Selvitys asemakaavatiedon versionhallinnasta ja validoinnista

Otso Helenius, OTSO.DEV

18. joulukuuta 2019

OTSO.dev
consult@otso.dev
+358 50 544 9174

Sisältö

OTSO.dev
consult@otso.dev
+358 50 544 9174

OTSO.dev
consult@otso.dev
+358 50 544 9174

Tiivistelmä

Tämä dokumentti on Paikkatietoalusta-hankkeen osana toteutettavaa Maankäyttöpäätökset-hanketta varten tuotettu selvitys asemakaavoitusprosesseissa käsiteltävän ja tuotettavan tiedon versionhallinnasta (elinkaaren hallinnasta) sekä validoinnista (laadunvarmistuksesta) Ympäristöministeriössä kehitteillä olevan kaavatietomallin kontekstissa. Selvityksen tuloksia hyödynnetään osana Maankäyttö- ja rakennuslain kokonaisuudistusta.

Selvityksessä perehdytään rakenteisen tietomallimuotoisen asemakaavasisällön versionhallintaan liittyviin hyötyihin, haasteisiin, sekä toteutustapoihin. Lisäksi tarkastellaan validoinnin toteutusta sekä sen hyötyjä, haasteita ja toteutustapoja osana versionhallittua suunnitteluprosessia.

Selvitys alkaa yleisluontoisella käsitteiden kuvauksella, josta siirrytään kansainvälisen vertailun sekä kansallisen suunnittelua koskevan tiedonhallinnan nykytilan kartoituksen kautta tuloksiin ja MRL-kokonaisuudistuksen osatavoitteiden ohjaamiseen ja niitä edistäviin suosituksiin.

Helsingissä

18. joulukuuta 2019

Otso Helenius

OTSO.dev
consult@otso.dev
+358 50 544 9174

1 Johdanto

Asemakaavan voidaan perustellusti sanoa olevan merkittävin maankäyttöpäätöstyyppi, sillä se kytkee ylempien kaavatasojen ohjauksen ja paikalliset reunaehdot konkreettisiin ja sitoviin rakennettua ympäristöä koskeviin muutoksiin. Hyväksytty asemakaava on tyypillisesti pitkällisen suunnittelu- ja vuorovaikutusketjun ja iteratiivisen suunnitelmien hiomisen lopputuote. Kaavan laadulliseen sisältöön kohdistuu jo nykyisessä lainsäädännössä suuri joukko vaatimuksia, joihin vastaavien ratkaisujen suoria ja epäsuoria vaikutuksia joudutaan arvioimaan varsin kattavasti.

Kaavaprosessin tiedonhallinnalta vaaditaan johdonmukaisuutta ja luotettavuutta, jotta se täyttää kyseiset vaatimukset. Näiden laadullisten kriteerien täyttyminen vuorostaan on riippuvainen ennen kaikkea versionhallinnan ja validoinnin – toisilta nimiltään elinkaari- ja laatusääntöjen – muodostaman perustan toiminnasta.

Näihin seikkoihin otetaan monien muiden ohella kantaa meneillään olevassa Maankäyttö- ja rakennuslain (MRL) kokonaisuudistuksessa, jonka tavoitteena on ”yksinkertaistaa alueidenkäytön suunnittelujärjestelmää, kehittää rakentamisen ohjausta, tukea kansalaisten mahdollisuuksia vaikuttaa omaa elinympäristöä koskevaan suunnitteluun ja päätöksentekoon sekä varmistaa, että lakiteksti on selkeä ja johdonmukainen”.[1] Kokonaisuudistuksen tavoitteisiin pyritään lukuisissa osahankkeissa, joista Paikkatietoalusta-hankkeen osana toteutettava Maankäyttöpäätökset-hanke keskittyy suunnittelujärjestelmän digiloikkaan. Sen osana toteutettu Kuntapilotti-projekti on tuottanut viiden kunnan kanssa prototyypin kaavoitukseen soveltuvista tietomallista ja -prosesseista, joiden on tarkoitus tuottaa harmonisoitua suunnittelutietoa kansalliselle Paikkatietoalustalle sekä kaavoituksen omiin että muiden rinnakkaisten prosessien ja toimijoiden tarpeisiin.

Tämä dokumentti on Maankäyttöpäätökset-hanketta varten tuotettu selvitys asemakaavoitusprosesseissa käsiteltävän ja tuotettavan tiedon versionhallinnasta ja validoinnista Ympäristöministeriössä kehitteillä olevan kaavatietomallin kontekstissa. Selvitys pyrkii kattavaan ja yleistajuiseen kuvaukseen versionhallinnan ja validoinnin roolista asemakaavoituksessa niin kansallisen tason, kuntien kuin muidenkin toimijoiden perspektiivistä. Tavoitteena on myös tuoda esiin kyseisten – varsin tarkkaan rajattujen – toimenpiteiden systeemisistä vaikutuksista suunnittelujärjestelmään, sekä lisätä teknisesti orientoituneen ja muun yleisön keskinäistä ymmärrystä aihealueen puitteissa.

Lisäksi selvitys kuvaa toimenpidesuosituksia, joiden katsotaan edesauttavan kaavoituksen versionhallinnan ja tiedon validoinnin onnistunutta toteutusta osana MRL-kokonaisuudistusta.

Selvityksen on tuottanut itsenäinen maankäytön suunnitteluteknologian konsulttiyritys OTSO.dev, ja sen ohjauksesta sekä seurannasta ovat vastanneet Maankäyttöpäätökset-hankkeen erityisasiantuntija Satu Taskinen ja projektinjohtaja Sakari Jäppinen Ympäristöministeriöstä. Työ on toteutettu 8.7.–2.12.2019 välisenä aikana.

[1] `https://mrluudistus.fi/`

OTSO.dev
consult@otso.dev
+358 50 544 9174

1.1 Tutkimuskysymykset

Selvityksessä vastataan seuraaviin tutkimuskysymyksiin:

1. Mitä merkittäviä hyötyjä ja haasteita liittyy kansallisen kaavatiedon versionhallinnan ja validoinnin käyttöönottoon ja toteuttamiseen?
2. Miten kaavatietomallin rakenne vaikuttaa tiedon versiontiin ja validointiin?
3. Miten maankäyttöpäätöstietojen versionhallinta ja validointi on toteutettu kansainvälisissä esimerkeissä? Mitä tietoja versioidaan ja validoidaan? Mitä hyvää ja huonoa toteutuksissa on?
4. Mitkä ovat keskeisten kansallisten toimijoiden valmiudet versionhallintaan ja validointiin?
5. Mitä tulee ottaa huomioon maankäyttöpäätöstietojen versioinnista ja validoinnista säädettäessä ja ohjeistaessa? Mitä instrumenttia tähän tulisi käyttää?

Tutkimuskysymyksiin vastataan suositusten muodossa luvussa 4. Suositukset pohjautuvat edeltävissä luvuissa johdettuihin tuloksiin, jotka on esitetty tiivistetyssä muodossa pykäliin jaettuina argumentteina.

1.2 Lähdeaineisto

Selvityksessä hyödynnetään Paikkatietoalusta-hankkeen osahankkeiden – erityisesti Kuntapilotti-hankkeen – julkaistujen tulosten sekä Tulevaisuuden maankäyttöpäätökset -hankkeen välitulosten lisäksi relevanttia akateemista tutkimuskirjallisuutta.

Olennaisen osan lähdeaineistoa muodostaa myös tämän selvityksen tekijän väitöskirjatyössään tuottama sisältö. Se koostuu kaavoitusprosessien rakenteen, tiedonhallinnan ja siihen käytettävän teknologian kehittämisestä ja tutkimuksesta niin Suomessa kuin ulkomailla. Aineistoa on tuotettu Aalto-yliopiston koordinoimissa konsortiohankkeissa sekä yhdessä KIRA-digi-kokeiluhankkeessa yhteistyössä yritysten, kuntien ja muiden viranomaistahojen kanssa.

1.3 Lähtökohdat

Selvitys pohjautuu sekä akateemisen soveltavan tutkimuksen että MRL-uudistuksen osana määriteltyihin arvovalintoihin, jotka toimivat tulosten ja annettavien suositusten pohjana.

1.3.1 Uudistuksen lähtökohdat

MRL-uudistuksen päätavoitteiden lisäksi selvityksen toteutusta ohjaavat Maankäyttöpäätökset- ja Tulevaisuuden maankäyttöpäätökset -hankkeiden (Taskinen, 2018; Taskinen, 2019b) keskeiset päämäärät. Niitä ovat vaatimus maankäytön suunnittelu- ja päätöstietojen rakenteisuuden, avoimuuden, semanttisen vakioinnin, laadunvarmistuksen, elinkaaren hallinnan ja yhteentoimivuuden lisäämisestä sekä kyseisen tiedon kustannustehokkaasta tuottamisesta kansalliselle Paikkatietoalustalle.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Tavoitteena on myös luoda helppokäyttöisiä palveluita, jotka tukevat sujuvaa vuorovaikutusta maankäyttöpäätösten valmistelussa ajantasaisen, ymmärrettävän ja helposti saavutettavan tiedon avulla. Lisäksi pyritään demokraattisen päätöksenteon ja oikea-aikaisen vuorovaikutuksen toteutumiseen ohjaamalla toimijat soveltamaan parasta mahdollista tietoa ja selkeitä johdonmukaisia päätöspolkuja.

1. Suunnittelun ja päätöksenteon täytyy pohjautua ymmärrettävään tietoon, jonka laatu, rakenne, käsitteet ja elinkaari ovat yhtenäisiä ja hallittuja.
2. Suunnittelussa tapahtuvan vuorovaikutuksen ja toimijoiden välisen tiedonvaihdon tulee hyödyntää yhteisiä harmonisoituja rajapintoja.
3. Suunnittelun täytyy pystyä ottamaan huomioon, vaikuttamaan sekä mukautumaan toimintaympäristössä tapahtuviin muutoksiin hallitusti.

1.3.2 Tutkimukselliset lähtökohdat

Tutkimuksellisen pohjan muodostaa oletus siitä, että maankäyttö- ja rakennuslain kokonaisuudistus on väistämättä rakenteeltaan *systeeminen*. Tällä tarkoitetaan sitä, että uudistuksen toteutusta ei voida hajauttaa ratkottavaksi sisällöltään itsenäisiin komponentteihin: esimerkiksi tämän selvityksen kontekstina olevan kaavatietomallin rakennetta ei siis voi pitää vain puhtaasti teknisenä määrittelykysymyksenä.

Tämä johtuu siitä, että kaikki uudistuksen osat vuorovaikuttavat keskenään sekä laajemmin uudistuksen kohteena olevan yhteiskunnan kanssa – osien vaikutusmekanismit eivät ole välttämättä suoria tai edes näkyviä, ja erityisesti pidemmän aikajänteen johdannaisvaikutusten ennakointi on haastavaa.

Kaavoitus on lisäksi *sosiotekninen* järjestelmä: kulloinkin saatavilla oleva teknologia määrittää alan tiedollisia rajoja (mitä suunnittelun kohteena olevasta yhteiskunnasta ja suunnittelun vaikutuksista siihen voidaan tietää) sekä arvovalintoja (millaisiin ratkaisuihin tulisi pyrkiä ja millä keinoin). Ala vuorostaan valitsee käyttöön ja hylkää erilaisia teknologioita kulloinkin vallitsevien tarpeiden, tavoitteiden ja reunaehtojen puitteissa.

On yleinen virhe olettaa, että teknologia on puhtaasti alisteista sitä käyttävän organisaation tai toimijan tavoitteille, ja että järjestelmäntoimittajan tarjoama ratkaisu voidaan täten valita vain teknillistaloudellisten kriteerien perusteella. Teknologian käyttö ei ole koskaan täysin sen suunnitellun toiminnallisuuden mukaisesti kontrolloitua ja ennakoitavaa, vaan vuorovaikutukseen liittyy aina epätarkoituksenmukaisia piirteitä ja ennakoimattomuutta, jotka vaikuttavat kyseiseen teknologiaan, sen käyttäjään, käyttötapaan ja seurauksiin (Verbeek, 2015).

Konkreettinen esimerkki siitä, miksi kaavatietomallin versionhallintaa ja validointia ei tule lähestyä vain teknisenä ongelmana löytyy järjestelmä- (engl. *systems engineering*, SE) ja ohjelmistotuotannon (engl. *software engineering*, SWE) puolelta. Vielä pitkälti 1990-luvulle asti suuriakin järjestelmäkokonaisuuksia toteutettiin olettaen, että asiakkaan tarpeet voidaan tulkita ja koota formaaleiksi järjestelmää koskeviksi vaatimuksiksi (engl. *requirements specification*), joiden toteuttaminen ohjelmoimalla tapahtuu arvoneutraalissa ympäristössä (Boehm, 2003). Toisin sanoen oletettiin, että järjestelmäntoimittaja kykenee sekä ymmärtämään asiakkaan tarpeet että toteuttamaan ratkaisun objektiivisesti.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Kuitenkin esimerkiksi Emam ja Koru (2008) selvityksessä 45-52 % projekteista oli epäonnistunut joko kokonaan tai osittain, ts. ylittänyt budjetin, myöhästynyt tai toteutunut ominaisuuksiltaan tai laadultaan vajavaisina. Forbesin yritysten digitalisaatiota koskevassa haastattelussa (B. Rogers, 2016) uudistusten arvioitiin epäonnistuvan 84 %:ssa tapauksista. Alan sisäisessä keskustelussa ohjelmistojen tuottavuuden heikkoa kasvua suhteessa tarjolla olevaan laskentatehoon sekä tiedon määrään on kuvattu termillä *software crisis* (Fitzgerald, 2012). Projektien epäonnistumisiin johtaneita syitä on tyypillisesti useampi kuin yksi, ja käytännössä jokainen liittyy juuri ohjelmistotuotannon inhimilliseen puoleen, kuten kommunikaatioon ja organisaatiokulttuuriin (Verner, Sampson ja Cerpa, 2008).

