäjälle suunnatun valmiin sovelluksen tuottamisessa.

Huoltolaskentaohjelmisto koostuu neljästä tiedostosta, joita ovat käyttöliittymä, pääohjelma, hutil.py (hutil-luokka) ja raportti.py (raportti-luokka). Pääohjelma luo huoltolaskentaohjelmiston käyttöliittymän ja reagoi käyttäjän toimiin. Huoltotilanneilmoituksiin liittyvä laskenta on hutil.py tiedostossa, joka sisältää Hutil-luokan määrittelyt. Raporttien tuottamiseen liittyvät luokat ja funktiot on koottu raportti.py -tiedostoon.

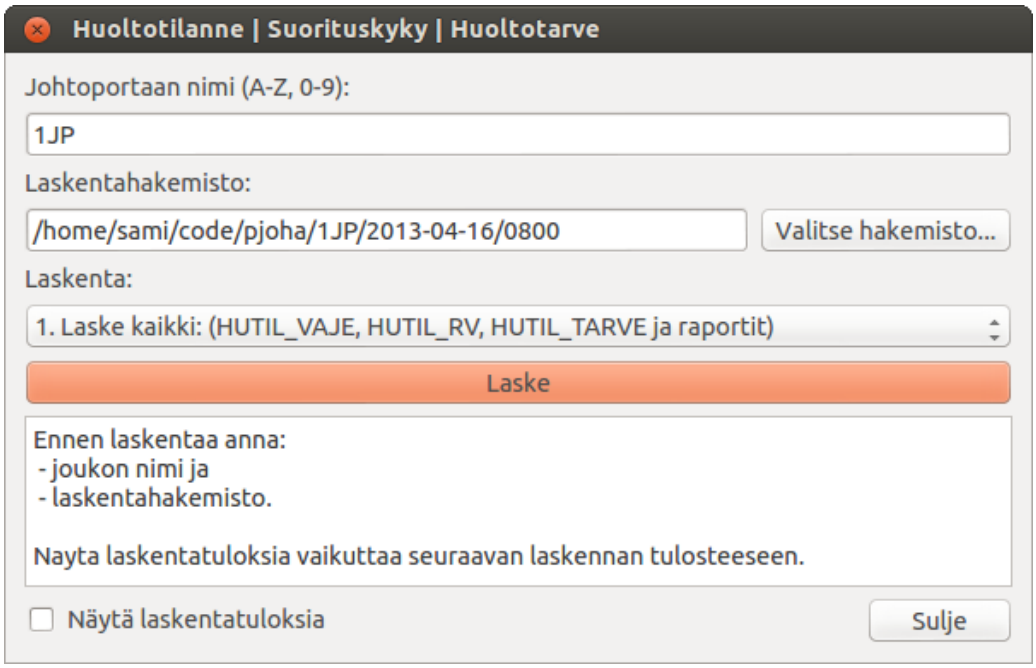
Huoltolaskentaohjelman käyttöliittymä toteutettiin Qt Designer-ohjelmalla, jonka avulla käyttöliittymän rakentaminen tapahtui graafisesti. Qt Designerilla

tehty xml-muotoinen käyttöliittymätiedosto muunnettiin päätteessä Python-ohjelmakoodiksi kuviossa 6 esitetyllä komennolla. (Summerfield 2010, 75.)



KUVIO 6 Käyttöliittymän kääntö pyuic4 ohjelmalla

Kuviossa 6 pyuic4 on ohjelman nimi. Komennolle annetaan kaksi argumenttia, joista jälkimmäinen on ohjelman pyuic4 syöttötiedosto, joka sisältää Qt Designerilla tehdyn käyttöliittymän. Ensimmäinen argumentti (-o <tiedoston nimi>) määrittää generoitavan python ohjelmakoodia sisältävän tiedoston nimen.



KUVIO 7 Huoltolaskentaohjelmiston käyttöliittymä

Käyttöliittymässä (kuviossa 7) laskentatoiminnot ovat käytettävissä johtoportaan nimen syöttämisen ja laskentahakemiston valinnan jälkeen. Käyttäjä valitsee haluamansa laskennan keskiosan valintalistasta. Laskenta käynnistyy välittömästi

valinnan jälkeen. Laske-painikeella voi käynnistää valintalistan mukaisen laskennan uudestaan. Jos käyttäjä valitsee ’Näytä laskentatuloksia’ -kohdan, niin alareunan tekstialueelle tulostetaan pieni otos laskentatuloksista laskennasta. Valinnan puuttuessa tekstialueelle tulostuu infoa laskennan kulusta (kuvio 7).

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  import sys, os, shutil
5  from PyQt4.QtCore import *
6  from PyQt4.QtGui import *
7
8  from pandas import Series, DataFrame
9  import pandas as pd
10 import numpy as np
11 from datetime import datetime
12
13 import xlrd
14 import openpyxl
15 import glob
16
17 import ui_huoltolaskentadlg
18 from hutil2 import Hutil
19 from htrv import raportti
```

KUVIO 8 Huoltolaskentaohjelmiston pääohjelman lähdekoodin alkuosa

Huoltolaskentaohjelmiston pääohjelman lähdekoodin alkuosassa tuodaan kaikki tarvittavat kirjastot ja modulit (kuvio 8). Rivillä 1 on ns. shebang-merkintä, jossa kerrotaan mistä python tulkki löytyy. Koska ohjelmakoodissa käytetään suomen kieltä, niin rivillä kaksi määritellään käytettävä merkistö. Rivien 2 ja 4 välissä olevat dokumentointiin ja tekijänoikeuksiin liittyvät tiedot on jätetty pois kuviosta 8. Pythonin peruskirjastot, kuten tiedostojen käsittely, tuodaan rivillä 4 ja graafisen käyttöliittymän edellyttämät kirjastot riveillä 5-6. Ohjelman laskennan kannalta oleelliset kirjastot tuodaan riveillä 8-11 ja Excel-tiedostojen käsittelyyn liittyvät kirjastot riveillä 13-15. Huoltolaskentaohjelmiston muut modulit tuodaan riveillä 17-19.

Huoltolaskentaohjelmiston alustamisen ohjelmakoodi on esitelty kuviossa 9. Rivillä 3 kutsutaan emoluokan rakentajaa, jonka jälkeen luodaan ja alustetaan instassimuuttujat riveillä 5-12. Sivulla 10 kuviossa 3 huoltotilannelaskennan ja huoltotarvelaskennan tuotettamille huoltotilanneilmoituksille luodaan omat oliot Hutil-luokasta kuviossa 9 riveillä 9-11. Huoltotilanneilmoitusten rakentaminen tapahtuu Hutil-luokan metodien kautta. Suorituskykyraportin rakentava olio luodaan rivillä 12. Käyttöliittymän muodostamiseen ja päivittämiseen liityvät metodit ovat riveillä 15-18.

```
1  class HuoltolaskentaDlg(QDialog,
       ui_huoltolaskentadlg.Ui_HuoltolaskentaDlg):
2
3      def __init__(self, parent=None):
4          super(HuoltolaskentaDlg, self).__init__(parent)
5          self.text = ''    # Tekstikentan sisalto
6          self.path = ''    # Tallennuspolku
7
8          # Olioiden luonti
9          self.vajeHutil = Hutil('VAJE')    # Luo olion - Vaje
10         self.riviHutil = Hutil('RV')      # Luo olion - Rivi
11         self.tarveHutil = Hutil('TARVE') # Luo olion - Tarve
12         self.skykyRapo = Raportti()
13
14         # GUI
15         self.setupUi(self)            # Luo kayttoliittyman
16         self.addLaskentaCBoxItems()  # Lisaa laskentojen nimet
17         self.muutaPushButton.setFocus(True)
18         self.updateUi()
```

KUVIO 9 Luokan HuoltolaskentaDlg alustusmetodi

Luokan rakentajassa on nähtävillä hyvin vähän käyttöliittymään liittyvää ohjelmakoodia, koska huoltolaskentaohjelmiston käyttöliittymän rakentamisesta huolehtii luokan HuoltolaskentaDlg alustusmetodin rivillä 14 kutsuma setupUi()-metodi (kuvio 9). Metodi hakee luokan HuoltolaskentaDlg argumenttien (rivillä 1) tietojen perusteella ui_huoltolaskentadlg.py tiedoston samasta työhakemistosta ja rakentaa sen perusteella käyttöliittymän.

Kuviossa 7 esitelty huoltolaskentaohjelmiston käyttöliittymän valintalistan sisältö luetaan tiedostosta, koska käytettävien laskentojen määrä ja nimet voivat muuttua kehitystyön edetessä. Valintalistan tietojen luku tiedostosta ja liittäminen käyttöliittymän valintalistan tietosisällöksi on esitetty kuviossa 10. Rivillä 12 poistetaan rivin lopusta viimeinen merkki (rivinvaihto) ennen rivin lisäämistä valintalistaan. Jos tiedoston lukemisessa tapahtuu virhe, niin

virhekäsittelyn try-except-finally -rakenne tulostaa virheilmoituksen käyttöliittymän tekstialueelle ja päätteeseen (rivit 15-16).

