Mikael Filppula

Tyyppileimikoinnin käyttötapaus osana metsävaratiedon palvelualustan toiminnallisia palveluita

Diplomityö

Tampereen Teknillinen Yliopisto

Automaatiotekniikan koulutusohjelma

2017

Tiivistelmä

MIKAEL FILPPULA: Tyyppileimikoinnin käyttötapaus osana metsävaratiedon palvelualustan toiminnallisia palveluita
Tampereen teknillinen yliopisto
Diplomityö, 44 sivua
Marraskuu 2017
Automaatiotekniikan DI-tutkinto-ohjelma
Automaation tietotekniikan pääaine
Tarkastajat: Risto Ritala, David Hästbacka

Avainsanat: Metsävaratieto, palvelualusta, tyyppileimikointi, webservice, diplomityö

Metsävaratiedon hyödyntämiseksi ja jalostamiseksi on kehitetty palvelualustaa, jonka tavoitteena on tarjota mahdollisimman ajantasaista metsävaratietoa käyttäjille tai käyttäjäsovelluksille. Palvelualustalla on käytössä toiminnallisia palveluita, jotka tarjoavat työkalut metsätiedon hakemiseen, päivittämiseen ja muuhun käsittelyyn. Tässä työssä tutkitaan, miten tyyppileimikointi, eli tietyntyyppisten hakkuumahdollisuuksien selvittäminen joltain alueelta, voitaisiin toteuttaa käyttämällä palvelualustan tarjoamia palveluita, ja mitä vaatimuksia, rajoitteita tai muita toimenpiteitä tyyppileimikointi vaatisi palvelualustalta.

Työssä muodostettiin koostettu tietorakenne, jonka tietosisällön avulla usean eri tietolähteen metsävaratieto voidaan yhdistää yhdeksi, tyyppileimikoinnille optimaaliseksi tietomuodoksi. Tietorakenne perustui alan kirjallisuuteen sekä metsäteollisuuden käyttämiin standardeihin ja tietokantoihin.

Tyyppileimikointia tutkittiin vertailemalla kahta eri algoritmivaihtoehtoa, joista yksi perustui tyyppiluokituksiin, jotka määriteltiin metsätiedon hilasoluille ennen algoritmin suorittamista, sekä similariteettilaskentaan, joka tehtäisiin algoritmin aikana. Tyyppiluokituksen tutkimiseksi työssä tehtiin suppea sovellus, jonka avulla huomattiin käytetyn tyyppiluokituksen vaikuttavan merkittävästi algoritmissa tarvittavien referenssisolujen määrään: 6^3 (216) eri luokituksen tapauksessa referenssisoluja tarvittiin 1150, jotta sama luokitus löytyi kaikille yhdeksästä kohdealueen hilasolusta 95% varmuudella.

Tyyppileimikointiin tarvittavan tietomäärän ja algoritmidemon perusteella datafuusion, ajantasaistuksen ja haun toiminnallisille palveluille esitettiin kolmea eri vaihtoehtoa tyyppileimikointia varten. Vaihtoehtoina olivat algoritmissa käytettävien tietokantojen rajoittaminen, kaikkien hilasolujen prosessoiminen ja tallentaminen ennen algoritmia ja suoritusajan pidentäminen, joka edellyttäisi notifiointi-palvelun käyttämistä algoritmin päätyttyä. Uudeksi palveluksi esitettiin kahta hilasolua vertailevaa palvelua.

Abstract

MIKAEL FILPPULA: Influence of Selective Logging's Use Case on Forest Data Service Platform's Functional Services
Tampere University of Technology
Master's thesis, 44 pages
November 2017
Degree Programme in Automation Engineering, MSc (Tech)
Major: Information Systems in Automation
Examiners: Risto Ritala, David Hästbacka

Keywords: Forestry, service platform, web service, selective logging, master's thesis

In hopes to utilize and refine forestry data, a service platform has been developed to provide access to the most updated data to its users. The users can access the data through the platform's functional services, which provide tools for data searching, updating and filtering. In this thesis, we examine how selective logging, or selecting forest areas that fulfill criteria for a specific logging method, could be performed by the functional services, and what constrains, requirements or other measures would be imposed by the use case to the service platform.

In this thesis, we design a composite data structure that combines the different data sources into a single, optimal data structure, which can be utilized in the selective logging. The data structure is based on the existing data structures, standards and forestry literature.

In addition, we examined two different approaches to the selective logging's algorithm. In the first variant, each individual forest cell receives a classification based on the cell's attributes. This classification is calculated before the algorithm, so comparing two cells reduces to whether the cells have the same classification or not. The second variant for the algorithm was calculating the similarity between the cells separately. To examine the first variant, we created a concise program that was used to observe the effects of different classifications and the number of reference cells. We conclude that even with a simple classification of 6^3 different classifications, 1150 reference cells were required for nine target cells to be correctly evaluated 95% of the times.

Based on the findings, we proposed three approaches for the functional services that would help performing the selective logging successfully. The approaches were limiting the used databases during the algorithm, processing and saving all the required forest cells before the algorithm and extending the performance time. The last approach also requires using the notification service on the platform. Finally, we proposed developing an additional functional service to that would compare two cells with each other.

Alkusanat

Tämä diplomityö tehtiin TTY:llä yhteistyössä Metsätehon kanssa vuonna 2017. Työssä tutkitaan mitä vaatimuksia, rajoitteita tai muita vaikutuksia tyyppileimikointi ja strateginen katkonnanohjaus luo metsävaratietoa käsittelevälle palvelualustalle.

Haluan kiittää professori Risto Ritalaa ja tutkijatohtori David Hästbackaa tuesta ja suuntaa-antavista neuvoista, jotka auttoivat tämän diplomityön valmistumiseen. Haluan kiittää myös kaikkia Metsäteholta työhön osallistuneita onnistuneesta yhteistyöstä.

Marraskuu 2017

Mikael Filppula

Sisällysluettelo

Kuvaluettelo

Taulukot

Termistö ja lyhenteet

Pohjapinta-ala (eng. basal area) – Puutilavuus hehtaaria kohden (m^3/ha).

Runkolukusarja – Puiden paksuusjakauma tietyllä alalla

RMSE – Root mean square error, neliöllisen keskiarvon virhe

DBH – ”Diameter at Breast Height”, puun läpimitta rinnankorkeudelta. Suomessa 1,3 metriä maanpinnasta.

Stand – metsäkuvio, metsäjakso

ABA – Area Based Approach, aluepohjainen laserkeilausmenetelmä

ITD – Individual Tree Detection, Yksinpuintulkinta

LiDAR – Light Detection And Ranging, yleinen termi laserkeilaus teknologialle

ALS – Airborne Laser Scanning, lentolaserkeilaus

MLS – Mobile Laser Scanning, laserkeilaus maastossa liikkuvalta alustalta

TLS – Terrestrial Laser Scanning, laserkeilaus maastossa liikkumattomalta alustalta

NN – Nearest Neighbour, lähin naapuri menetelmä

SAR – Synthetic Aperature Radar

XML – Extensible Markup Language

JSON – JavaScript Object Notation

HPR – Harvested Production, StanForD2010-standardin viesti

FO – Forest Object, eräs Metsäkeskuksen standardin viesti

FD – Forest Data, eräs Metsäkeskuksen standardin viesti

IETF – Internet Engineering Task Force

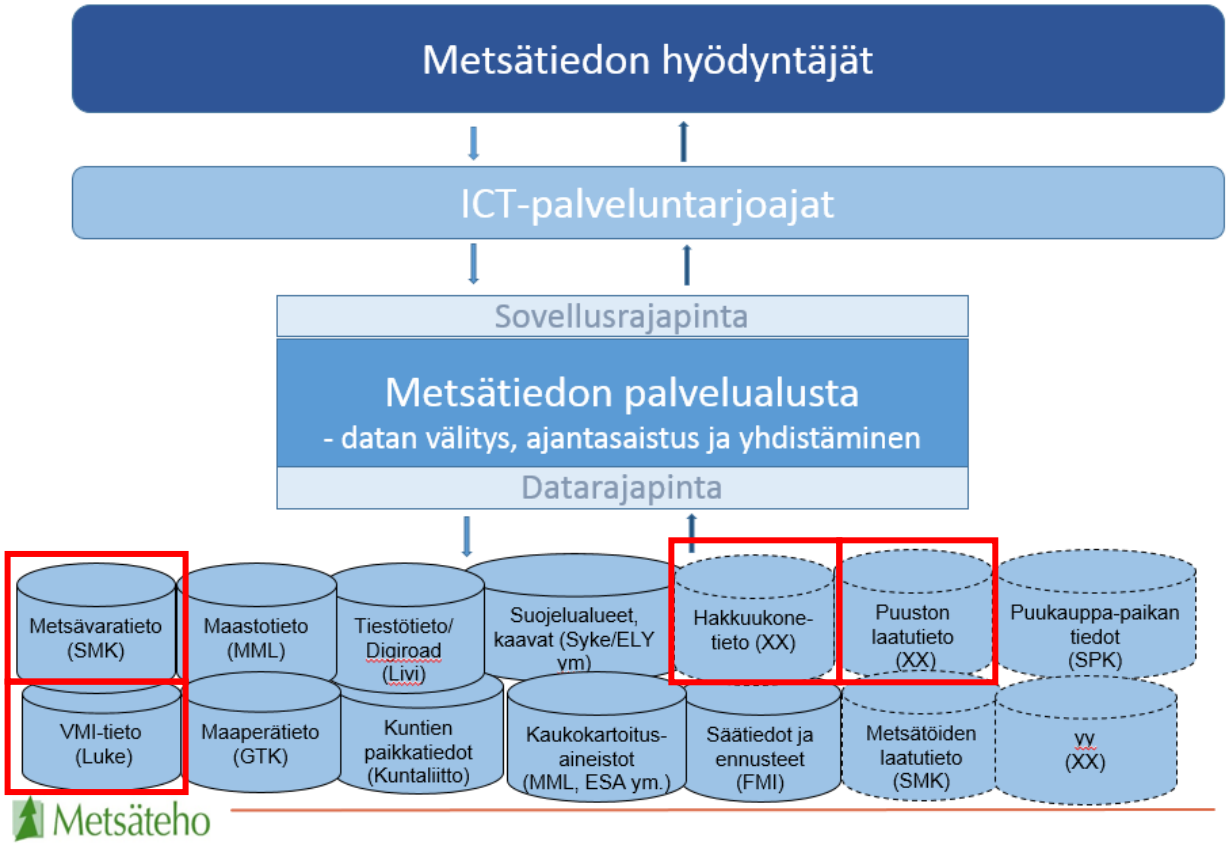
1. Johdanto

1.1 Tausta

Metsätalous on yksi Suomen merkittävimmistä työllistäjistä ja vientialoista. Luonnonvarakeskuksen [1] laatiman tilaston mukaan vuodelle 2015 Suomen 22,8 milj. hehtaarista metsää saatiin 56 milj. kuutiometriä puuta. Samana vuonna metsäteollisuuden tuotteiden vienti oli 11,2 miljardia euroa, mikä vastasi Tilastokeskuksen [2] mukaan noin 20% Suomen koko viennistä, eli oli Suomen suurin vientiala. Metsäteollisuuden tärkeä rooli on edellyttänyt kehittämään uusia metsienkartoitusmenetelmiä ja estimointialgoritmeja kustannusten optimoinnin ja strategisen metsäsuunnittelun työkaluiksi. Esimerkiksi lentolaserkeilaukselle, jota on käytetty onnistuneesti maastomittauksessa, on kehitetty estimointialgoritmeja kartoittamaan puustotunnuksia keilauksen pistepilvestä. Tämä on mahdollistanut suurien metsäalojen kartoittamisen kustannustehokkaasti ja riittävällä tarkkuudella metsäinventointia ja puunkorjuuta varten. [3] Yksityiselle metsäyhtiölle raa'an metsätiedon hankkiminen perinteisellä maastomittauksella on työlästä ja kallista. Lisäksi raakatiedon, kuten lentolaserkeilauksen pistepilven, prosessoiminen vaatii tarkkaan laadittuja ja erikoistuneita metsäkohtaisia malleja. Jo olemassa olevien metsätietolähteiden yhdistäminen on haastavaa, eikä metsätieto ole aina suoraan hyödynnettävässä muodossa.

Vuonna 2014 suomalainen metsäteollisuuden tutkimus- ja kehitysyritys Metsäteho aloitti ”Forest Big Data”-projektin (FBD), jonka tavoitteena on mahdollisimman automaattinen työlaadun hallintajärjestelmä puunkorjuuta varten. Projektin osatavoitteita ovat hakkuukoneiden kehittäminen metsätiedon keräämisessä, tiedonkeruun helpottaminen strategisen metsäsuunnittelun tarpeisiin, sekä ”täsmämetsätalous”, jossa metsäsuunnittelu voidaan toteuttaa yksittäisten puiden tasolla. Osana projektia, Metsäteho aloitti vuonna 2016 suunnittelemaan palvelualustaa, jonka tavoitteena on yhdistää heterogeenisen mittausdatan useasta eri tietolähteestä ja tarjota kerätystä metsätiedosta homogeeninen, ajantasainen tietolähde suunnittelua varten. Palvelualusta toimisi yhtenä rajapintana kaikkiin metsäinventoinnille tarpeellisiin tietolähteisiin, tarjoten puustotiedon lisäksi myös palvelun omia estimointimalleja ja datafuusiota tietolähteiden yhdistämiseksi ja ajantasaistamiseksi. Yksi palvelualustan käyttötarkoituksista on mahdollistaa tyyppileimikointi, eli tietyt puutunnukset täyttävän hakkuualueen etsiminen ja siitä saatavan puusaannon estimoiminen.

Tämän diplomityön tavoitteena on tutkia tyyppileimikoinnin optimaalista toteutusta ja miten se tulisi vaikuttamaan palvelualustan toiminnallisten palveluiden toteutuksiin. Diplomityön kannalta palvelualustaa sivuaa ”Metsäkonetiedon välitys keskitettyyn tietokantaan ja tietokantasovelluspilotti” – hanke. Sen tavoitteena on luoda perusteet laajamittaisen metsäkoneilla kerättävän tiedon kokoamiseksi ja hyödyntämiseksi, sekä kehittää ja testata menetelmät ja käytännöt hakkuukoneiden tuottaman datan hankkimiseksi yhtenäisellä tavalla. Osa diplomityössä esitellyistä tietokannoista perustuvat kyseisen hankkeen vielä keskeneräisiin pilotteihin.



Kuva 1: Palvelualustan konseptikuva ja diplomityön tärkeimmät tietokannat korostettuna. Alkuperäinen kuva on Metsäteholta.

Kappaleessa 2 esitellään taustatietoa tyypillisestä metsätiedosta ja keräysmenetelmien tarkkuuksista. Kappaleessa 3 kuvataan, kuinka metsävaratietoa siirretään ja tallennetaan nykyisten standardien ja tietokantojen avulla. Kappaleessa 4 syvennytään vielä tarkemmin tiedonsiirron tekniikkaan ja tutkitaan, luovatko ne joitain rajoitteita palvelualustalle. Kappaleessa 5 esitellään palvelualustan konseptia ja toiminnallisia palveluita tarkemmin, sekä tyyppileimikoinnin käyttötapaus. Kappaleessa 6 pyritään kuvaamaan optimaalinen koostettu tietorakenne palvelualustan sisäiseen tiedonsiirtoon. Kappale 7 ja 8 käsittelevät tyyppileimikoinnin algoritmin toteutusta, joista ensimmäinen käsittelee vaihtoehtoisia toteutustapoja ja jälkimmäinen työssä tehdyn algoritmidemon tuloksia. Lopulta kappaleessa 9 pohditaan miten tehdyt havainnot vaikuttavat toiminnallisten palveluiden toteutuksiin ja annetaan työstä yhteenveto.

1.2 Tutkimuskysymykset

Tämän diplomityön päätavoite on selvittää metsätiedon palvelualustan toiminnallista käytettävyyttä tyyppileimikoinnin näkökulmasta. Työn tuotoksena saadaan koostettu tietorakenne, jolla metsätietoa voidaan käsitellä palvelualustan sisällä sekä arviot, miten toiminnallisten palveluiden toteutukset riippuvat tyyppileimikoinnin käyttötapauksen asettamista vaatimuksista. Tavoitteiden pohjalta laadittiin kolme tutkimuskysymystä:

- Millainen koostettu tietorakenne olisi optimaalinen palvelualustan käytettäväksi tukemaan tyyppileimikointia? Mitä attribuutteja siihen sisältyy?

Kysymykseen etsitään vastausta tutkimalla palvelualustan käyttötapauksia, alan kirjallisuutta ja teollisuudessa käytettyjä metsätietostandardeja ja tietokantoja, ja tarkastelemalla niiden yhteneväisyyksiä. Tällä tavoin tietorakenne mukailisi mahdollisimman montaa teollisuudessa käytettyä valmista tietorakennetta, mikä helpottaa tietorakenteen käyttöä datafuusiossa ja tyyppileimikoinnissa. Koska diplomityö keskittyy tutkimaan tietorakennetta tyyppileimikoinnin strategisen suunnittelun kannalta, muut kuin metsätiedon attribuutit - kuten omistajuus tai dynaamiset olosuhdetiedot - tutkitaan vain pinnallisesti. Kysymykseen laaditaan vastauksena tietorakenteen XML (Extensible Markup Language) skeema tai vastaava esitys.

- Miten tyyppileimikoinnin algoritmi kannattaisi toteuttaa?

Tyyppileimikoinnille on esitetty eri algoritmivaihtoehtoja, ja tässä työssä tutkitaan, kuinka nämä eri algoritmit soveltuisivat käytettäväksi ja miten algoritmin valinta näkyisi toiminnallisissa palveluissa. Tyyppileimikoinnissa tarvittavan tyyppiluokituksien määrittäminen vaatii laajempaa tutkimusta eikä sitä yritetä tässä työssä ratkaista. Työssä tyyppiluokituksen oletetaan olevan mahdollinen, ja sille kehitetään yksinkertainen ”dummy”-toteutus simuloimaan luokittelua. Kysymyksen vastauksena algoritmista annetaan suppeahko demo Javalla.

- Miten rajapinnan toiminnalliset palvelut kannattaisi toteuttaa tyyppileimikoinnin näkökulmasta?

Tässä työssä laaditaan pohja toiminnallisten palveluiden toteutukselle, eli mietitään, millaisia vaatimuksia, rajoitteita ja kustannuksia palveluilla on, ja miten ne skaalautuvat. Nämä ominaisuudet ovat riippuvaisia tyyppileimikoinnin algoritmin toteutuksesta. Vastauksessa otetaan myös kantaa rajapinnalle optimaalisista teknologioista, vaikka rajapinnan toteutus jääkin sovelluskehittäjän päätettäväksi.

2. Metsävaratieto

Tässä kappaleessa esitellään metsäteollisuuden terminologiaa, metsätietoa, ja tärkeimpiä metsään liittyviä mittausmenetelmiä. Kappaleen ensimmäinen osa käsittelee strategisen metsäsuunnittelun metsä- ja paikkatietoa, jälkimmäinen osa esittelee mittaustapoja, joilla raakadataa kerätään metsästä. Hakkuukoneiden keräämää saantotietoa ei esitellä erikseen tässä työssä, vaan tyydytään toteamaan siihen sisältyväksi kaadettujen puiden mittasuhteet, puutavaralaji ja laatu.

Metsäsuunnittelu jaetaan strategiseen ja operatiiviseen suunnitteluun. Strategisessa suunnittelussa etsitään alueita, joissa hakkuut ovat taloudellisesti kannattavia. Operatiivinen suunnittelu keskittyy tietyn alueen hakkaamiseen, jolloin tiedot yksittäisistä puista ja dynaamisista olosuhteista ovat tarpeen. Tässä työssä keskitytään strategiseen suunnitteluun.

2.1 Metsätieto

2.1.1 Puustotunnukset

Strategisen suunnittelun perustana ovat puustotunnukset, eli tutkittavan alueen puustoa kuvaavat suureet. Puustoa voidaan kuvata seuraavilla attribuuteilla:

- Puulaji ja runkomuoto
- Puunkorkeus ja puun keskikorkeus tietyllä alalla
- Puun läpimitta rinnankorkeudelta (DBH)
- Puun kokonaistilavuus, sekä tukkiosan ja kuituosan tilavuudet
- Puiden lukumäärä tietyllä alalla
- Runkolukusarja eli puiden paksuusjakauma tietyllä alalla
- Pohjapinta-ala
- Puun sijainti
- Alimman oksan korkeus
- Elävän latvuksen raja
- Puuston ikä

Tärkeimpiä mittoja ovat puun korkeus ja läpimitta rinnankorkeudelta, joksi Suomessa määritellään 1,3 metriä maanpinnasta. Näiden mittojen ja puun runkomuodon avulla voidaan arvioida puun kokonaistilavuus. Pohjapinta-alalla tarkoitetaan puiden yhteenlaskettua läpimittaa pinta-alayksikköä kohti, yksikkönä m2/ha.