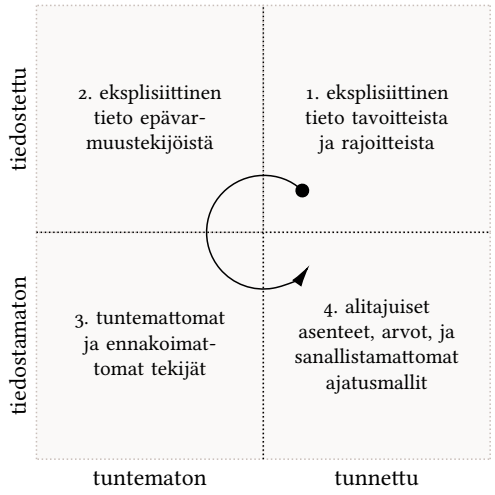
Järjestelmä- ja ohjelmistotuotannossa on kasvavassa määrin opittu tunnistamaan juuri vaatimusmäärittelyjen kehityksen kautta laajempia systeemisiä haasteita jotka tulee ottaa huomioon (katso esim. Dick, Hull ja Jackson, 2017; ISO/IEC/IEEE, 2018; Lauesen, 2018; O'Regan, 2017). Pohjimmiltaan kyse on yhteismitattomuuden tunnistamisesta: ohjelmistot ovat selkeärajaisia keinotekoisia järjestelmiä, jotka koostuvat tunnistettavista osista ja tarkkaan määritellyistä riippuvuuksista, mutta niitä käytetään laajalti osana sosioekonomisia järjestelmiä, jotka taas ovat orgaanisia ja joustavia (Becker et al., 2016). Tämä yhteismitattomuus ilmenee järjestelmien kehitystyössä mm. seuraavin tavoin:

1. Asiakasorganisaation on vaikea tunnistaa ja sanallistaa, mitä kehitettävältä järjestelmältä tarkalleen halutaan.
2. Vaatimukset keskittyvät herkästi optimaaliseen prosessimäärittelyyn tai pinnalla oleviin käytännön haasteisiin systeemisten ongelmien jäädessä piiloon.
3. Organisaatiossa on tyypillisesti useita tahoja (ylläpitäjät, päälliköt, asiantuntijat jne.), joiden tarpeet tai toiveet kehitettävää järjestelmää kohtaan ovat joiltain osin ristiriidassa.
4. Organisaatio kykenee vain harvoin arvioimaan kehitettävää järjestelmää koskevien toiveiden toteutuskelpoisuutta, sillä järjestelmänkehitys ei kuulu sen omaan erikoisosaamiseen.
5. Organisaatio ilmaisee tarpeensa oman viitekehyksensä (engl. *domain*) kielellä ja käsitteistöllä, jotka lähes aina eroavat vaatimusmäärittelyä tekevän tahon viitekehyksestä.
6. Vaatimusmäärittelyssä sekoitetaan usein vaatimukset (mitä järjestelmän tulee tehdä) ja toteutus (kuinka järjestelmä suorittaa vaatimuksen mukaisen tehtävän).
7. Järjestelmäkehityksessä halutaan tyypillisesti minimoida muutokset organisaation rakenteeseen ja toimintaan, mikä rajaa huomattavasti tarjottavien ratkaisujen joukkoa.

Kuvassa 1 on esitetty näiden haasteiden taustalla vaikuttavia tekijöitä. Järjestelmäuudistusten suunnittelua ohjaavat tyypillisesti *tiedostetut tunnetut* (kuvassa kenttä nro. 1) tekijät: vaatimusmäärittelyssä pyritään kuvaamaan *funktionaalisilla vaatimuksilla*, mitä järjestelmän tulisi tehdä, sekä *ei-funktionaalisilla vaatimuksilla* ehdot, joiden puitteissa järjestelmän tulee täyttää ensiksi mainitut.

Laadukkaassa suunnittelussa pyritään myös tunnistamaan *tiedostetut tuntemattomat* (2) eli vaikutustekijät, joiden suuntaa ja suuruutta ei tunneta, ja suunnittelemaan järjestelmä joko immuuniksi tai riittävän joustavaksi, jotta se toimii myös näiden epävarmuustekijöiden puitteissa. Harvinaisempi, yleensä kriittisiä järjestelmiä koskeva suunnittelu pyrkii

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 1: Nelikenttä tietämisen laadullisesta luokittelusta. Nuoli kuvastaa järjestystä yleisimmin projekteissa huomioidusta kategoriasta (1) harvinaisimpaan (4).

ratkaisemaan, miten järjestelmän tulisi reagoida *tiedostamattomiin tuntemattomiin* (3) tekijöihin eli nk. *mustiin joutseniin*, joiden olemassaoloa saati vaikutusta ei osata ennakoida. Kaikista pienimmälle huomiolle jäävät tyypillisesti *tiedostamattomat tunnetut* (4) tekijät, eli asenteet, arvot ja toimintamallit, jotka ovat niin kiinteä osa toimintaa, ettei niitä tiedosteta ja osata kyseenalaistaa.

Edellä kuvatut seikat eivät siis ole alasidonnaisia, vaan yleisen tason järjestelmänkehityksen haasteita, jotka tulee näin ollen ottaa huomioon myös maankäytön suunnittelun kentällä.

Kehitysprojektien puitteissa mahdollisia konkreettisia parannuskeinoja ovat mm. (Becker et al., 2016, s. 61–62) kuvaamat kestävän vaatimusmäärittelyn periaatteet, Lauesen (2018) tapaustutkimuksessa kuvattu ongelmalähtöisen vaatimusmäärittelyn (engl. *problem-based requirements engineering*) hyödyntäminen, ketterät (engl. *agile*) tiiviiseen vuorovaikutukseen nojaavat kehitysmenetelmät kuten Scrum ja Kanban sekä jatkuvan toimituksen ja integraation menetelmät (DevOps).

Vaikka näillä menetelmillä voidaan parantaa kehitettävän järjestelmän ja sitä käyttävän organisaation kykyä varautua kuvassa 1 ilmeneviin haasteisiin, suunnitellaan suuria tietojärjestelmiä tyypillisesti edelleen enemmän vesiputousmallia muistuttavassa lineaarisessa kontekstissa (määrittely → toteutus → testaus → käyttöönotto).

Järjestelmäkehityksen kaltaisten muutosprosessien käynnistäminen ja läpivienti on raskasta, tulosten, aikataulun sekä kustannusten ennakointi haastavaa, ja muutos herättää tyypillisesti muutosvastarintaa asiakasorganisaation sisällä.

Koska projektiluonteinen järjestelmäkehitys nähdään tyypillisesti ”pakollisena pahana”, joka edeltää käyttöönottoa ja sen mukanaan tuomia hyötyjä, saatetaan määrittelyjä ja erityisesti niitä toteuttavia ratkaisuja tehdä vain kehitysprojektin aikana vallinneisiin tarpeisiin sekä työn laajuuden hallitsemiseksi että kustannussyistä. Nykytilaa palvelevat *ad-hoc*-ratkaisut johtavat kuitenkin vähitellen järjestelmän teknisen velan (engl. *technical debt*)

OTSO.dev
consult@otso.dev
+358 50 544 9174

kasvuun – toisin sanoen siihen, että järjestelmän toiminnan muuttaminen myöhemmin vastaamaan muuttunutta toimintaympäristöä on yhä kalliimpaa ja vaikeampaa.

Valtaosa tässä kappaleessa kuvatuista toimintaan ja muutokseen kohdistuvista ongelmista liittyy kuuluisaan *Maslow'n vasarana* tunnettuun kognitiiviseen vinoumaan[2]. Abraham Maslowin popularisoima vertauskuva[3] vasarasta kuvaa ehdollistumisen vaikutusta ajatteluun ja toimintaan: vaikka itse ratkottavat ongelmat vaihtelisivat luonteeltaan, ohjaavat ammatillinen kokemus, tietopohja ja saatavilla olevat työkalut käsittelemään niitä vakiintuneen toimintamallin ja käsitekehyksen puitteissa.

Kyseisen *business-as-usual*-toimintakulttuurin sijaan organisaatioon tulisi olla sisäänrakennettuna kyky reflektoida ja analysoida omaa toimintaansa, muodostaa sen pohjalta jatkuvasti päivittyvä ajantasakuva muutostarpeista ja ohjata niiden toteutumista asteittain.

4. Järjestelmien tiedonhallinta ja käyttötapaukset eivät ole vain teknisiä kysymyksiä, vaan niitä tulee käsitellä sosioteknisenä haasteena.
5. Eteenpäin vietävien ratkaisujen tulee muodostua ja toteutua alan toimijoiden muodostaman verkoston toimivuuden ehdoilla, ei sektori- tai hierarkialähtöisesti.
6. Ratkaisujen tulee ohjata organisaatioita kohti jatkuvaa omasta toiminnasta oppimista sekä resilienssiä toimintaympäristön muutoksille.

1.3.3 Tiedonhallinnan nykytila asemakaavaprosesseissa

Johdanto-osuuden lopuksi tarkastelemme selvityksen tuottajan meneillään olevaan väitöskirjatutkimukseen pohjaten kaavoituksen tiedonhallinnan nykytilaa lyhyesti.

Yhteenvetona voidaan todeta kuntien ja konsulttien tiedonhallinnan asemakaavojen valmisteluprosessien sisällä olevan hyvin henkilö- ja tilannesidonnaista. Tämä on merkittävä havainto koska sillä on huomattavia vaikutuksia kaavoituksen tehokkuuteen, virhealttiuteen ja syntyvien kaavojen sisällölliseen laatuun.

Nykyinen lainsäädäntö keskittyy määrittelemään pääpiirteittäin virallisten julkistettavien aineistojen muodon ja laadullisen sisällön, sekä niiden valmistelua koskevia ehtoja kuten selvitykset, vaikutusten arvioinnin ja osallistumisen ja vuorovaikutuksen toteutumisen. Näiden ehtojen täyttyminen jätetään laissa kuntien sisäisen prosessimäärityksen ja sen toteutuksen varaan, mikä on ajan myötä kunkin kunnan resurssien ja toimintakulttuurin puitteissa muovannut varsin vaihtelevia käytäntöjä. Lain kuvaamassa laajassa kontekstissa (mm. edellytysten luominen "hyvälle elinympäristölle") määriteltyjen tavoitteiden täyttäminen toimintaympäristön monimutkaistuessa on johtanut siihen, että kaavoitusta pidetään kasvavassa määrin yhä monimutkaisempana ja raskaampana (Rinkinen, 2017; Jama et al., 2018).

[2] Vinouma ja sen variaatiot tunnetaan myös nimillä *Einstellung* (saks.), Déformation professionnelle (ransk.) ja Job conditioning (engl.).

[3] "I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."

OTSO.dev
consult@otso.dev
+358 50 544 9174

Kaavojen valmistelutyö on perinteisesti nojannut ammatilliseen erityisosaamiseen, useiden eri alojen asiantuntijoiden yhteistyöhön sekä hiljaiseen tietoon suunnitteluympäristöjen ja kaavaratkaisujen historiasta. Sukupolvien väliset erot näkyvät selvimmin siinä, että nuoremmat nojaavat merkittävästi vanhempaa sukupolvea enemmän suositeltuja käytäntöjä kuvaaviin ohjeistuksiin, tarkistuslistoihin sekä uusien työkalujen oma-alotteiseen pilotointiin.

Varsinaisia juuri kaavasuunnittelun ja kaavojen valmistelun tiedonhallintaa varten kehitettyjä alakohtaisia työkaluja ei ole olemassa. Tyypillisesti kunnissa tiedonhallinta rakentuu varmennettua paikka-, rekisteri- ja tilastotietoa hallinnoivien järjestelmien sekä suunnitteluprosessissa tarvittavien rakenteetonta (*strukturoimatonta*) tietoa hallinnoivien yleisten projektinhallinta- tai projektipankkiohjelmistojen varaan.

Juuri suunnittelussa käytettävä ja tuotettava rakenteeton tieto on suunnittelua tukevan teknologian kannalta haastavaa. Esimerkiksi tilattu selvitys on tyypillisesti PDF-dokumentti, jossa kuvailevan leipätekstin oheen on upotettu taulukoita, analyysiohjelmistosta otettuja kuvakaappauksia sekä rasteri- tai vektorikarttoja tulosten tulkitsemiseksi. Monia sisällöllisiä asioita viestitään esimerkiksi sanallisesti, paikkatiedosta irroitettuun aineistoon lisättävillä annotaatioilla, havainnollistavilla kaaviokartoilla, taulukoilla ja valokuvilla.