```
1  def addLaskentaCBoxItems(self):
2     '''Lisaa laskentaCBox:n sisallon (nimet)'''
3
4     try:
5        fh = open('gui/lask ennat', 'r')
6
7        if fh:
8           lista = fh.readlines()
9
10          for rivi in lista:
11             if rivi[0] != '#':
12                self.laskentaCBox.addItem(rivi[:-1])
13
14    except IOError as e:
15       self.text = unicode('%s\n' % e)
16       print '%s\n' % e
17
18    finally:
19       self.updateUi()
```

KUVIO 10 Huoltolaskentaohjelmiston addLaskentaCBoxItems-metodi

Käyttöliittymän widgettien päivittäminen esitellään kuviossa 11. Metodia updateUi kutsutaan aina käyttäjän tekeminen käyttöliittymävalintojen yhteydessä, mikä takaa käyttöliittymän säilymisen ehjänä kaikissa tilanteissa.

```
1     def updateUi(self):
2        '''Paivittaa kayttoliittyman widgetit'''
3
4        # Tekstikenttien tekstit
5        self.hakemistoLineEdit.setText(self.path)
6        self.textEdit.setText(self.text)
7
8        # Widgettien 'enablointi'.
9        if self.path == '':
10          self.laskentaCBox.setEnabled(False)
11          self.laskePushButton.setEnabled(False)
12
13       else:
14          self.laskentaCBox.setEnabled(True)
15          self.laskePushButton.setEnabled(True)
16          self.laskePushButton.setFocus(True)
17
18       if self.laskentaCBox.count() == 0:
19          self.laskePushButton.setEnabled(False)
20          self.hakemistoLineEdit.setEnabled(False)
21          self.muutaPushButton.setEnabled(False)
22          self.infoCheckBox.setEnabled(False)
```

KUVIO 11 Huoltolaskentaohjelmiston updateUi-metodi

Menettely mukailee MVC ohjelmointiparadigmaa etäisesti, koska tiedot (M), ja käyttöliittymä (V) on eroteltu toisistaan. Käyttöliittymän päivittäminen alkaa tekstikenttien sisällön päivittämisellä. Laskentahakemiston sisätö luetaan self.path-muuttujasta ja tekstialueen self.text-muuttujasta (rivit 5-6). Ohjelmiston eri vaiheissa funktiot tallentavat tekstikentissä esitettävää sisältöä em. muuttujiin. Laskennan valintalista ja laskenta-painike eivät ole käytettävissä, jos laskentahakemistoa ei ole valittuna (rivit 10-11). Käyttöliittymän kaikki widgetit, pl. Cancel ja Ok -painikkeet, kytketään pois käytöstä riveillä 18-22.

Huoltolaskentaohjelmiston käynnistyksen jälkeen käyttäjän on ensimmäiseksi valittava hakemisto laskentaa varten. Laskentahakemisto sisältää laskennan lähtötietona käytettävät huoltotilanneilmoitukset (vaje ja tuki). Kuvion 12 rivillä 5 on hakemiston valintadialogin (rivi 8) otsikkopalkissa esitettävä teksti. Käyttäjän valitseman hakemiston hakemistopolku tallennetaan rivillä 8 self.path-muuttujaan.

```
1  def valitseHakemisto(self):
2      '''Tallentaa kayttajan valinnan mukaisen
           hakemistopolun.'''
3
4      # QFileDialogi-funktion argumenttien arvot.
5      otsikko = 'Valitse laskentahakemisto!'
6
7      # Poimi Hutil tiedostojen tallennushakemisto
8      dn = QFileDialog.getExistingDirectory(self, otsikko,
           '/home/')
9
10     # Hakemistopolun tallennus
11     self.path = unicode(dn)
12
13     # Kayttoliittyman paivitys
14     self.updateUi()
```

KUVIO 12 Huoltolaskentaohjelmiston valitseHakemisto-metodi

Huoltolaskentaohjelmiston laskenta on keskitetty laske-metodiin, joka esitellään kuviosa 13. Rivillä 5 luetaan käyttöliittymästä käyttäjän valitsemaa laskentaa vastaava indeksi, jota käytetään if-elif -rakenteen vertailuarvona laskentafunktion valinnassa. Rivillä 6 luetaan käyttöliittymästä infoCheckBoxin arvo, jota käytetään valittavan laskentafunktiokutsun toisena parametrina. Info muuttujan arvo vaikuttaa laskentafunktioissa siihen, mitä ne tulostavat käyttöliittymän tekstialueelle.

```
1  def laske(self):
2     '''SLOT - tuottaa laskentatiedostot.'''
3
4     # Kayttajan valintojen poiminta
5     indeksi = self.laskentaCBox.currentIndex()
6     info = self.infoCheckBox.isChecked()
7
8     ### Valinnan mukaisen laskennan suorittaminen
9
10    # Kaikki laskennat
11    if indeksi == 0:
12       text = self.vajeHutil.laadiHutil(self.path, info)
13       text += self.riviHutil.laadiHutil(self.path, info)
14       text += self.tarveHutil.laadiHutil(self.path, info)
15       text += self.skykyRapo.laadiSkyky(self.path,
             self.jopo, self.hjko, info)
16       text += raportti.laskeSkykyraportti(self)
17
18    elif indeksi == 1:
19       text = 'Tappio- ja kulutusennustelaskentaa ei ole
             toteutettu!'
20
21    # Huoltotilanne (HUTIL_VAJE)
22    elif indeksi == 2:
23       text = self.vajeHutil.laadiHutil(self.path, info)
24
25    # Huoltotilanne (HUTIL_RV)
26    elif indeksi == 3:
27       text = self.riviHutil.laadiHutil(self.path, info)
28
29    # Huoltotarve (HUTIL_TARVE)
30    elif indeksi == 4:
31       text = self.tarveHutil.laadiHutil(self.path, info)
32
33    # Suorituskyky
34    elif indeksi == 5:
35       text = raportti.laskeSkykyraportti(self)
36       text = self.skykyRapo.laadiSkyky(self.path, self.jopo,
             self.hjko, info)
37
38    # Tilausten generointi
39    elif indeksi == 6:
40       text = 'Tilausten generointia ei ole toteutettu!'
41
42    self.text = unicode(text)
43
44    # Kayttoliittyman paivitys
45    self.updateUi()
```

KUVIO 13 Luokan HuoltolaskentaDlg:n laske-metodi

Kuviossa 13 riveillä 9-33 kutsutaan käyttäjän valinnan mukaista laskentafunktiota. Laskentafunktioiden paluuarvot kootaan text-muuttujaan, joka lopuksi tallennetaan self.text-muuttujaan. Työn kehityksellinen luonne (pilotti) huomioitiin käyttöliittymän toteutukssa siten, että uusien laskentafunktioden lisääminen ei vaikuta käyttöliittymän ulkoasuun. Uuden laskennan nimi tai kuvaus lisätään laskennat-tiedostoon (kuvio 10) omalle riville, josta se luetaan käyttöliittymän valintalistaan. Uuden laskentafunktion kutsu liitetään laske-metodiin (kuvio 13).

Huoltolaskentaohjelmiston käynnistävä main-funktio esitellään kuviossa 14. Rivillä 3 luodaan PyQt-ohjelma, jolle välitetään ohjelman argumentit. Rivillä 4 luodaan HuoltolaskentaDlg-luokan mukainen ui-olio, joka tuodaan rivin 4 show() funktiolla näkyväksi. PyQt-ohjelman tapahtumasilmukan käynnistys ja ohjelman lopetus ovat rivillä 9.