Puun laatusuureita ovat mm. alimman oksan korkeus ja elävän latvuksen raja. Ne antavat lisätietoa puuston biomassasta ja auttavat estimoimaan puiden terveyttä. Yksittäiset puun korkeudet ja alimpien oksien korkeudet antavat lisätietoa puuston biomassasta. Ne voidaan mitata vain työläästi käsimenetelmin, joten niitä harvoin mitataan koko puustolta. Puuston iän avulla voidaan päätellä myös sieltä saatavaa puulaatua.

Kustannussyistä johtuen kaikkia puustoattribuutteja ei voida mitata yksittäisistä puista, vaan metsäalueiden puustotunnukset ovat tyypillisesti koealueista estimoituja keskitunnuksia. Runkolukusarjan ja pohjapinta-alan määrittäminen pelkillä keskitunnuksilla on kuitenkin epätarkkaa, ja niiden estimoimista on tutkittu mm. Weibull- ja k-MSN-algoritmeilla. Ehdolla on myös menetelmä, jossa hilalle muodostettaisiin yksittäiset puut koealatiedon perusteella. [4] [3]

2.1.2 Muut metsätiedot

Puustotunnukset kuvaavat hyvin puuston laatua ja mittasuhteita mittaushetkellä. Kun puustotunnuksia halutaan estimoida aiempien mittausten ja kasvumallien perusteella, tarvitaan tietoa myös puuston kasvuolosuhteista. Olosuhdetieto voidaan jakaa dynaamiseen ja staattiseen olosuhdetietoon. Dynaaminen olosuhdetieto vaihtelee riippuen ajankohdasta. Tyypillisiä suureita ovat esimerkiksi roudan paksuus ja sademäärä. Dynaamista olosuhdetietoa käytetään lähinnä operatiivisessa suunnittelussa eikä sitä siksi oteta huomioon suunniteltaessa koostettua tietorakennetta. Staattinen olosuhdetieto on vastaavasti harvoin muuttuvaa, ja siihen kuuluu mm. maalajiryhmä, maankosteus, ja kivisyysindeksi.

Metsien käyttöä voi rajoittaa erilaiset suojelualueet ja kaavat, sekä omistajatiedot. Lisäksi metsien mittaustiedoille määritetään mittaustapa ja mittauksen päivämäärä, jotka antavat kontekstia tiedon tuoreudesta, tarkkuudesta ja oikeellisuudesta. Nämä ovat esimerkkejä metsävaroihin liittyvästä metatiedosta.

2.1.3 Paikkatieto

Metsäalueen attribuutit voidaan ilmaista teoriassa kolmella eri paikkatiedon muodolla: kuvioina, hilasoluina tai yksittäisistä puista koostuvana tietorakenteena. Riippumatta esitysmuodosta, paikkatiedon koordinaatit ovat sidottuna johonkin tasokoordinaattijärjestelmään, joista Suomessa käytetään WGS84- ja ETRS-TM35FIN-referenssikoordinaatistoa.

Kuviot voidaan esittää mielivaltaisenkokoisina monikulmioina. Esimerkki metsäalueiden kuvioinnista on esitetty kuvassa 2. Metsäalueiden lisäksi monikulmioilla määritellään myös kiinteistöjen, suojelualueiden, järvien, peltojen ja muiden kaavoitusten aluerajat. Monikulmiot esitetään tyypillisesti vektorimuodossa, eli kulmapisteiden koordinaatit listataan myötä- tai vastapäiväisessä järjestyksessä kuvailumenetelmästä riippuen. Metsäteollisuudessa käytetään termiä mikrokuvio kuvastamaan pientä, automaattisesti määriteltyä metsäkuviota. Metsätiedon tavoitetiladokumentissa [5] mainitaan mikrokuvion tavoiteltavaksi keskikooksi 0,25 – 0,50 hehtaaria. Varsinaiset metsäkuviot muodostetaan yhdistämällä mikrokuviot aluerajoihin (kuten omistusrajat ja vesistöt) sekä rajaamalla kuvioita manuaalisesti.

Kuva 2: Metsäkuviot [5]

Hilasoluilla viitataan metsätaloudessa puustotiedon perusyksikköinä käytettyihin 16x16 metrisiin metsäsoluihin, joilla voidaan muodostaa koko Suomen metsät käsittävä hilajoukko. Strategisessa suunnittelussa hilasolusta halutaan metsävaratietona ainakin solun puuston keskitunnukset (lukumäärä, korkeus, läpimitta) puulajeittain, sekä kasvupaikkatunnuksia. Hilaruutu yksilöidään esimerkiksi

Metsäkeskuksen tietokannassa sille annettavan hilanumeron avulla, josta solun fyysinen sijainti voidaan päätellä. Numeron laskukaava on seuraava: Hilaruudun koordinaateista vähennetään aluksi referenssikoordinaatiston origo, jonka jälkeen metriset X ja Y koordinaatit jaetaan 16 eli hilaruudun sivunpituudella. Lopuksi saadut arvot asetetaan peräkkäin Y-arvo ensin, joka muodostaa hilasolun lopullisen numeron. Konkreettinen laskutoimitus voisi olla esimerkiksi:

X_{origo} = 50000, Y_{origo} = 660000, X_{min} = 529296, Y_{min} = 6976184

$X_{diff} = X_{min} - X_{origo}$ = 479296, $Y_{diff} = Y_{min} - Y_{origo}$ = 376176

$X_{col} = X_{diff} / 16$ = 29956, $Y_{row} = Y_{diff} / 16$ = 23511

Solun numero = ’23511’ + ’29956’ = ’2351129956’

Yksittäisten puiden sijainnilla ei ole merkitystä strategisessa suunnittelussa, joka keskittyy tyypillisesti hilatason tai kuviotason puustotietoon. Puiden yksilöllinen seuranta on kuitenkin mainittu yhtenä edellytyksenä täsmämetsätalouden saavuttamiseksi; seuraamalla puiden kasvua yksittäisten puiden tasolla voidaan valita vain kaikkein optimaalisimmat puut hakkuihin. Yksinkertaisin tapa ilmaista puun sijainti on pelkkänä koordinaattipisteenä, mutta ongelmana on näiden koordinaattien paikantaminen muuten kuin perinteisillä kenttämittausmenetelmillä. Kuten tulemme huomaamaan, lentolasertekniikoilla voidaan estimoida puun sijainti latvustosta, mutta tähän mennessä estimaatit eivät ole olleet tarkkoja. Hakkuukone paikantaa puun hakkuukoneen GPS-paikantimen ja kouran asennon perusteella, mutta sijaintitieto muuttuu hakkuusuunnittelun kannalta hyödyttömäksi, koska puu kaadetaan paikannuksen yhteydessä. Vanhaa sijaintitietoa voidaan luultavasti käyttää muiden paikannusmenetelmien kalibrointiin. Holopainen et al. [3] esittävät, että korjuukoneet tallentavaisivat ensimmäisen harvennuksen yhteydessä metsään jäävien puiden sijainnit maastolaserkeilauksen avulla.

2.2 Puustotunnusten mittausmenetelmät

2.2.1 Perinteiset kenttämittaukset

Vaikka erilaiset kartoittamisteknologiat ovat tehostaneet metsävaratiedon hankkimista, puustotunnuksia kerätään edelleen ns. perinteisillä menetelmillä, joilla tarkoitetaan tunnusten mittaamista suoraan puista käsityökaluilla. Kenttämittauksissa määritetään tyypillisesti puulaji, korkeus ja läpimitta, ja mittaukset kohdistuvat yhteen puuhun kerrallaan koealueella. Koealueen tuloksista voidaan johtaa puustotietojen estimaatteja kuten puustotunnusten keskiarvot, runkolukusarja ja pohjapinta-ala sekä tutkitulle koealueelle että suuremmille alueille metsikkötasolta alueelliselle ja jopa laajemmin. Pienetkin mittausvirheet yksittäisten puiden mittauksissa voivat tästä syystä moninkertaistua etenkin suurempia alueita estimoidessa. [6]

Kenttämittauksien tarkkuuksista on tehty useita tutkimuksia, joista tässä työssä tarkastellaan kahta: Hyyppösen ja Roiko-Jokelan [7] vanhempaa tutkimusta vuodelta 1978 sekä Luoma et al. [8] tuoreempaa tutkimusta vuodelta 2017. Molemmissa selvitettiin korkeuden ja DBH-läpimitan mittausvirhettä. Mittauksien harhaa ei voitu kuitenkaan määrittää, koska puiden tarkkoja mittoja oli käytännössä mahdotonta selvittää. Hyyppösen ja Roiko-Jokelan tutkimuksessa [7] puut mittasi kaksi kokenutta mittaajaa, jotka käyttivät korkeuden mittaamiseen mittatankoa, suuntakehää ja hypsometriä, ja läpimitan mittaamiseen mittasaksia. Vastaavasti Luoma et al. tutkimuksessa [8] mittaajia oli neljä, jotka käyttivät korkeuden mittaamiseen klinometriä ja läpimitan mittaamiseen läpimittanauhaa. Jälkimmäisessä tutkimuksessa tähdennettiin myös ristimittausmenetelmän käyttöä, eli läpimitta mitattiin kahdesti niin,

että mittauskohdat olivat kohtisuorassa kulmassa toisiinsa nähden. Tällöin puunmuodolla oli pienempi vaikutus mittausvirheeseen. [7] [8]

Luoma et al. [8] tutkimuksessa mittaajien välinen keskihajonnan keskiarvo puiden läpimitoissa oli 0,3 cm (1,5%) ja puiden korkeuksissa 0,5 m (2,9 %). Suurimmat yksittäisten mittausten erot mittaajien välillä olivat 2,1 cm läpimitalle ja 4,2 m korkeudelle. Tutkimuksessa kuitenkin mainitaan 80,6% puiden läpimitoista eronneen alle 1 cm mittaajien välillä. Mittaajien käyttämä ristimittausmenetelmä paransi läpimitan mittaustarkkuutta: Läpimitan keskihajonnan keskiarvo oli yhden mittauksen menetelmällä 2,2% eli noin 1,5 kertaa suurempi kuin ristimittausmenetelmällä. Hyyppösen ja Roiko-Jokelan [7] tutkimuksessa läpimitan keskihajonta oli 0,27 cm (1,4%) ja korkeuden keskihajonta vaihteli 0,2m – 1,08m (1,6% - 7,7%) välillä riippuen mittausmenetelmästä. Molempien suureiden mittauksessa puulajilla on suurin merkitys mittaustarkkuuteen: Luoma et al. [8] mukaan männyn kaarna vaikeuttaa sen läpimitan määrittämistä verrattuna kuuseen ja koivuun, kun taas koivun lehdistö vaikeuttaa puun latvan ja siten puunkorkeuden määrittämisessä. Lisäksi korkeuden mittauslaitteella oli suuri merkitys, kuten tutkimuksessa [7] huomattiin. Mittaajien kokeneisuus vähentää virhettä, joten molempiin tutkimuksiin oli valittu kokeneet mittaajat. [7] [8]

2.2.2 Lentolaserkeilaus

Lento- tai ilmalaserkeilaus, johon viitataan lyhenteellä ALS (Airborne Laser Scanning), perustuu keilauksen infrapunapulssien tulkintaan laserin heijastuessa takaisin puiden lehdistä, puunrungoista sekä muusta kasvillisuudesta. Sen avulla metsästä voidaan luoda kolmiulotteinen pistepilvi, jota analysoimalla voidaan estimoida puustoattribuutteja. Laserkeilauksella saatava puustotieto on riittävän tarkkaa strategista metsäsuunnittelua varten, ja sitä on käytetty yksityismetsien inventointiin Suomessa vuodesta 2010 lähtien. Yksityisyritykset sekä Metsähallitus ovat käyttäneet ALS:ää jopa tätä aikaisemmin. Lentolaserkeilaus menetelmiä on kaksi: yksinpuintulkinta (ITD, individual tree detection) ja aluepohjainen menetelmä (ABA, area-based approach). [3]

Aluepohjaisessa menetelmässä otosyksikkö on hilasolu. Menetelmässä puustotunnukset määritetään pistepilvestä laskettujen tunnusten sekä maastokoealoilta tarkasti määritettyjen puustotunnusten tilastollisen riippuvuuden perusteella. Yleistä mallia riippuvuudelle ei ole, sillä keilausparametrit, kuten lentokorkeus, keilauskulma ja pulssitiheys, vaikuttavat malliin keilauskohtaisesti. Siksi riippuvuusmalli joudutaan määrittämään aineistokohtaisesti, ja malleja on tutkittu mm. regressioanalyysilla, NN-ennustusmenetelmillä, bayesilaisella lähestymistavalla sekä neuroverkoilla. Ilmakuvia voidaan hyödyntää mallien tukimateriaalina estimoinnin parantamiseksi. [9] [10]

Yksinpuintulkinnassa pistepilvestä erotellaan lokaalit maksimit, jotka voidaan tulkita puiden latvoiksi. Maksimista voidaan mitata puun pituus. Tosin näin määritetyssä pituudessa on ilmennyt systemaattista aliarviota, koska etenkin lehtipuissa keilauksen pulssit läpäisevät ylimmät lehdet ennen kunnollista heijastumista. Latvoista voidaan määrittää myös puun sijainti, mutta tunnistustulokset ovat olleet toistaiseksi epäluotettavia: keskimäärin pistepilvestä tunnistetaan lokaaleja maksimeja 60-70% puiden lukumäärästä, mutta vaihteluväli on noin 20%-90%, ja puiksi tunnistetuista maksimeista pahimmillaan jopa puolet eivät ole olleet todellisia puita [9]. Virheet kohdistuvat kuitenkin ensisijaisesti alemman jakson pienempiin puihin; korkeimmat puut pystytään paikantamaan luotettavasti. Puun läpimittaa ei saada suoraan pistepilvestä, mutta sitä voidaan ennustaa Holopaisen ym. [3] mukaan allometrisillä malleilla, lokaaleilla regressiomalleilla sekä NN-menetelmillä. Pistepilvestä voidaan arvioida kohteen puulaji noin 74-96% tarkkuudella [9]. Yksinpuintulkinnassa maastoreferenssin tarve on pienempi verrattuna ABA:an ja

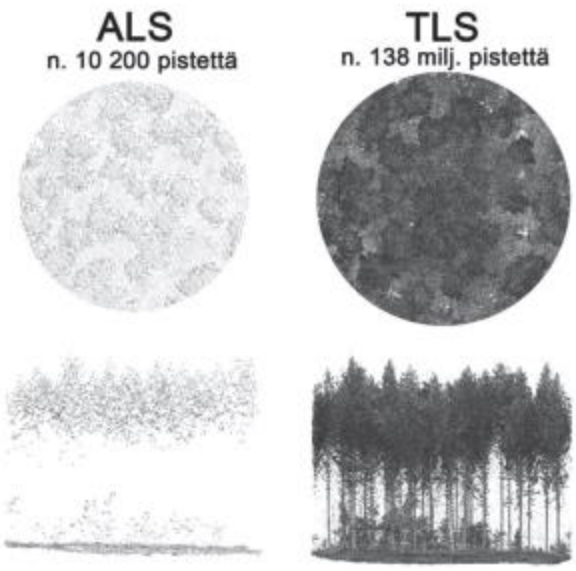
runkolukusarjaa voidaan päätellä osittain suoraan saadusta pistepilvestä, mutta se vaatii ABA:an verrattuna korkeampaa pulssitiheyttä (> 2 pulssia / m2), joka edellyttää tehokkaampia ja kalliimpia laitteita. [3] [9] [10]

Toistaiseksi metsäsuunnittelussa on käytetty ABA-menetelmää sen tarkemman estimointikyvyn ja halvempien laitteiden ansiosta. Sillä on myös saavutettu joissain tutkimuksissa puustotietoja estimoidessa yhtä hyviä tarkkuuksia kuin perinteisiin maastomittauksiin perustuvalla kuvioittaisella arvioinnilla. Holopaisen ym. kirjassa [3] mainitaan erään tällaisen tutkimuksen saaneen keskiläpimitan, keskipituuden ja tilavuuden keskineliövirheen neliöjuureksi (RMSE) ABA-menetelmällä 9,5 %, 5,3 %, ja 9,8. Vastaavasti ITD:llä päästiin Maltamo et al. tutkimuksessa [11] k-MSN estimoinnilla 5,2 %, 2,0 % ja 11% RMSE-arvoihin. Tutkimus kuitenkin perustui pieneen määrään manuaalisesti tai puoliautomaattisesti määritettyjä puita ja kohdistui pelkästään mäntyihin. Tästä syystä tulosten spekuloitiin kuvastavan saavutettavan tarkkuuden ylärajaa ja viittaisi ITD:llä päästävän ABA:a parempiin estimaatteihin samalla kalibrointi- ja maastomittausmäärällä. ITD-menetelmän etuna on myös runkoluvun saaminen suoraan mittaustuloksista sekä teoriassa yksittäisten puiden paikantaminen. Sekä ABA:ssa että ITD:ssä on esiintynyt pientä systemaattista aliarviota puiden korkeuksien mittauksessa. [3]

2.2.3 Maastolaserkeilaus

Puustoa voidaan laserkeilata myös maanpinnalta. Staattisen keilausalustan keilauksesta käytetään termiä TLS (Terrestrial Laser Scanning), kun taas liikkuvan keilausalustan, kuten auton, hakkuukoneen tai repun, keilauksesta käytetään termiä MLS (Mobile Laser Scanning). Molempien toimintaperiaate on sama kuin ALS:n, mutta koska keilaus kohdistuu lentolaserkeilausta paljon pienemmälle osalle puustoa ja pidemmän aikaa, saadut pistepilvet ovat huomattavasti tiheämpiä ja tarkempia. ALS ja TLS pistepilvien erot on esitetty kuvassa 3. Maastolaserkeilaus keskittyy myös hieman eri puustoattribuutteihin kuin ALS: Ilmasta tutkitaan erityisesti puiden pituutta ja tiheyttä, kun taas TLS/MLS tutkii puiden läpimittaa, latvusten leveyttä ja aluskasvillisuutta. [3] [12]

Maastolaserkeilauksen tarkkuus on samaa suuruusluokkaa kuin perinteisissä kenttämittauksissa. Bauwens ym. [4] keilasivat ja analysoivat kahdeksan koealueen puiden läpimitat kolmella keilausmenetelmällä. Yksinkertainen keilaus ei tuottanut hyviä tuloksia (RMSE 13,4%), mutta useamman keilauksen pistepilvistä saatiin kahdella muulla menetelmällä perinteisiä kenttämittauksia vastaavia tuloksia (RMSE 4,7% ja 4,1%). Yksinkertainen keilausmenetelmä oli kuitenkin paljon nopeampi; keilaus kesti korkeintaan 10 minuuttia, kun taas muilla menetelmillä aikaa saattoi kulua yli tunti. Yksinkertaisen keilauksen pistepilvi pystyttiin analysoimaan noin 10 minuutissa, kun muilla menetelmillä analysointiin kului yli tunti. Perinteisillä mittausmenetelmillä puutunnusten mittaamiseen kului 30 minuuttia ja tulosten analysointiin 10 minuuttia. [4]



Kuva 3: ALS ja TLS pistepilvien vertailu [12]

TLS/MLS soveltuu huonosti suurien metsäalueiden kartoittamiseen pienen keilausalansa vuoksi, joka rajoittuu maanpinnalta havainnoitavaan metsään. Metsien inventoinnin sijasta Holopainen ym. [3] esittävät TLS/MLS:ää hyödynnettävän operatiivisessa suunnittelussa harvennettavien puiden valitsemisessa, jäävän puuston arvioimisessa sekä runkojen optimaalisen katkonnan tarkentamisessa. On myös ehdotettu, että keilauskamera asennettaisiin suoraan metsäharvesteriin, joka keilaisi ympäröivää metsää ja/tai metsäkoneen muodostamaa ajouraa ajonaikaisesti.

2.2.4 Satelliittikuvat

Laserkeilauksen lisäksi metsävaroja voidaan estimoida SAR-satelliittien (Synthetic Aperture Radar) ottamien puuston kuvien perusteella. Satelliittikuvien etuina ovat kuvien saatavuus, ajankohtaisuus ja halvemmat kustannukset kuin ALS-tyyppisellä kuvaamisella. Kuvien heikkoutena on huomattavasti epätarkempi paikkaresoluutio verrattuna laserkeilaukseen. Holopainen ym. [10] esitettävät satelliittikuvien mahdollisiksi käyttötarkoituksiksi metsien äkillisten muutosten päivittäisen seurannan, esimerkiksi selvittämään myrskytuhoja tai hakkujälkeä, sekä ALS:n tulkinnan tukemisen. Artikkelissa mainitaan myös mahdollisuuksista tulkita satelliittikuvista metsätunnuksia interferometrisellä korkeusmittauksella, jossa kuvista voidaan luoda kolmiulotteisia malleja satelliitin ottaessa kuvat eri suunnista. Tekstissä kuitenkin korostetaan, että satelliittikuvista tulkituilla pistepilvillä on paljon laserkeilausta harvempi resoluutio, mikä johtaa huonompaan estimointitarkkuuteen. Satelliittikuvat tuskin tulevat korvaamaan nykyisiä laserkeilausmenetelmiä lähitulevaisuudessa.