On tärkeää tiedostaa, että tiedonhallinnan ongelmat eivät suinkaan ole puhtaasti sisäsyntyisiä, vaan seurausta myös toimijoiden välisten yhteisten rajapintojen puuttumisesta ja sitä seuraavasta pakosta hyväksyä käsiteltäväksi rakenteetonta tietoa. Vastaavat tiedon rakenteen, sisällön ja hallinnan epäjohdonmukaisuutta koskevat ongelmat sekä niistä seuraavat virheet ja tehottomuus riivaavat sekä julkisen että yksityisen puolen toimijoita.

Rakenteetonta vapaamuotoista sisältöä hyödynnetään ja tuotetaan kaavaprosesseissa yleisesti ottaen runsaasti. Pohjalla vallitsevia syitä on kuvattu alla:

1. Rakenteeton tieto on heterogeenista ja yhteismitatonta, eikä sen rakenteistamiseen ole valmiita suunnittelunaikaisia malleja, työkaluja tai prosesseja.
2. Suunnittelun sisällöllinen kommunikointi eri osapuolille ihmiselle ymmärrettävässä (engl. *human-readable*) muodossa on kaavoituksessa prioriteetti – eri materiaalien tulkitseminen käsin ja koostaminen vapaamuotoisesti mm. havainnollisiksi kaavioiksi, kartoiksi ja esityksiksi täyttävät kommunikaatiota tukevat nykyiset käytännön tarpeet.
3. Valmistelutyön kognitiivisen kuormittavuuden[4] ja aikataulupaineiden vuoksi on monessa tilanteessa myös merkittävästi nopeampaa jalostaa ja viestiä tietoa vapaasti sanallisessa ja visuaalisessa muodossa, vaikka se aiheuttaa runsaasti johdannaista manuaalista työtä, lisää virheiden mahdollisuutta ja sitoo henkilöresursseja myöhemmissä vaiheissa.

Prosessin sisäinen suunnitteluargumentaatio ja viittaukset työnaikaiseen aineistoon[5] ovat yllä mainittujen seikkojen vuoksi käytännössä puhtaasti rakenteettomia. Tietoa pyritään

[4]Suunnittelijat sekä muut valmistelutyöhön osallistuvat joutuvat käsittelemään suurta määrää muodoltaan ja sisällöltään hyvin monenlaista tietoa. Tämä kuormittaa työmuistia huomattavasti ja heikentää siten pidempään jatkuessaan työntekijöiden kykyä suoriutua tehtävistä tehokkaasti ja virheettömästi.

[5]Työnaikaisella aineistolla tarkoitetaan kaikkea sitä tietoa koostavaa ja tulkitsevaa epävirallista materiaalia jonka avulla suunnitelmaa edistetään. Tähän joukkoon lasketaan skissit, tiedon jäsentelyyn käytetyt taulukot, diagrammit, työnaikaiset 3D-mallit jne.

OTSO.dev
consult@otso.dev
+358 50 544 9174

jalostamaan siten, että suunnitelman sisältö jalostuu prosessin edetessä kasvavassa määrin yhdeksi kokonaisuudeksi, joka voidaan tulkita ilman laajoja ja pitkiä viittausketjuja kaikkiin sen pohjana olevan työnaikaisen aineiston eri versioihin. Poikkeuksen tekevät itsenäisiä kokonaisuuksia muodostavat ulkoiset aineistot kuten selvitykset, joihin voidaan viitata helposti läpi prosessin. Hyödynnettäviä työkaluja on käytössä runsaasti, aina graafisen alan ohjelmistoista rakennusalan mallinnus-, työlistojen hallinta- ja toimisto-ohjelmiin.

Prosessimuistia pyritään kartuttamaan tyypillisesti tallentamalla työnaikaisia tietoja organisaation sisäverkkoon esimerkiksi kaavavaiheiden tai päivämäärien mukaan jäsenneltynä. Toinen yleinen ratkaisu on käyttää projektinhallintaohjelmistoa, jossa tiedostoihin liitetään päivämäärän ja tallentajan lisäksi muita yleishyödyllisiä kuvailevia metatietoja. Lisäksi joidenkin tiedostojen sisällöstä voidaan rajatuissa määrin lukea metatietoa automaattisesti. Nämä ratkaisut eivät kuitenkaan kykene luomaan ja ylläpitämään versioitujen tiedostojen sisällön osien välisiä viittauksia, mistä johtuen tiedon sisäisen eheyden ja muutosten johdonmukaisuuden varmentaminen jää edelleen ihmiselle.

Yllä mainittu sisällön jalostaminen muodostaa henkilö- ja tilannesidonnaisen mustan laatikon kaavaprosessin sisälle, tehden jo tuotetun työn reflektoinnista sekä sen sisällöllisestä varmentamisesta haastavaa. Tämä tarkoittaa tyypillisesti sitä, että keskellä kaavaprosessia aloittavien työntekijöiden on mahdotonta ilman intensiivistä perehdytystä ymmärtää työnaikaista aineistoa. Henkilöstön vaihtuminen kesken prosessin pakottaa joissain tapauksissa aloittamaan jo tehdyn työn uudelleen.

Julkisuuteen päätyneitä esimerkkejä kaavaprosessin aikana eteenpäin kopioituvista virheistä ja katoavista tiedoista on vain vähän, sillä sisältöä varmentamassa on lukuisia tahoja ja haastattelujen perusteella virheiden tunnistamisprosentti on korkea. Pohjimmainen ongelma ei kuitenkaan ole virheiden päätyminen virallisiin dokumentteihin, vaan tiedonhallinnan menetelmät, joiden seurauksena niitä syntyy.

Kaavaprosessin toiminnallisia ongelmia ei kuitenkaan voi ratkaista puuttumalla vain organisaatioiden sisäisiin käytäntöihin ja teknologiaan, sillä kaavoituksessa tarvittavaa tietoa tuotetaan lukuisissa rinnakkaisissa lupa-, sopimus-, hallinta-, seuranta- ja suunnitteluprosesseissa. Ulkopuolista tietoa ei usein ole saatavilla juuri prosessin tarvitsemalla hetkellä, saatavilla olevan tiedon varmuusaste ei ole riittävä kaavaa koskevien pysyvien ratkaisujen tekemiseksi, tai tieto on rakenteetonta ja sen jatkojalostus on hankalaa.

Kaavoituksen käytäntöjen moninaisuutta ylläpidetään myös osittain tarkoituksellisesti. Asemakaavat ovat väistämättä ajan ja paikan suhteen uniikkeja suunnitelmia, ja tämän riippuvuuden vuoksi prosessien yhtenäistämistä vastustetaan tyypillisesti tilannekohtaisen jouston tarpeeseen ja yhtenäistettyjen prosessien jäykkyyteen vedoten.

7. Nykyiset kaavojen valmisteluun käytettävät työkalut eivät tarjoa valmistelutyöhön prosessitukea.
8. Kaavoituksen työnaikainen tiedonhallinta on merkittävissä määrin henkilö- ja työkalusidonnaista.
9. Kaavaprosesseihin tuotavan rakenteettoman tiedon määrä muodostaa nykyisellään merkittävän haasteen työkalujen ja työnkulkujen uudistamiselle.
10. Kaavaprosesseissa on runsaasti riippuvuuksia muiden rinnakkaisten prosessien aikataulusta ja sisällöntuotannosta.

OTSO.dev
consult@otso.dev
+358 50 544 9174

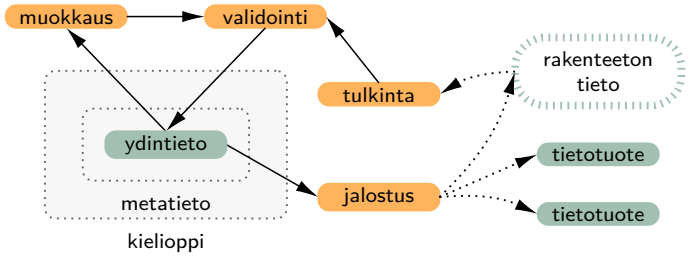
2 Käsitteistö

Edellä aihetta avattiin systeemisestä ja käytännönläheisestä näkökulmasta. Tässä luvussa siirrytään astetta teoreettisemmalle tasolle ja kuvataan selvityksen ydinkäsitteiden määritelmät, roolit ja keskinäiset riippuvuussuhteet.

2.1 Rakenteinen tieto

Kuten edellä todettiin, sekä rakenteisen tiedon puute että rajalliset kyvyt tuottaa ja hyödyntää sitä tehokkaasti ovat merkittäviä kaavoitusprosessien toimintaa rajoittavia tekijöitä. Jotta rakenteisen tiedon keskeinen rooli sekä sen käyttöön siirtymisen välttämättömyys voidaan perustella, luodaan aluksi lyhyt katsaus sen määritelmään ja käyttötapauksiin.

Yksinkertaistettuna rakenteisuudella tarkoitetaan digitaalista tietoa, jota voidaan tuottaa, muokata ja hyödyntää joutumatta tulkitsemaan sen sisältöä manuaalisesti ihmisvoimin (Witten, Bainbridge ja Nichols, 2010). Rakenteinen tieto koostuu itse varsinaisen sisällön (ydintiedon) lisäksi *metatiedosta*, joka kuvailee sen laatua, rakennetta, luotettavuutta, käyttötapaa, siihen liittyviä käyttöoikeuksia tai muita ominaisuuksia (Greenberg, 2005; ISO/IEC/IEEE, 2004).



Kuva 2: Yksinkertainen havainnekuva rakenteisen tiedon käyttämisestä

Joissain tapauksissa rakenteen määrittely voi olla hyvin selkeää: esimerkiksi yksinkertaisen lämpötila-anturin tapauksessa voidaan pitää jossain määrin kiistattomana, että tallennetun tiedon tulisi sisältää itse lämpötila-arvon lisäksi ainakin lämpötilayksikkö, kirjausaika sekä mittauspaikan sijainti ollakseen hyödyllinen. Kaavoituksen kaltaisessa monialaisessa ympäristössä vuorostaan on huomattavasti vaikeampaa ratkaista, missä laajuudessa hyödynnettyä ja tuotettua tietoa tulisi rakenteistaa ja miten se tulisi toteuttaa.

Kvantitatiivinen tieto on usein jo lähtökohtaisesti rakenteista, sillä mitattavaa tietoa tuottavien laitteiden ja järjestelmien ulosanti on vakioitu niiden käyttökohteen mukaisesti jo suunnitteluvaiheessa. Kvalitatiivisen tiedon rakenteistaminen taas vaatii joko manuaalista ihmisvoimin tehtävää työtä, heuristiikkaa tai esimerkiksi koneoppimisen menetelmiä.

Siinä missä yleisesti käytetyt (esimerkiksi juuri fysikaaliset) suureet on kehitetty lähtökohtaisesti mahdollisimman universaaleiksi, kaavoituksen kaltaisen erityisalan käsitteet ovat muotoutuneet pitkän ajan kuluessa muun yhteiskunnan kehityksen ohessa ja ovat täten paikkaan, aikaan, lakeihin ja käytäntöihin sidottuja. Tämä tarkoittaa käytännössä myös sitä, että määritelmät ja itse niiden pohjalla olevat käsitteet ja niiden keskinäiset

OTSO.dev
consult@otso.dev
+358 50 544 9174

suhteet ovat aina jossain määrin liukuvia, rajoiltaan epäselviä, tai ristiriitaisia. Ala voi toimia niiden varassa pitkäänkin nojaamalla vakiintuneisiin käytäntöihin, kokemusperäiseen asiantuntijuuteen sekä niiden varassa tehtyihin *ad-hoc*-ratkaisuihin.

Rakenteisen tiedon käyttöönotto ja hyödyntäminen tähtää kyseisen kaltaiseen toimintaympäristöön liittyvien epävarmuuksien minimointiin sekä toiminnan laadun ja tehokkuuden parantamiseen (West, 2011). Erityisesti tietotyötä tekevien – mutta kasvavassa määrin kaikkien – organisaatioiden toiminnan laadun ja yhteiskuntavastuun kannalta on olennaista selvittää rakenteisen tiedon hyödyntämisen mahdollisuudet tehostaa organisaatioiden resurssien käyttöä, prosessien läpinäkyvyyttä ja organisaatio-oppimista sekä vähentää toiminnan henkilöriippuvuuksia, kognitiivista kuormaa ja vinoumia.

Rakenteisen tiedon ei tarvitse olla avoimeen kuvauskieleen tai rakenteeseen pohjautuvaa, mutta tyypillisesti näin on, sillä saman avoimen standardin hyödyntäminen tekee tiedon teknisestä yhdistämisestä luontevaa. Lisäksi kaikki tietoa tuottavat toimijat pääsevät hyötymään helposti toistensa tuottamasta tiedosta. Rakenteisen tiedon tyypillisiä rajapintoja käsitellään tarkemmin kappaleessa 2.1.2.