```
1  def main():
2
3      # Luodaan PyQt ohjelma
4      app = QApplication(sys.argv)
5
6      # Luodaan HuoltolaskentaDlg:n mukainen kayttoliiittyma
7      ui = HuoltolaskentaDlg()
8
9      # Tehdaan kayttoliittyma nakyvaksi
10     ui.show()
11
12     # Ohjelman lopetus
13     sys.exit(app.exec_())
14
15 if __name__ == '__main__':
16     main()
```

KUVIO 14 Huoltolaskentaohjelmiston main-metodi

4.3 Huoltotilanneilmoitusten tuottaminen

Huoltotilannelaskennan rakentaminen tapahtui kahdessa vaiheessa. Ensimmäisessä vaiheessa rakennettiin useiden huoltotilanneilmoitusten tietojen yhdistäminen. Toisessa vaiheessa liitettiin huoltotilannelaskenta huoltolaskentaohjelmiston osaksi. IPython Notebook:lla tuotettu ohjelmakoodia piti muokata ennen kuin se oli lisättävissä huoltolaskentaohjelmiston osaksi. Ohjelmakoodiin lisättiin piirteitä olio-ohjelmoinnista, mikä mahdollistaa jatkossa paremman ohjelmakoodin osien uudelleen käytettävyyden, laajennettavuuden ja ylläpidettävyyden.

Huoltotilannelaskenta on tilanneraporttien tuottamisen ensimmäinen vaihe kuviossa 3 (s. 10). Huoltotilannelaskennan syötteinä ovat perusyksiöiden huoltotilanneilmoitukset, joista huoltotilannelaskenta tuottaa kaksi huoltotilanneilmoitusta. Ensimmäinen on joukkoyksikön oma huoltotilanneilmoitus (HUTIL_VAJE), joka on tietosisällöltään vastaava laskennan lähtötietoina käytettyjen perusyksiköiden huoltoilmoitusten kanssa. Toinen on suorituskykylaskennan syötteenä käytettävä versio huoltotilanneilmoituksesta (HUTIL_RV), jonka tietosisältö eroaa edellisestä vain perusyksiköistä esitettävän tiedon osalta. Kuviossa 5 rivistä seitsemän alkaen esitetäänkin rivivahvuustieto vajetiedon asemesta.

4.3.1 Hutil-luokka

Huoltotilanneilmoituksia tuottavat (kuvio 3, s. 10) huoltotilannelaskenta (vaje, rv), huoltotarvelaskenta (tarve) sekä tappio- ja kulutusennustelaskenta. Niitä tuottava laskenta on kokonaisuudessaan Hutil-luokan laadiHutil-metodissa, koska kaikilla huoltotilanneilmoituksilla on sama perusrakenne. Huoltolaskentaohjelmiston käynnistyessä HuoltolaskentaDlg-luokan rakentaja luo kolme Hutil-luokan oliota yksi kutakin laskennalla tuotettavaa huoltotilanneilmoituksen versiota kohden (kuvio 15). HuoltolaskentaDlg-luokan rakentaja esitellään kuviossa 9 (s. 24).

```
1    # Olioiden luonti (HUTIL, SKYKY JA TARVE)
2    self.vajeHutil = Hutil('VAJE')   # Vaje
3    self.riviHutil = Hutil('RV')    # Rivivahvuus
4    self.tarvehutil = Hutil('Tarve')   # Tarve
```

KUVIO 15 Hutil-luokan olioiden luonti

Hutil-luokan olion tyyppi annetaan olion luonnin yhteydessä parametrina (kuviot 15 ja 16). Huoltolaskentaohjelmiston pääohjelman eli luokan huoltolaskentaDlg laske()-metodista kutsutaan laadiHutil-metodia riveillä 10-12, 17, 19, ja 21 (kuvio 13, s. 27). Näistä rivit 12 ja 21 liittyvät huoltotarvelaskentaan. Kuviossa 17 on esimerkki Hutil-luokan laadiHutil()-metodin kutsusta.

```
1 class Hutil:
2
3     def __init__(self, tyyppi = None):
4         self.tyyppi = tyyppi
5         self.sheets = []   # Excel-tiedoston valilehtien nimet
6         self.units = {}    # Sailio joukolle unit = {}
7         self.summary = {} # Kooste
```

KUVIO 16 Luokan Hutil alustusmetodi

```
1     # Huoltotilanneilmoitus (HUTIL_VAJE)
2     elif indeksi == 2:
3         text = self.vajeHutil.laadiHutil(self.path, info)
```

KUVIO 17 Esimerkki luokan Hutil laadiHutil-metodin kutsusta

Luokan Hutil rakentaja (kuvio 16 rivit 4-9) luo huoltotilannelaskennassa tarvittavat instanssimuuttujat. Huoltotilanneilmoitusten tyyppejä ovat jko, tuki, vaje, rv ja tarve. Tyyppi vaikuttaa mm. ohjelman haaratumisiin.

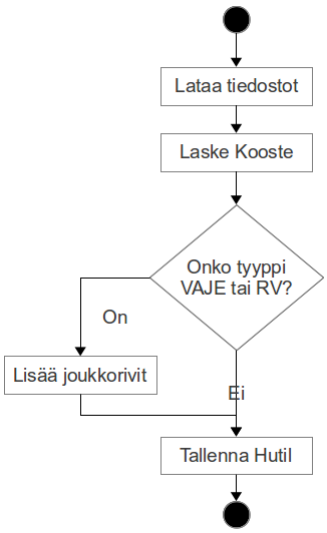
Huoltotilanneilmoituksen välilehtien nimet tallennetaan lista-tyyppiseen instassimuuttujaan sheets. Perusyksiköiden huoltotilanneilmoituksien tietosisältö kopioidaan unit-sanakirjaan, joka edelleen taltioidaan units-sanakirjaan. Laskennan tuotoksena syntyvän huoltotilanneilmoitusten koosteen (vaje) tietojen tallentamista varten luodaan summary-sanakirja.

Sanakirjat ovat tietorakenteita, jotka koostuvat avain-arvo -pareista. Tietoa haetaan ja tallennetaan avaimien avulla. Sanakirjan ’unit’ avaimina käytetään huoltotilanneilmoituksen välilehtien nimiä ja arvoina DataFrame-olioita, joihin on kopioitu vastaavien välilehtien tietosisältö. DataFrame on perusrakenteeltaan 2-uloitteinen taulukko. Välilehtien nimiä ovat mm. ’ase’ ja ’ajon’ (esiteltiin luvussa 3.2.1). Sanakirjan ’units’ avaimina käytetään tiedostojen nimistä poimittuja joukkojen nimiä ja arvoina unit-sanakirjoja. (Kasurinen 2009, s. 131-139.)

```
1    units = {'1JK' : {'ase' : df_ase, 'ajon' : df_ajon},
2             '2JK' : {'ase' : df_ase, 'ajon' : df_ajon}}
```

KUVIO 18 Esimerkki Pythonin sisäkkäisistä sanakirjoista

Kuvion 19 vuokaavio esittelee huoltotilanneilmoitusten eri versioiden koostamisen vaiheet. Huoltotarvelaskennan tuotoksena olevan huoltotilanneilmoituksen (tarve) laskenta on yksinkertaisempi, koska sen tietorakenteesta puuttuvat perusyksiköiden rivit (kuvio 5, s. 14). Sen sisältönä ovat ainoastaan Excel-työkirjan rivit 1-6 (nimike, mv, tv, rv, vaje ja vaje-%).



KUVIO 19 Huoltotilannelaskennan kulku

Huoltotilanneilmoitus rakennetaan luokan Hutil-luokan laadiHutil-metodissa kutsumalla useampaa funktiota, jotka vastaavat kuvion 19 vuokaaviota. Paremman hallittavuuden vuoksi laadiHutil-metodi on pilkottu useampaan funktioon, kuin pelkkä toistettavien osien modulointi olisi edellyttänyt (kuvio 20). Seuraavissa kappaleissa esitellään huoltotilannelaskennan neljä vaihetta ensin loppukäyttäjän näkökulmasta ja sen jälkeen ohjelmistoteknisestä näkökulmasta. Jälkimmäisessä korostetaan pandas-kirjaston ja Excel-kirjaston osia.

```
1     def laadiHutil(self, path, info):
2         '''Laskee Excel-tiedostot: HUTIL_VAJE, HUTIL_RV ja
              HUTIL_TARVE'''
3
4         ### Kaikki tyyppit
5
6         # Luetaan HUTIL-tiedostot
7         self.lueTiedostotHutil(path)
8
9         # Laskee rivien 1-5 tiedot
10        self.laskeKooste()
11
12
13        ### Vain tyyppit: VAJE ja RV
14
15        # Lisataan joukkorivit
16        if self.tyyppi == 'VAJE' or self.tyyppi == 'RV':
17            self.lisaaJoukkorivit()
18
19
20        ### Kaikki tyyppit
21
22        # Tallennetaan kooste
23        self.tallennaHutil(path)
24
25        # Muodostetaan kayttoliittymaan tulostettava ote
26        text = self.myToString(info)
27
28        return text
```

KUVIO 20 Hutil-luokan laadiHutil-metodi

Ensimmäisessä vaiheessa (kuvio 20 rivi 5) lueTiedostotHutil-funktio lukee käyttäjän valitsemasta laskentahakemistosta perusyksiköiden huoltotilanneilmoitukset ja taltioi niiden sisällön pythonin sarjallisiin muuttujiin, joista käytetään nimeä sanakirja (eng. dictionary).

Aluksi olion tyypin perusteella luodaan maski, jota käytetään tiedostojen valitsemiseen laskentahakemistosta. Maskia vastaavien tiedostojen sisällöt luetaan DataFrame-olioihin kahdessa sisäkkäisessä ohjelmasilmukassa. Joukon nimi poimitaan tiedoston nimestä ja sitä käytetään avaimena, kun joukon tiedot sisältävä olio tallennetaan units-sanakirjaan. Olion vajeHutil parseXls-funktio (kuvio 21) palauttaa parametrien mukaisen tiedoston välilehden sisällöstä luomansa DataFrame-olion. DataFrame-olion taulukossa rivien indeksinä käytetään Excel-välilehden sarakkeen A tietoja, antamalla toisena olevan 'index_col'-parametrin arvoksi nolla. Rivillä seitsemän poistetaan DataFrame-olion talukosta rivit alkaen kuudennesta rivistä. Metodi fillna täyttää DataFrame-olion taulukon tyhjän solun ylemmän rivin vastaavan sarakkeen arvolla. Huoltotilanneilmoituksissa rivin kolme ('TV') arvo voi olla tyhjä, joilloin laskennassa käytetään rivin 2 ('MV') arvoa.