3. Metsävaratiedon tallennus ja tiedonsiirto

Tässä kappaleessa esitellään suomalaisia metsävaratiedon standardeja ja tietokantoja, selvitetään niiden käytetyt tietorakenteet, ja vertaillaan niitä toisiinsa. Tarkoituksena on tarjota pohjatieto mahdollisimman kattavan koostetun tietorakenteen luomiseksi, jota voidaan hyödyntää palvelualustalla. Koostettua tietorakennetta käsitellään tarkemmin kappaleessa 6.

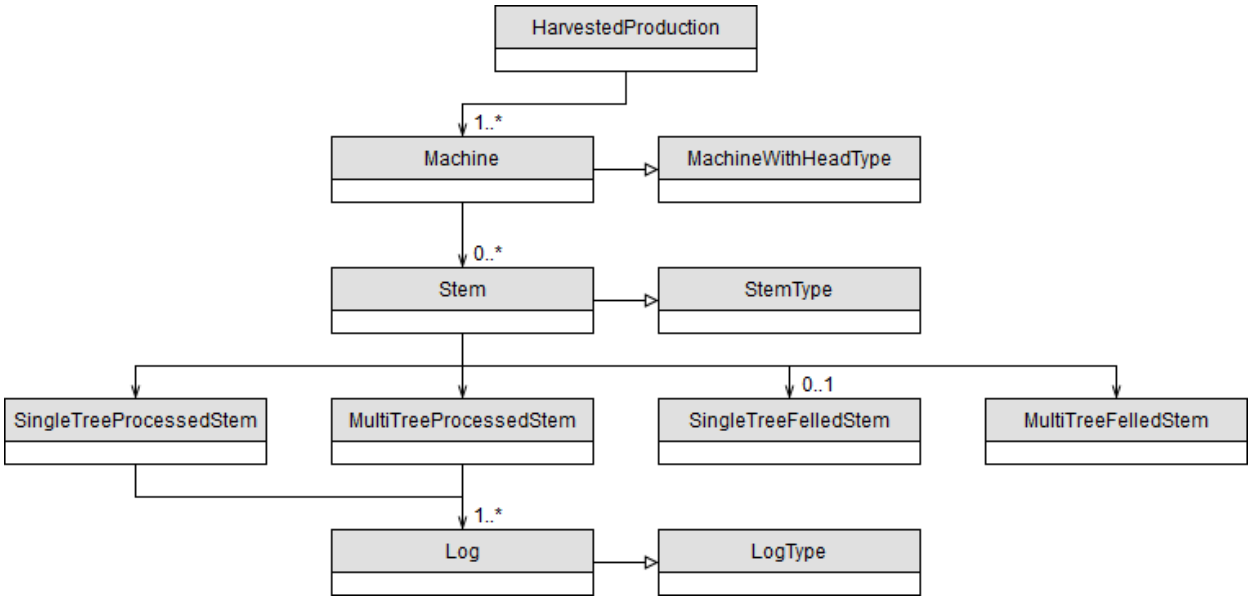
3.1 Standardit

Tähän työhön on poimittu kaksi suomalaista metsävaratiedon standardia: StanForD2010 ja Metsäkeskuksen metsävaratiedon standardi. Niistä tehdyt havainnot perustuvat niiden XML-skeemoihin [13] ja [14]. XML-skeemoja esitellään lyhyesti luvussa 4.1.

3.1.1 StanForD2010

StanForD2010 on Metsätehon, Skogforskin ja konevalmistajien kehittämä standardi. Se on alun perin lähtöisin vanhemmasta StanForD-standardista, mutta tässä työssä StanForD:lla viitataan aina StanForD2010-standardiin. Sitä käytetään metsähakkuiden hallinnointiin ja raportointiin, ja se perustuu metsäkoneille, puille, pölkyille ja muille objekteille annettaviin identiteetteihin, joiden avulla objekteja voidaan yksilöidä ja seurata. Tiedonsiirto kategorisoidaan viestityyppeihin viestien käyttötarkoituksien perusteella. Tälle diplomityölle tärkein viestityyppi on HPR (Harvested Production), jolla siirretään hakkuiden tukkipuiden tietoja. Standardissa käytetään WGS84-koordinaatistoreferenssiä. [15]

HPR käsittelee yksityiskohtaisesti hakkuukoneen käsittelemiä puunrunkoja ja niistä saatuja pölkkyjä. Yksittäinen HPR viesti kohdistuu aina yhteen hakkuuseen, muttei välttämättä vain yhteen hakkuukoneeseen; viestiin voi sisällyttää kaikkien hakkuuseen osallistuneiden hakkuukoneiden keräämät hakkuutiedot. HPR:n viestirakenne on esitetty yksinkertaistettuna kuvassa 4. Kuvan lukumääräsuhteet perustuvat XML-skeemassa mainittuihin arvoihin. SingleTreeProcessedStem:lle, MultiTreeProcessedStem:lle ja MultiTreeFelledStem:lle ei löytynyt lukumääräsuhteita.



Kuva 4: Yksinkertaistettu kuva HPR:n viestirakenteesta.

HarvestedProduction sisältää metatiedon StanForD2010-versiosta, sekä tiedot pinta-alan (ha), läpimitan (cm), pituuden (cm), tilavuuden (m^3) ja painon (kg) mittayksiköistä. Siihen sisältyy kuvassa mainittu Machine, MessageHeader, joka sisältää metatiedon viestin luojasta ja päivämäärästä, ja Extension, jolla viestiin voi lisätä vapaamuotoista lisätietoa. Machine viittaa hakkuukoneeseen ja periytyy MachineWithHead-luokasta, joka sisältää laitteen valmistajan ja mallin. Itse Machine sisältää tiedot hakkuun puulajeista, metatietoa työkoneesta sekä mitä puita hakkuukoneella kaadettiin tai käsiteltiin. Työkoneella kaadetut ja prosessoidut puut sisältyvät Stem-luokkaan. StemType, josta Stem periytyy, sisältää puunrunkoa tai runkoja yksilöiviä avaimia, korjuupäivämäärän, sijainnin (korjuukoneen sijainti + kouran asento) sekä muita lisätietoja. Stem:n sisältämä puutieto on jaettu kuvassa esitetysti neljään luokkaan. Luokkiin tallennetaan rungon DBH, tilavuus ja rungosta katkotut pölkyt SingleTreeProcessedStem:n ja MultiTreeProcessedStem:n tapauksessa. Kuten StemType, LogType sisältää pölkyn yksilöiviä avaimia sekä lisätietoa pölkyn leikkaustavasta, mutta lasketut mittasuhteet sisältyvät Log-luokkaan. Edellä esitetyn hierarkian takia yhdestä pölkystä voidaan jäljittää, kuka pölkyn on katkonut, milloin se on katkottu, mistä puunrungosta ja millä hakkuukoneella.

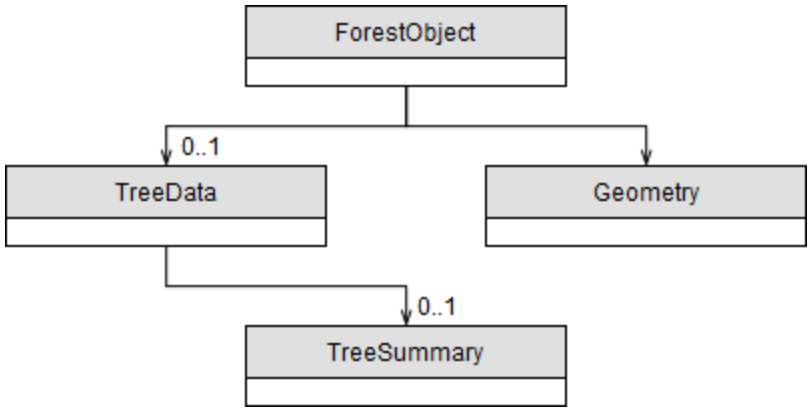
HPR:ssä ei tallenneta puun tarkkaa koordinaattitietoa tai muuta paikkatietoa. Tämä voi hankaloittaa puustotiedon yhdistämistä muihin tietolähteisiin pelkästään HPR-tietorakenteen avulla.

3.1.2 Metsäkeskuksen metsävaratiedon standardi

Metsävaratiedon standardi on Suomen vanhin metsätietostandardi, jonka ensimmäinen versio valmistui vuonna 2009. Se sisältää määrittelyt mm. puustolle, maaperälle, hakkuiden puutavaralajeille ja käytön rajoituksille. Standardi sisältää StanForD:n tapaan myös hakkuiden toiminnanohjauksen ja puukaupan tiedot, mutta tässä työssä käsitellään vain metsävaratiedon esitykseen tarkoitettuja XML-skeemoja. Standardissa käytetään TRS-TM35FIN koordinaatistoreferenssiä.

Standardin skeemapaketissa on kaksi kandidaattia metsävaratiedon siirtämiseen: ForestObject (FO) ja ForestData (FD). Molemmilla voidaan esittää jonkin metsäkuvion aluerajat ja puustoattribuutit. FO:lla esitetään aina yksittäinen kuvio, kun taas FD sisältää yhden omistajaryhmän useita metsäkuvioita. FO on sisällöltään yksinkertaisempi eikä sisällä metsäkuviolle tehtyjä operaatioita tai erikoisominaisuuksia. Seuraavaksi esitellyt hierarkiat perustuvat skeemapaketin versioon 9.24, joka on toistaiseksi uusin virallistettu ja lukittu versio Seuraava 10.13 versio on toistaiseksi vielä työn alla. Skeemojen laajuuden takia kuvista on jätetty pois puustokerroksia käsittelevät luokat.

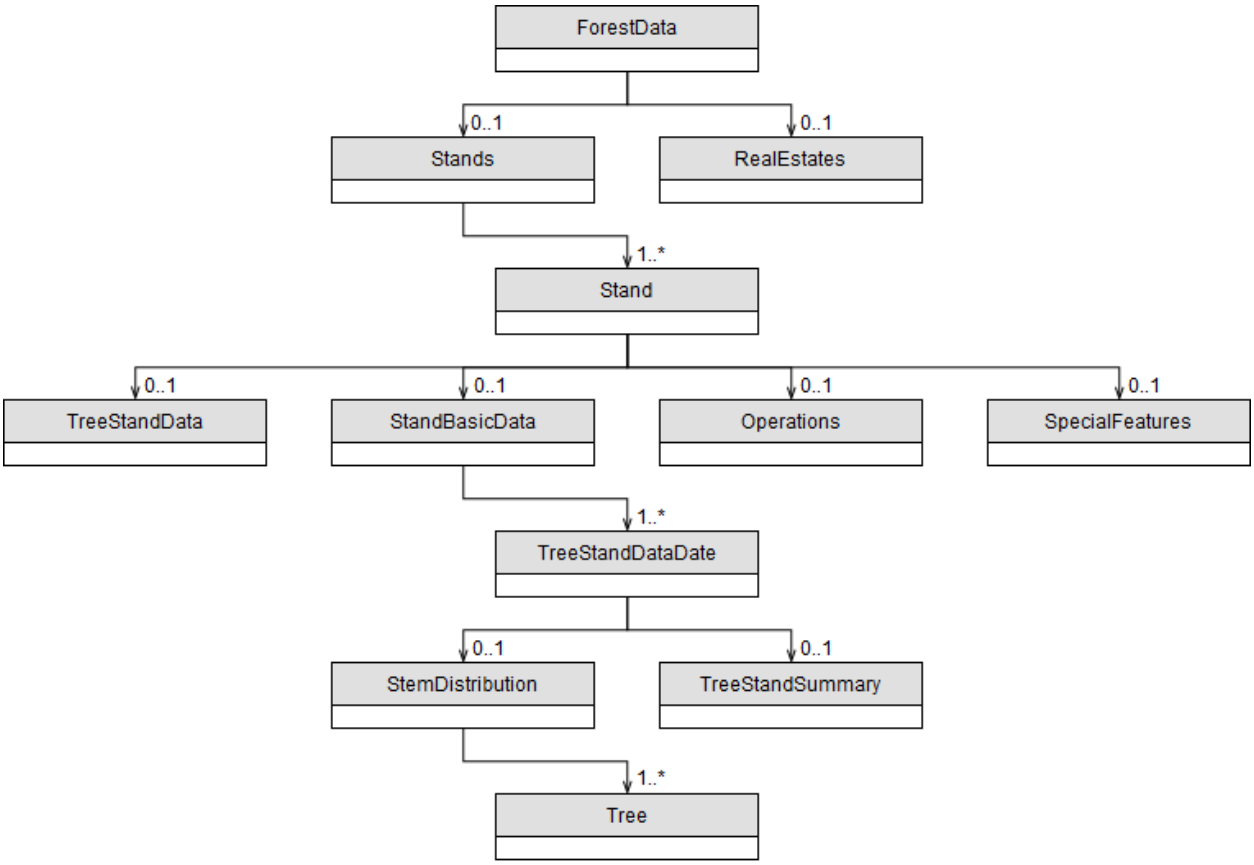
Kuvassa 5 on esitetty ForestObject:n rakenne. FO sisältää paljon yleistä tietoa kuviosta, kuten sen maalajin ja pääpuulajin, sekä kuvion puustotunnukset (TreeData) ja alueen rajat (Geometry). Vaihtoehtoina rajageometrian esittämiseksi ovat monikulmio ja monikulmiojoukko, mutta ei hilasolua. Puustotietoon kuuluu alueen puuston keskitunnukset (TreeSummary) ja puujaksot (Storeys). Jaksoihin on sisällytetty yksittäisten puiden sijainnit, mutta ei muita puukohtaisia attribuutteja.



Kuva 5: Yksinkertaistettu ForestObjectin rakenne

Kuvassa 6 on esitetty FD:n rakenne. Hierarkian ylimpänä kerroksena on ForestPropertyData. Yksittäinen instanssi koostuu omistetuista metsäkuvioista (Stands) ja sen omistajaryhmän tiedoista (RealEstates). Metsäkuviot luonnollisesti koostuvat yhdestä tai useammasta metsäkuviosta (Stand), joille on määritelty puustotiedot (TreeStandData), kuvion perustiedot (StandBasicData), tehdyt toimenpiteet (Operations) ja erikoisuudet (SpecialFeatures). Perustiedot käsittävät kaiken ei-puustotiedon metsäkuviosta, kuten alueen sijainnin, pinta-alan ja muodon, maaperätiedon, harvennus- ja ojitusvuodet, sekä toimenpiderajoitukset. Toimenpiteet on jaettu Operations-luokassa kahteen päätyyppiin, metsähoitoon ja hakkuuseen. Lisäksi se sisältää mm. toimenpiteille ehdotetut ajankohdat, hakkuissa saatu puumäärä tai sen arvio, ja toimenpiteen tarkennukset. Puustotiedot koostuvat kuvion nykyisestä ja menneistä puustotiedoista. TreeStandDataDate käsittää yhden ajankohdan puustotiedot, sisältäen ajankohdan lisäksi puuston runkolukusarjan (StemDistribution), yhteenvedon puustoattribuuteista (TreeStandSummary) ja puustokerrokset. Yhteenveto sisältää puuattribuuttien keskitunnuksia, puiden lukumäärän, pohjapinta-alan, tilavuuden, kaupallisen arvon, eri puunosien biomassat, sekä valtapuulajin. StemDistribution koostuu yhdestä tai useammasta puusta (Tree), joille on määritelty mm. mittasuhteet, laji, ikä, tilavuus (kokonais-, tukki- ja kuitupuutilavuudet) sekä viimeisin muokkauspäivämäärä. Vaikka puita voidaan yksilöidä ID:n avulla, ForestData ei tue puun sijaintitietoa.

ForestObject:ssa metsäkuvio ilmoitetaan monikulmion ja kuviotyypin avulla. ForestDatassa alue voi koostua useammasta monikulmiosta, koska esitettävä alue voi rajata useasta pienalueesta koostuvan metsäalueen omistusrajat. Hilasoluja ei voida esittää hilanumeroilla. Metsätietostandardi on kuitenkin siirtymässä Metsäkeskuksen mukaan [14] GeoPackage-formaattiin ja tukemaan hilasolumuotoista tiedonsiirtoa. Kyseinen standardi on valmis, mutta sitä ei tätä työtä tehdessä oltu hyödynnetty laajamittaisessa käytössä eikä se ollut saatavilla tutkittavaksi.



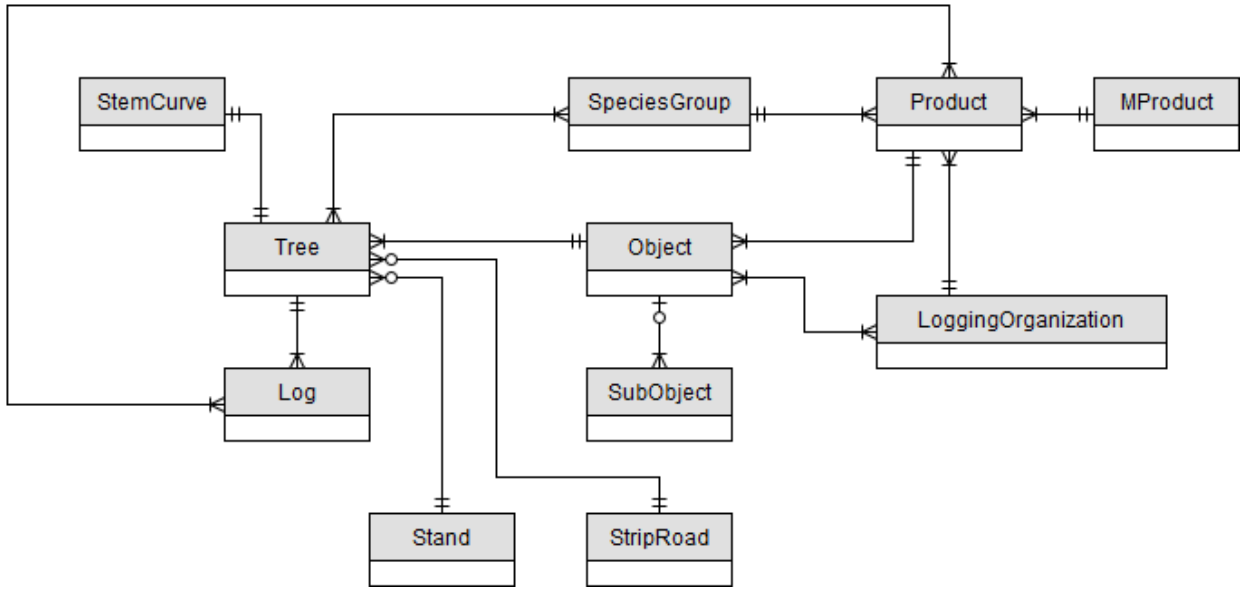
Kuva 6: Metsätietostandardin ForestData:n rakenne

3.2 Tietokannat

3.2.1 Hakkuukonetietokanta

Metsätehon hakkuukonetietokantaan on koottu saantotiedot tehdyistä hakkuista. Se on osa palvelualustan tietokantapilottia, ja oli tämän diplomityön aikana vielä kehitteillä. Sillä ei tiettävästi ole tarkkaa päivityssykliä, vaan saantotietoja lisätään tietokantaan aina, kun hakkuu on prosessoitu. Koska kyseessä on SQL-tietokanta, sen rakenteesta voidaan muodostaa relaatiokaavio, joka on esitetty kuvassa 7. Kaavio saattaa olla vanhentunut tai puutteellinen, koska tietokantaa kehitettiin vielä työtä tehdessä.

Rakenne mukailee pitkälti StanForD2010-standardin HPR-viestirakennetta: Työkoneen saanto lajitellaan puulajityypeittäin puihin ja niistä saatuihin tukkeihin. HPR-viestirakenteesta poiketen puut kuuluvat yhteen kuvioon (Stand) ja ajouraan (StripRoad), josta saantotieto on peräisin. Näiden kahden luokan sisältöä ei oltu tätä työtä tehdessä vielä laadittu, mutta muiden luokkien sisällöt vastasivat lähes täysin StanForD2010-standardia.



Kuva 7: Hakkuukonetietokannan relaatiokaavio. Alkuperäisen kaavion on laatinut Metsäteho.

3.2.2 Metsäkeskuksen tietokanta

Metsäkeskuksen (SMK) tekemä metsien inventointi perustuu kaukokartoitukseen (laserkeilauksella tai ilmakuvauksella), koealamittauksiin ja kohdennettuun maastoinventointiin. Metsäkeskuksen tietokanta sisältää näistä mittausdatoista tulkittuja hilasolukohtaisia puustotunnuksia. SMK:n tavoitteena on inventoida 1,5 milj. hehtaaria vuotuisesti, jolloin sama alue tulisi tulkittua minimissään 10 vuoden välein. Ote yhden hilasolun tietosisällöstä on esitetty taulukossa 1.