2.1.1 Ontologiat

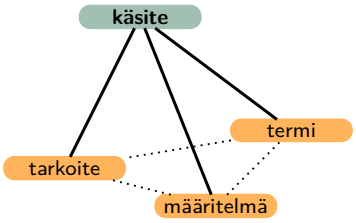
Alan siirtyessä tuottamaan ja hyödyntämään rakenteista tietoa suhteellisen yksinkertaisia aineistoja laajemmassa mittakaavassa täytyy rakenteen pohjaksi luoda *ontologia* eli määrittely käytettävistä käsitteistä ja niiden välisistä suhteista.

Kuten yllä mainittiin, organisaatiot ja yksilöt toimivat jo luonnostaan pitkälle sisäistettyjen implisittisten mallien varassa. Ontologioiden rakentaminen ja hyödyntäminen rakenteisen tiedon avulla pyrkii tekemään nämä mallit näkyviksi, ja sen seurauksena parantaa toiminnan laatua muun muassa seuraavin tavoin (Ontolog, 2012):

1. tekemällä eksplisiittisiä ja ymmärrettäviä (esimerkiksi tilastolliseen analyysiin tai inferenssiin perustuvia) sekä implisiittisiä mutta olennaisia oletuksia ja havaintoja järjestelmistä, joita käytämme ja joiden osana toimimme,
2. edesauttamalla järjestelmien ja tiedon vuorovaikutusta merkityssisällön (semanttisen) yhteentoimivuuden kautta,
3. parantamalla käytettävien mallien suunnittelua, mukauttamista ja uudelleenkäyttöä,
4. vähentämällä kehitys- ja operatiivisia kustannuksia,
5. parantamalla päätöksenteon järjestelmien (engl. *Decision Support Systems*) toimintaa,
6. auttamalla tiedon hallinnassa (engl. *knowledge management*) ja tuottamisessa (engl. *knowledge discovery*), sekä
7. auttamalla mukautuvampien järjestelmien kehittämisessä.

Ontologiatyö lähtee lähes aina alakohtaisista tarpeista, ja useat ontologiat ovatkin täysin itsenäisiä alan piiriin (engl. *domain*) rajoittuvia kokonaisuuksia. Itsenäisten ontologioiden käyttöä tukee se, että muiden alojen ja laajemman yleisön kanssa kommunikointi tapahtuu yleensä vain pienellä osalla tiedosta, jota ala itse hyödyntää ja tuottaa. Tyypillisesti viestinnässä on tarvetta yhdistää, yksinkertaistaa ja yleistajuistaa sisältöä vastaanottajalle. Rakenteisen ”raakatiedon” tarjoaminen palvelee vain rajattuja käyttötapauksia sen lisäksi, että siihen saattaa liittyä tietoturvariskejä.

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 3: Käsitteen suhde sitä määrittäviin tekijöihin

Yksikäsitteisyyden säilyttävän kommunikaatiotavan kehittäminen osapuolten välille on ylipäätään hyvin vaikeaa. Kaksi alaa voi esimerkiksi käyttää samasta termistä hyvin erilaisia määritelmiä, ja usein pelkkä määritelmän kuvaus ei itsessään riitä avaamaan, millaisesta käsitteestä on kyse. Tällöin tarvitaan alan sisäistä hiljaista tietoa kaikkien käsitteeseen liittyvien konnotaatioiden ymmärtämiseksi.

Rakenteisen tiedon määrän sekä tietojärjestelmiä yhdistävän infrastruktuurin kasvun myötä yhteentoimivuuden parantamiseen on alettu kiinnittää kasvavissa määrin huomiota. Siiloutumisen ja raja-aitojen muodostamien haasteiden ylittämiseen on kehitetty lukuisia nk. *ylätason ontologioita*. Esimerkiksi BFO (Basic Formal Ontology[6]), UFO (Unified Foundational Ontology[7]) sekä SUMO (Suggested Upper Merged Ontology[8]) tarjoavat yhtenäisen selkärangan, jonka varaan useat eri alat voivat rakentaa keskenään yhteensopivia ontologioita.

Ylätason ontologiat ovat tyypillisesti hyvin raskaita käyttää, sillä yleispätevyyden ja laajan kattavuuden saavuttamiseksi niissä joudutaan määrittelemään korkean abstraktiotason filosofisia käsitteitä, joiden varaan tarkemmat alakohtaiset käsitteet rakennetaan.

Yleisin este yhteisten ontologioiden käyttöönotolle laajassa mittakaavassa onkin niiden hyödyntämiseen tarvittavan määrittelytyön vaativuus. Esimerkkinä voidaan käyttää BFO-ontologian *spatial region* -luokan määritelmää:

> A *spatial region* is a *continuant entity* that is a **continuant_part_of** space_R as defined relative to some frame R.

Varmistuakseen siitä, onko esimerkiksi *maankäyttöalue* luokan *spatial region* aliluokka, täytyy perehtyä yllä olevan määritelmän kautta luokkaan *continuant entity*, sen yläluokkaan *entity*, sekä niille määriteltyihin lukuisiin relaatioihin ja koko ontologiaa koskeviin universaaleihin.

Alakohtaiset tietotarpeet ovat kuitenkin usein hyvin käytännönläheisiä, eikä ilmaisuvoimaisen laajan ontologian käyttöönotosta saada välittömiä operatiivisia hyötyjä. Hyödyt alkavat realisoitua vasta kun riittävästi tietoa on saatu tulkittua kyseisen ontologian varaan, ja tämä puolestaan vaatii usean organisaation pitkäjänteistä ja johdonmukaista sitoutumista toimintakulttuurien ja käytetyn teknologian uudistamiseen. Tulkinta on myös kallista: West (2011) arvioi yhtenäistämisessä tarvittavien rajapintojen kustannusten voivan olla jopa 25–70% verrattuna koko tiedonhallintajärjestelmän kehityskustannuksiin.

[6] `https://github.com/BFO-ontology/BFO`
[7] `https://ontouml.readthedocs.io/en/latest/intro/ufo.html`
[8] `https://github.com/ontologyportal/sumo`

OTSO.dev
consult@otso.dev
+358 50 544 9174

Usean itsenäisen toimijan omaehtoinen sitoutuminen samaan ylätason ontologiaan ilman yhtenäistä koordinoivaa kansallista tai ylikansallista elintä on hyvin epävarmaa niin kauan, kun ontologian koko ei ole ylittänyt selvästi toimijan omaa käyttöönoton kustannushyötyrajaa. Valtaosa käyttöönottoon asti päätyvistä laajoista ontologioista onkin esimerkiksi teollisuuden standardointielinten tai poliittisen ohjauksen tulosta.

Operatiivisten hyötyjen ja laajan yhteentoimivuuden tasapainottamisen seurauksena nämä ontologiat – esimerkkeinä Euroopan komission EIF (European Interoperability Framework[9]) ja Yhteentoimiva Suomi[10] -alustan ontologiat – asettuvat alakohtaisten ja universaalien ontologioiden välimaastoon.

Niiden laaja käyttöönotto tarkoittaa tyypillisesti pitkäaikaista ja laajaa sitoutumista yhteen monoliittiseen ja tiukasti rajattuun viitekehykseen. Käyttöönottoon liittyy tällöin merkittäviä systeemisiä riskejä, sillä operatiivisen ennakoitavuuden vuoksi ontologian täytyy olla käyttövalmis – toisin sanoen sisällöltään vakaa ja kattava. Mikäli ontologiassa havaitaan käyttöönoton jälkeen puutteita tai virheitä, on niiden korjaaminen hidasta ja vaikeaa, sillä kaikki ontologiaa hyödyntävät toimijat joutuvat tekemään päivityksen koordinoidusti. Lisäksi joudutaan sopimaan tavasta kuvata virheellisen version mukaan tuotettu tieto korjatun ontologian mukaiseksi.

Ylätason ontologioista poiketen näissä ontologioissa päädytään usein kuvaamaan pääosin asioiden nykytilaa operatiivisen puolen tarpeiden vuoksi. Ontologian joustamattomuus suhteessa ympäröivän maailman muutoksiin johtaa kuitenkin pitkällä aikavälillä niiden välisen käsitteellisen kuilun ja päivittämisen kustannusten kasvuun.

Myös ontologian määrittelemän käsiteavaruuden rajallisuus muodostaa omat riskinsä. Tallennettaessa reaalimaailman sisältöä ontologian mukaiseksi, yksinkertaistetaan (*redusoidaan*) se ontologian määrittelemään malliin. Mikäli malli osoittautuu myöhemmin riittämättömäksi eikä samaa tietoa ole enää mahdollista kerätä, on arvokasta sisältöä menetetty pysyvästi.

Yhtä lailla esimerkiksi vapaamuotoinen suunnittelusisältö joudutaan redusoimaan ontologian mukaiseksi kokonaisuudessaan sellaisissakin tapauksissa, joissa korrektia tai luontevaa käsiterakennetta ei ole olemassa. Tämä rajoite ruokkii kahdenlaisia haitallisia toimintamalleja. Ensinnäkin se ehdollistaa toimijoita ajattelemaan ratkottavia ongelmia ontologian rajojen puitteissa, vaikka se heikentäisi ratkaisujen laatua. Toisekseen se ohjaa kohti rinnakkaisen hiljaisen tiedon tuottamista sekä termistön väärinkäyttöä niitä vastaavien tarkoitteiden laajentuessa kattamaan vain toimijoiden ymmärtämiä konnotaatioita.

Riskien realisoituminen tarkoittaa käytännössä paluuta kappaleessa 1 kuvattuun toiminnan nykytilaan.

Näistä haasteista huolimatta ontologioiden käyttöön ei tule suhtautua yksioikoisesti asiana, joka ainoastaan luo enemmän ongelmia kuin ratkaisee. Hyödyntämistä ei toisaalta saa myöskään pitää itseisarvona olettaen, että toteutuksen tavasta riippumatta tulos on aina jossain määrin nettopositiivinen verrattuna käyttöönottoa edeltävään tilanteeseen. Onnistumisen kannalta sekä määrittelyä että toteutusta voidaankin pitää kriittisyydeltään ja painoarvoltaan yhtä tärkeinä vaiheina.

[9] `https://ec.europa.eu/isa2/eif_en`
[10] `https://yhteentoimiva.suomi.fi/fi/`

OTSO.dev
consult@otso.dev
+358 50 544 9174

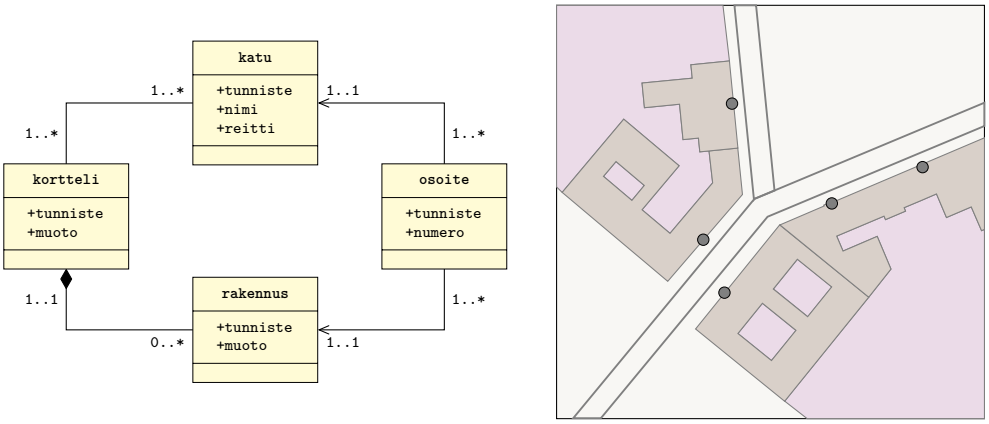
Seuraavaksi käydään läpi ontologioita hyödyntävän rakenteisen tiedon määrittelyyn käytettäviä menetelmiä, sen mahdollistavia järjestelmiä sekä niiden vaikutuksia toteutuksen laatuun ja käytettävyyteen.

11. Ontologisten mallien määrittelyssä täytyy ennakoida niiden käyttöönoton muodostamat sisäiset, toimijoiden väliset ja ajalliset rajoitteet.

2.1.2 Mallit ja skeemat

Kun käsitteellinen ontologinen malli formalisoidaan, syntyy *looginen malli*, jonka sisäinen johdonmukaisuus voidaan varmentaa. Loogisia malleja voidaan rakentaa lukuisin eri tavoin, mutta ylivoimaisesti suosituin on UML-standardi[11]. Se koostuu lukuisista eri kaaviotyypeistä, joista erityisesti *luokkakaaviota* käytetään hyvin yleisesti kaikenlaisten tieto- ja järjestelmärakenteiden kuvaamiseen. UML on kielenä hyvin monikäyttöinen ja soveltuu rakenteiden ja suhteiden lisäksi myös esimerkiksi toiminnan kuvaamiseen.

UML-kaavion peruselementtejä ovat *luokat*, jotka edustavat joitain reaalimaailman käsitettä tai ilmiötä yleisellä tasolla (esimerkiksi *katu*). Luokkien rajojen puitteissa esiintyvää vaihtelua edustavat attribuutit (esimerkiksi *nimi*). Luokista voidaan johtaa tietyillä attribuuttiarvoilla varustettuja *olioita* (*instansseja*), jotka edustavat jollain ajanhetkellä olemassa olevia yksilöitä. Yleensä useamman luokan välille on myös mielekästä määritellä niiden keskinäinen hierarkkinen tai muu suhde. Lisäksi luokkien määrittely voi sisältää kuvauksen niiden toiminnoista (luokan *metodit*).