```
def lueTiedostotHutil(self, path):
    '''Lukee ja tallentaa tiedostojen tiedot valilehdittain'''

    ### Luodaan tiedostojen lukemisen maski

    # maskin alkuosa
    fileMask = path + '/*HUTIL'

    # Maskin loppuosa, kun tyyppi on VAJE tai RV
    if self.tyyppi == 'VAJE' or self.tyyppi == 'RV':
        fileMask += '_JKO.xls'
        # Luetaan maskin mukaiset tiedosto
        self.lueTiedostot(fileMask)

    # Maskin loppuosa, kun tyyppi on TARVE
    elif self.tyyppi == 'TARVE':
        # HUTIL_Tuki (jaettava materiaali)
        fileMask += '_TUKI.xls'
        self.lueTiedostot(fileMask)

        # Hutil_Vaje (materiaali vaje)
        fileMask = path + '/*HUTIL_VAJE.xls'
        self.lueTiedostot(fileMask)


def lueTiedostot(self, fileMask):
    '''Poimii silmukassa maskiin sopivat tiedostot'''

    for fh in glob.glob(fileMask):
        xls = pd.ExcelFile(fh)  # Luetaan xls tiedosto
        self.sheets = xls.sheet_names # Tallen. taulukkonimet

        unitname = self.poimiJoukonNimi(fh) # Poimi joukon nimi
        unit0 = {}              # Valiaikainen muuttuja (dict)

        for sheet in self.sheets:         # Luetaan taulukot
            if sheet != 'prek':
                unit0[sheet] = self.parseXls(xls, sheet)

        self.units[unitname] = unit0 # Lisataan joukkolistaan


def poimiJoukonNimi(self, fh):
    '''Poimitaan joukon nimi tiedoston nimesta'''

    parts = fh.split('_') # Pilkotaan polku
    unitname = parts[1:2] # poimitaan toinen kentta
    return unitname[0]


def parseXls(self, xls, sheetName):
    '''Parse xls'''

    # Luetaan taulukot
    parsedS = xls.parse(sheetName, index_col=0, na_values=[0])

    # Taydennetaan puuttuva data; puuttuvat taydennetaan
       edellisella
    parsedS = parsedSheet.ix[:5].fillna(method='pad')

    return parsedSheet
```

KUVIO 21 Luokan Hutil tiedostojen latausfunktiot

Toisessa vaiheessa joukkoyksikön huoltotilanneilmoituksen eri välilehtien riveille 2-6 tuotetaan myöhemmin tallennettavat sisällöt (kuvio 22). Ensin luodaan summary instanssimuuttujalle huoltoilmoituksen mukainen rakenne, ja sen sisältö nollataan. Kunkin perusyksikön taulukoiden tiedot lisätään silmukassa summaryn taulukoihin, jos huoltotilanneilmoituksen tyyppi on vaje tai rv. Tyypin ollessa tarve, joukkoyksikön huoltojoukon tiedoista vähennetään joukkoyksikön kumulatiivinen vaje, eli tyyppiä vaje olevan huoltotilanneilmoituksen tiedot.

Lopuksi joukkoyksikön huoltotilanneilmoituksen riville kuusi (VAJE-%) lasketaan tuleva lukuarvo. Lukuarvo lasketaan rivien kolme (RV) ja neljä (TV) arvoista. Huoltotilanneilmoituksen henk-välilehden vajeprosentin laskennalle laskeVajeprosentti-funktiossa on oma laskenta, koska henk-välilehden rakenne poikkeaa muiden välilehtien rakenteesta.

```
def laskeKooste(self):
    '''Funktio lisaa silmukassa huoltotilanneilmoitusten
       valilehtien sisallot tuotettavan
       huoltotilanneilmoituksen sisaltoon. Taulukoiden
       vastinsolut lasketaan yhteen.'''

    # Luodaan tuotettava huoltotilanneilmoituksen tuottavat
       DataFrame-oliot, joiden taulukot nollataan.
    self.luoHutil()

    if self.tyyppi == 'VAJE' or self.tyyppi == 'RV':

        ### Lisataan kumulatiivisesti joukkojen tiedot

        # key=un=joukonimi, value=uv=joukon olio
        for un, uv in self.units.iteritems():

            # sn=valilehdennimi, sv=valilehden olio
            for sn, sv in uv.iteritems():

                # lisataan koosteen arvoihin. Jos tyhja nolla (0)
                self.summary[sn] = self.summary[sn].add(sv,
                    fill_value=0)

    elif self.tyyppi == 'TARVE':

        ### Valilehdittain laskentaan: TARVE = TUKI - VAJE
        for k in self.sheets:
            self.summary[k] = self.units['HK'][k] -
                (self.units['TSTOS'][k])

    # Laskentaan koosteen vajeprosentit.
    self.laskeVajeprosentti()
```

KUVIO 22 Luokan Hutil koosteen laskentafunktiot

Kolmannessa vaiheessa (kuvio 23) lisaaJoukkorivit-funktio lisää perusyksiköiden tiedot joukkoyksikön huoltotilanneilmoituksen kaikkiin välilehtiin. Lisäyksen jälkeen rivit laitetaan aakosjärjestykseen. Perusyksiköiden tiedot sijaitsevat huoltotilanneilmoituksessa rivistä seitsemän alkaen.

```
def lisaaJoukkorivit(self):
    # Kaannetaan taulukko
    self.transposeSheets()

    # Lisataan joukkojen tiedot
    for unit, value in self.units.iteritems():
        for shname, shobj in self.summary.iteritems():
            shobj[unit] = value[shname].ix[self.tyyppi] #VAJE/RV

    # Kaannetaan taulukko (palautus alkuperaiseen asentoon)
    self.transposeSheets()

    # Indeksoidaan DataFrame uudestaan
    self.reindexKooste()


def transponseSheets(self):
    '''Kaannetaan taulukot'''
    for key, value in self.summary.iteritems():
        self.summary[key] = self.summary[key].T


def reindexKooste(self): # ''' DataFrame:n
    Uudelleenindeksointi'''
    # Listamuutujien alustukset
    rivitHenk = ['MV','KV','RV','VAJE','VAJE-%']
    rivitMuut = ['MV','TV','RV','VAJE','VAJE-%']
    rivitJko = []

    # Poimii joukkojen nimet units-sanakirjasta ja lisaa ne
        rivitJko muuttujaan
    for key, value in self.units.iteritems():
        rivitJko.append(key)

    # Jarjestaa aakkosjarjestykseen
    rivitJko.sort()
    for item in rivitJko:
        rivitHenk.append(item)
        rivitMuut.append(item)

    # Uudelleenindeksoi kaikki taulukoiden indeksit (rivit)
    for key, value in self.summary.iteritems():
        if key == 'henk':
            self.summary[key] =
                self.summary[key].reindex([rivitHenk])
        else:
            self.summary[key] =
                self.summary[key].reindex([rivitMuut])
```

KUVIO 23 Luokan Hutil joukkorivien lisäysfunktiot

Lisättävä tieto on joko vaje tai rivivahvuus sen mukaan onko tyyppi muuttujan arvo 'VAJE' vai 'RV'. Ennen perusyksiköiden tietojen lisäämistä tehdään self.summaryn taulukoille transpoosi, eli taulukon rivit muutetaan sarakkeiksi ja

sarakkeet riveiksi (kuvio 23). Transpoosin jälkeen silmukassa lisätään summaryn taulukoihin uusina sarakkeina units-sanakirjan joukkojen tiedot. Perusyksiköiden tietojen lisäämisen jälkeen summaryn taulukoiden alkuperäinen muoto palautetaan tekemällä transpoosi uudelleen. Transpoosi tehdään, koska ei löydetty muuta keinoa lisätä rivejä taulukkoon.

Neljännessä vaiheessa (kuvio 24) tallennetaan DateFrame-olioiden sisällöt Excel-työkirjaan. Joukkoyksikön huoltotilanneilmoituksen aiemmat versiot poistetaan laskentahakemistosta. Poistolla varmistetaan, että laskentahakemistossa on vain yksi versio kustakin laskennan tuottamasta huoltoitilanneilmoituksesta. Aiempi versio tiedostosta pitää poistaa ohjelmallisesti, koska tiedoston nimen sisältämän aikaleiman vuoksi uusi tiedosto ei korvaa vanhaa (luku 3.2.2). Tiedoston nimi muodostetaan aikaleimasta, vakiotekstistä (HUTIL), käyttöliittymässä annetusta johtoportaan nimestä ja tarkentimesta.