Taulukko 1: Esimerkki Metsäkeskuksen hilamuotoisesta metsävaratiedosta

Sarake	**Kuvaus**	**Esimerkki**
Kuvio_ID	Kuvion tunniste	10_1
HILARUUTU_ID	Hilaruudun ID	104116416
HILARUUTU_NRO	Hilaruudun numero	651718088
KASVUP_LK	Kasvupaikkaluokka	2
MAALAJI	Maalaji	20
T_MA	Mäntyjen ikä, v	94
G_MA	Mäntyjen pohjapinta-ala m2/ha	13,42
N_MA	Mäntyjen lukumäärä/ha	200
D_MA	Mäntyjen keskiläpimitta, mm	35,69
H_MA	Mäntyjen keskipituus, cm	25,35
V_MA	Mäntyjen tilavuus m3/ha	118,95
T_KU	Kuusien ikä, v	75
G_KU	Kuusien pohjapinta-ala m2/ha	12,95
N_KU	Kuusien lukumäärä/ha	323
D_KU	Kuusien keskiläpimitta, mm	27,61
H_KU	Kuusien keskipituus, cm	21,29
V_KU	Kuusien tilavuus m3/ha	145,09
T_LP	Lehtipuun ikä, v	55
G_LP	Lehtipuun pohjapinta-ala m2/ha	0,5
N_LP	Lehtipuun lukumäärä/ha	24
D_LP	Lehtipuun keskiläpimitta, mm	19,82
H_LP	Lehtipuun keskipituus, cm	18,84
V_LP	Lehtipuun tilavuus m3/ha	18,44
CELLSIZE	Hilaruudun koko m2	256

Koska tietokantaan tallennetaan puustotietoa hakkuista riippumattomasti, siihen ei sisälly saantotietoa, eli esimerkiksi tukkien tilavuuksia. Dokumentaatiosta ei myöskään löytynyt suoraa mainintaa hilanumeroiden referenssikoordinaatistosta. Toisaalta koska kyseessä on Suomen Metsäkeskuksen tietokanta, tietokannassa oletettavasti käytetään ETRS-TM35FIN-referenssiä eli samaa kuin metsävarastandardissa.

Metsätieto 2020 tavoitetilan raportissa [5] mainitaan Metsäkeskuksen koealakohtaiset puiden korkeuden, läpimitan, pohjapinta-alan ja tilavuuden RMSE-% tavoitetasot. Koska yhden koealan pinta-ala (tyypillisesti 9 metrin säteinen ympyrä) on lähes sama kuin hilasolun, tavoitetasojen voidaan olettaa rinnastuvan myös hilamuotoisen metsävaratiedon tarkkuuksiin. Tavoitetasot on koottu taulukkoon 2.

Taulukko 2: Tärkeimpien puustotunnuksien RMSE-% tavoitetasot.

h = korkeus, dbh = puun läpimitta rinnankorkeudelta, ppa = pohjapinta-ala, v = tilavuus.

	h	dbh	ppa	v
Kokonaispuusto	10	15	20	20
Pääpuulaji	10	20	30	30
Alueen yleisin havupuulaji	15	25	35	35
Toinen havupuulaji	15	25	40	40
Lehtipuu	20	30	45	45

3.2.3 LUKE:n MVMI

Luonnonvarakeskuksen (LUKE) monilähteinen valtakunnan metsien inventointi (MVMI) käyttää hyväksi maastotietojen lisäksi Landsat TM -satelliittikuvia ja muita numeerisia tietolähteitä, ja sisältää rasterikuvia koko Suomen metsävaroista hilamuodossa. Rasterikuvat ovat julkisesti saatavilla GeoTIFF-muodossa. Aineiston kapasiteetin vuoksi metsävaratieto on jaoteltu UTM200-karttalehtijakoon. Yhdellä kyselyllä tietokannasta voi hakea vain yhden karttalehtijaon mukaisen alueen (esimerkiksi M4), ja hakutulokseen sisältyy vain yksi puustosuure, esimerkiksi männyn tilavuus.

Kuvassa 8 on esitettynä ote MVMI:n sisällöstä visualisoituna QGIS-ohjelmalla. Kuvaan on otettu kaksi puustosuuretta: männyn tilavuus ja koivun tilavuus. Yksi pikseli kuvassa vastaa yhtä metsävaratiedon hilasolua. Pikselin sävy kuvastaa tilavuuden arvoa; tummempi sävy ilmaisee suurempaa arvoa.

Kuva 8: MVMI-rasterikuva. Vasemmalla puoliskolla on esitettynä männyn tilavuus, oikealla puoliskolla koivun.

Kuvasta voi havaita myös järvet ja tiet. QGIS:llä ei voinut tutkia yksittäisen solun arvoa, vaan havainnointi rajoittui solun väriarvoon. Kuvan tummin väri vastaa 168 m^3/ha:n tilavuutta.

3.3 Tietokantojen ja standardien vertailu

Alla on taulukoitu esiteltyjen standardien ja tietokantojen vastaavuuksia ja eroja. Taulukko 3 käsittää tietorakenteiden metatiedon, kun taas taulukkoon 4 on koottu yleisimmät puustotunnukset, sekä miten ja missä yksikössä ne tallennetaan tietorakenteisiin. Myös yksittäisten puiden seuranta otettiin mukaan vertailuun, vaikkei se varsinaisesti kuulu strategiseen metsäsuunnitteluun. Standardien ja tietokantojen soveltuvuutta koostettuun tietorakenteeseen pohditaan tarkemmin kappaleessa 6.

Taulukko 3: Tietokantojen ja standardien vertailu metatasolla

	StanFord2010	**FO/FD**	**HakkuukoneTK**	**MetsävaraTK**	**MVMI**
Formaatti	XML	XML	XML	OGC GeoPackage tai XML	GeoTIFF
Koordinaattijärjestelmä	WGS84	ETRS-TM35FIN	WGS84	ETRS-TM35FIN	ETRS-TM35FIN
Metsäkuviot hilasoluina	Ei	Ei	Ei	Kyllä	Kyllä
Metsäkuviot monikulmioina	Ei	Kyllä	Ei	Ei	Ei
Yksittäisten puiden seuranta	Kyllä	FO: Vain sijainti / FD: Vain attribuutit	Kyllä	Ei	Ei
Sisältää puustomittauksia	Ei	Kyllä	Ei	Kyllä	Kyllä
Sisältää hakkuutietoa	Kyllä	Kyllä	Kyllä	Ei	Ei

Formaatilla tarkoitetaan tiedonsiirtomuotoa, jossa standardia yleensä käytetään tai missä muodossa tietokannasta voidaan hakea tietoa. HakkuukoneTK:n formaatti perustuu olettamukseen, että siinä käytetään lopulta samaa XML-pohjaista StanForD-2010-standardia kuin mihin se perustuu.

Taulukko 4: Tietokantojen ja standardien vertailu tärkeimpien puustoattribuuttien kannalta

	StanFord2010	**FO/FD**	**HakkuukoneTK**	**MetsävaraTK**	**MVMI**
Korkeus tai pituus, yksikkö	Yksittäinen puu, cm	Yksittäinen puu ja alueen keskiarvo, m	Yksittäinen puu, cm	Lajeittain keskiarvo, cm	Rasterikuva
Läpimitta, yksikkö	Yksittäinen puu, mm	Yksittäinen puu ja alueen keskiarvo, cm	Yksittäinen puu, mm	Lajeittain, mm	Alueen keskiarvo, cm
Tilavuus (koko puu), yksikkö	Yksittäinen puu, m^3	Yksittäinen puu ja alueen keskiarvo, m^3/ha	Yksittäinen puu, m^3	Lajeittain, m^3/ha	Lajeittain, m^3/ha
Tilavuus (tukkiosuus), yksikkö	Yksittäinen puu, m^3	Yksittäinen puu ja alueen keskiarvo, m^3/ha ja %	Yksittäinen puu, m^3	Ei	Lajeittain, m^3/ha
Tilavuus (kuituosuus), yksikkö	Yksittäinen puu, m^3	Yksittäinen puu ja alueen keskiarvo, m^3/ha	Yksittäinen puu, m^3	Ei	Lajeittain, m^3/ha
Puiden lukumäärä	Hakkuukohtainen	Kuviokohtainen	Hakkuukohtainen	Solukohtainen	Solukohtainen
Runkolukusarja	Ei	Kyllä	Ei	Kyllä	Ei
Pohjapinta-ala	Ei	Kyllä	Ei	Kyllä	Kyllä
Yksittäisten puiden sijainti	Pistekoordinaatti + ID	FO: Pistekoordinaatti / FD: ID	Pistekoordinaatti + ID	Ei	Ei
Puuston ikä	Ei	Yksittäinen puu ja alueen keskiarvo	Ei	Lajeittain	Alueen keskiarvo

4. Tiedonsiirtoteknologiat

Palvelualusta tulee hyödyntämään Amazonin Azure-pilven adaptereita, joiden avulla useat rajapintateknologiat voivat käyttää samoja palveluita. Tämä tekee teknologioiden vertailusta tässä työssä hieman redundanttia, koska asiakassovellus lopulta päättää menetelmästä palvelualustan sijaan. Tämän työn kannalta tiedonsiirtoteknologioihin on kuitenkin oltava edes pintapuolinen ymmärrys, jotta työn tuloksia voi tulkita ja pohtia.

4.1 Merkkauskielet

Jotta tietoa voidaan siirtää esimerkiksi HTTP:n avulla järkevästi, tietoon täytyy sisällyttää kontekstia. Merkkauskielet määrittelevät tiedonsiirrolle tarpeelliset viestirakenteet, joiden avulla tietoa voidaan siirtää verkon yli. Palvelualustalle soveltuvista merkkauskielistä tässä työssä esitellään lyhyesti kaksi avoimen standardin formaattia: XML ja JSON (JavaScript Object Notation).

Extensible Markup Language eli XML on W3C:n valvoma ja suosittelema standardi, jota käytettiin esimerkiksi StanForD:n ja metsäkeskuksen standardeissa. XML:ssä tietoa käsitellään elementteinä, joille voidaan antaa nimen lisäksi attribuutteja, jotka erottelevat ne toisista samannimisistä elementeistä. Alla on esitetty lyhyt esimerkki XML-dokumentista, joka on poimittu kirjasta [16].

```
<?xml version="1.0" encoding="UTF-8"?>
<customers>
	<customer ID=”C001”>
		<name>Acme Inc.</name>
		<phone>12345</phone>
	</customer>
	<customer ID=”C002”>
		<name>Star Wars Inc.</name>
		<phone>23456</phone>
	</customer>
</customers>
```

Jotta XML-dokumentti voidaan käsitellä yksikäsitteisesti ja validoida riippumatta dokumentin vastaanottajasta, dokumenttirakenne voidaan määritellä aiemmissa kappaleissa mainitun XML-skeeman avulla. Skeemaan voidaan määritellä elementtien nimien lisäksi myös elementtien tietotyypit, elementtihierarkia ja lukumääräsuhteet. Käytetty skeema mainitaan dokumentin alussa heti version ja merkistökoodauksen jälkeen. Skeemoja voidaan myös yhdistellä esimerkiksi ottamalla vanhasta skeemasta elementtityyppi ja määrittelemällä se toiseen skeemaan. Näin esimerkiksi tehdään myöhemmin koostetun tietorakenteen kappaleessa 6.3, jossa koko Owners-elementti on otettu Metsävarastandardin FD:n skeemasta.

JavaScript Object Notation (JSON) perustuu Javascript-ohjelmointikieleen. Se ei varsinaisesti ole merkkauskieli, vaan sillä viitataan JavaScript-kielellä tehtyyn tietorakenteeseen/olioon, jonka avulla tieto siirretään. Elementtien sijasta tiedon esityksessä JSON:ssa käytetään samaa arvopari käytäntöä kuin JavaScriptissä ja monissa muissa ohjelmointikielissä. Edellä esitetty XML-esimerkki olisi JSON-formaatissa seuraava:

```
{ ”customers” : {
	”customer” : {
		”id” :  “C001”,
		“name” : “Acme Inc.”,
		“phone” : “12345”
	},
	“customer” : {
		“id” : “C002”,
		“name” : “Star Wars Inc.”,
		“phone” : “23456”
	}
}}
```

Alun perin JSON:lla ei ollut skeemoja. IETF:n viimeisin luonnos [17] määrittelee samat ominaisuudet arvopareille kuin XML-skeema elementeilleen, mutta toistaiseksi se ei ole vielä standardi.

JSON:ia suositaan sen helppolukuisuuden, tiiviimmän koon, ja paremman suorituskyvyn ansiosta. Toisaalta helppolukuisuudelle on vaikea antaa numeerista arvoa, eikä XML välttämättä kaikissa käyttötapauksissa jää suorituskyvyltään tai pakkauskooltaan JSON:sta [18].

4.2 Paikkatiedon formaatit

Kuten kappaleessa 2.1.3 kuvattiin, hilasolujen sijainti esitetään hilanumeron avulla, josta solun koordinaatit voidaan johtaa laskennallisesti. Hilasolujen lisäksi metsäalueita voidaan esittää monikulmioiden vektorimonikulmioina. Vaikka palvelualustalla metsävaratietoa tullaan pääasiallisesti käsittelemään hilasolumuotoisena, asiakassovellukset tulevat esittämään kyselyjä, joissa paikkatieto annetaan monikulmioina. Paikkatiedon esittämiseksi on kehitetty formaatteja, joista tässä työssä esitellään kaksi XML-yhteensopivaa formaattia, SVG ja GML, sekä yksi JSON-yhteensopiva formaatti, GeoJSON.

Scalable Vector Graphics (SVG) on avoin ja alustaneutraali formaatti, joka on tarkoitettu ensisijaisesti kaksiulotteisten vektorikuvien esitykseen. SVG kuitenkin mahdollistaa koordinaatistoreferenssin siirtämisen viestipohjassaan niin, että kuviot voidaan yhdistää muuhun paikkatietoon. Pienenä heikkoutena SVG:ssä on, ettei sillä voida esittää yksittäisiä pisteitä.

Geography Markup Language (GML) on tarkoitettu paikkatiedon tiedonsiirtoon. Se on avoin ja alustaneutraali viitekehys kuten SVG, ja sillä voidaan välittää kuvioiden koordinaatistoreferenssi XML-attribuuttina. GML:n geometrisille muodoille on määritetty kolme vektorimuotoista objektia: piste, viivajono ja monikulmio. Näiden avulla voidaan käytännössä esittää kaikki tarpeellinen paikkatieto, ja siksi GML soveltuu erinomaisesti metsäsuunnittelussa tarvittavan paikkatiedon esittämiseen yksittäisistä puista moniosaisiin kohdealueisiin ja ajouriin, ja sitä käytetäänkin jo nyt metsätietostandardeissa.

GeoJSON on GML:n tavoin yksinomaan paikkatiedon formaatti. Sille on määritelty neljä objektia geometrisille muodoille: piste, viivajono, monikulmio ja monikulmiojoukko. Monikulmioiden ulkoreuna ilmaistaan antamalla monikulmion kulmien koordinaatit vastapäiväisessä järjestyksessä, ja monikulmiossa mahdolliset reiät ilmaistaan monikulmiona antamalla kulmien koordinaatit myötäpäiväisessä järjestyksessä. Kuten GML, GeoJSON tarjoaa kaikki tarpeelliset ominaisuudet, joita palvelualustalla voitaisiin tarvita.

4.3 Web Service protokollat

Web Servicellä tarkoitetaan verkonkautta saatavaa ohjelmointirajapintaa (verkko)palveluun, jota tarjotaan hyödynnettäväksi muihin ohjelmiin tai tietojärjestelmiin. Tässä työssä tutkitaan kahta Web Service periaatetta, REST ja SOAP/WSDL. Molemmat ovat alustariippumattomia, mutta SOAP on XML-pohjainen, kun taas REST ei määrittele käytettävää merkkauskieltä.

REST eli Representional State Transfer ei ole varsinaisesti protokolla vaan arkkitehtuurimalli. Se perustuu Fieldingin väitöskirjaan [19], jossa hän määrittelee REST-tyyppiselle (RESTful) arkkitehtuurille viisi vaatimusta. Näistä ensimmäinen on asiakas-palvelin tyylinen arkkitehtuuri. Palvelin, joka tarjoaa rajapintansa kautta palveluja, kuuntelee ja vastaa asiakasprosessien sille tekemiin kyselyihin. Tällä tavoin yhtä palvelua voi hyödyntää useampi kuin yksi asiakasprosessi. Toinen vaatimus on tilattomuus, eli tarjotut palvelut toimivat palvelimella aina samalla tavalla riippumatta edeltäneistä kyselyistä. Ainoastaan asiakas on tietoinen tehdyistä kyselyistä. Kolmas vaatimus on välimuistin käyttäminen. Kun asiakassovellus hakee palvelimelta jotain, haetulle tiedolle voidaan määrittää, voiko sen tallentaa välimuistiin. Jos asiakassovellus myöhemmin tarvitsee samaa tietoa uudestaan, se voi hakea tiedon välimuistista tekemättä uutta kyselyä palvelimelle, näin vähentäen tietoliikennettä. Fieldingin mukaan tämä parantaa tehokkuutta, skaalaavuutta ja käyttäjän suorituskykyä sen kustannuksella, että välimuistissa oleva tieto voi olla vanhentunutta. Neljäs vaatimus on, että palvelimen tarjoama rajapinta on sama riippumatta sitä kutsuvasta asiakasprosessista. Tämä yksinkertaistaa ja helpottaa rajapinnan käyttöönottoa, mutta luonnollisesti tekee rajapinnasta tehottomamman kuin tiettyä asiakassovellusta varten räätälöity rajapinta. Viides vaatimus on kerroshierarkia. Sillä rajoitetaan komponentteja näkemästä palvelinkerrosten taakse, eli mitä palveluja komponenttien käyttämät palvelimet mahdollisesti käyttävät. Tällä vältetään niin kutsuttu ”legacy code” eli vanhan koodin käyttäminen asiakasprosessien puolella, josta jotkin vanhat prosessit voivat olla edelleen riippuvaisia. Kerrokset voivat myös auttaa tasaamaan tietoliikenteen kuormitusta, mutta aiheuttavat väistämättä suurempia viiveitä, jos komponenttien on haettava käytetty palvelu useamman kerroksen läpi. [19]

Fielding [19] mainitsee myös kuudennen, vapaaehtoisen vaatimuksen, jota hän kutsuu nimellä ”Code-on-Demand”. Se sallii asiakasprosessien lataavan suoritettavaa ohjelmakoodia palvelimelta, mikä sallii järjestelmän joustavamman käytön huonomman näkyvyyden kustannuksella. Lisäksi Fielding on korostanut blogissaan [20], että REST-arkkitehtuurin rajapinnan palvelujen pitäisi löytyä hypertekstilinkkien avulla. Toisin sanoen asiakassovelluksella ei tarvitse olla ennestään tietoa palvelimen tarjoamista palveluista, vaan se voi hakea palvelut palvelimelta saatavien linkkien avulla.

SOAP on W3C:n standardoima tapa siirtää XML-pohjaista tietoa webpalvelussa. Se on alun perin lyhenne sanoista Simple Object Access Protocol, mutta nykyään sitä pidetään omana terminään eikä lyhenteenä. SOAP on aikoinaan kehittynyt XML-RPC:stä ja keskittyi ainoastaan HTTP:n kautta liikkuvaan tiedonsiirtoon. Nykyään SOAP on laajentunut käyttämään muitakin tiedonsiirtoprotokollia, mutta käyttää edelleen XML:ää merkkauskielenään.

Yksinkertaisuudessaan SOAP käärii lähetettävän XML-dokumentin ”kirjekuoreen”. Viestiin voi lisätä vapaaehtoisen otsakkeen (Header) kuten HTTP:ssä, johon voi sisällyttää mm. attribuutit actor ja mustUnderstand. Actor määrittää tarkasti viestin vastaanottajan, kun taas mustUnderstand-arvon ollessa voimassa viestin vastaanottajan on hylättävä viesti, jos se ei osaa tulkita otsaketta. Kolmantena attribuuttina mainittakoon myös encodingStyle, jolla voidaan määrittää SOAP-viestissä käytetty koodaustyyli. EncodingStyle voidaan mainita myös otsakkeen ulkopuolella.

WSDL (Web Service Description Language) määrittää itse Web Servicen tarjoamat palvelut. Sillä ilmaistaan mikä viesti palvelulle pitää lähettää, mihin osoitteeseen se lähetetään ja minkälaisen viestin palvelu palauttaa. WSDL ei määrittele vastaanotettavan tai lähetettävän viestiä konkreettisesti, vaan vain abstraktilla tasolla. Tyypillisesti sitä käytetään yhdessä SOAP:n kanssa, josta tulee termi SOAP/WSDL.