Kuva 4: Vasemmalla on esitetty yksinkertainen UML-malli joka kuvaa neljä eri luokkaa, niiden attribuutit sekä väliset relaatiot. Oikealla on skeemaa hyödyntävä esimerkki kartan muodossa esitettynä.

Kuvan 4 esimerkissä[12] on kuvattu looginen malli, joka määrittelee mm. kaikkia kortteleita edustavan luokan `kortteli`, jonka attribuutteina ovat `tunniste` (uniikki nimi tai *avain*

[11] engl. *Unified Modelling Language*
[12] Tämä ja muut luvun esimerkit eivät edusta ns. parhaita mallinnusperiaatteita, vaan niiden ainoa tarkoitus on havainnollistaa kappaleessa kuvailtuja rakenteita ja käsitteitä.

OTSO.dev
consult@otso.dev
+358 50 544 9174

jolla jokin tietty kortteli voidaan erottaa kaikista muista kortteleista), sekä `muoto` (korttelin fyysinen muoto). Malli määrittelee esimerkiksi, että kaikki rakennukset ovat korttelien osia, ja että jokainen osoite on riippuvainen tietystä rakennuksesta ja kadusta (ei voi esiintyä ilman niitä). Jokainen karttakuvassa näkyvä rakennus, tontti, osoite ja katu on siis omaa luokkaansa edustava *olio*, joka noudattaa luokalle ja sen suhteille asetettuja rajoitteita.

Kun halutaan hyödyntää johonkin ontologiaan pohjautuvaa mallia konkreettiseen tiedon tallentamiseen ja käsittelyyn, puhutaan *fyysisen mallin* määrittelystä. Esimerkiksi tietokantojen tapauksessa puhutaan tällöin skeemasta, joka esittää loogisen mallin rajoitteita tietokannan ymmärtämään muotoon tulkittuna. Lisäksi se määrittelee, missä muodossa loogisen mallin tieto kantaan tallennetaan. Käsitteenä *skeema* on kuitenkin kontekstisidonnainen, ja esimerkiksi XML-muotoinen looginen malli on myös nimeltään skeema.

Käsitteiden tarkoilla määritelmillä ei kuitenkaan ole suurempaa merkitystä tämän selvityksen kannalta. Olennaisempaa on ymmärtää, mihin kysymyksiin eri tasoiset mallit vastaavat.

Määriteltäessä ontologiaa käsitetasolla on pohjimmiltaan vastattu kysymykseen: ”Millaisista asioista voidaan puhua?”. Loogisen mallin määrittelyssä on lisäksi vastattu kysymykseen: ”Miten kyseisten käsitteiden avulla voidaan tuottaa korrektia puhetta?” Skeeman toteutuksessa vastataan lopuksi kysymykseen: ”Kuinka puhetta tuotetaan ja tulkitaan tietyn teknologian avulla?”

Seuraavaksi tarkastellaan muutamia malleja, jotka edustavat erilaisia näkökulmia rakenteisen tiedon tallentamiseen ja käsittelyyn. Tarkoituksena on havainnollistaa, millaisia mahdollisuuksia erilaiset teknologiat tarjoavat ontologisten mallien toteutukseen, mutta myös niiden sisältämiä rajoitteita.

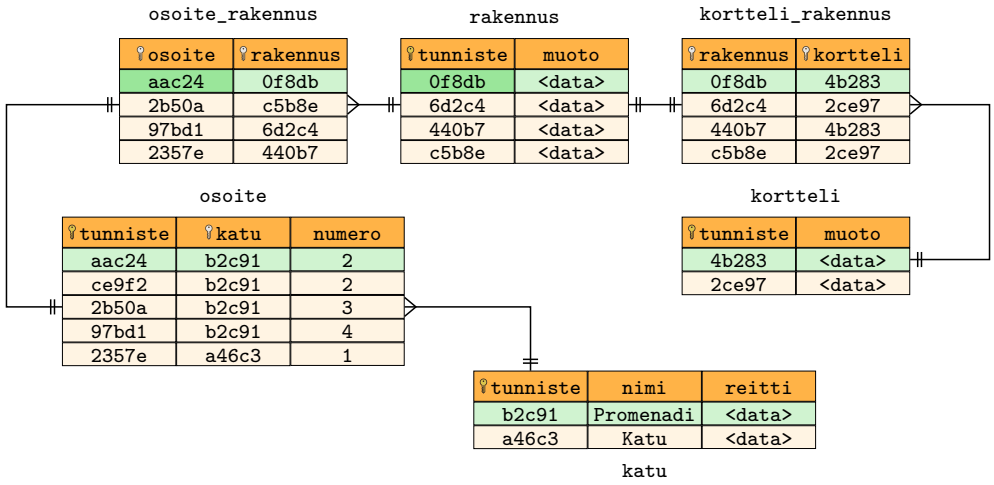
Relaatiomalli Yleinen vakiintunut tapa toteuttaa laajoja rakenteista tietoa käsitteleviä tietojärjestelmiä on hyödyntää *relaatiomalliin* pohjautuvaa tietokantaohjelmistoa (nk. SQL-tietokanta). Siinä skeeman rakenne on toteutettu yhteen tai useampaan tauluun (*relaatioon*), joiden sarakkeet määrittävät niihin tallennetun tiedon tyypin ja rivit edustavat yksittäisiin olioihin – esimerkiksi tiettyyn rakennukseen – liittyviä tietoja. Tiedon hajauttaminen useaan tauluun on mahdollista rivien välille luotavien viittausten avulla.

Relaatiotietokantoja hyödynnetään tyypillisesti suurten tietomassojen käsittelyyn, sillä niiden avulla vakiomuotoista tietoa voidaan säilöä ja käsitellä hyvin tehokkaasti. Esimerkiksi valtaosa paikkatietoalustojen tietokannoista on relaatiomuotoisia.

Kuvassa 5 on tulkittu kuvan 4 skeema sekä kartalla esitetty sisältö relaatiomuotoon. Aiemmin mainitut olioita yksilöivät tunnisteet löytyvät keltaisella avaimella (🔑) merkityistä sarakkeista, ja edeltävässä UML-kaaviossa esitetyt suhteet (viittaukset oliosta toiseen, esimerkiksi osoitteeseen sidottu rakennus) valkoisella avaimella (🔑) merkityistä.

Relaatiomuotoon toteuttaminen on vaatinut itse luokkien olioiden tallentamiseen tarvittavien taulujen (`osoite`, `katu`, `kortteli` ja `rakennus`) lisäksi tauluja, jotka edustavat olioiden välisiä suhteita: esimerkiksi taulussa `kortteli_rakennus` kuvataan jokaiseen kortteliin sisältyvät rakennukset.

OTSO.dev
consult@otso.dev
+358 50 544 9174



osoite	rakennus
aac24	0f8db
2b50a	c5b8e
97bd1	6d2c4
2357e	440b7

tunniste	muoto
0f8db	<data>
6d2c4	<data>
440b7	<data>
c5b8e	<data>

rakennus	kortteli
0f8db	4b283
6d2c4	2ce97
440b7	4b283
c5b8e	2ce97

tunniste	katu	numero
aac24	b2c91	2
ce9f2	b2c91	2
2b50a	b2c91	3
97bd1	b2c91	4
2357e	a46c3	1

tunniste	muoto
4b283	<data>
2ce97	<data>

tunniste	nimi	reitti
b2c91	Promenadi	<data>
a46c3	Katu	<data>

Kuva 5: Kuvan 4 skeema relaatiotietokannan tauluilla kuvattuna. Vihreällä on korostettu kaikki rivit, joiden tiedot yhdessä kuvaavat yhden olion (rakennus `0f8db`) tietoja ja sen suhdetta ympäristöönsä. Nuolien tehtävä on vain auttaa lukemaan taulujen välisiä viittauksia visuaalisesti. Itse tiedon voidaan ajatella olevan puhtaasti ”taulukkomuotoista”.

”Ylimääräiset” taulut johtuvat nk. *normalisaatioperiaatteista*, joiden tarkoituksena on minimoida relaatiokantaan tallennettavan tiedon toisteisuus eli samaa asiaa kuvaavan tiedon löytyminen useasta paikasta. Ilman normalisointia toteutettava relaatiokanta on yleensä nopeampi etenkin tietoa luettaessa, mutta toteutustapa johtaa toisteisuuteen, mikä taas altistaa sisällön epäjohdonmukaisuuksille – pahimmassa tapauksessa sille, että kannasta löytyy kaksi eri arvoa samalle asialle. Normalisoinnin käyttö kutenkin hidastaa tietokannasta lukemista erityisesti silloin, kun tietoa on runsaasti.

Esimerkin tarkoituksena on osoittaa, että relaatiokannan tapa tallentaa tietoa on monissa tapauksissa itse tiedon ”luontaisesta” rakenteesta poikkeava – ilmiötä kutsutaan nimellä olio-relaatioyhteensopimattomuus (engl. *impedance mismatch*).

Näistä haasteista huolimatta relaatiokantojen käyttöä puolustaa niiden kyky toimia nk. ACID-ehdon[13] täyttäminen. Yksinkertaistettuna kyseisen ehdon täyttäminen takaa sen, että usean henkilön käyttäessä ja muokatessa tietokannan tietoja yhtä aikaa, pystyy tietokanta jatkuvista muutoksista huolimatta antamaan aina sisäisesti johdonmukaisia vastauksia käyttäjille. ACID-ehto on yksi pääasiallinen syy siihen, että tiedon varmuutta ja vakautta painottavissa käyttötapauksissa toteutukseen valitaan usein relaatiotietokanta.

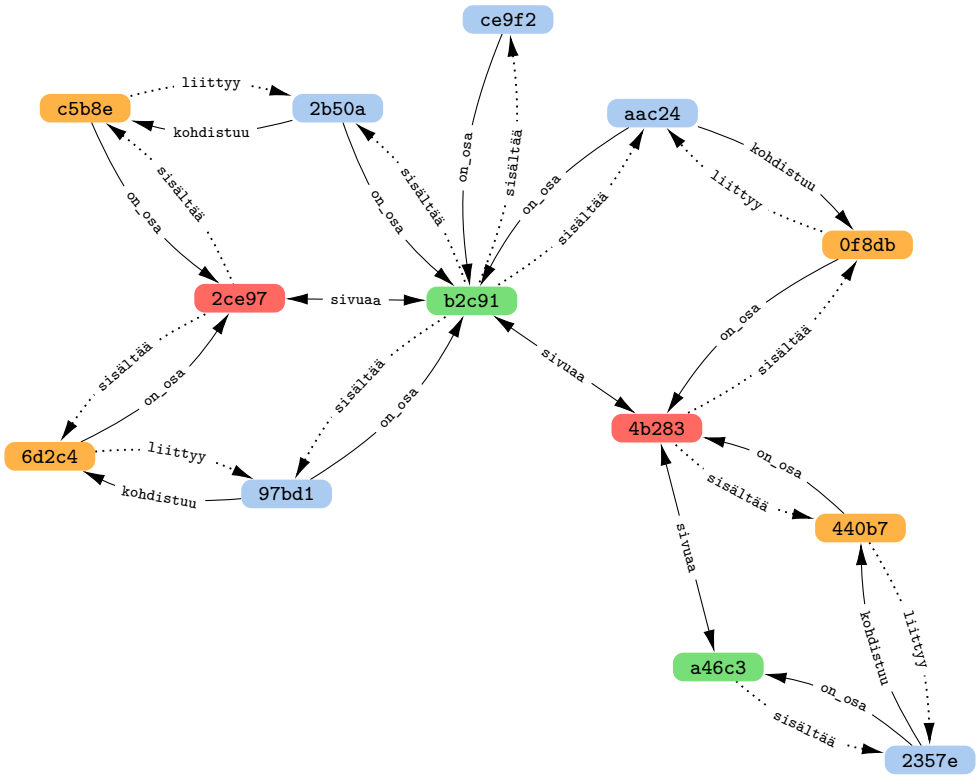
Kilpailevat mallit Rakenteisen tiedon hallintaan on relaatiomallin lisäksi saatavilla nykyään runsaasti kilpailevien nk. NoSQL-mallien varaan rakennettuja hyvin vaihtelevia tietokantaratkaisuja. Näitä ovat muun muassa:

- graafitietokannat,
- avain–arvo-tietokannat,
- dokumenttitietokannat sekä
- saraketietokannat

[13] engl. *Atomicity, Consistency, Isolation, Durability*

OTSO.dev
consult@otso.dev
+358 50 544 9174

Tarve näille malleille on syntynyt usein tilanteista, joissa relaatiomalli ei ole enää skaalautunut tehokkaasti yhä suuremmaksi kasvavien nopeasti muuttuvien ja sisällöltään moninaisten tietojen käsittelyyn. Monet NoSQL-toteutukset rakentuvat nk. BASE-ehdon[14] varaan – ne eivät takaa että tieto on sisäisesti johdonmukaista tai että se olisi jatkuvasti saatavilla kokonaisuudessaan. ACID-ehdon tiukoista vaatimuksista luopuminen on järkevää käyttötapauksissa, joissa tarve kyetä prosessoimaan esimerkiksi sata miljoonaa tietuetta hyvin lyhyessä ajassa on niin tärkeää, että käsittelyssä voidaan hyväksyä yksi tulkintavirhe tuhatta tietuetta kohti.