```
def tallennaHutil(self, path, jopo, hjko, info):
    '''Tallentaa tex-tiedoston'''

    ### Tallennuksen osavaiheiden funktioiden kutsut

    # Poistetaan tiedoston aiemmat versiot
    self.poistaVanhaRaportti(path)

    # Muodostetaan tiedostonimen aikaleima
    aikaleima = self.teeAikaleima()

    # Muodostetaan tiedostonimi
    tiedostonimi = self.teeTiedostonimi(aikaleima, jopo)

    ### Tallennetaan tiedostoon

    # Luodaan eraanlainen 'kahva'
    writer = pd.ExcelWriter(tiedostonimi)

    # Lisataan writeriin, jos valilehden nimi ei ole 'prek'
    for sheet in self.sheets:
        if sheet != 'prek':

            self.summary[sheet].to_excel(writer,
                sheet_name=sheet)

    # Kirjoitus levylle
    writer.save()
```

KUVIO 24 Luokan Hutil tallennusfunktiot

Viimeisessä vaiheessa (kuvio 25 rivi 17) myToString()-funktio palauttaa käyttöliittymän alaosan tekstialueelle tulostettavan tekstin. Funktio koostaa kaikkien taulukoiden sisällöstä otteen. Olion tyyppi määrittää, kuviossa 25 riveillä 7-13, taulukoista otteeseen poimittavan rivin. Funktio palauttaa otteen, jos käyttäjä on käyttöliittymässä valinnut 'Näytä laskentatuloksia'. Oletuksena funktio palauttaa rivillä viisi rakennetun tiedotusviestin.

```
1    def myToString(self, info):
2       '''Palauttaa kayttoliittymaan tulostettavan tekstin.'''
3
4       tyyppi = unicode(self.tyyppi)
5       text = 'HUTIL_' + tyyppi + ' - Laskenta onnistui!\n'
6
7       if info == True:
8          if self.tyyppi == 'VAJE':
9             ra = 3; rl = 4
10         elif self.tyyppi == 'RV':
11            ra = 2; rl = 3
12         elif self.tyyppi == 'TARVE':
13            ra = 1; rl = 2
14         text = unicode(self.summary['henk'].ix[ra:rl,:4]) +
              '\n'
15         # ...Poistettu muut valilehdet
16         text += unicode(self.summary['kmat'].ix[ra:rl,:2])
              + '\n'
17
18      return unicode(text)
```

KUVIO 25 Luokan Hutil myToString-metodi

4.3.2 Tappio- ja kulutusennustelaskenta

Tappio- ja kulutusennustelaskentaa käytetään suunnittelussa ja toimeenpanossa. Huoltolaskentaohjelmistoon toiminto lisätään kolmannessa vaiheessa.

Laskennan syötteenä käytetään huoltotilanneilmoituksia (jko ja tuki) ja skenaariotiedostoa. Suunnitteluvaiheessa laskennan syötteenä käytettävät huoltotilanneilmoitukset ovat suunnittelijoiden arvioon perustuvia. Laskennan tuotoksena on uusi huoltotilanneilmoitus, josta on vähennetty tappiot ja kulutukset. Tappioiden ja kulutuksen laskennassa käytetään skenaariotiedoston arvoja.

Syötteenä käytetystä huoltotilanneilmoituksesta vähennetään skenaariotiedoston mukaiset tappiot ja kulutukset. Tuloksena on uusi huoltotilanneilmoitus.

Tässä luvussa esitettävät kaavat on johdettu täydennysoppaan ja

pakkausrekisterin tiedoista laskentafunktioiden ja edelleen laskentakirjaston pohjaksi. Kulutuslaskennan avulla laskentaa eri varusteiden kulutusta aikayksikköä, yleensä vuorokautta, kohti. Kulutuslaskenta perustuu oletettuun taistelun kiivauteen, joka muutetaan yhtälön 1 kaltaisella yhtälöllä tuliannoksiksi. Yhtälön (1) kertoimet ovat varustekohtaisia, ja niitä ylläpidetään varustus-tiedostossa (taulukko 7, s. 16).

$$T = ax^3 + bx^2 + cx + d \tag{1}$$

Laskennan välivaiheiden vähentämiseksi ensimmäisessä vaiheessa lasketaan varusteiden pakkaustiedosta kuormalava- ja massakertoimet. Näitä varustekohtaisia kertoimia käytetään laskennan seuraavissa vaiheissa, kuten kulutuslaskennassa kuormalavojen lukumäärän laskentaan.

Varusteen kuormalavakerroin (k_m) lasketaan varusteen pakkaustiedoista siten, että varustekohtaisesti tuliannos k_{lt} jaetaan pakkauksen laukausmäärän k_{lp} ja kuormalavan pakkausmäärän (k_{pl}) tulolla.

$$k_l = \frac{k_{lt}}{k_{lp}k_{pl}} \qquad , jossa \tag{2}$$

k_l kuormalavakerroin
k_{lt} laukausta per tuliannos
k_{lp} laukausta per pakkaus
k_{pl} pakkausta per kuormalava.

Varusteen massakerroin (k_m) lasketaan varusteen pakkaustiedoista siten, että varustekohtaisesti tuliannoksen k_{lt} ja kuormalavan massa k_{lm} tulo jaetaan pakkauksen laukausmäärän k_{lp} ja kuormalavan pakkausmäärän (k_{pl}) tulolla.

$$k_m = \frac{k_{lt}k_{lm}}{k_{lp}k_{pl}} \qquad , jossa \tag{3}$$

k_m massakerroin
k_{lt} laukausta per tuliannos
k_{lm} kuormalavan massa
k_{lp} laukausta per pakkaus
k_{pl} pakkausta per kuormalava.

Varusteen määrä vastaava massa (asetyypin ammusten massa) lasketaan pakkausrekisterin ja huoltotilanneilmoituksen tiedoista siten, että massa on varusteen lukumäärän n, tuliannoksen T ja massakertoimen k_m tulo.

$$m = nTk_m \qquad , jossa \tag{4}$$

m massa
n varusteen lukumäärä
T varusteen tuliannos
k_m varusteen massakerroin.

Varusteen määrä kuormalavoina (asetyypin ammusten kuormalavoina) lasketaan pakkausrekisterin ja huoltotilanneilmoituksen tiedoista siten, että kuormalavat on varusteen lukumäärän n, tuliannoksen T ja massakertoimen k_l tulo.

$$L = nTk_l \qquad , jossa \tag{5}$$

L kuormalavojen lukumäärä
n varusteen lukumäärä
T varusteen tuliannos
k_l varusteen kuormalavakerroin.

Kuormalavojen (EURO) kuljettamiseen tarvittava 20 jalan merikonttien määrä lasketaan kaavalla:

$$K = \frac{L}{k_e}, jossa \tag{6}$$

K konttien lukumäärä
L kuormalavojen (EURO) lukumäärä
k_e yhden kontin max kuormalavamäärä (EURO).

Kuormalavojen (FIN) kuljettamiseen tarvittava konttien määrä lasketaan kaavalla (7), jossa siirrettävien kuormalavojen lukumäärä (L) jaetaan konttiin maksimissaan lastattavalla FIN-kuormalavamäärällä (k_f):

$$K = \frac{L}{k_f} \qquad , jossa \tag{7}$$

K konttien lukumäärä
L kuormalavojen (FIN) lukumäärä
k_e yhden kontin max kuormalavamäärä (FIN).

4.4 Raporttien tuottaminen

Huoltolaskennan vaiheita esittelevän kuvion 3 (sivulla 10) mukaan raportteja tuottavat suorituskykylaskenta ja huoltotarvelaskenta. Tässä luvussa esitellään raporttien muodostamisen toiminto suorituskykyraportin avulla. Suorituskykylaskennalla tuotetaan joukkoyksikön huoltotilanneilmoituksesta (rv) suorituskykyraportti, jossa numeerinen tieto on ryhmitelty ja visualisoitu. Suorituskykyraportteja tuotetaan eri tarvitsijoille huomioiden niiden erilaiset tietotarpeet.

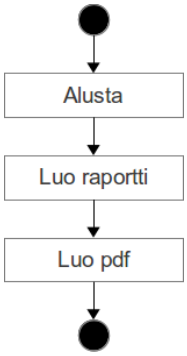
Raportteja tuottava laskenta on jaettu kahteen Raportti-luokan metodiin, joita ovat laadiSkyky ja laadiHtarve. Huoltolaskentaohjelmistossa luodaan raportti-luokan mukainen skykyRapo-olio. Raportti-luokan alustus-metodissa alustetaan raporttien tuottamisessa tarvittavat instanssimuuttujat (kuvio 26). Luokan laadiSkyky-metodin kutsumat funktiot käyttävät suoraan instassimuuttujia tietojen lukemiseen ja kirjoittamiseen funktiokutsujen parametrien ja funktioiden paluuarvojen asemesta.