REST:n ja SOAP/WSDL:n vertailu on vaikeaa, koska kyseessä on hyvin erilaiset tiedonsiirtomenetelmät. REST:n eduiksi voidaan lukea riippumattomuus tiedonsiirtoteknologiasta (paitsi HTTP), yksinkertaisempi sovelluskehitys, SOAP/WSDL:ää kevyempi rakenne, ja se mukailee paremmin Internetin toimintaperiaatteita ja mallia. SOAP/WSDL pystyy sen sijaan toimimaan hajautetuissa järjestelmissä, kun REST olettaa pisteestä-pisteeseen tiedonsiirron. SOAP/WSDL on pitkälle standardoitu, ja sillä on sisäänrakennettujen turvamekanismien lisäksi käytössä joukko WS-* standardeja, joilla voidaan parantaa tiedonsiirron tietoturvaa, viestien eheyttä ja transaktioita. [21]

5. Palvelualusta

Nykyään metsäkoneilla kerätään hakkuiden yhteydessä suuria määriä tietoa, jota voitaisiin hyödyntää tarkentamaan perinteisillä menetelmillä laskettuja metsätunnuksia ja korjuuolosuhdetietoja. Hakkuukoneiden mittausdataa on jo hyödynnetty leimikoiden puustotunnusten ennustamisessa sekä puutavaralajien katkontavaihtoehtojen tarkasteluun, mutta toistaiseksi näitä tietoja ei ole ollut mahdollista jakaa käyttäjille julkisessa jaossa olevan tietovaraston tai järjestelmän kautta. Ratkaisuksi ongelmaan on esitetty metsävaratiedon palvelualustaa.

Kuten johdannossa mainittiin, palvelualustan tavoitteena on yhdistää eri tietolähteistä kerättävä metsätieto ja olosuhdetieto mahdollisimman ajankohtaiseksi, homogeeniseksi metsätiedoksi, jota loppukäyttäjät voivat hyödyntää omassa suunnittelutyössään. Palvelualustan kehitys koostuu useasta eri alihankkeesta, johon tämä työ liittyy metsäkonetiedon tietokantahankkeen kautta.

Kappaleessa 5.1 käsitellään lyhyesti palvelualustan rakennetta ja sen tarjoamia palveluita. Kappaleessa 5.2 esitellään metsävaratietojen analysoinnin käyttötapaus, johon kuuluu tämän työn kannalta olennaisesti tyyppileimikoinnin algoritmi.

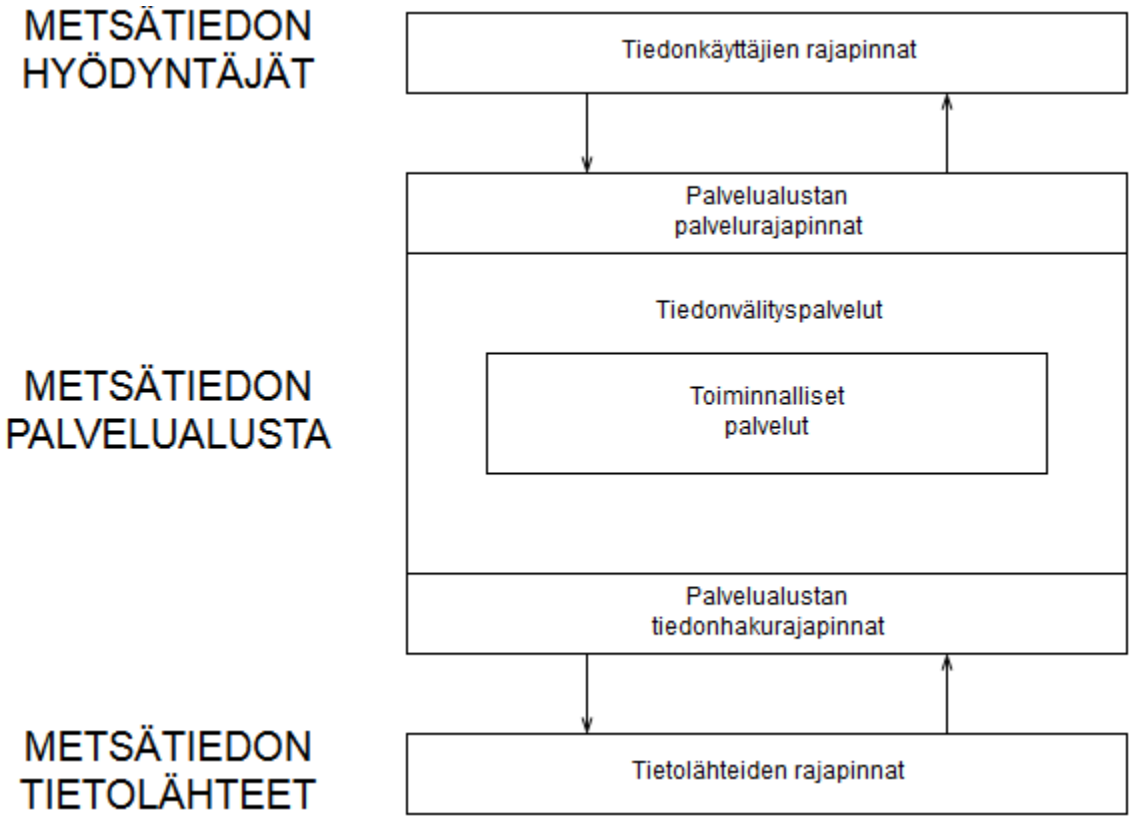
5.1 Rakenne ja palvelut

Palvelualustan rakenne koostuu kahdesta rajapinnasta, toinen tietolähteisiin ja toinen tiedonkäyttäjiin, sekä tiedonvälityspalveluista ja toiminnallisista palveluista. Palvelualustan periaate on esitettynä kuvassa 9.

Tiedonhakurajapinnassa palvelualusta esittää metsätietoa koskevia informaatiopalvelupyyntöjä. Tieto on tallennettuna palvelualustasta riippumattomiin tietokantoihin. Vaikka palvelualustalla voi olla vaikutusta uusien tietokantojen rajapintoihin, nykyisten tietokantojen rajapinnat määrittelevät ensisijaisesti teknologiat ja protokollat, joilla metsätieto siirretään palvelualustalle. Tiedonkäyttäjien rajapinnassa palvelualusta on vastaa palvelupyyntöihin. Palvelurajapintojen toteutus ei ole sidottu yhteen teknologiaratkaisuun: Modernit palvelukehykset, kuten Amazonin Azure-pilvi, tarjoavat adaptereita, jotka mahdollistavat eri rajapintateknologioiden käyttämisen. Adapterit antavat käyttäjäsovellusten kehittäjille enemmän valinnanvaraa oman rajapintansa toteutukseen.

Tiedonvälityspalvelut koostuvat nimensä mukaisesti palvelualustan sisäisen tiedonvälityksen organisoinnista. Palveluihin kuuluvat mm. tietoturva, käyttäjähallinta, sanomaseuranta, palveluiden hallinta ja orkestrointi, sekä laskutus. Lisäksi tiedonvälityspalvelut ylläpitävät palvelualustan omia tietovarastoja, jotka on tarkoitettu sekä väliaikaisen että pitkäaikaisen tiedon tallentamiseen. Vaikka tiedonvälityspalvelut ovat tärkeä osa palvelualustaa, ne eivät kuulu tämän diplomityön fokukseen ja jätetään siksi tarkemmin tutkimatta.

Toiminnalliset palvelut ovat palvelualustan kiinnostavimmat ominaisuudet tälle työlle. Ne luovat pohjan palvelualustan sovelluksille, ja niiden avulla palvelualusta muuttaa metsätietolähteistä kerätyt aineistot käyttäjälle hyödynnettävään muotoon. Toiminnalliset palvelut eivät ole suoraan loppukäyttäjän saatavilla, mutta ovat saatavilla epäsuorasti palvelualustan rajapintojen takana. Alla on esitelty lyhyesti ydinosa toiminnallisista palveluista.



Kuva 9: Palvelualustan periaatekuva

- **Haku:** Palvelualusta hakee sille määritellyn alueen metsätiedot.
- **Suodatus:** Palvelualustalla voidaan suodattaa metsäalueita. Esimerkiksi hakutuloksesta voidaan suodattaa pois ne alueet, joihin käyttäjällä ei ole oikeuksia tai joissa ei kasva puustoa. Jos suodatus kohdistuu johonkin puustotunnukseen, palvelun esivaatimuksena voi olla monikulmioesityksen muuttaminen ensiksi hilasolumuotoon. Suodatus voi kohdistua tiettyjen puustotunnusten lisäksi myös olosuhde-, omistajuus- tai rajoitetietoon.
- **Ajantasaistaminen:** Koska jotain metsätietolähteitä ei päivitetä kuin kerran muutamaan vuoteen, tietokantoihin tallennettu tieto ei ole täysin ajantasaista. Ajantasaistamisella annetaan mahdollisimman tarkka estimaatti nykyhetken metsätiedosta kasvu- ja ennustusmallien avulla. Jos puusto on käsitelty edellisen mittauksen jälkeen, mittaustieto ei kuvaa nykytilaa lainkaan, joten kasvumallien lisäksi tietoa ajantasaistetaan toimenpidetapahtumia koskevan tiedon perusteella.
- **Datakonversio:** Konversiolla voidaan muuttaa metsätunnusten yksikköjä. Konversioon kuuluu myös koordinaatistoreferenssin vaihtaminen.
- **Datafuusio:** Tietokantojen metsätieto voi olla erittäin vaihtelevaa tarkkuudeltaan ja tuoreudeltaan. Datafuusion tarkoituksena on yhdistää kaikkien tietokantojen metsätieto yhdeksi koherentiksi kokonaisuudeksi, jota tutkimalla saadaan mahdollisimman tarkka arvio nykyhetken metsistä.
- **Paikkakonversio:** Palvelualusta voi tarvittaessa muuttaa paikkatiedon vektorimuotoiset monikulmiot algoritmeille helpommin käsiteltävään hilamuotoon. Tätä muunnosta kutsutaan

”vector-to-grid”-muunnokseksi. Muunnos voidaan tehdä myös toiseen suuntaan, eli muuttaa hilasolujoukko takaisin monikulmioiksi.

- **Notifiointi:** Metsätiedon prosessointi useasta eri tietolähteestä voi olla pitkäkestoinen prosessi. Tämä voi aiheuttaa ongelmia sellaisissa sovelluksissa, joiden pitää jatkua pitkien prosessointitaukojen jälkeen. Notifioinnin avulla näille sovelluksille voidaan ilmoittaa prosessoinnin valmistumisesta.

5.2 Metsävaratietojen analysoinnin käyttötapaus

Yksi palvelualustan käyttötapauksista on ”Metsävaratietojen analysointi hakkuumahdollisuuksien arviointia, puunhankinnan strategista suunnittelua ja katkonnan ohjausta varten.” Siinä metsäyhtiö saa ajantasaista tietoa kohdealueen metsävaroista joko puuhuoltoa tai hakkuumahdollisuuksia varten. Saatua tietoa voidaan hyödyntää mm. oston strategisessa ja operatiivisessa suunnittelussa, puutavaran tehdastoimitusten suunnittelussa, korjuun ja kuljetuksen resurssisuunnittelussa, kuljetusvaihtoehtojen suunnittelussa sekä ohjatessa katkontaa dynaamisesti korjuukohteiden luokitteluun perustuvan puuston ennustetun jalostusarvon mukaan. Käyttötapauksen sovellus tarjoaa käyttäjälle liittymän mm. hakujen ja valintojen tekemiseen. Tästä käyttötapauksesta työhön on rajattu strateginen suunnittelu katkonnanohjausta varten ja tyyppileimikointi.

Kirjauduttuaan palvelualustaan, käyttäjä tai sitä edustava sovellus määrittää sovelluksessaan suunniteltavana olevan kohdealueen, hakkuutoimenpiteen ja muut metsävaratietojen laskennan valintatekijät. Valintatekijöitä voivat olla esimerkiksi omistajaryhmä, puulaji tai jokin muu puustoa rajoittava suodatin. Käyttäjä voi lisäksi valita vain ne metsävaroja kuvaavat tunnukset, jotka hän haluaa laskettavaksi, sekä millä jaottelulla tiedot lasketaan. Tämän jälkeen käyttäjä antaa kyselyn käsiteltäväksi palvelualustaan.

Hakkuumahdollisuuksien laskenta perustuu hilatasoiseen metsävaratietoon, joka saadaan mm. Metsäkeskuksen metsävaratiedosta. Hilamuotoisen tiedon puuttuessa voidaan hyödyntää tukimateriaalina myös MVMI:n hiladataa, muita tutkittuja kaukokartoitusaineistoja tai metsäsuunnittelun kuviotietoja. Riippumatta mitä käyttäjä on määrittänyt suodatettavaksi, laskennassa otetaan huomioon metsäkäsittelyä rajoittavat kaava- ja suojelutiedot, luontokohdetiedot sekä muut rajoittavat tai ohjaavat kriteerit. Laskennan tuloksena käyttäjä saa sovellukseen lasketut puustotunnukset ja saantoestimaatit kohdealueesta.

6. Koostettu tietorakenne

Koostetun tietorakenteen tehtävänä on toimia palvelualustalla hilamuotoisen metsätiedon ja saantotiedon yhdistävänä tietomuotona. Sen avulla toiminnalliset palvelut voivat käsitellä prosessoitavaksi tarkoitettua metsätietoa ilman ylimääräistä käsittelyä. Kappaleissa 6.1 ja 6.2 tutkitaan tietorakenteen vaatimuksia ja nykyisten tietorakenteiden soveltuvuutta, kun taas 6.3 esittelee toteutuksen. Tässä työssä annettu tietorakenne on kuitenkin vain luonnos eikä välttämättä vastaa palvelualustalla käytettävää lopullista toteutusta.

6.1 Tietorakenteen vaatimuksia

Tyyppileimikointia varten tarvitaan hilakohtaisia puuston keskitunnuksia ja hakkuutietoa. Sekä puustotunnukset että hakkuutiedot pitää jaotella puulajeittain vähintään mänty-kuusi-lehtimetsä-jaottelulla, jonka lisäksi hakkuutiedot pitää jaotella myös hakkuutoimenpiteen mukaan.

Tietorakennetta käytetään ensisijaisesti palvelualustan sisäisessä kommunikaatiossa. Sitä voisi hyödyntää myös rajapintojen toteutuksessa, mutta niihin on optimaalisempaa laatia paremmin käyttötarkoituksia varten tehdyt ratkaisut. Tietorakenteen käytettävyyttä voidaan parantaa soveltamalla metsävaratiedon standardeja tai mukailemalla metsätietokantojen tietorakenteita täysin omien tietorakenteiden sijasta, jotta puustotunnusten tallentaminen tietorakenteeseen olisi vaivattomampaa.

Puustotunnusten virhearviot on saatava osaksi datafuusiota. Vaikka Metsäkeskuksen omat RMSE-arvot on esitetty luvun 3.2.2 taulukossa 2, virhearvot voivat poiketa muissa tietokannoissa esiintyvistä virheistä jo mittaustavankin vuoksi. Tästä syystä virhearvot pitäisi sisällyttää osaksi tietorakennetta vähintään vapaaehtoisena lisätietona. Tarvittaessa taulukkoa voidaan käyttää täydentämään puuttuvia tarkkuustietoja datafuusiossa ja ajantasaistuksessa, jos haetulle puustotunnukselle ei saada virhearviota muista tietolähteistä.

6.2 Nykyisten tietorakenteiden sopivuus

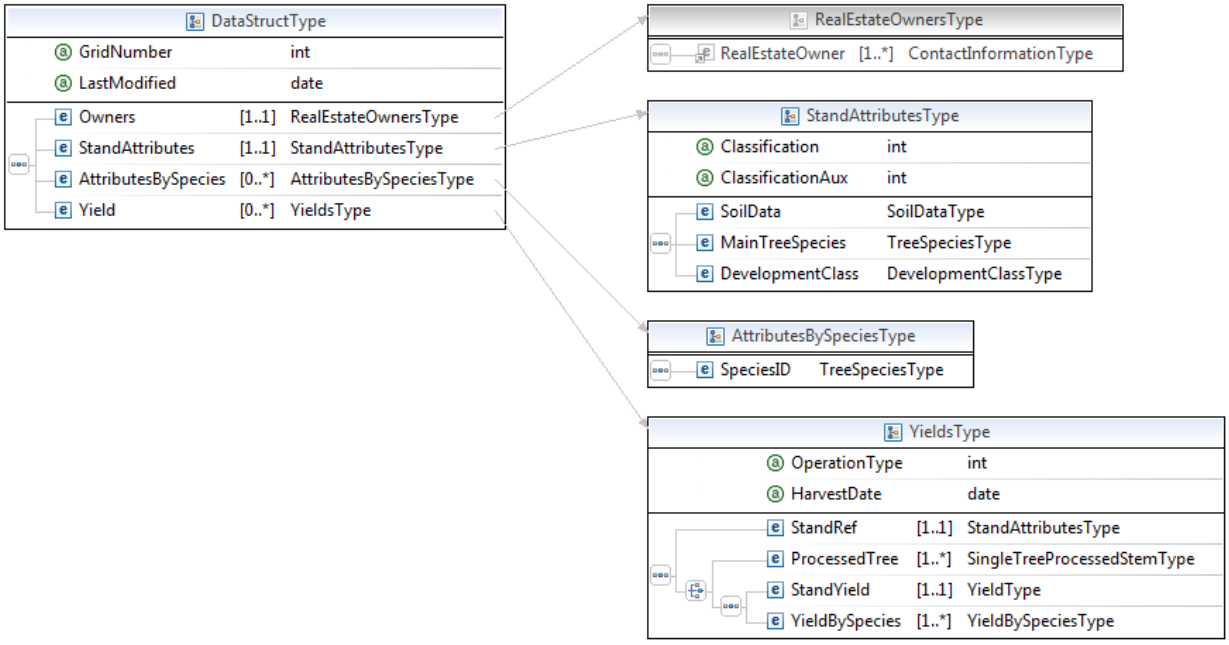
Koska StanForD:ia käytetään hakkuiden puustotiedon siirtämiseen, sillä ei saada siirrettyä kaikkea tarpeellista tietoa vielä hakkaamattomista puista. Esimerkiksi se ei käsittele tietoa hakkuualueesta muuten kuin mistä puut on kaadettu, mikä voi vaikeuttaa hilapohjaista tallennusta tai runkolukusarjan selvittämistä. Vaikka StanForDiin on mahdollista sisällyttää lisätietoa, ei ole optimaalista vakiinnuttaa StanForDia hilanumero siirtotavaks. Tästä syystä StanFordin HPR:ää ei voida suoraan käyttää rajapintana palvelualustan ja metsävaratietokantojen välillä.

MVM-datan haunaikainen prosessoiminen voi olla liian hidast. Vaikka haku kohdistuisi vain yhteen puustotunnukseen, rasterikuvan on kooltaan kymmenien megatavuja, ja sisältää tuhansia hilasoluja. MVMI:stä saatava tieto on kuitenkin stabiilia, sillä tietokantaa päivitetään vain 2-3 vuoden välein, joten se voitaisiin prosessoida palvelualustalla taustalla. Prosessoinnissa on huomioitava myös ajantasaistus, sillä MVMI:tä päivitetään yleensä edeltävän vuoden puustotiedoilla.

Jotta kaikkia tietokantoja ja standardeja voitaisiin hyödyntää, palvelualustalla täytyy tehdä muunnoksia jompaankumpaan koordinaatistojärjestelmistä.