Kuva 6: Kuvien 4 ja 5 sisältö graafitietokantana. Yksittäisten olioiden attribuutit on jätetty esittämättä tilan säästämiseksi ja luokat on värikoodattu seuraavasti: ● `rakennus`, ● `kortteli`, ● `tie`, ● `osoite`. Kuvan 4 mukaiset relaatiot on esitetty tavallisilla nuolilla, potentiaaliset sisältöä rikkaammin kuvaavat relaatiot katkoviivanuolilla.

Monissa NoSQL-toteutuksissa ei ole mahdollisuutta ottaa kehitettyä ontologista mallia suoraan käyttöön, sillä niissä ei lähtökohtaisesti ole sisäänrakennettuna mekanismia tietomalliin liittyvien johdonmukaisuusehtojen ylläpitämiseen – mahdolliset rajoitteet toteutetaan ohjelmistotasolla tapauskohtaisesti. Näitä NoSQL-malleja ei käsitellä tässä selvityksessä.

Perheeseen kuuluu kuitenkin myös ACID-ehdon täyttäviä malleja, joiden kehittämisen pohjalla on ollut tarve käsitellä tietoa, joka ei taivu relaatiomallin käyttöön luontevasti. Näistä tarkasteluun on otettu maankäytön suunnittelun ja versionhallinnan kontekstissa mielenkiintoinen graafitietokanta.

Kuvassa 6 on esitetty edellä kuvatun UML-esimerkin sisältö graafitietokannan muodossa.

[14] engl. *Basically Available, Soft state, Eventual consistency*

OTSO.dev
consult@otso.dev
+358 50 544 9174

Vertaamalla graafin rakennetta kuvan 4 karttaan, voidaan nähdä, että graafi kuvaa hyvin alueelle sijoitettujen kohteiden topologisia suhteita: esimerkiksi korttelia kuvaavasta noodista nähdään helposti `sisältää`-relaation kautta siihen kuuluvat rakennukset.

Kuten kuvasta voi päätellä, graafitietokantaa ei ensisijaisesti harkita käytettäväksi, mikäli tarkoituksena on säilöä suuria lista- tai taulukkomuotoisia tietomääriä (tähän esimerkiksi relaatiomalli on luontevampi). Vastaavasti relaatiomallin käyttö muuttuu hyvin työlääksi, mikäli sillä halutaan käsitellä monimutkaisia topologioita.

Implisiittiset mallit Implisiittisillä malleilla tarkoitetaan tietoa joka on teknisessä mielessä rakenteista muttei sisältönsä osalta koneluettavaa.

Tyypillinen esimerkki implisiittisestä mallista on jonkin suunnitteluun käytettävän ohjelmiston tiedostoformaatti. Formaatin tehtävänä on säilöä tietoa tehokkaasti kyseisen ohjelmiston ehdoilla, ts. teknisesti helposti hyödynnettävässä muodossa. Tallennustehokkuuden maksimoimiseksi – usein myös immateriaalioikeuksien suojelemiseksi – tiedostot ovat *binäärimuotoisia*, eivät siis rakenteisen tiedon tapaan ihmisluettavia.

Tämän seurauksena tiedon rakenne on usein tiiviisti kytköksissä ohjelman sisäisiin tietorakenteisiin, eikä sen hyödynnettävyys muissa ohjelmistoissa ja käyttötapauksissa ole kovin korkea. Tyypillisesti tällaiset formaatit ovat myös määrittelyltään suljettuja: niiden rakennekuvausta ei ole avoimesti saatavilla, ja formaattia hallinnoi yksi kaupallinen taho.

Useissa ohjelmistoissa on mahdollisuus viedä (engl. *export*) tuotettua tietoa myös avoimien standardien mukaisiin formaatteihin. Operaatiossa kuitenkin menetetään usein osa tietosisällöstä, joko siksi, että avoin yleinen formaatti ei tue kaikkia ohjelman sisäisiä rakenteita tai siksi, että ohjelmistovalmistaja haluaa sitouttaa käyttäjät tuotteeseen estämällä tiedon helpon peilauksen muihin ohjelmistoihin tai järjestelmiin.

Rajapintamallit SQL- ja NoSQL-mallien tapauksessa tieto sijaitsee käytännössä aina tietokannassa, ja sen kanssa vuorovaikutetaan käyttötapauksia varten kehitetyillä ohjelmistoilla nk. rajapintakirjastoa (API, engl. *Application Programming Interface*) käyttäen.

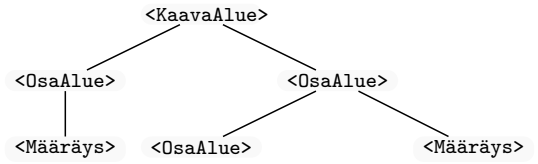
Kun tietokanta sijaitsee palvelimella, toteutetaan rajapinta yleensä ratkaisuilla (REST, SOAP, paikkatiedon tapauksessa WFS jne.), joiden kautta tietoon pääsee käsiksi ohjelmistojen lisäksi helposti myös esimerkiksi selaimella, ja joiden avulla tieto on helppo tallentaa johonkin yleisesti tarkoitukseen käytettyyn ihmisluettavaan formaattiin (CSV, JSON ja XML tai vastaava).

Erityisesti käyttötapauksissa, joissa tietokannalla on laajempi joukko käyttäjiä, on olennaista, että vastaanottajat kykenevät tulkitsemaan kannasta haetun tiedon rakenteen ja merkityksen. Kun tieto viedään kannasta ulos esimerkiksi XML-formaatissa, on itse tiedon rakenne arvoineen kuvattu kyseiseen tiedostoon, ja saadun tiedon eheys voidaan tarkistaa validoimalla se XSD-skeemaa vasten.

Tämän ansiosta kaksi osapuolta voivat vaihtaa tietoa yhteismitallisesti järjestelmiensä välillä ilman, että kummankaan osapuolen tarvitsee tietää, millainen järjestelmä toisella osapuolella on käytössä. Kahden järjestelmän yhteentoimivuuden mahdollistava abstraktiotasoa kuvataan yleensä käsitteellä "*loose coupling*".

OTSO.dev
consult@otso.dev
+358 50 544 9174

```
<KaavaAlue>
  <OsaAlue>
    <Määräys></Määräys>
  </OsaAlue>
  <OsaAlue>
    <Määräys></Määräys>
    <OsaAlue>
    </OsaAlue>
  </OsaAlue>
</KaavaAlue>
```



Kuva 7: XML-muotoisen tiedon serialisoitu esitys, sekä sitä vastaava puurakenne

Tässä kappaleessa tarkastelu keskitetään XML:ään, sillä se on yksi harvoista laajalti tuetuista formaateista, jotka tukevat monipuolista rakenteista sisältöä sekä laajaa validointia. XML kuten muutkin tässä mainitut tekstimuotoiset ihmisluettavat formaatit ovat serialisoituja; sisältö on tallennettu yksiulotteisena, eli kaikki sisältö sijaitsee yhdellä janalla tietyn etäisyyden päässä tiedoston alusta laskettuna. Rakenne on analogisesti vastaava yhteen litaniaan kirjoitetun tekstin kanssa (josta on jätetty pois tekstin ladontaan liittyvät rivinvaihdot, kappaleet ja jako sivuihin).

Samaan tapaan kuin erilaiset tietokantojen rakenteet suosivat tietyn tyyppistä tietoa toisen kustannuksella, myöskään rajapintaan käytettävä formaatti ei ole vinoumista vapaa. Kuvassa 7 on havainnollistettu XML-tiedoston sisällön lisäksi sen topologista rakennetta: formaatin rajoitteena on se, että kaikki tieto sijaitsee hierarkkisesti yhdestä elementistä haarautuvassa puussa (engl. *nested structure*).

Rakenteen rajoitteena on se, että elementti A voi sisältää lukuisia muita elementtejä, mutta näistä kukin kuuluu väistämättä vain A:han – elementillä ei siis voi olla useampaa ”vanhempaa”. Tätä rajoitetta voidaan jossain määrin kiertää attribuuttien avulla tehtävillä viittauksilla ja ylimääräisillä luokilla[15]. (Chaudhuri, Glace ja Wilson, 2006)

12. On olennaista tiedostaa, että sekä tiedon tallennukseen käytettävät järjestelmät (nk. *backend*) että rajapintaratkaisut sisältävät erinäisiä rajoitteita.
13. Vaikka ratkaisut määrittävätkin teknistä toteutusta, niiden ei tule vaikuttaa ontologioiden ja mallien sisältöön ja rakenteisiin.

2.2 Versionhallinta

Tässä kappaleessa käsitellään tiedon versionhallintaa (engl. *version control*) pintapuolisesti ja yksinkertaistettuna siinä määrin kuin myöhempien kappaleiden sisällön osalta on tarpeellista.

[15]Esimerkiksi korttelit ja niihin kuuluvat rakennukset voidaan kaikki esittää mallissa relaatioiden avulla rinnakkaisina itsenäisinä elementteinä, ja ”kuuluu kortteliin” -suhde esittää erillisillä elementeillä, joissa yksi attribuutti on varattu jonkin korttelin tunnisteelle ja toinen jonkin rakennuksen tunnisteelle. Rakennuksen kuulumista kortteliin ilmaistaisiin siis esimerkiksi näin: `<KuuluuKortteliin kortteli= "0f8dc"rakennus = "f2049"/>`.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Käsitteenä *versionhallinta* tarkoittaa yksinkertaistettuna jonkin itsenäisen tietokokonaisuuden (*aggregaatin*) sisältöön kohdistuvien muutosten tallentamista ja seurantaa. Aggregaatti voi olla esimerkiksi yksittäinen tiedosto, olio (tiettyä kohdetta kuvaava tietue) tai tietokanta. Versionhallinta voidaan toteuttaa tallentamalla joko vain aggregaatin muuttuneita osia (nk. *deltoja*) tai koko sisältö, jolloin tiedon toisteisuus (*redundanssi*) kasvaa.[16] Deltojen hyödyntäminen on haastavaa tai mahdotonta silloin, kun tieto on esimerkiksi ohjelmistojen omissa binääriformaateissa (kts. kappale 2.1.2).

Aggregaatteja voi olla seurannassa useita, esimerkiksi kun halutaan hallita jonkin suunnitteluprojektin hakemistorakennetta ja siihen tallennettuja tiedostoja. Muuttuneen sisällön lisäksi seurataan useimmiten, milloin muutos on tapahtunut ja kuka sen on tehnyt. Seurannalla tarkoitetaan kirjaimellisesti sitä, että olemassaolevan tiedon korvautuessa uudella se ei katoa vaan arkistotuu, ja näiden kahden tilan välille luodaan viittaus (uusi tieto ”osaa kertoa”, mitkä aiemmat tiedot se on korvannut). Nämä työkalut hyödyntävät pääosin aina deltoja, eikä niiden tarkoitus ei ole vain dokumentoida tehtyä työtä – vähintään yhtä tärkeää on niiden hyödyntäminen työn tehokkaaseen hajauttamiseen ja koordinointiin.

Versionhallintaa harjoitetaan vaihtelevassa laajuudessa myös kaikissa tietokantatoteutuksissa, jotka on suunniteltu seuraamaan yhteen tai useampaan olioon kohdistuvia muutoksia. Tietokannoissa versionhallintaa hyödynnetään tyypillisesti muutosten kirjausta myöhempää analyysia tai muuta käyttöä varten.

Yllä kuvattu jaottelu kuvastaa versionhallinnan mekanismien kontekstisidonnaisuutta. Ensiksi mainitussa priorisoidaan joustoa ja kykyä tulkita ja muovata tietoa tilannekohtaisesti muuttuvien tarpeiden pohjalta. Jälkimmäisessä priorisoidaan tiedon yhteismitallisuutta, yksikäsitteisyyttä ja vakautta. Tässä selvityksessä näistä kahdesta perspektiivistä käytetään käsitteitä *vapaamuotoinen* ja *rakenteinen* versionhallinta.

Näiden kohteiden ulkopuolella versionhallintaan törmätään nykyään varsin tavanomaisissa tietotyön tehtävissä, kuten tekstinkäsittelyssä ja tiedostonhallinnassa. Monet tekstinkäsittelyohjelmat tallentavat käyttäjästä riippumatta deltoja muokattavasta dokumentista, mahdollistaen esimerkiksi muutosten vertailun ja erheellisesti poistetun sisällön palauttamisen. Lisäksi työntekijöiden väliseen tiedonvaihtoon ja ryhmätyöhön tarkoitetut nk. Groupware- ja CMS-ohjelmistot säilyttävät tallennetuista tiedostoista tyypillisesti useita kokonaisia edeltäviä versioita.