```
1  class Raportti:
2     def __init__(self):
3     '''Luokan Alustaja (rakentaja)'''
4
5        # Suorituskykyraportin rakentamiseen
6        self.tex = ''
7
8        # Palautettava viesti
9        self.txt = ''
10
11        # tex-tiedostojen tmp-hakemisto
12       self.tmpdir0 = '/tmp'
13       self.tmpdir1 = 'htarve3'
14
15       # tex-tiedostojen tmp-polku
16       self.tmppath = self.tmpdir0 + '/' + self.tmpdir1
17
18       # Tiedostonimen aikaleima
19       self.aikaleima = ''
20
21       # Skyky-raporttitiedoston nimi
22       self.tiedostonimi = ''
23
24       # polku + tiedostonimi
25       self.tallennuspolku = ''
```

KUVIO 26 Luokan Raportti alustus-metodi

Kuvion 27 vuokaavio esittelee suorituskykyraportin muodostavan laadiSkyky-metodin vaiheet. Pääohjelmasta kutsutaan vain skykyRapo-olion

laadiSkyky-metodia, joka rakentaa neljän vaiheen kautta suorituskykyraportin. Kuviossa 28 esitellään kunkin vaiheen toteuttavan funktion kutsu laadiSkyky-metodissa.



KUVIO 27 Luokan Raportti laadiSkyky-metodin kulku

```
1    def laadiSkyky(self, path, jopo, hjko, info):
2
3        self.alusta()          # Luo tmp-hakemiston
4        self.luoSisalto()      # Luo raportin muutuvan sisallon
5        self.luoPdf(path)      # Luo pdf-tiedoston
6
7        # Rakentaa palautettavan viestin
8        myToString(path, jopo, hjko, info)
9
10       return unicode(self.txt) # palauttaa viestin
```

KUVIO 28 Luokan Raportti laadiSkyky metodi

Ensimmäisessä vaiheessa alusta-funktio poistaa raportin vanhat välivaiheiden tiedostot ja hakemistot, minkä jälkeen funktio luo tarvittavat hakemistot uudestaan. Hakemistorakenne luodaan, koska seuraavassa vaiheessa raporttifunktiot tallentavat tuottamansa kuva- ja tekstitiedostot hakemistorakenteeseen. Hakemistorakenne määritetään huoltolaskennan kehittämisen kolmannessa vaiheessa.

Toinen vaihe on raportin luomisen tärkein vaihe. Sen aikana luodaan raportin osat usealla raporttifunktioilla (kuvio 29). Tuotettava raportti on tyypiltään latex-dokumentti, mikä mahdollistaa sen kokoamisen useasta kuva- ja tekstitiedostosta. Ensimmäisenä kutsuttava raporttifunktio luo raportin päädokumentin alkuosan. Seuraavat raporttifunktiot tuottavat tallentavat tuotoksensa vaiheessa yksi luotuun hakemistorakenteeseen. Lisäksi raporttifunktioden paluuarvoina olevat tekstit liitetään päädokumenttin.

Päädokumenttiin liitettävät tekstiosat ovat input-komentoja, joita käytetään käännösvaiheessa ulkopuolisten tiedostojen lisäämiseen päädokumenttiin. Viimeinen raporttifunktio lisää päädokumenttiin raportin loppuosan.

```
1     def luoSisalto()
2        '''Rakentaa raportin sisallon'''
3        self.tex += raporttifunktio01()
4        self.tex += raporttifunktio02()
5        #...
6        self.tex += raporttifunktioNN()
```

KUVIO 29 Luokan Raportti luoSisalto-funktio

Opinnäytetyössä toteutettiin päädokumentin alku- ja loppuosat luovat raporttifunktiot, mikä mahdollisti toiminnon esittelyn loppukäyttäjän näkökulmasta. Huoltolaskentaohjelmaan tulevien raporttifunktioden sisältö määritellään ja toteutetaan huoltolaskennan kehittämisen kolmannessa vaiheessa, eikä aihetta sen vuoksi käsitellä enempää tässä opinnäytetyössä.

Kolmannessa vaiheessa (kuvio 30) luoPdf-funktio luo latex-dokumentista pdf-dokumentin. Rivin 6 komento annetaan kahdesti, koska muutoin luotavan pdf-tiedoston sisällysluettelo jää tyhjäksi.