6.3 Toteutus

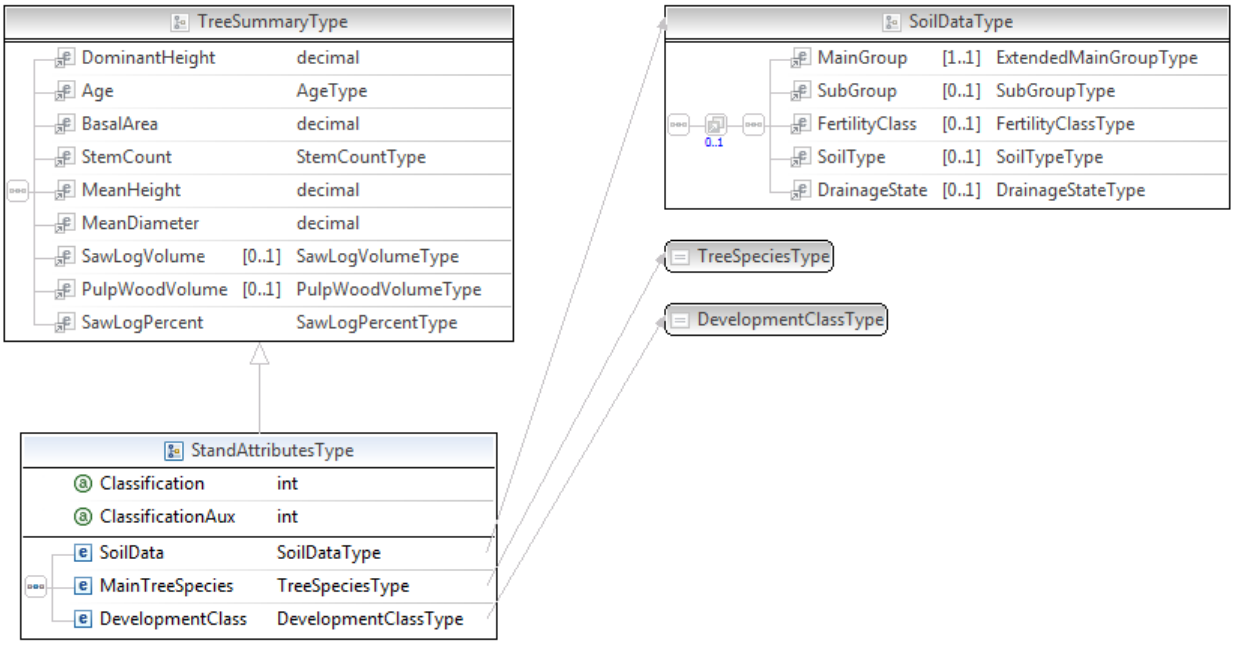
Tietorakenteen attribuutit pohjautuvat Metsäkeskuksen hilapohjaiseen metsätietoon ja sen rakenne perustuu pääosin Metsäkeskuksen metsästandardiin (versio 9). Tietorakenteesta tehtiin XML-skeema, jonka yleisrakenne on esitetty kuvassa 10. Tietorakenteen XML-skeema on annettu liitteessä 1, mutta sen tutkiminen vaatii myös edellä mainitun metsästandardin XML-skeeman, joka on saatavilla lähteestä [14], sekä StanForD2010:n XML-skeeman.



Kuva 10: Koostetun tietorakenteen yleisrakenne

Tietorakenteen pääelementti on DataStruct. Se kuvastaa yhtä metsätiedon hilasolua ja koostuu neljästä elementistä: StandAttributes, AttributesBySpecies, Owners ja Yield. Kuten Metsäkeskuksen tietokannassa, hilasolut yksilöidään hilanumeron avulla, mutta tietokannasta poiketen DataStructiin ei ole sisällytetty kuvion ID:tä eikä hilanumeron ID:tä, koska ne voivat poiketa eri tietokantojen välillä. Hilanumeron referenssikoordinaatistoksi oletetaan ETRS-TM35FIN. Datastructiin kuuluu myös viimeisin muokkauspäivämäärä, joka annetaan Date-tyyppisenä attribuuttina.

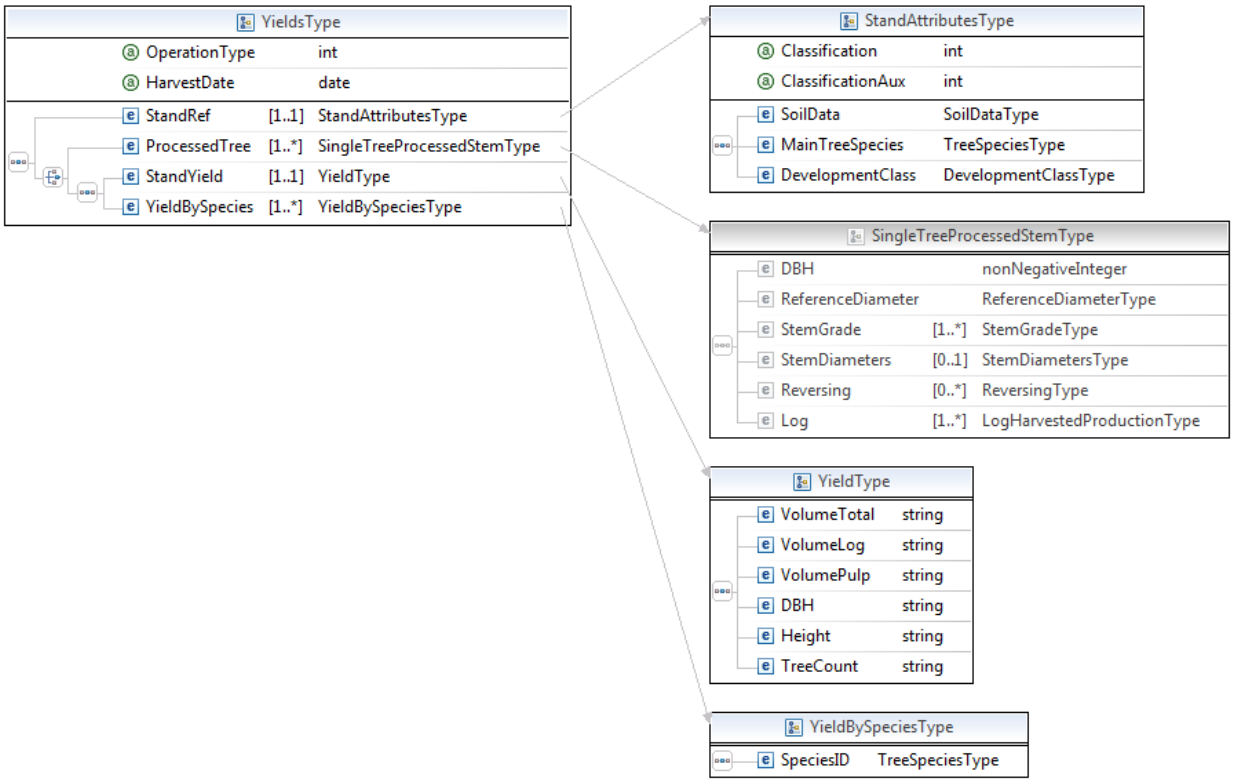
Owners on tyypiltään Metsäkeskuksen standardin RealEstateOwners, jota käytetään ForestDatassa. RealEstateOwners on kuitenkin vain esimerkki mahdollisesta omistajuustiedosta; luokka kattaa pääasiallisesti kaikki tärkeimmät omistajuustiedot, mutta myös paljon lisätietoa, joita ei tarvita toiminnallisissa palveluissa tai loppukäyttäjän saatavaksi.



Kuva 11: StandAttributes-elementin rakenne. AttributesBySpecies periytyy myös TreeSummaryType:stä, mutta sillä on omana elementtinään vain puulaji, johon puustotunnukset kohdistuvat.

Hilasolun puustotunnukset sijaitsevat StandAttributes ja AttributesBySpecies elementeissä. Edeltävä sisältää solun keskitunnukset, kun taas jälkimmäinen yhden puulajin puustotunnukset hilasolussa. Molemmat periytyvät Metsäkeskuksen standardin TreeSummaryType-luokasta, joka oli osa aiemmin esiteltyä ForestObjectia. StandAttributes on pakollinen osa hilatietoa, mutta jos solussa ei kasva puustoa, puulajikohtaisia puustotunnuksia ei tarvitse tallentaa tietorakenteeseen. StandAttributes-elementtiin tallennetaan myös maastotunnukset (metsävarastandardin SoilData), pääpuulaji, kehitysjakso ja mahdollinen luokitus tyyppileimikointia varten. Pelkkiin puustotunnuksiin kohdistuvan luokituksen lisäksi kuviosta on myös lisäluokitus (ClassificationAux), joka kuvastaa metsän kasvuolosuhteita, kuten maaperää. Saantotietoihin tallennetaan hakkuuta edeltäneen metsän StandAttributes nimellä StandRef. Tällä tavoin tietorakenteen hilasolua voidaan käyttää tyyppileimikoinnissa referenssisoluna.

Yields sisältää tiedot yhdestä hilasolussa tehdystä hakkuutoimenpiteestä. Se koostuu edellä mainitusta StandRef:stä, puulajeittain eritellyistä saantotiedoista, ja hilasolun kokonaissaannosta. Näiden lisäksi sillä on attribuutteina hakkuutoimenpiteen tunnus ja päivämäärä, jolloin hakkuu suoritettiin. Saantotiedon antamiseksi on annettu kaksi vaihtoehtoa. Ensimmäisenä vaihtoehtona on antaa prosessoidut puut StanForD:n SingleTreeProcessed-luokan avulla. Toinen vaihtoehto on jakaa saantotieto puulajeittain ja hilasolun kokonaissaantoon ja antaa ne StandYield- ja YieldBySpecies-luokkien avulla, kuten esitetty kuvassa 12. Sekä StandYield että YieldBySpecies ovat päätyypiltään YieldType, mutta jälkimmäiseen on tallennettu myös saannon puulaji. YieldTypen rakenne perustuu Metsäkeskuksen tietokantaan.



Kuva 12: Saantotiedon rakenne

6.4 Huomioita

Puulajit määritellään metsästandardin TreeSpeciesType:llä. Teollisuudessa vaikuttaisi olevan tapana yhdistää kaikki lehtipuut yhteen ryhmään, jolloin lehtipuiden erottelukyky menetetään. Yhdistetylle tai muuten määrittelemättömälle lehtipuutiedolle suositellaan tässä työssä käytettäväksi koodia ”9”, eli ”Muu lehtipuu”.

StanForD:n SingleTreeProcessed-tyyppinen saantotieto voi olla hieman redundanttia, jos sitä ei anneta sellaisenaan loppukäyttäjälle. Toiminnallisissa palveluissa on helpompi käsitellä valmiiksi perustunnuksiksi muutettua saantotietoa kuin prosessoida StanForD:n tietorakenteesta tunnuksia ajonaikaisesti, etenkin kun halutaan muodostaa suurien kohdealueiden kokonaisestimaatteja. Toisaalta StanForD:n avulla voidaan antaa saantotiedosta yksityiskohtaisempaa tietoa kuin perustunnuksilla, joiden prosessoimiseen voi kadota osa hyödyllisestä lisätiedosta.

DataStructin LastModified-attribuutti ilmaisee tuoreinta metsätietoa, johon puustotunnukset perustuvat. Datafuusiolla pyritään muodostamaan mahdollisimman tarkka estimaatti senhetkisestä metsästä, mutta mitä vanhempaa dataa algoritmilla on käytössä, sitä epävarmempia puustoattribuutit ovat.

7. Tyyppileimikoinnin algoritmi

Suunnitellusti tyyppileimikoinnin algoritmin syötteenä on tutkittavan kohdealueen rajat ja hakkuutoimenpide, joka halutaan toteuttaa kohdealueella, sekä referenssialue. Algoritmin tuloksena saadaan kohdealueen saantoestimaatti, joka kuvastaa millaista puutavaraa hakkuutoimenpiteellä kohdealueelta saadaan.

Kohdealue on se alue, jonka arvioitua saantoa tietyllä hakkuutoimenpiteellä halutaan tutkia tyyppileimikoinnin avulla. Referenssialueella tarkoitetaan aluetta, jolta etsitään puustotunnuksiltaan vastaavia hilasoluja ja joiden puusto on käsitelty valitulla hakkuutoimenpiteellä. Kohdealueen puutavaraa voidaan estimoida näiden hilasolujen saantotiedon perusteella. Referenssialueen on oltava riittävän suuri, jotta se sisältäisi riittävän kattavasti kohdealueen hilasoluja vastaavia puustotunnuksia ja samaan aikaan etsityn hakkuutoimenpiteen saantotietoja. Kohdealueen ei välttämättä tarvitse kuulua referenssialueen osajoukkoon, vaan referenssialue voi olla siitä irrallinen.

Tässä työssä tyyppileimikoinnin algoritmin teoriasta esitellään kolmea kokonaisuutta: tyyppiluokittelua, referenssialueen määrittelytapaa ja palvelualustan kannalta optimaalisinta algoritmia tyyppileimikoinnille.

7.1 Tyyppiluokitus

Kuten aiemmin on mainittu, tässä työssä ei laadita tyyppileimikoinnissa käytetyn luokittelun, eli tyyppiluokittelun, määrittelyalgoritmia. Tyyppiluokittelun toteutusmahdollisuuksia on kuitenkin spekuloitava, sillä luokittelu on olennainen osa referenssialueen määrittelyä ja tyyppileimikoinnin algoritmia.

Ensimmäinen luokitteluvaihtoehto on hilasolukohtainen tyyppiluokitus. Tyyppiluokitus on jokin diskreetti arvojoukko, joka voidaan antaa kuvastamaan solun puustotunnuksia. Solujen välisiä luokituksia vertaamalla voidaan nopeasti etsiä samankaltaisia soluja. Tyyppiluokitus tullaan todennäköisesti määrittämään soluille palvelualustan taustaprosessina ajonaikaisen määrittämisen sijasta, mikä vähentää ajonaikaisia laskutoimituksia, ja siten parantaa algoritmin suorituskykyä. On esitetty, että puustotunnuksien lisäksi muodostettaisiin niistä erillinen tyyppiluokitus myös kasvuolosuhteiden perusteella, jolla voidaan joissain tapauksissa parantaa saantoestimaatin tarkkuutta.

Toisena vaihtoehtona on ns. similarisuuden tutkiminen, jossa kahden hilasolun puustotunnuksien välinen samankaltaisuus pyritään määrittämään haun aikana. Koska täysin kohdealueen hilasoluja vastaavia referenssisoluja tuskin löytyy tarpeeksi Suomen metsistä, solujenvälistä similariteettia verrataan johonkin raja-arvoon, jonka ylittyessä hilasolujen voidaan todeta olevan tarpeeksi samankaltaisia keskenään algoritmia varten. Koska similariteetti tutkitaan haun aikana, menetelmän suorituskyky voi olla huomattavasti huonompi kuin tyyppiluokituksen, toisaalta menetelmä ei vaadi ylimääräistä tallennustilaa palvelualustalta.

7.2 Referenssialueen määrittely

Riippumatta luokittelutavasta, kohdealuetta täytyy verrata hakkuutietoja sisältävään referenssialueeseen. Tyyppiluokittelun tavoin referenssialueen määrittely sisältää omat haasteensa ja on aiheena liian laaja tähän työhön. Referenssialueen määrittelylle on esitetty kahta eri vaihtoehtoa: kohdealueriippuvainen hakukohtainen määrittely ja vakioaluemäärittely.

Hakukohtaisessa määrittelyssä referenssialue kohdistuu kohdealueen ympärille ja minimoidaan haetun alueen koon perusteella. Mitä suurempi kohdealue sitä suurempi referenssialue määritetään. Samalla optimoidaan vertailtavien soluparien määrä, mikä minimoi tarvittavaa laskentaa palvelualustalla ja sovelluksessa suhteessa soluparien määrään. Menetelmän riskinä kuitenkin on, ettei kohdealuetta ympäröivästä metsästä saada hakkuutietoa, jolloin referenssialueen muodostaminen ei ole mahdollista tai sen määrittämiseen menisi paljon turhaa aikaa.

Vakioaluemäärittelyssä tietyn alueen kohdealueille annetaan aina samat referenssisolut. Suomen metsät voisi jaotella esimerkiksi sektoreiksi MVMI-lohkojen perusteella, ja kullekin sektorille määritettäisiin oma referenssialueensa. Vakioalueiden avulla referenssialuetta ei tarvitse määrittää haun aikana ja hakkuutiedon saatavuus voidaan varmistaa. Referenssialueen kokoa voitaisiin myös optimoida kohdealueen mukaan kuten hakukohtaisessa määrittelyssä. Toisaalta jos haku kohdistuu erikoiseen puustoon tai olosuhdeympäristöön, huonosti valituilla referenssisoluilla alueiden vertailukelpoisuus voi kärsiä. Tätä ongelmaa ei olisi, jos referenssialue valittaisiin kohdealueen ympäriltä, olettaen että kyseistä puustoa on kaadettu kohdealueen ympäriltä aiemminkin.

7.3 Tyyppileimikoinnin algoritmi

7.3.1 Algoritmivaihtoehdot

Tyyppiluokitusmenetelmien perusteella laadittiin kaksi algoritmivaihtoehtoa. Molemmille algoritmivaihtoehdoille on laadittu pseudokoodit kuvaamaan karkeasti algoritmin toimintaa molemmissa tapauksissa. Alkuperäiset pseudokoodit on laadittu tästä diplomityöstä riippumattomasti[1], mutta niiden esitysmuotoa on muokattu tässä työssä. Referenssialueen määrittely ei näy algoritmissa suoraan, vaan sillä on vaikutusta lähinnä algoritmin tuloksiin. Tyyppiluokitustapa muuttaa algoritmin toimintaperiaatetta hieman, mutta algoritmin rakenne on pääosin samanlainen käytettäessä tyyppiluokitusta tai similariteettilaskentaa. Tyyppiluokitusta käyttävän algoritmin pseudokoodi on esitettynä ensimmäiseksi.

Input: Kohdealueen vektoriesitys (RoL), referenssialueen vektoriesitys (RoR), käsittelytapa (OM)

Output: Saantoestimaatti kohdealueelta

1: Tee Vector2Grid-muunnos RoL:lle (RoLg) ja RoR:lle (RoRg).

2: Etsi referenssialueelta ne hilasolut (RoRgHarvester), joista tietokannassa on OM-käsittelytavan saantotietoa.

3: Hae tietokannasta kohdealueen hilasolujen puustotunnustyypit omaan vektoriin (FtypeRoLg).

4: Hae tietokannasta referenssialueen hilasolujen puustotunnustyypit omaan vektoriin (FtypeRoRg).

[1] R.Ritala, yksityinen tiedonanto, Platform 2 –projekti.

5: Alusta references, relevantRoRg ja n. n = 0.

6: Tee jokaista elementiä vektorissa FtypeRoRg:ssa kohden:

6.1: Kasvata n:n arvoa yhdellä.

6.2: Etsi FtypeRoRg(n) kanssa samaa puustotyyppiä olevien hilasolujen sijainnit. FtypeRoLg:sta vektoriin Pos.

6.3: Jos samaa puustotyyppiä olevia soluja löytyi eli Pos ei ole tyhjäjoukko:

6.3.1: Alusta k. k = 0.

6.3.2: Tee jokaiselle elementille vektorissa Pos:

6.3.2.1: Kasvata k:n arvoa yhdellä.

6.3.2.2: Lisää references-vektoriin RoLg:n arvo kohdassa Pos(k) ja RoRg:n arvo kohdassa n.

6.3.3: Tallenna RoRg:n n. hilasolu relevantRoRg:hen, jos sitä ei ole siellä ennestään.

7: Hae kaikkien relevantRoRg:hen tallennettujen hilasolujen attribuutit, tallenna ne dataHarvester vektoriin.

8: Alusta kokonaissaantoestimaatti (EY), dataSet ja j. EY = 0, j = 0.

9: Tee jokaista elementtiä RoLg:ssa kohden:

9.1: Kasvata j:n arvoa yhdellä.

9.2: Tyhjennä dataSet.

9.3: Etsi referenssihilasolut (relevantRefs), joilla on sama luokitus kuin Rolg:llä kohdassa j.

9.4: Jos relevantRefs on tyhjä:

9.4.1: Käsittele poikkeustilanne, jossa kiinnostavan alueen hilasolulle ei saatu saantoestimaattia.

9.5: Muuten

9.5.1 Tee jokaista elementtiä i vektorissa relevantRefs kohden:

9.5.1.1 Lisää dataSet:iin dataHarvester:n elementti kohdasta relevantRefs(i).

9.6 Laske solun saantoestimaatti algoritmilla dataSet:n, RoLg:n alkion kohdassa j ja relevantRefs avulla.

9.7 Lisää saatu estimaatti EY:hyn.

10: Anna kokonaissaantoestimaatti EY.

Ensimmäiseksi kohdealue ja sille määritelty referenssialue muutetaan hilasolumuotoon palvelualustan vector-to-grid-muunnoksella. Muunnoksen jälkeen muodostetuille hilasoluille haetaan niiden tyyppiluokitukset. Seuraavaksi kohdassa 6 kerätään *references* –listaan samaa puustotyyppiä olevat hilasolut kohdealueelta ja referenssijoukosta, sekä *relevantRoRg* –listaan ne referenssihilasolut, joiden puustotyyppi eli tyyppiluokitus on täysin sama kuin jossain kohdealueen solussa. Kun kaikki olennaiset hilasolut on haettu referenssialueelta, voidaan hakea niiden saantotiedot kohdassa 7, ja muodostaa kohdealueen kokonaissaantoestimaatti kohdassa 9. Estimaatin sisältö koostuu samoista attribuuteista kuin koostetun tietorakenteen YieldType-luokka (kuva 12).

Similariteettilaskentaa käyttävä versio tarvitsee parametreikseen kohdealueen, referenssialueen ja käsittelytavan lisäksi similariteettimitan kynnysarvon sekä puustoattribuutit, joilla rajoitetaan tutkittavaa similariteettia. Kynnysarvon määritelmä ei ole vielä yksiselitteinen, mutta tässä työssä sen voidaan olettaa olevan joku prosenttiluku. Similariteettilaskentaa käyttävä algoritmi on esitetty seuraavaksi.