Versionhallinnaksi lasketaan usein myös vaiheiden, päivämäärien, juoksevan numeroinnin mukaan luotujen hakemistojen tai tiedostojen nimiin lisättävien päätteiden (*suffiksien*) tapaiset manuaaliset tavat jäsentää tietoa. Niitä kuvaa kuitenkin paremmin laveampi käsite *versiointi*, sillä vaikka tiedosta luodaan lukuisia eri versioita, ei samalla synny vakaata ja virheiltä suojattua muutoshistoriaa, joka yksikäsitteisesti yhdistäisi versiot toisiinsa ja suojaisi vanhoja versioita muutoksilta (sisältö, nimi, aikaleima, sijainti).

Seuraavaksi käsittelemme ylläolevan jaottelun mukaisesti vapaamuotoisen ja rakenteisen tiedon versionhallinnan menetelmiä.

[16] Deltoja tallennettaessa ei haaskata tallennustilaa niiden osien säilyttämiseen jotka eivät ole muuttuneet. Redundanssia halutaan kuitenkin joissain tapauksissa kasvattaa sisällön ”varmuuskopioimiseksi”.

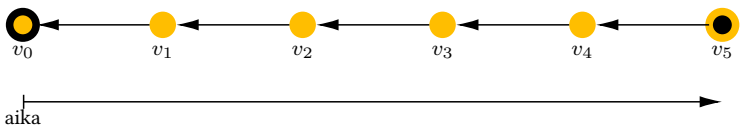
OTSO.dev
consult@otso.dev
+358 50 544 9174

2.2.1 Vapaamuotoinen versionhallinta

Tässä mallissa tyypillinen pienin aggregaatti on tiedosto, suurin taas kaikki tiedostot sisältävä hakemistopuu (esimerkiksi projektikansio sisältöineen). Käyttötapauksena on yleensä jonkin monia tiedostoja ja mahdollisesti hakemistoja sisältävän projektin (esimerkiksi ohjelmistotuote tai asemakaava) versionhallinta. Tiedostojen roolina on projektin sisällön luokittelu mielekkäisiin hallittaviin kokonaisuuksiin. Vapaamuotoisuudella tarkoitetaan tässä yhteydessä sitä, että versionhallintaan käytettävä työkalu ei itsessään lähtökohtaisesti ota kantaa tiedostojen sisältöön tai hakemistorakenteeseen.

Keskeisin versionhallinnan rakenne on *puu* (tässä tapauksessa suunnattu graafi[17], käytännössä analoginen suhteessa kappaleessa 2.1.2 kuvattuun graafiin), jossa solmut (seuraavien kappaleiden kuvissa ympyrät) esittävät *muutoksia* ja niitä yhdistävät kaaret (kuvissa nuolet) muutoksien välistä *järjestyssuhdetta*.

Kyseinen puurakenne yhdessä kaikkien aggregaattien jokaisen version kanssa muodostavat tietovarannon (engl. *repository*), ts. projektin koko muutoshistorian kaikkine sisältöineen.



Kuva 8: Yksinkertainen *mock-up*-esimerkki yhden aggregaatin sisällön lineaarisesta versionhallinnasta

Lineaarinen malli Yksinkertaisin tämän tyyppisen versionhallinnan malli on lineaarinen. Tyypillinen esimerkki sen toteutuksesta on esimerkiksi tekstinkäsittelyohjelma, joka tallentaa taustalla käyttäjän dokumenttiin tai taulukkoon tekemiä muokkauksia (*kumulatiivisesti*) kasvavaan muutoshistoriaan.

Kuvassa 8 on esitetty versionhallinnan puu, joka on syntynyt jonkin aggregaatin (v_0) työstämisestä. v_0 on yksikäsitteisesti puuhun ensimmäiseksi tallennettu versio, sillä siitä ei lähde kaarta mihinkään muuhun versioon. v_1 – kuten jokainen järjestyksessä sitä seuraava versio – on syntynyt sitä edeltävän version sisältöön tehtyjen muokkausten tallentamisesta. Versio v_5 on sekä ajallisesti että järjestyksessä yksikäsitteisesti viimeisin, sillä yhdenkään muun version kaari ei pääty siihen.

Tästä riippuvuussuhteesta seuraa, että puuhun tallentuva versiohistoria on pysyvä (engl. *immutable*), eikä sitä voi tahallisesti tai vahingossa vääristää ilman ketjun katkeamista. Pysyvyyttä voidaan havainnollistaa esimerkillä: jos henkilöt A ja B ryhtyvät kukin itsenäisesti muokkaamaan kuvan 8 versiota v_5 ja A ehtii tallentaa muutoksensa versioksi v_6, ei B enää voi suoraan tallentaa omaansa, sillä se ei pohjaudu nykyiseen puun uusimpaan (A:n tallentamaan) versioon v_6 vaan edeltävään versioon v_5. Jos suora tallentaminen sallittaisiin, B kirjoittaisi A:n version päälle tarkastamatta sen sisältöä. (Berliner, 1990)

[17] Graafi ei sisällä silmukoita (se on *asyklinen*): jokaisesta muutoksesta (solmusta) pääsee nuolten (kaarien) kautta tarkastelemaan vain yhtä tai useampaa järjestyksessä sitä edeltänyttä muutosta. Kaaria seuraamalla päädytään aina graafiin ensimmäiseksi tallennettuun versioon.

OTSO.dev
consult@otso.dev
+358 50 544 9174

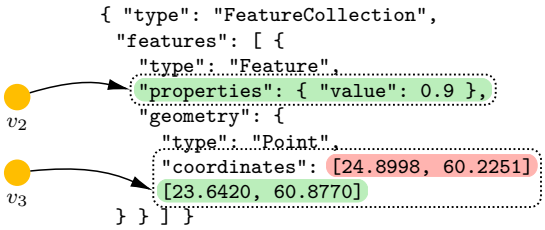
Yllä esitetty konfliktitilanne voidaan ratkaista joko optimistisella (nk. *copy-modify-merge*) tai pessimistisellä (nk. *lock-modify-unlock*) mallilla. Optimistisessa mallissa toimitaan yllä kuvatun mukaan: muokkaaminen on vapaata, ja mahdollinen ristiriita ratkaistaan vasta tallennusvaiheessa yhdistämällä käyttäjän tekemät muutokset tallennushetken uusimpaan versioon. Pessimistisessä mallissa muokattava aggregaatti lukitaan yksinoikeudella muokkaajan käyttöön aina tallentamiseen asti, jolloin konflikteja ei pääse alun perinkään syntymään[18].

Optimistinen malli soveltuu paremmin tilanteisiin joissa muutosten yhdistäminen on useimmiten triviaalia (voidaan useimmiten automatisoida), ja yhtäaikainen (asynkroninen) työskentely on tärkeää. Pessimistinen malli soveltuu tilanteisiin, joissa samaan aggregaattiin kohdistuvien muutosten yhdistäminen on joko mahdotonta tai sitä halutaan muista syistä välttää. Projektin sisällön hajauttaminen useampaan aggregaattiin pienentää todennäköisyyttä kahden käyttäjän muokkauksien törmäykselle, mutta liian pitkälle vietynä hankaloittaa itse projektin mielekästä hallintaa.

Aggregaatit ja muutos Ennen monimutkaisempien mallien käsittelyä on vielä syytä avata aggregaatteihin versionhallinnassa kohdistuvien muutosten luonnetta.

Alla aihetta on avattu kahden yksinkertaisen esimerkin kautta, joista ensimmäinen käsittelee aggregaattien sisältöä ja toinen niiden olemassaoloa.

Ensimmäisessä esimerkissä muokkauksen kohteena on aggregaatti, jonka sisältönä on GeoJSON-muotoon serialisoitua paikkatietoa (serialisoinnista puhuttiin alustavasti kappaleessa 2.1.2).



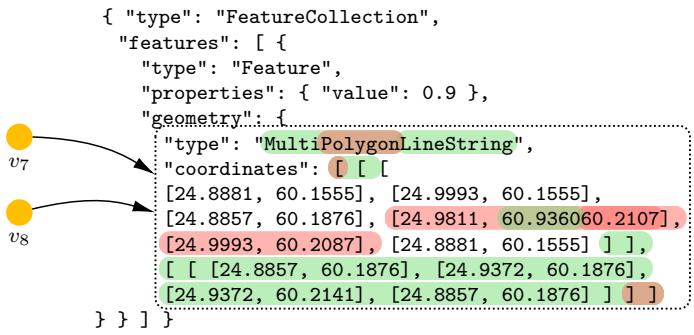
Kuva 9: Yksinkertainen paikkatietoa sisältävä GeoJSON-tiedosto, johon on tehty kaksi ei-risteävää muutosta. Vihreällä merkityt kohdat ovat lisättyä sisältöä, punaisella merkatut poistettua.

Ennen muokkausta v_2 kuvan 9 sisältö on kuvannut yhtä pistettä maan pinnalla koordinaateissa $60{,}2251°$N, $24{,}8998°$E. Kyseisessä muokkauksessa siihen on lisätty attribuutti nimellä *value* sekä arvolla $0{,}9$, ja lopuksi muokkauksessa v_3 pisteen sijainti on muutettu arvoon $60{,}8770°$N, $23{,}6420°$E. Koska muokkaukset eivät ole päällekäisiä (eivät risteä), ne voidaan haluttaessa toteuttaa optimistisella mallilla ja yhdistää automaattisesti.

Kuvassa 10 edellistä aggregaattia on muokattu versiosta v_3 versioon v_6 asti, jonka jälkeen siihen kohdistuu kaksi risteävää muutosta. Versio v_6 sisältää yhden monikulmio-objektin (`Polygon`), jonka toinen muutos on vaihtanut kahdesta monikulmiosta koostuvaksi (`MultiPolygon`), ja toinen on muuttamassa sitä murtoviivaksi (`LineString`).

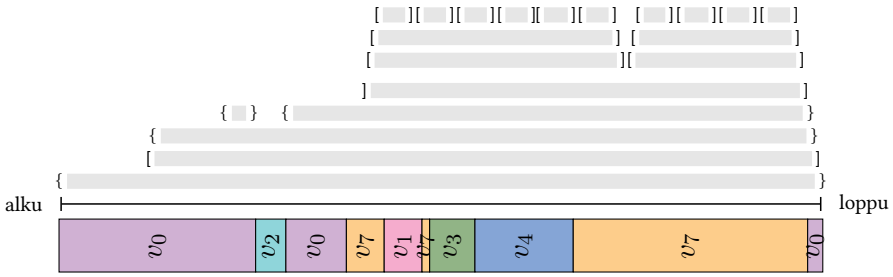
[18] Pessimistinen malli on käytössä myös esimerkiksi rakennusalan suunnitteluohjelmistoissa kuten Revit Architecturessa

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 10: Edeltävän kuvan tiedosto, mutta käyttäjien tekemät muokkaukset risteävät useammassa kohdassa.

Kyseisessä tilanteessa tarvitaan käyttäjän tekemää arviota siitä, millä tavoin v_8 muokkaus yhdistetään edeltävään, jotta tulos on sisällöltään tarkoituksenmukainen. Yhdistämistapoja ovat edellisen muutoksen sisällön ylikirjoitus, muutosten yhdistäminen tai viimeisimmän muutoksen hylkääminen.



Kuva 11: Kuvan 10 serialisoitu sisältö muokkauksen v_7 ajanhetkellä. Janan yläpuolella on esitetty aggregaatin sisältämän GeoJSON-tietorakenteen sisäkkäiset haka- ja aaltosulkeilla rajatut osat, alapuolella sen sisältöön kohdistuneiden muutosten rajat. Yksinkertaisuuden vuoksi sisältö on jätetty pois ja ainoastaan lohkojen rajat esitetään. Avain-arvo -parien muodostamat rajat on myös jätetty pois.

Kuvassa 11 on esitetty kuvan 10 sisältö version v_7 ajanhetkellä. Kyseisen version ”otos”sisällöstä on kaikista edeltävistä muutoksista koostettu aggregaatti. Koska missä tahansa muutoksessa on voitu lisätä tai poistaa mielivaltainen määrä merkkejä tiedostosta, ei versionhallinta pysty lähtökohtaisesti takaamaan että serialisoitu sisältö olisi jokaisen version kohdalla validi GeoJSON-tiedosto.

Tilanne voidaan kuitenkin korjata, sillä tallennushetken operaatioon voidaan liittää nk. *hook*, eli kutsu esimerkiksi ulkoiseen ohjelmaan tai ohjelmakirjastoon, joka suorittaa halutun toiminnon ennen muutoksen tallentamista. Esimerkiksi GeoJSON-syntaksin tarkistimen kytkeminen osaksi operaatiota tarjoaisi mahdollisuuden tarkastaa sisältö ja hylätä version tallentaminen, mikäli sisällön rakenne on viallinen.

Sisällön rakenteen validoinnin lisäksi osa vapaamuotoisen versionhallinnan työkaluista (esimerkiksi Git[19]) laskee automaattisesti muutoksen sisällöstä, muutoksen tallentaneen käyttäjän tiedoista ja muutoksen pohjalla olevasta versioketjusta nk. tiivistearvon (engl. *hash*).