```
1  def luoPdf(self, path):
2     '''Luo pdf-tiedoston tex-tiedostosta'''
3
4     # Tallennnetaan nykyinen tyohakemisto
5     mycwd = os.getcwd()
6
7     # Vaihdetaan tyohakemistoksi tex-tiedoston hakemisto
8     os.chdir(self.tmppath + '/report/') #'/tmp/report/')
9
10    # Muodostetaan systeemikutsu-string
11    text = 'pdflatex ' + self.tallennuspolku
12
13    # Luodaan pdf-tiedosto kahdesti, koska sisallysluettelo
14    # muodostetaan kaanoksen yhteydessa
15    os.system(unicode(text))
16    os.system(unicode(text))
17
18    # Kopioidaan pdf-tiedosto path-muuttujan mukaiseen
         paikkaan
19    src = self.tallennuspolku[:-3] + 'pdf'
20    dst = path
21    shutil.copy2(unicode(src), unicode(dst))
22
23    # Palautetaan alkuperainen tyohakemisto
24    os.chdir(mycwd)
```

KUVIO 30 Luokan Raportti luoPdf funktio

5 POHDINTA

Opinnäytetyö oli luonteeltaan kehitystyö, jossa suunniteltiin ja rakennettiin huoltolaskentaohjelmisto. Ohjelmiston avulla pilotoitiin huoltolaskennan automatisointia. Työn tavoitteet tarkentuivat kehitystyön edetessä erilaisten kokeilujen kautta.

Opinnäytetyön päätuotteita ovat huoltotilanneilmoitus (Excel-työkirja), laskennan toimintamalli ja huoltolaskentaohjelmisto. Koska opinnäytetyö on laajemman kokonaisuuden, huoltolaskennan kehittämisen, ensimmäinen vaihe, niin tuotteet saavat lopullisen muotonsa huoltolaskennnan kehittämisen toisen vaiheen testauksen ja opinnäytetyössä laaditun kokonaissuunnitelman ohjaamana kolmannessa vaiheessa.

Työn tavoitteena oli rakentaa huoltolaskentaohjelmisto (ensimmäinen versio 1.0), joka automatisoi huoltotilanneilmoitusten yhdistämisen sekä tuottaa suorituskyky- ja huoltotarveraportteja huoltotilanneilmoitusten tiedoista.

Tarkestaltaessa työn tavoitetta ja työn tuloksia voidaan todeta, että tavoitteet toteutuivat tärkeimmiltä osin. Tavoitteet saavutettiin huoltotilanneilmoitusten yhdistämisen automatisoinnin osalta täysimääräisesti, mutta vain osittain raportoinnin ja laskennan dokumentoinnin osalta. Laskentamallissa olevat tilausten genrointi sekä tappio- ja kulutusennustelaskenta toteutetaan huoltolaskennan kehittämisen kolmannessa vaiheessa. Raporttien tarkoituksena on havainnollistaa loppukäyttäjille huoltotarvelaskennan automatisoinnin ja visualisoinnin mahdollisuudet.

Huoltotilanneilmoitusten yhdistämisen automatisoinnin tarkoituksena oli poistaa työläs ja virhealtis työvaihe. Tämä mahdollistaa sen, että huoltohenkilöstöllä on enemmän aikaa tulosten analysointiin. Huomioonottaen rajaukset, jotka esiteltiin johdannossa, tavoite saavutettiin. Toteutettu huoltolaskentaohjelmisto tuottaa kahdeksasta perusyksikön huoltotilannilmoituksesta joukkoyksikön huoltotilanneilmoituksen kaikki kolme versiota (vaje, rv ja tarve) ja suorituskykyraportin rungon noin sekunnissa.

Kolmannessa vaiheessa lisättävät raporttifunktioden määrä, ja niiden sisältämän laskennan laajuus voivat vaikuttaa kaikkien tuotosten tuottamiseen kuluvan ajan kasvamisen joihinkin sekunteihin. Arvio perustuu työtä ennen ja sen aikana tehtyihin samalla toteutustavalla tehtyihin testauksiin. Suorityskykyraportin tuottamisen mekanismia voidaan käyttää jatkossa myös huoltotarveraportin

tuottamiseen. Molempien raporttien varsinaisen sisällön tuottavat raporttifunktiot, toteutetaan kokonaisuunnitelman mukaisesti testausken jälkeen kolmannessa vaiheessa.

Opinnäytetyön oheistavoitteena oli kerätä ja kehittää huoltolaskennassa käytettäviä laskentakaavoja ja algoritmeja. Tässä työssä esitellään joitakin huoltotarvelaskennassa ja tappio- ja kulutusennustelaskennassa tarvittavista laskentakaavoista. Laskentakaavojen kehittämiseen ja dokumentoimiseen ei kyetty käyttämään niin paljon aikaa kuin oli suunniteltu.

Opinnäytetyön tulosten arviointia varten on työssä toteutetun huoltolaskentaohjelmiston ominaisuuksia tai toimintoja verratava työn tavoitteisiin sekä asiakas- ja ohjelmistovaatimuksiin (luku 2.2).

Opinnäytetyö täytti huoltolaskentaohjelmistolle asetetut asiakasvaatimukset yhtä lukuunottamatta. Asiakasvaatimukseen (AV-2), jossa vaadittiin ilmoituksen toimivan tilauksena, ei tässä työssä otettu kantaa, koska se ei ollut tekijän päätettävissä. Huoltotilanneilmoitusten perusteella voidaan ennakoida tulevat tilaukset, mutta ilmoituksissa ei ole tällä hetkellä riittävän yksityiskohtaisia tietoja, mitä tilaus edellyttää. Asiakasvaatimus (AV-2) toteutetaan edellä mainituista syistä vasta kokonaissuunnitelman kolmannessa vaiheessa.

Työssä luotu huoltotilanneilmoitus soveltuu perusyksikkötasalla henkilö-, kalusto- ja materiaaliresurssien seurantaan siten, että joukkueiden vajeet päivitetään Excel-työkirjaan. Työkirja laskee perusyksikön resurssivajeiden summat ja resurssien rivivahvuudet määrävahvuudesta. Jos joukolle on määritetty jokin muu tavoitevahvuus, niin sitä käytetään laskennassa määrävahvuuden asemesta.

Huoltolaskentaohjelmisto tuottaa ohjelmistovaatimuksien yksi ja seitsemän (OV-1, OV-7) mukaisesti koosteet huoltotilanneilmoituksista. Koosteet tallennetaan Excel-työkirjoina käyttäjän määrittämään laskentahakemistoon (OV-2, OV8). Ohjelmisto tuottaa koosteita kolme erilaista versiota kolmeen eri jatkokäyttöön. Huoltolaskentaohjelmisto tuottaa raporttien rungot (OV-4), mutta ei huoltotilanneilmoituksesta laskettua raporttien sisältöä (OV-3).

Opinnäytetyön laskentamallin suunnitteluvaiheessa suunniteltiin huoltotilanneilmoituksen rakenne ja laskennan kokonaisuus. Tämän perusteella luotiin kolmannessa luvussa esitetyt huoltolaskennan käyttötilanteet. Huoltotilanneilmoituksen rakenteen kehittäminen ja testaaminen veivät suunniteltua enemmän aikaa, koska rakennetta kehitettiin työn loppuun asti.

Suunniteltua enemmän aikaa veivät myös python kirjastojen opettelu ja laskenta-algoritminen kehittäminen.

Huoltolaskennan automatisoinnilla parannetaan huoltotilannetietojen kokoamisen luetettavuutta ja nopeutta sekä mahdollistetaan erilaisten suorituskyky- ja huoltotarveraporttien tuottaminen huoltotilanneilmoituksista. Automatisoinnin avulla asiantuntijoille jää enemmän aikaa laskentatulosten eli erilaisten suorituskyky- ja huoltotarveraporttien analysointiin ja arviointiin.

Opinnäytyössä kehittetty huoltotilanneilmoituksen rakenne ja huoltotilanneilmoituksista tuotettavien koosteiden (vaje, rv ja tarva) laskenta mahdollistavat huoltotilanneilmoituksen muokkaamisen mm. harjoituksen tarpeita vastaaviksi. Huoltotilanneilmoituksen välilehtien nimet, varusteiden nimet ja varusteiden lukumäärät ovat muokattavissa ilman ohjelmakoodin muutoksia. Kaikilla perusyksiköillä on kuitenkin oltava käytössä sama versio huoltotilanneilmoituksesta. Välilehtien määrän muuttaminen ja välilehtien rivien 2-6 muokkaaminen vaatii ohjelmakoodin muuttamista. Mikäli muokattavuus halutaan säilyttää jatkossa, se on huomioitava raporttien sisällön tuottavien raporttifunktioiden suunnittelussa. Vaikka opinnäytetyön kehyksenä käytettiin perusyksikkö-joukkoyksikkö tasoa, toteutus on skaalattavissa kaikille organisaatiotasoille.

Työn edetessä käytiin keskustelua siitä, jätetäänkö huoltotilanneilmoituksen rakenteeseen kuormalava- ja massarivit ainakin kriitisen materiaalin välilehdelle ja mahdollisesti asevälilehdelle. Rivien tarkoituksena on auttaa hahmottamista konkretisoimalla vajerivin määrä täydennyksien ja kuljetusten näkökulmista. Tämän hetkisen suunnitelman mukaan vajeiden muuttaminen kuormalavoiksi ja massoiksi on huoltotarveraporteissa. Perusteena on se, että näin huoltotilanneilmoituksen välilehdet eivät poikkea toisistaan ja siten monimutkaista koosteiden laskentaa.

Työssä tuotettua lähdekoodia kehitetään jatkossa mm. siten, että luodaan selkeämpi luokkahierarkia. Nykyisen kahden luokan aiheuttamaa toistoa ja päällekäisyyttä poistetaan luomalla yhteinen yläluokka. Yläluokkaan kerätään tiedostojen käsittelyyn liittyvät toiminnot, kuten luku, tallennus sekä aikaleimojen ja tiedostonimen luonti. Huoltolaskentaohjelmistosta testauksessa löydetyt virheet on dokumentoitu ja ne korjataan ohjelmiston seuraavaan versioon.

Sotilasympäristön perusluonteeseen liittyvät yllättävät ja monesti kertaluonteiset