Input: Kiinnostavan alueen vektoriesitys (RoL), referenssialueen vektoriesitys (RoR), käsittelytapa (OM),similariteettimitan kynnysarvo (Thrsd), similariteettilaskentaan valitut puustoattribuutit (ForAttr)

Output: Saantoestimaatti kiinnostavalta alueelta

1: Tee Vector2Grid-muunnos RoL:lle (RoLg) ja RoR:lle (RoRg).

2: Etsi referenssialueelta ne hilasolut (RoRgHarvester), joista tietokannassa on OM-käsittelytavan saantotietoa.

3: Muodostetaan tietorakenne (FDRoLg) kiinnostavan alueen hilasolujen puustotunnuksista indeksoituna RoLg:n mukaan järjestykseen.

4: Muodostetaan tietorakenne (FDRoRgHarvester) referenssialueen hilasolujen puustotunnuksista indeksoituna RoRgHarvester:n mukaan järjestykseen

5: Alusta relevantSimilarities, relevantRoRg, nL ja nR. nL = 0.

6: Tee jokaiselle elementille RoLg:ssa:

    6.1: Kasvata nL:n arvoa yhdellä.

    6.2: nR = 0

    6.3: Tee jokaista elementtiä vektorissa RoRgHarvester kohden

        6.3.1: Kasvata nR arvoa yhdellä.

        6.3.2: Laske hilasolujen FDRoLg(nL) ja FDRoRgHarvester(nR) välinen similariteetti (simMeas) algoritmilla ForAttr:n perusteella.

        6.3.3: Jos simMeas > Thrsd

            6.3.3.1 Lisätään relevantSimilarities-luetteloon RoLg(nL) ja RoRg(nR) pari, sekä niiden välinen similariteetti simMeas.

6.3.3.2: Tallenna RoRg:n n. hilasolu relevantRoRg:hen, jos sitä ei ole siellä ennestään.

7: Hae kaikkien relevantRoRg:hen tallennettujen hilasolujen attribuutit, tallenna ne dataHarvester vektoriin.

8: Alusta kokonaissaantoestimaatti (EY), dataSet, similarities ja j. EY = 0, j = 0.

9: Tee jokaiselle elementille RoLg:ssa:

9.1: Kasvata j:n arvoa yhdellä.

9.2: Tyhjennä dataSet ja similarities.

9.3: Etsi referenssigridisolut (relevantRefs), joilla on riittävä similariteetti kuin Rolg:llä kohdassa j.

9.4: Etsi similariteetit (relevantSimils), jotka vastaavat kohdassa 9.3 etsittyjä soluja.

9.5: Jos relevantRefs on tyhjä:

9.5.1: Käsittele poikkeustilanne, jossa kiinnostavan alueen hilasolulle ei saatu saantoestimaattia.

9.6: Muuten tee jokaista elementtiä i vektorissa relevantRefs kohden:

9.6.1 Lisää dataSet:iin dataHarvester:n elementti kohdasta relevantRefs(i).

9.6.2 Lisää similarities-listaan relevantSimils:n elementti kohdasta i

9.6 Laske solun saantoestimaatti algoritmilla dataSet:n, similarities:n, RoLg:n alkion kohdassa j ja relevantRefs avulla.

9.7 Lisää saatu estimaatti EY:hyn.

10: Anna kokonaissaantoestimaatti EY.

7.3.2 Pohdintaa algoritmeista

Tyyppiluokituksen haasteena on luokkien lukumäärän asettaminen: Jos luokkia on liikaa, saman tyyppiluokituksen hilasoluja on vaikeampi löytää, jos taas liian vähän, saantoestimaattien tarkkuus kärsii. Similarisuusluokittelun haasteena on määritellä, miten kahden metsäsolun välinen similarisuus ylipäätään lasketaan, vieläpä niin, että laskenta ei aiheuta merkittäviä viiveitä. Sama ongelma syntyy myös, jos solujen puustotyyppien ei tarvitse olla samat vaan vain lähellä toisiaan – miten määritellään luokkien erojen metriikka.

Algoritmi antaa estimaatin koko kohdealueelta, mutta saanto voi olla jakautunut epätasaisesti alueelle tai keskittynyt vain osaan hilasoluista. Koodissa lasketaan välityloksena kohdassa 9.6 hilasolukohtaiset saantoestimaatit, mitä voisi hyödyntää saantojakauman tarkastelussa. Estimaatin tarkkuuteen vaikuttaa vahvasti attribuuttien mittaustarkkuus, ja siksi estimaatin virhearvion voisi antaa kokonaisestimaatin yhteydessä.

Algoritmissa ei oteta kantaa siihen, miten käsitellään hilasoluja, joiden leimikkotyyppiä vastaavia, saantotietoja sisältäviä soluja ei ole referenssidatassa. Helpointa on antaa kokonaissaantoestimaatin lisäksi ne hilasolut tai hilasolujen lukumäärä, joille ei löytynyt saantoestimaattia. Vaihtoehtoisesti voidaan arvioida puuttuvien hilasolujen vaikutusta lähinaapurisolujen perusteella, jos kohdealueen hilasolut muistuttavat tarpeeksi toisiaan.

Algoritmin toimivuudesta on vaikea antaa arviota pelkän pseudokoodin perusteella. Tätä tutkittiin tarkemmin pienen demon avulla, jota esitellään seuraavassa kappaleessa.

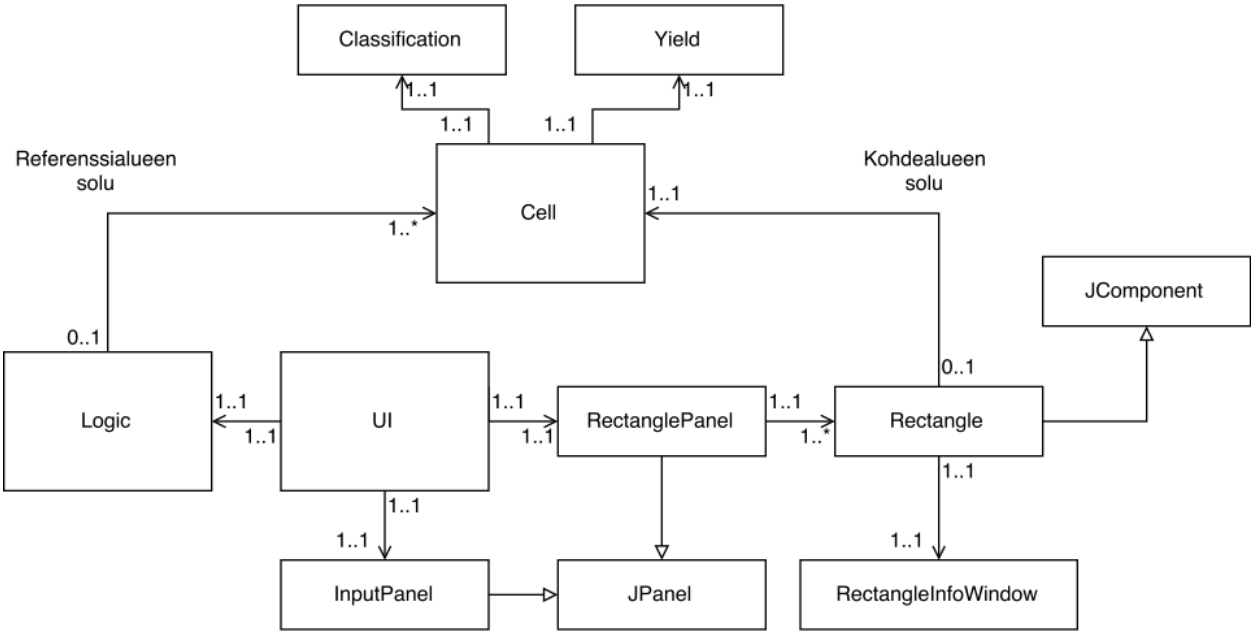
8. Tyyppileimikoinnin demo

Tyyppileimikoinnin algoritmin testaamiseksi tehtiin pieni demosovellus. Se toteutettiin Javalla (JDK:n versio 1.8.0-131) ja kirjoitettiin Eclipse-kehitysympäristössä (versio Kepler Service Release 2). Koska similariteettilaskennan toteuttaminen n-ulotteisessa ympäristössä ilman syvällistä perehtymistä alan teoriaan on haastavaa, tyydyttiin työssä toteuttamaan vain valmiiksi laskettuun tyyppiluokitukseen perustuva versio algoritmista. Referenssialue koostuu satunnaisesti generoiduista hilasoluista, jolloin kohdealueen hilasolut ovat riippumattomia referenssialueesta. Menettelytapa emuloi puustotunnuksiltaan mahdollisimman monipuolista referenssialuetta, eli muistuttaa enemmän vakioaluemäärittelyä kuin kohdealueriippuvaista määrittelyä.

Demossa käyttäjällä tai käyttäjää edustavalla sovelluksella oletettiin olevan oikeudet tutkia kohdealueen ja referenssialueen soluja, sekä oikeudet suorittaa haku. Lisäksi vector-to-grid-muunnos, datafuusio ja tyyppiluokitus oletettiin tehdyiksi sekä referenssisoluille että kohdealueen soluille ennen algoritmin ajamista.

8.1 Ohjelman rakenne

Demon luokkakaavio on esitetty kuvassa 13. Tärkeimmät luokat ovat UI, Logic ja Cell. Loput luokat täydentävät käyttöliittymän toiminnallisuutta tai sisältävät hilasolukohtaisia tunnuksia.



Kuva 13: Demon luokkakaavio

UI-luokka sisältää käyttöliittymän ja syötteiden käsittelyn. Syötteet käsitellään InputPanel:in kautta, kun taas hilasolut esitetään RectanglePanel:in kautta. Molemmat periytyvät Javan JPanel-oliosta. Kohdealueen solut, jotka esitetään käyttöliittymässä värillisinä nelikulmioina, tallennetaan Rectangle-luokkaan, jotka RectanglePanel kokoaa hilaksi. Solutiedot Rectanglessa tallennetaan Cell-luokan avulla. Jotta kohdealueen solut pysyisivät mahdollisimman monipuolisina, näiden solujen luokitukset on kovakoodattu eikä generoida satunnaisesti.

Logic:ssa sijaitsee itse algoritmi, jonka lisäksi siellä generoidaan referenssialue. Referenssisolujen määrä määritetään käyttöliittymästä, ja solut generoidaan aina uudelleen, kun algoritmi aloitetaan. Kuten kohdealueen solut, referenssisolut ovat myös Cell-luokan instansseja.

Cell sisältää hilasolun luokituksen ja saannon, ja käsittelee hilasoluun kohdistuvat operaatiot. Hilasolu kuuluu joko referenssialueen soluihin tai kohdealueen soluihin, mutta ei molempiin samanaikaisesti. Luontivaiheessa soluun generoidaan aina jokin luokitus, mutta saanto generoidaan vain referenssialueen soluille. Kohdealueen solut saavat saantotiedon Logic-luokan kautta, kun algoritmi on ajettu läpi.

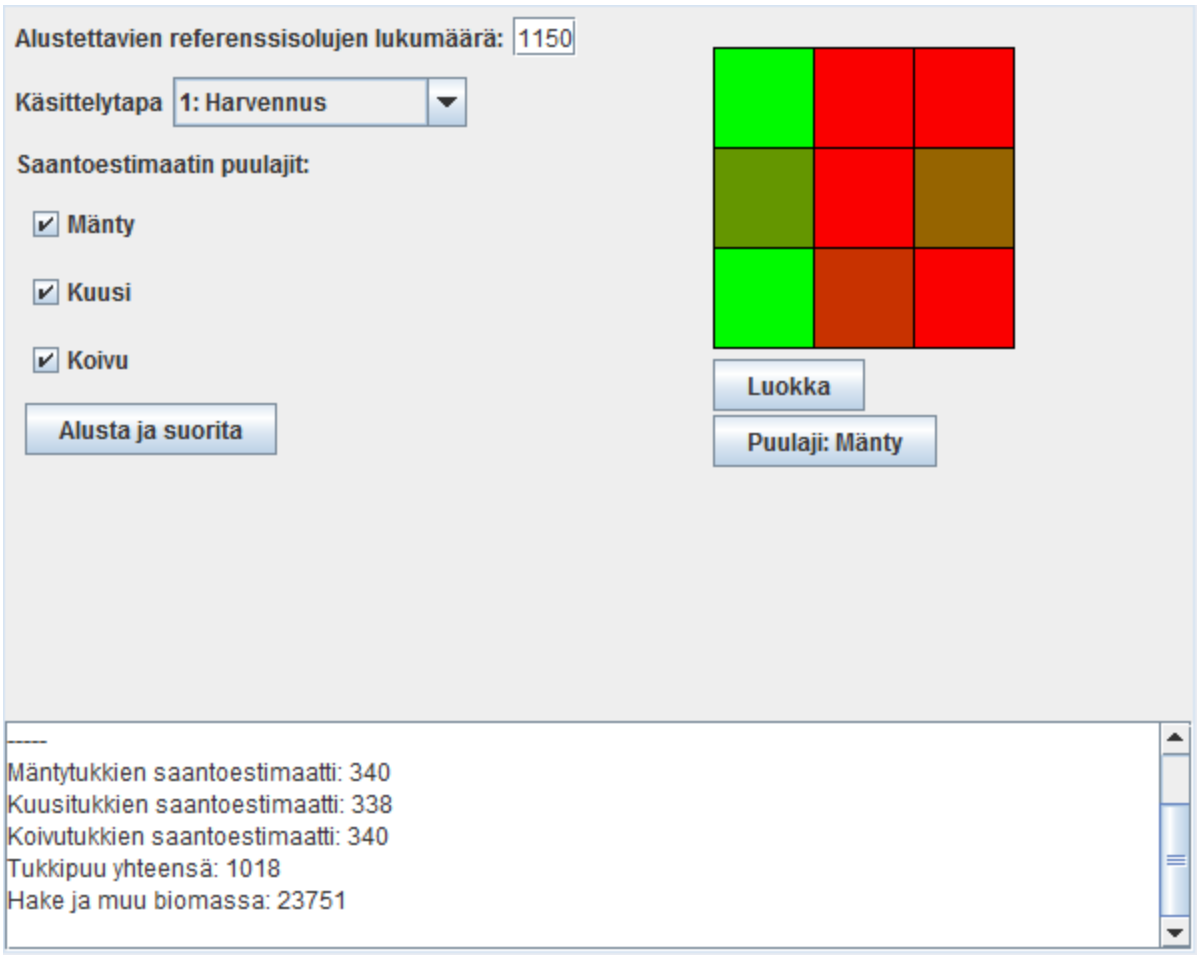
Tyyppiluokittelu, joka sijaitsee Classification-luokassa, koostuu kolmen kokonaisluvun vektorista. Luvut kuvastavat männyn, kuusen ja lehtipuun määrää hilasolussa. Kukin niistä voi saada arvon nollasta viiteen (0 - 5), mikä tarkoittaa 216 mahdollista eri tyyppiluokituksen variaatiota. Saantotieto sisältää vastaavasti tukkimäärät mainitulle kolmelle puutyypille sekä hakkuun muun biomassan. Referenssisolun saantotieto on riippuvainen sen luokituksesta: mitä suurempi luokituksen arvo, sitä suurempi saanto saadaan kyseistä puutyyppiä. Biomassan määrä on suoraan sidonnainen puulajiluokitusten arvoihin, tosin siihen generoidaan pientä hajontaa.

8.2 Toiminnallisuus

Ohjelman tarkoitus on testata pseudokoodin 1 mukaista tyyppileimikoinnin algoritmia. Sen peruskäyttötapaus on yksinkertaisuudessaan seuraava: käyttäjä antaa alustusparametrit, ajaa algoritmin ja saa kohdealueelle kokonaissaantoestimaatin. Väärät alustusparametrit antavat virheilmoituksen. Algoritmista poiketen käyttäjä saa myös solukohtaiset saantoestimaatit. Ohjelman käyttöliittymä on esitetty kuvassa 14.

Käyttäjä voi valita käyttöliittymästä generoitavien referenssisolujen määrän, käsittelytavan/hakkuutoimenpiteen sekä tyyppiluokittelussa käytettävät puutyypit. Puutyyppien valinta ei kuulu lopulliseen algoritmiin, vaan sitä käytettiin tutkimaan luokittelun vaikutusta saantoestimaattiin. Käyttäjä voi suorittaa algoritmin ”Alusta ja Suorita”-painikkeen avulla. Algoritmin tuottama kokonaissaantoestimaatti annetaan kuvan alareunan konsolissa. Käyttöliittymässä on lisäksi esillä kohdealueen hilasolut, jotka esitetään yhdeksänä nelikulmioina ikkunan oikeassa yläkulmassa. Kukin nelikulmio visualisoi solun luokitusta tai saantoa määritetylle puulajille. Mitä vihreämmäksi solu on värjätty, sitä suurempi sen luokitus/saanto on kyseisellä puutyypillä. Musta väri ilmaisee, ettei visualisoitavaa solutietoa ole saatavilla. Käyttäjä voi vaihtaa tarkasteltavaa puulajia ja suuretta käyttöliittymän painikkeilla.

Nelikulmion painaminen avaa solukohtaisen infoikkunan, josta voi tarkastella solun luokittelua ja saantotietoa. Solun saantotiedoissa ilmaistaan selkeästi, jos algoritmia ei ole vielä ajettu tai referenssialueelta ei saatu solulle saantotietoa. Tiedoista ilmenee myös, mistä referenssisoluista saantotieto on saatu. Referenssisoluja esitetään maksimissaan neljä. Jos solulla on tätä enemmän referenssisoluja, annetaan listan päätteeksi ”++”.



Kuva 14: Demon käyttöliittymä

8.3 Huomioita demosta

Algoritmi vaikutti toimivalta, mutta tehtyjen havaintojen perusteella tyyppiluokittelu voi olla haastava toteuttaa yhtä tarkkana kuin pseudokoodissa. Jo yhdeksälle kohdealueen solulle hyvin suppealla luokittelulla tarvittiin arviolta yli 100 kertaa enemmän referenssisoluja, jotta kaikille kohdealueen hilasoluille löytyisi saantotieto 90% todennäköisyydellä. Todellisuudessa referenssisoluja voidaan tarvita vähemmän, kun solujen leimikkotyypit perustuvat oikeisiin metsämittauksiin eikä satunnaisgenerointiin ja ovat kohdealuepohjaisessa määrittelyssä myös kohdealueen läheltä, vähentäen entisestään kasvuolosuhdepohjaista variaatiota. Toisaalta oikeat kohdealueet ovat paljon suurempia ja luokittelu on luultavasti kattavampi kuin demon pelkkiin puutilavuuksiin perustuva luokittelu. Similariteettilaskenta tai pehmeämmät luokitterurajat voisivat auttaa vähentämään tarvittavien referenssisolujen määrää, tosin estimaattien tarkkuudet voivat huonontua. Demossa tyyppiluokittelun yksinkertaistaminen eli vain yhteen tai kahteen puutyyppiin perustuva luokittelu antoi luokittelun ulkopuolelle jääneistä puutyypeistä virheellisiä saantotietoja.

Myös puuttuvien referenssitietojen vaikutusta testattiin. Demossa hilasolua, jolle ei löydy saantotietoa, ei oteta mukaan lopulliseen estimaattiin. Tällä ei ole vaikutusta hilasolujen keskiarvoon, mutta voi antaa pessimistisen kuvan kohdealueen kokonaissaannosta. Referenssittömät solut voi huomata nopeasti solukohtaisista saantotiedoista ja pääikkunan kohdealueen visualisoinnista, jossa referenssittömät solut

esitetään mustina. Kokonaissaantoestimaatista ei voi havaita, että saantotietoja puuttuu, vaan tästä on ilmoitettava erikseen.

Koska jokaista referenssisolua vertaillaan jokaisen kohdealueen solun kanssa, algoritmin kertaluokka on n^2. Demossa tämä ei näkynyt algoritmin suoritusajassa, mutta suuria kohdealueita analysoidessa algoritmi voi olla hyvin hidas. Tyyppiluokittelun algoritmi voidaan luultavasti nopeuttaa nlog(n)-kertaluokkaan hyvällä tietokantasuunnittelulla ja indeksoinnilla, jos palvelualustan tietokanta on relaatiotietokanta. Similariteettilaskentaa on vaikeampi optimoida samalla tavalla, koska solujenvälistä samankaltaisuutta ei voida määrittää ennalta. Jos solujen similarisuuteen vaikuttaa selvästi jokin avainattribuutti kuten puun läpimitta, yhtenä ratkaisuna voi olla referenssisolujen järjestäminen kyseisen attribuutin avulla tasapainotetuksi hakupuuksi. Tällöin osa selvästi kohdesolun avainattribuutin kannalta poikkeavista referenssisoluista voidaan jättää vertailematta ja laskennan kertaluokka saadaan lähemmäksi nlog(n):ää.

9. Pohdinta

9.1 Toiminnallisten palveluiden vaatimukset

Koostetun tietorakenteen, pseudokoodien ja demon perusteella voidaan tutkia tyyppileimikoinnin käyttötapaukseen tarvittavia toiminnallisia palveluita. Tarkastelu kohdistuu erityisesti siihen, miten tyyppileimikointi vaikuttaa palveluiden suorituskykyyn, skaalautuvuuteen ja tarkkuuteen.

Molemmissa algoritmeissa vector-to-grid-paikkakonversio muuttaa käyttäjän antaman kohdealueen hilapohjaiseen muotoon. Muunnoksen suorituskyky ja skaalautuvuus tuskin tulevat olemaan käyttötapauksen suorituskykyä rajoittavina tekijöinä, mutta palvelun tarkkuuteen täytyy kiinnittää erityistä huomiota. Käyttäjän monikulmio voi koostua useasta eri monikulmiosta, tai toisiaan leikkaavista monikulmioista. Monikulmioiden tulkitsemiseksi rajapintaan pitää määritellä tarkasti niiden esitystavat, sekä miten kuviot tulkitaan hilamuotoiseksi. On myös huomioitava, miten tulkitaan tilanne, jossa monikulmio leikkaa solua: tulkitaanko hilasolun saantotieto kokonaan kokonaisestimaattiin, jos monikulmio leikkaa solua riittävästi (esim. yli puolet solun pinta-alasta) vai vain leikkaavan alueen osalta.

Tyyppileimikoinnin suorituskyvyn kannalta olennaisimmat palvelut ovat datafuusio, ajantasaistus ja hilasolujen haku. Jos referenssialueet määritellään vakioaluepohjaisesti, puustotietojen käsittely voidaan suorittaa palvelualustalla taustaprosessina ennen hakua, jolloin haunaikainen prosessointi vähenee. Referenssisolut voidaan luultavasti tallentaa johonkin palvelualustan omaan tietokantaan koostetun tietorakenteen mukaisesti. Kohdealueiden ja kohdealuepohjaisten referenssisolujen käsittely haun aikana voivat hidastaa tyyppileimikoinnin suorituskykyä etenkin silloin, kun puustotiedot haetaan useasta eri tietolähteestä. Alla on esitetty muutama vaihtoehto ongelman ratkaisuksi.

- **Rajoitetut tietolähteet:** Haunaikana datafuusiosta jätetään pois esiprosessoitavat tai muuten hitaat tietolähteet, kuten LUKE:n MVMI. Ajantasaistus tehdään vain tarvittaessa. Tällä tavoin hausta ei tule liian pitkäkestoista käyttäjälle. Toisaalta datafuusiosta tulee epätarkempi, ja osaa tietolähteistä ei välttämättä käytetä lainkaan.
- **Kaikkien hilasolujen tallennus:** Datafuusio suoritetaan taustaprosessina, jonka jälkeen kaikki hilasolut tallennetaan palvelualustalle. Haettaessa hilasoluja ne voidaan antaa suoraan prosessoitavaksi palvelualustan tietovarastosta ilman erillistä datafuusiota. Ratkaisu kuitenkin skaalautuu huonosti, kun hilasolujen määrä lähenee oletettua miljoonaluokkaa; solujen tallentaminen vaatii paljon tallennuskapasiteettia, eikä puustotietojen hakeminen suuresta (SQL-)tietokannasta ole välttämättä nopeaa.
- **Suoritusajan pidentäminen:** Sallitaan prosessin käyttää niin paljon aikaa kuin se tarvitsee algoritmiin tarvittavien palveluiden suorittamiseen. Tämä ratkaisu on hyvin varmasti hitaampi kuin edelliset, vaatii ylimääräistä tallennustilaa tulosten väliaikaiseen varastointiin, ja voi monimutkaistaa asiakasohjelmien toteutusta, joiden täytyy jotenkin saada tulokset käsiinsä algoritmin päätyttyä.

Jos algoritmissa käytetään vakioaluepohjaista referenssimäärittelyä, paras ratkaisu on luultavasti kahden ensimmäisen ratkaisun yhdistelmä: Kohdealueen solut voidaan hakea rajoitetuista tietolähteistä, kun taas kaikki referenssisolut voidaan tallentaa palvelualustalle algoritmia varten. Hyödyllisimmät referenssisolut voidaan suodattaa entisestään esimerkiksi MVMI:n sektoreiden avulla riippuen kohdealueesta. Kohdealueriippuvaisessa määrittelyssä suoritusajan pidentäminen vaikuttaa olevan paras vaihtoehto,

mutta hyvällä tietokantasuunnittelulla kaikkien hilasolujen tallentaminen voi myös osoittautua hyväksi ratkaisuksi.

Toistaiseksi demossa alueiden suodattaminen on sivutettu. Kohdealue on suodatettava ainakin ennen datafuusiota, ja koska aluekaavat tallennetaan perinteisesti monikulmioina, suodatus kannattaa ajoittaa myös ennen paikkakonversiota. Suodatus voi olla tärkeä osa kohdealuekohtaista referenssialuemäärittelyä, jos referensseistä halutaan karsia solut, joista ei ole saantotietoa, tai muuten kelvottomat alueet pois similariteettilaskennan optimoimiseksi.

Notifiointi tulee tärkeäksi osaksi tyyppileimikointia, jos suoritusajan pidentäminen osoittautuu ainoaksi käyttökelpoiseksi vaihtoehdoksi. Notifioinnilla ei ole merkittäviä suorituskyvyllisiä tai skaalautuvuus ongelmia, mutta sen on pystyttävä ylläpitämään tietoa hauista, jotta tulokset voidaan lähettää oikealle osapuolelle.

9.2 Uudet palvelut

Kriittisenä tyyppileimikoinnin osana on kahden hilasolun samankaltaisuuden arviointi. Samankaltaisuuden ”asteille” on annettu vaihtoehdoiksi tyyppiluokittelu ja similariteettilaskenta, mutta toistaiseksi tässä työssä ei ole otettu kantaa, miten itse vertailu voisi tapahtua algoritmin sisällä. Jotta mahdollista vertailua voitaisiin käyttää laajemmassa skaalassa ja muissa käyttötapauksissa, voisi olla tarpeen kehittää erillinen toiminnallinen palvelu kahden hilasolun vertailemiseksi. Yksinkertartaisimmillaan palvelulle annetaan kaksi hilasolua vertailtavaksi, johon palvelu antaa ulostulona binäärisen kyllä/ei vastauksen, tai sitten palvelu antaa solujen samankaltaisuuden numeroarvon, esimerkiksi prosenttiluvun, jota voidaan verrata käyttötarkoituksesta riippuen johonkin raja-arvoon.

Tyyppileimikointi on tässä työssä lähtenyt periaatteesta, jossa käyttäjä antaa haluamansa kohdealueen ja saa sille saantoestimaatin, joka perustuu datafuusiolla muodostettuihin saantotietoihin referenssialueelta. Toisaalta, leimikoinnin voisi suorittaa myös käänteisellä periaatteella: käyttäjä antaa vanhan hakkuualueen tai teoreettisen hakkuualueen, johon hän saa vastauksena sellaiset kohdealueet, joissa hän voi ja saa suorittaa samankaltaisia hakkuita. Käyttöskenaario on skaalaltaan paljon työssä esiteltyä tyyppileimikointia laajempi ja vaatisi enemmän prosessointiaikaa, mutta ei vaikuttaisi vaativan mitään uutta toiminnallisuutta palvelualustalta käyttötapauksen toteuttamiseksi.

9.3 Yhteenveto

Työn tavoitteena oli tutkia tyyppileimikoinnin toteutusta palvelualustalla kolmen tutkimuskysymyksen avulla:

- Millainen koostettu tietorakenne olisi optimaalinen palvelualustan käytettäväksi tukemaan tyyppileimikointia? Mitä attribuutteja siihen sisältyy?

Koostettu tietorakenne muodostettiin alan kirjallisuuden sekä tietokantojen rakenteiden perusteella. Tietorakenne perustuu metsäalalla käytettyihin hilasoluihin, joihin voidaan tallentaa solukohtaisten metsävaratietojen lisäksi mm. maaperätietoja, hakkuutoimenpiteitä ja omistajuustietoja.

- Miten tyyppileimikoinnin algoritmi kannattaisi toteuttaa?

Tyyppileimikoinnin algoritmille esitettiin kahta eri variaatiota, joista toinen perustui tyyppiluokitukseen ja toinen similariteettilaskentaan. Tyyppiluokitusta tutkittiin tarkemmin toteuttamalla demosovellus, jolla voitiin havainnoida luokituksen dimensioiden vaikutusta tulosten tarkkuuteen ja oikeellisuuteen. Demon

avulla havaittiin myös luokittelun ja kohdealueen koon vaikutus tarvittavien referenssisolujen määrään, joka kasvoi merkittävästi mitä laajempaa luokitusta tyyppileimikoinnissa käytettiin.

- Miten rajapinnan toiminnalliset palvelut kannattaisi toteuttaa tyyppileimikoinnin näkökulmasta?

Toiminnallisille palveluille määritettiin rajoituksia ja vaatimuksia, jotka pohjautuivat demossa tehtyihin havaintoihin sekä esiteltyihin tietokantoihin. Merkittävimmät vaatimukset kohdistuivat datafuusioon, ajantasaistukseen ja hilasolujen hakuun, jotka vaativat usean eri tietolähteen metsävaratietoa nopeasti. Näille palveluille esitettiin ratkaisuiksi tietolähteiden rajoittamista, kaikkien hilasolujen tallentamista tai algoritmin suoritusajan pidentämistä. Työssä esitettiin kehitettäväksi myös palvelua, jolla kahta hilasolua voitaisiin vertailla keskenään.

Työn aihepiiriin jäi kuitenkin paljon kehitettävää. Osa huomioista jäi melko pinnallisiksi ja kohdistuivat enimmäkseen sellaisiin puustotietoa käsitteleviin aineistoihin ja tietokantoihin, jotka olivat työtä kirjoittaessa vielä työn alla. Näiden aineistojen lopulliset tietorakenteet ja palvelualustan suorituskyky vaikuttavat pitkälti algoritmin toimintaperiaatteisiin ja suorituskykyyn, joista annettiin tässä työssä vain karkeita arvioita. Lisäksi työssä keskityttiin enimmäkseen puustotietoon ja sivutettiin pitkälti olosuhdetieto, jolla voi olla merkittävä vaikutus saantoestimaattien laatuun. Työ on myös valitettavan teoriapainotteinen.

Lähdeluettelo

[1] Suomen Luonnonvarakeskus, ”Tilastotietokanta,” 2017. [Online]. Available: http://statdb.luke.fi/PXWeb/pxweb/fi/LUKE/?rxid=001bc7da-70f4-47c4-a6c2-c9100d8b50db. [Haettu 30 Maaliskuu 2017].

[2] Suomen Tilastokeskus, ”Kotimaankauppa,” 9 Joulukuu 2016. [Online]. Available: http://tilastokeskus.fi/tup/suoluk/suoluk_kotimaankauppa.html. [Haettu 30 Maaliskuuta 2017].

[3] M. Holopainen, J. Hyyppä, M. Vastaranta ja H. Hyyppä, Laserkeilaus metsävarojen hallinnassa, Helsinki: Helsingin Yliopiston metsätieteiden laitos, 2013.

[4] S. Bauwens, H. Bartholomeus, K. Calders ja P. Lejeune, ”Forest Inventory with Terrestrial LiDAR: A Comparison of Static and Hand-Held Mobile Laser Scanning,” *Forests,* osa/vuosik. 7, nro 6, 2016.

[5] Arbonaut, ”Metsätieto 2020 Tavoitetila,” 2015.

[6] A. Kangas ja M. Maltamo, Forest Inventory: Methodology and Applications, Dordrecht: Springer, 2006.

[7] M. Hyppönen ja P. Roiko-Jokela, ”Koepuiden mittauksen tarkkuus ja tehokkuus,” *Folia Forestalia,* nro 356, pp. 1-25, 1978.

[8] V. Luoma, N. Saarinen, M. A. Wulder, J. C. White, M. Vastaranta, M. Holopainen ja J. Hyyppä, ”Assessing Precision in Conventional Field Measurements of Individual Tree Attributes,” *Forests,* osa/vuosik. 8, nro 2, pp. 38-54, 2017.

[9] J. Vauhkonen, V. Kankare, T. Tanhuanpää, M. Holopainen ja M. Vastaranta, ”Puuston runkolukusarjan ja laatutunnusten mittaus kaukokartoituksella - esiselvitys ja käytännön testi,” Metsäteho, Vantaa, 2013.

[10] M. Holopainen, M. Vastaranta ja J. Hyyppä, ”Outlook for the Next Generation’s Precision Forestry in Finland,” *Forests,* nro 5, pp. 1682-1694, 2014.

[11] M. Maltamo, J. Peuhkurinen, J. Malinen, J. Vauhkonen, P. Packalén ja T. Tokola, ”Predicting tree attributes and quality characteristics of Scots pine using airborne laser scanning data,” *Silva Fennica,* osa/vuosik. 43, nro 3, pp. 507-521, 2009.

[12] M. Holopainen, M. Vastaranta ja J. Hyyppä, ”Yksityiskohtaisen metsävaratiedon tuottaminen - kohti täsmämetsätaloutta?,” *Metsätieteen aikakausikirja,* nro 4, pp. 229-234, 2014.

[13] Skogforsk, Metsäteho, *StanForD2010 standardin XML-skeema,* 2010.

[14] Suomen metsäkeskus, ”Metsätietostandardit esittelysivusto,” 2017. [Online]. Available: http://www.metsatietostandardit.fi/fi/standardit. [Haettu 9 Toukokuuta 2017].

[15] T. Räsänen, ”StanForD 2010:n ensimmäinen versio hyväksytty,” Elokuu 2011. [Online]. Available: http://www.metsateho.fi/stanford-2010n-ensimmainen-versio-hyvaksytty/. [Haettu 20 Syyskuuta 2017].

[16] B. Joshi, Beginning XML with C# 2008 From Novice to Professional, Berkley, California: Apress, 2008.

[17] IETF, ”JSON Schema: A Media Type for Describing JSON Documents,” 21 Huhtikuuta 2017. [Online]. Available: http://json-schema.org/latest/json-schema-core.html. [Haettu 26 Syyskuuta 2017].

[18] D. Lee, ”Fat Markup: Trimming the Fat Markup Myth one calorie at a time,” tekijä: *Proceedings of Balisage: The Markup Conference 2013*, Montréal, Canada, 2013.

[19] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, California: University of California, 2000.

[20] R. T. Fielding, ”REST APIs must be hypertext-driven,” 20 Lokakuuta 2008. [Online]. Available: http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven. [Haettu 19 Toukokuuta 2017].

[21] B. Spies, ”Web Services, Part 1: SOAP vs. REST,” 2 Toukokuuta 2008. [Online]. Available: http://www.ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest. [Haettu 27 Syyskuuta 2017].

Liitteet

Liite 1: Koostetun tietorakenteen XML-skeema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/DataStruct"
xmlns:tns="http://www.example.org/DataStruct" elementFormDefault="qualified"
xmlns:Q1="http://standardit.tapio.fi/schemas/forestData/realEstate"
xmlns:Q2="http://standardit.tapio.fi/schemas/forestData/forestobject"
xmlns:Q3="http://standardit.tapio.fi/schemas/forestData/common"
xmlns:Q4="http://standardit.tapio.fi/schemas/forestData/Stand"
xmlns:Q5="urn:skogforsk:stanford2010">

    <import schemaLocation="../StanForD/HarvestedProduction_V3p3.xsd"
namespace="urn:skogforsk:stanford2010"></import>
    <import schemaLocation="../V9/StandBasicData.xsd"
namespace="http://standardit.tapio.fi/schemas/forestData/Stand"></import>
    <import schemaLocation="../V9/ForestEnumeratives.xsd"
namespace="http://standardit.tapio.fi/schemas/forestData/common"></import>
    <import schemaLocation="../V9/ForestObject.xsd"
namespace="http://standardit.tapio.fi/schemas/forestData/forestobject"></import>
    <import schemaLocation="../V9/RealEstate.xsd"
namespace="http://standardit.tapio.fi/schemas/forestData/realEstate"></import>
    <element name="DataStruct" type="tns:DataStructType"></element>
```

```
<complexType name="DataStructType">
       <sequence>
            <element name="Owners" type="Q1:RealEstateOwnersType" maxOccurs="1"
minOccurs="1"></element>
            <element name="StandAttributes"
                  type="tns:StandAttributesType" maxOccurs="1" minOccurs="1">
             </element>
             <element name="AttributesBySpecies"
                  type="tns:AttributesBySpeciesType" maxOccurs="unbounded"
                  minOccurs="0">
             </element>
             <element name="Yield" type="tns:YieldsType"
                  maxOccurs="unbounded" minOccurs="0">
             </element>
       </sequence>
       <attribute name="CellNumber" type="int"></attribute>
        <attribute name="LastModified" type="date"></attribute>
    </complexType>

    <complexType name="AttributesBySpeciesType">
       <complexContent>
             <extension base="Q2:TreeSummaryType">
                  <sequence>
                        <element name="SpeciesID"
                              type="Q3:TreeSpeciesType" maxOccurs="1"
minOccurs="1">
                        </element>
                        <element name="Uncertainty" type="tns:UncertaintyType"
maxOccurs="1" minOccurs="0"></element>
                  </sequence>
             </extension>
       </complexContent>
    </complexType>

    <complexType name="YieldBySpeciesType">
       <complexContent>
             <extension base="tns:YieldType">
                  <sequence>
                        <element name="SpeciesID"
type="Q3:TreeSpeciesType"></element>
                  </sequence>
             </extension>
       </complexContent>
    </complexType>
```

```
    <complexType name="YieldAverageType">
       <sequence>
             <element name="VolumeTotal" type="string"></element>
             <element name="VolumeLog" type="string"></element>
             <element name="VolumePulp" type="string"></element>
             <element name="DBH" type="string"></element>
             <element name="Height" type="string"></element>
             <element name="TreeCount" type="string"></element>
       </sequence>
    </complexType>

    <complexType name="StandAttributesType">
        <annotation>
             <documentation>Hilasolun puustotunnukset</documentation>
        </annotation>
        <complexContent>
             <extension base="Q2:TreeSummaryType">
                   <sequence>
                         <element name="SoilData" type="Q4:SoilDataType"
maxOccurs="1" minOccurs="1"></element>
                         <element name="MainTreeSpecies"
                               type="Q3:TreeSpeciesType" maxOccurs="1"
minOccurs="1">
                         </element>
                         <element name="DevelopmentClass"
                               type="Q3:DevelopmentClassType" maxOccurs="1"
minOccurs="1">
                         </element>
                         <element name="Uncertainty"
                               type="tns:UncertaintyType" maxOccurs="1"
minOccurs="0">
                         </element>
                   </sequence>
                   <attribute name="Classification" type="int">
                         <annotation>
                               <documentation>Tyyppileimikoinnin puustotunnuksien
luokitus</documentation>
                         </annotation></attribute>
                   <attribute name="ClassificationAux" type="int">
                         <annotation>
                               <documentation>Tyyppileimikoinnin kasvuympäristön
tai olosuhteiden luokitus</documentation>
                         </annotation></attribute>
             </extension>
       </complexContent>
    </complexType>
```

```
<complexType name="YieldsType">
	<sequence>
		<element name="StandRef" type="tns:StandAttributesType"
			maxOccurs="1" minOccurs="1">
		</element>
		<choice>
			<element name="ProcessedTree"
				type="Q5:SingleTreeProcessedStemType"
maxOccurs="unbounded"
				minOccurs="1">
			</element>
			<sequence><element name="StandYield" type="tns:YieldType"
maxOccurs="1" minOccurs="1">
			</element><element name="YieldBySpecies"
type="tns:YieldBySpeciesType" maxOccurs="unbounded" minOccurs="1">
		</element></sequence>
		</choice>
	</sequence>
	<attribute name="OperationType" type="int"></attribute>
	<attribute name="HarvestDate" type="date"></attribute>
    </complexType>

    <complexType name="YieldType">
      <sequence>
		<element name="VolumeTotal" type="string"></element>
		<element name="VolumeLog" type="string"></element>
		<element name="VolumePulp" type="string"></element>
		<element name="DBH" type="string"></element>
		<element name="Height" type="string"></element>
		<element name="TreeCount" type="string"></element>
      </sequence>
    </complexType>

    <complexType name="UncertaintyType">
      <annotation>
		<documentation xml:lang="fi">Puustotunnusten RMSE:t (%).</documentation>
      </annotation>
      <sequence>
		<element name="DiameterRMSE" type="string"></element>
		<element name="HeightRMSE" type="string"></element>
		<element name="VolumeRMSE" type="string"></element>
		<element name="BasalAreaRMSE" type="string"></element>
      </sequence>
    </complexType>
</schema>
```

Liite 2: Tyyppileimikoinnin demon lähdekoodi

Saatavilla linkistä: https://www.dropbox.com/s/t3fix491ubs4nz4/Tyyppileimikointi_demo.zip?dl=0