tietotarpeet, joihin ei ole olemassa toimintamallia tai laskentaohjelmaa. Tietojärjestelmiä ei saada milloinkaan vastaamaan täysin kaikkiin tietotarpeisiin. Parempi vaihtoehto on valita helppokäyttöisten, mutta toiminnoiltaan huonompien työkalujen sijasta vaativampi, mutta monikäyttöisempi työkalu. Edellä mainittu vaativampi työkalu voisi olla tässäkin työssä esitelty IPython Notebook, joka soveltuu hyvin erilaisten aineistojen muokkaukseen, analysointiin ja visualisointiin. Työkalu mahdollistaa toistuvien tietotarpeiden käsittelyjen taltioimisen ja nopean kertaluonteisten ratkaisujen kehittämisen. Työkalun käyttöä rajoittaa se, että ei ole käytettävissä tarvittavaa tietoaineistoa tai käytettävissä oleva tietoaineisto on huonolaatuista.

Edellä mainittu työkalu edellyttää, että joukkoyksikön tasolta alkaen esikunnassa olisi oltava henkilö, jonka tehtävänä olisi tuottaa esikunnan henkilöstön tietotarpeita vastaavia raportteja tai tietoa. Työkalun käyttö vaatii tehtävään sijoitettavalta henkilöltä erityisosaamista ja soveltuvia koulutusaloja henkilölle voisivat olla mm. matematiikka, fysiikka ja ohjelmointitekniikka.

Opinnäytetyön taustalla on pidemmän aikaa jatkunut huoltolaskennan kehittämistarve erilaisine kokeiluineen. Työn alkuperäisenä tavoitteena oli tuottaa huoltolaskentaohjelmisto, joka olisi ollut nykyisin käytössä olevien taulukkolaskentatyökalujen päivitetty ja muokattattava versio. Aiheeseen perehtymisen myötä seuraavassa vaiheessa ohjelmiston fokusta siirrettiin yksityiskohtaisesta laskennasta koko toimintavaihtoehdon huoltotarpeen laskentaan ja eri toimintavaihtoehtojen vertailuun. Lisäksi ohjelmiston käyttö rajattiin vain suunnitteluprosessin kolmanteen vaiheeseen, jossa käsitellään toimintavaihtoehdot. Kolmennessa vaiheessa työn painopiste rajattiin siten, että keskitytään vain yhteen toimialaan, mutta aiemmasta poiketen huomioiden kaikki päätason prosessia. Tähän vaikutti pääasiassa se, että toimialojen välillä havaittiin riittävän vahva analogia; muiden lisääminen myöhemmin nuoduttaa samoja sääntöjä kuin nyt toteutettuksi valittu toimiala. Huoltolaskennan kehittämisessä opinnäytetyön keskiöön nostettiin lopulta huoltotilanneilmoituksista lähtevä laskenta, koska tilanteen seurananna osana oleva huoltotilanteen seuranta on kuitenkin perusteena kaikelle muulle laskennalle niin tilanneraporteille kuin ennusteraporteillekin.

Opinnäytetyön merkittävimmäksi haasteeksi muodostui työn kehityksellinen luonne ilman selkeää tavoitetta. Toisaalta kehityksellinen luonne oli työn suola. Työn edistymisen myötä kokemuspohjaisesti voitiin tarkentaa tavoitetta.

LÄHTEET

Kasurinen, J.P. 2009. Python 3 ohjelmointi. Jyväskylä: WSOYpro/Docendo-tuotteet.

McKinney. W. 2012. 1. painos. Python for Data Analysis. Sebastopol CA: O'Reilly Media

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

Summerfield, M. 2010. 4. painos. Rapid GUI Programming with Python and Qt. Prentice Hall

LaTeX project team. 2010. An introduction to LaTeX. [viitattu 18.4.2013]. Saatavissa: http://latex-project.org/intro.html.

Wikipedia. 2012a. Python. [viitattu 18.4.2013]. Saatavissa: http://en.wikipedia.org/wiki/Python_%28programming_language%29

Wikipedia. 2012b. Latex. [viitattu 18.4.2013]. Saatavissa: http://fi.wikipedia.org/wiki/Latex.

Riverbank. 2012. What is PyQt. [viitattu 18.4.2013]. Saatavissa: http://www.riverbankcomputing.co.uk/software/pyqt/intro

Wikipedia. 2012c. Qt kehitysympäristö. [viitattu 18.4.2013]. Saatavissa: http://fi.wikipedia.org/wiki/Qt_(kehitysympäristö).

Wikipedia. 2012d. PyQt. [viitattu 18.4.2013]. Saatavissa: http://en.wikipedia.org/wiki/Pyqt.

Digia. 2012. Product Qt. [viitattu 18.4.2013]. Saatavissa: http://qt.digia.com/Product/

Kuvat

Taulukot

LIITE 1

Nimi	Laadi tappio- ja kulutusennuste
Versiohistoria	0.1
Osallistujat	Tappio- ja kulutusennustelaskentaan osallistuu huoltopäällikkö..
Tuloehdot	Sotapelin tulokset käytettävissä.
Kuvaus	Tavoitteena on tuottaa perusyksiköiden huoltotilanneilmoituksista uudet huoltotilanneilmoitukset, joista on vähennetty tappio- ja kulutuslaskennan mukaiset tappiot ja kulutukset. Käyttäjä avaa huoltolaskentaohjelmiston, valitsee laskentahakemiston ja käynnistää tappio- ja kulutusennustelaskennan.
Tuotokset	huoltotilanneilmoitus (arvio).
Poikkeukset	-
Muut vaatimukset	Huoltotilannetasot, porrastukset ja PHT-tasot on määritelty.

Nimi	Laadi huoltotilanneilmoitus
Versiohistoria	0.1
Osallistujat	Käyttötapaukseen osallistuu joukkoyksikön huoltopäällikkö..
Tuloehdot	Toiminnon tuloehtona on, että perusyksiköiden huoltotilanneilmoitukset ovat saatavilla.
Kuvaus	Tavoitteena on tuottaa koostettu joukkoyksikön huoltotilanneilmoitus perusyksiköiden huoltotilanneilmoituksista. Koosteen laskennassa yhdistetään kaikkien perusyksiköiden huoltotilanneilmoitukset ja luodaan koko johtoportaan huoltotilanneilmoitus. Käyttäjä avaa huoltolaskentaohjelmiston, valitsee laskentahakemiston ja käynnistää huoltotilannelaskennan.
Poikkeukset	-
Muut vaatimukset	Huoltotilannetasot, porrastukset ja PHT-tasot on määritelty.

Nimi	Laadi Huoltotarveraporti
Versiohistoria	0.1
Osallistujat	Huoltotarvelaskentaan osallistuu joukkoyksikön huoltopäällikkö .
Tuloehdot	laskennan tuloehtona on, että joukkoyksikön huoltotilanneilmoituksen tarve-versio on käytettävissä. Muutoin tulee olla perusyksiköiden huoltotilanneilmoitukset on päivitetty kuten myös huoltojoukon huoltotilanneilmoitus (tuki).
Kuvaus	Tavoitteena on tuottaa huoltotarveraportti joukkoyksikön huoltotilanneilmoituksesta (tarve). Käyttäjä laskee johtoportaan huoltotilanneilmoituksen (vaje), jota käyttää yhdessä huoltojoukon huoltotilanneilmoituksen (tuki) kanssa huoltotarveraportin laskennan syötteenä. Laskennan tuotoksina ovat huoltotarveraportti ja huoltotilanneilmoitus (tukitarve). Käyttäjä avaa huoltolaskentaohjelmiston, valitsee laskentahakemiston ja käynnistää huoltotarvelaskennan. Tavoitteena on tuottaa joukoyksikön huoltotilanneilmoituksen tarve-versio. Tukitarpeen laadintaan osallistuu joukkoyksikön huoltopäällikkö. Toiminnon tuloehtona on, että huoltojoukon huoltotilanneilmoitukset ovat käytettävissä. Käyttäjä avaa huoltolaskentaohjelmiston, valitsee laskentahakemiston ja käynnistää tukitarvelaskennan.
Poikkeukset	-
Muut vaatimukset	Alajohtoportaiden huoltotilanneilmoitukset ja huoltojoukon huoltotilanneilmoitus (tuki) ovat päivitetty.

Nimi	Laadi tukitarve
Versiohistoria	0.1
Osallistujat	Tukitarvelaskentaan osallistuu joukkoyksikön huoltopäällikkö .
Tuloehdot	lToiminnon tuloehtona on, että huoltojoukon huoltotilanneilmoitukset ovat käytettävissä ja joukkoyksikön huoltotilanneilmoituksen tarve-versio on käytettävissä.
Kuvaus	Tavoitteena on tuottaa joukoyksikön huoltotilanneilmoituksen tarve-versio. Laskennan tuotoksena on huoltotilanneilmoitus (tarve). Käyttäjä avaa huoltolaskentaohjelmiston, valitsee laskentahakemiston ja käynnistää huoltotarvelaskennan. . Käyttäjä avaa huoltolaskentaohjelmiston, valitsee laskentahakemiston ja käynnistää tukitarvelaskennan.
Poikkeukset	-
Muut vaatimukset	Alajohtoportaiden huoltotilanneilmoitukset ja huoltojoukon huoltotilanneilmoitus (tuki) ovat päivitetty.

Nimi	Suorituskykylaskenta
Versiohistoria	0.1
Osallistujat	Suorityskykylaskentaan osallistuu joukkoyksikön huoltopäällikkö
Tuloehdot	Toiminnon tuloehtona on, että huoltotilanneilmoituksen rv-versio (KT-2:n tuotos) on käytettävissä.
Kuvaus	Tavoitteena on tuottaa suortituskykyraportti joukkoyksikön huoltotilanneilmoituksesta. Käyttäjä avaa huoltolaskentaohjelmiston, valitsee laskentahakemiston ja käynnistää suorituskykylaskennan. Käyttäjä käynnistää huoltolaskentaohjelmiston, valitsee hakemiston, jossa huoltotilanneilmoitukset on taltioituna. Valinnan jälkeen ohjelmisto laskee koosteen, eli johtoportaan huoltotilanneilmoituksen (rv), ja tallentaa sen samaan hakemistoon alajohtoportaiden huoltotilanneilmoitusten kanssa. Käyttäjä laskee johtoportaan huoltotilanneilmoituksen (rv), jota käyttää suorituskykyraportin laskennan syötteenä. Laskennan tuotoksena on suorituskykyraportti.
Poikkeukset	-
Muut vaatimukset	Alajohtoportaan huoltotilanneilmoitukset on päivitetty.

Nimi	Tilausten generointi
Versiohistoria	0.1
Osallistujat	Tilausten generiointiin osallistuu joukkoyksikön huoltopäällikkö.
Tuloehdot	Toiminnon tuloehtona on, että huoltotarvelaskennan tuottaman huoltotilannilmoituksen tarve-version lukemat on päivitetty suorituskykyraporttien, huoltotarveraporttien sekä tappio- ja kulutusennusteiden analyysin pohjalta.
Kuvaus	Tavoitteena on tuottaa tilaukset joukkoyksikön huoltotilanneilmoituksesta. Käyttäjä luo tilaukset operatiivisen (vast.) käyttäjän ohjauksen pohjalta. Ennen ohjausta operatiivinen käyttäjä on analysoinut tappio- ja kulutusraportin, suorituskykyraportin sekä huoltotarveraportin vaikutukset.
Poikkeukset	-
Muut vaatimukset	-