[19] `https://git-scm.com/`

OTSO.dev
consult@otso.dev
+358 50 544 9174

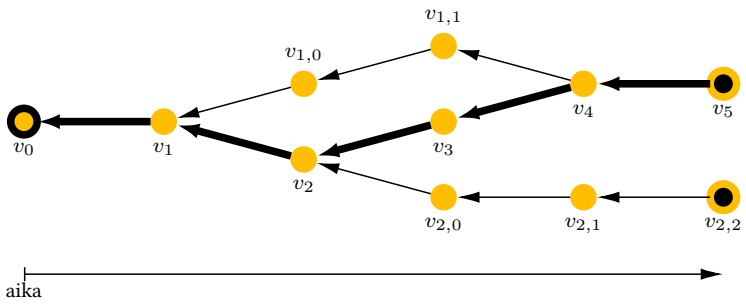
Tiivisteen ideana on laskea halutusta sisällöstä vakiomittainen luku, jonka arvo on aina identtinen samalle syötteelle ja (käytännössä) uniikki jokaiselle erilaiselle syötteelle. Tämän ansiosta sitä voidaan käyttää muutosten varmentamiseen, sillä pienikin muutos alkuperäisessä sisällössä tuottaa täysin poikkeavan tiivisteen arvon.

```
f3e63f6c769a26952fd97f3dafa310096cd7aec4
192d904c7c16179492154bedc817775ac1b3296c
```

Listaus 1: ”`kaavatietomalli`” -merkkijonosta laskettu SHA1-tiiviste, jonka alla ”`kaavatiet0malli`” -merkkijonosta laskettu vastaava

Tiivistettä voidaan käyttää kryptografisesti varmentamaan, ettei tieto ole muuttunut kahden osapuolen välissä siirtyessään, tai osana tietoon tehdyn muutoksen sähköistä allekirjoittamista.[20] Näistä suunnitelman muutosten validointia koskevista ominaisuuksista puhutaan lisää kappaleessa 2.3.

Haarautuva malli Kuva 12 esittää tilannetta, jossa uusien versioiden jalostaminen puun mistä tahansa versiosta (engl. *branching*) on sallittu.



Kuva 12: *Mock-up*-esimerkki haarautuvasta versionhallinnasta. Kaavio ei ota kantaa rinnakkaisissa haaroissa tehtyjen muutosten (esim. v_5 ja $v_{2,2}$) keskinäiseen järjestykseen. Yksi puun sisältä löytyvistä lineaarisista poluista (kuvaa 8 vastaava polku) on esitetty korostettuna.

Tärkeä huomio on, että jokainen muutoksia sisältävä polku vastaa itsessään kuvan 8 lineaarista tilannetta[21]. Olennaisin ero kuvan 12 tapauksessa on se, että vaikka jokaisessa yksittäisessä haarassa työskentely on lineaarisen mallin mukaisesti rajoitettua, eivät rinnakkaiset haarat millään tavoin rajoita toisiaan. Aiemman esimerkin henkilöistä A voi nyt jalostaa versiosta v_5 version v_6, ja samanaikaisesti henkilö B voi jalostaa versiosta $v_{2,2}$ version $v_{2,3}$ A:n muokkauksesta riippumatta, sillä kummallakin haaralla on nyt oma itsenäinen muutoshistoriansa.

Tämän ansiosta työstettävä kokonaisuus voidaan jakaa mielekkäisiin osiin, joita työstetään asynkronisesti. Edellä kuvattu tietojen automaattinen tai manuaalinen yhdistäminen tehdään vasta sitten, kun kahdessa haarassa tehty työ halutaan koota takaisin yhtenäiseksi kokonaisuudeksi. Haarojen hyödyntämiseen on olemassa lukuisia erilaisia strategioita,

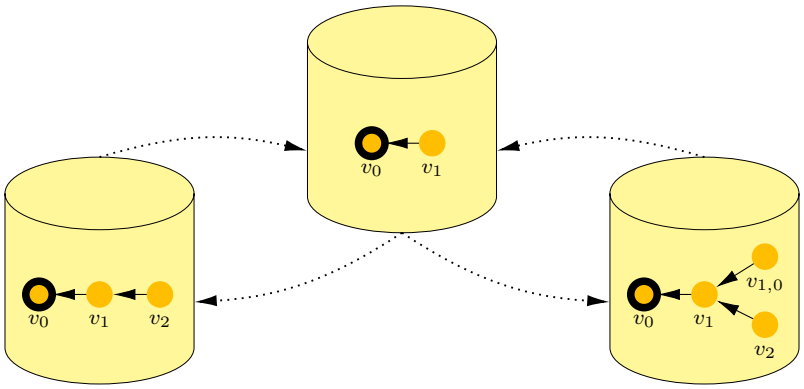
[20] Tiivisteiden laskemiseen ja allekirjoituksiin on saatavilla lukuisia laajalti käytettyjä ohjelmistokirjastoja ja työkaluja, kuten `shalsum`, libgcrypt (`https://gnupg.org/software/libgcrypt/`) tai Bouncy Castle (`https://bouncycastle.org/`).

[21] Valittaessa mikä tahansa versio (paitsi v_0) ja liikuttaessa nuolien suunnan mukaisesti versioon v_0, muodostuu kuvan 8 mukainen lineaarinen historia.

OTSO.dev
consult@otso.dev
+358 50 544 9174

joiden soveltuvuus riippuu työstettävän kokonaisuuden laadusta ja työtä tekevän ryhmän dynamiikasta ja koosta.

Hyvin yleinen strategia on käyttää yhtä haaraa koko projektin nk. *vakaiden muutosten* tallentamiseen (engl. *stable branch*). Ideana on jakaa projekti sisällön varmuusasteen ja toiminnallisuuden mukaan erillisiin haaroihin, joista tiedon jalostuessa sitä tuodaan hallitusti kohti vakaata haaraa. Näin voidaan varmistua siitä, että ainakin yksi haara sisältää aina vain sisällöltään johdonmukaista varmennettua tietoa.



Kuva 13: Esimerkki saman projektin työstämisestä useassa hajautetussa tietovarannossa

Keskitetty ja hajautettu malli Versionhallinta voidaan toteuttaa joko *keskitetysti* tai *hajautetusti*. Keskitetty malli on useimmiten intuitiivisin: siinä käytössä on yksi ja vain yksi auktorisoitu tietovaranto esimerkiksi työryhmän yhteisellä palvelimella. Sisällön parissa työskentelevät peilaavat haluamansa työstettävän osajoukon sieltä itselleen ja lähettävät lopuksi muutoksensa takaisin yhdistettäväksi.

Hajautetussa mallissa (kuva 13) peilataan työstettäväksi aina koko tietovaranto sisältöineen, jolloin jokainen käyttäjä työskentelee oman itsenäisen tietovarantonsa parissa. Hajautetussa mallissa työskentely on lähtökohtaisesti näkökulmasidonnaista: työstettävän kokonaisuuden laatu ja työhön osallistuvien yksilöiden ja organisaatioiden keskinäinen työnjako määrittävät esimerkiksi sen, kenen hallinnassa olevaa tietovarantoa pidetään auktoritatiivisena (mihin valmistuva sisältö lopulta kumuloituu).

Kuten sanasta *hajautus* voidaan päätellä, lisää tämä metodi tiedon toisteisuutta – tosin vain järjestelmän tasolla (kopioitavien tietovarantojen kautta), ei yksittäisten muutosten tasolla (jotka tallennetaan edelleen pääosin deltoina). Sille on kuitenkin hyvä peruste, sillä hajautuksen ansiosta järjestelmään ei muodostu keskitetyn versionhallinnan tapaan yhtä heikkoa lenkkiä, jonka vikaantuminen voisi muodostua katastrofaaliseksi. Vaikka paikallisesti tehdyt muutokset menetettäisiin esimerkiksi levyjärjestelmän vikaantumisen vuoksi, voidaan koko projektin peilausta edeltävä historia kopioida muilta osapuolilta ja työtä jatkaa kyseisestä pisteestä eteenpäin.

Hajautettu malli on käytännön sovelluksissa – erityisesti suurissa kehitysprojekteissa – osoittautunut erittäin tehokkaaksi tavaksi hallita tiedonjalostusta. (Chacon ja Straub, 2019) Yhteistä kaikille keskitetyn ja hajautetun versionhallinnan työnkuluille on se, että ne antavat työstettävästä sisällöstä kattavan laadullisen, organisatorisen ja ajallisen läpileikkauk-

OTSO.dev
consult@otso.dev
+358 50 544 9174

sen. Käytännöllisesti katsoen kaikki nykyinen ohjelmistotuotanto lepää versionhallinnan varassa, ja olisi virhe ajatella sen tarkoittavan, että versionhallintaan käytettävät työkalut soveltuvat vain lähdekoodin käsittelyyn. Esimerkiksi Potvin ja Levenberg (2016) tutkimuksessa Google hallinnoi noin miljardin tiedoston kokoista kehitettävien ohjelmistoprojektien tietovarantoa, josta lähdekoodin osuus on noin 9 miljoonaa, loppujen koostuessa asetustiedostoista, dokumentaatiosta ja työssä tarvittavasta datasta.

2.2.2 Rakenteinen versionhallinta

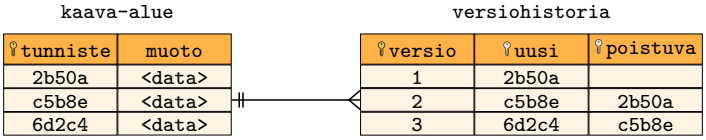
Vapaamuotoisessa versionhallinnassa ohjelmisto ylläpitää versiohistorian graafia ja siihen tallentuvien aggregaattien deltoja. Ohjelmisto pitää huolen siitä, että versiohistoria on aina topologisesti yhtenäinen ja suoritetut operaatiot (haarojen yhdistämiset, uusien versioiden luominen jne.) toteutuvat odotetusti. Ohjelmiston tehtävänä ei ole kuitenkaan määrittää tai kontrolloida, miten käyttäjä kyseistä puurakennetta ja sen sisälle automaattisesti syntyviä viittauksia hyödyntää.

Toteutettaessa versionhallintaa rakenteisesti, on toteutuksen pohjana tyypillisesti tietokanta. Tietokanta voi huolehtia skeeman ja ACID-periaatteen avulla, että muutokset säilyttävät kannan sisällön järkevänä, mutta tyypillisesti tietokannat eivät sisällä varsinaista tukea muutosten versionhallinnalle – etenkään hajautetusti.

Tyypillisesti tietokantaan tehdyt muutokset tallentuvat lokitietoihin, jolloin ”versiohistoriaa” voidaan kulkea taaksepäin toteuttamalla kyseiset muutokset käänteisesti ja käänteisessä järjestyksessä, kunnes tietokannan sisältö vastaa jotain tiettyä aiempaa ajanhetkeä. Menetelmä on kuitenkin raskas, rajoitettu ja tarkoitettu pääasiassa virheiden korjaamiseen.

Kun halutaan toteuttaa vapaamuotoista versionhallintaa imitoiva toiminnallisuus, joudutaan toteutus yleensä tekemään tapauskohtaisesti, sillä erilaisten käyttötarkoituksten, historiallisten syiden sekä rakenteellisten rajoitteiden vuoksi tietokantasovellukset eivät tavanomaisesti sisällä kattavaa versionhallinnan toiminnallisuutta.

Merkittävänä vedenjakajan voidaan siis pitää sitä, että versionhallinnan työkalut huolehtivat muutosketjun, tietokantasovellukset taas tiedon rakenteen johdonmukaisuudesta.

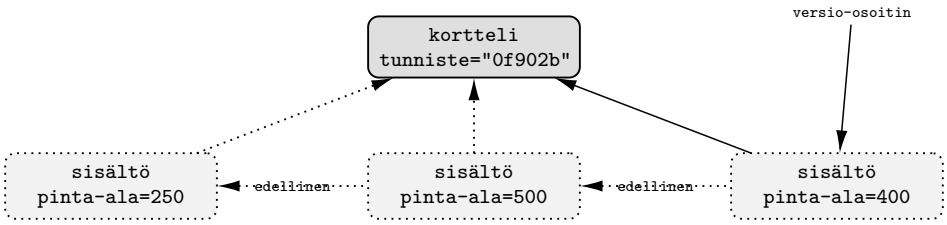


Kuva 14: Yksinkertainen skeeman avulla toteutettu yhden objektin versiohistorian seuranta relaatiotietokantaan.

Varsinaisen versionhallintaohjelmiston toiminnallisuuksien toteuttaminen tietokantaan olisi suuritöinen ja virheille altis projekti. Yksinkertaista versionhallinnan toiminnallisuutta voidaan toteuttaa esimerkiksi kuvan 14 mallin mukaisesti. Toteutus ei salli hajautusta eikä joustavia haarautumiseen pohjautuvia työnkulkuja.

Kuvassa 15 on esimerkki graafitietokantaan toteutetusta versionhallinnasta. Tässä mallissa haarautuva muutoshistoria on mahdollinen, mutta sen toteuttaminen vaatii lukuisia

OTSO.DEV
consult@otso.dev
+358 50 544 9174



Kuva 15: Graafitietokantaan viittauksilla toteutettu versiohistorian seuranta. Punainen noodi edustaa kohdetta sekä sen pysyvää tunnistetta, siihen viittaa useampi attribuuttinoodi jotka edustavat muutoksia. Yksittäinen viittausosoitin toimittaa kirjanpitäjän paikkaa ja osoittaa aina uusimpaan versioon.

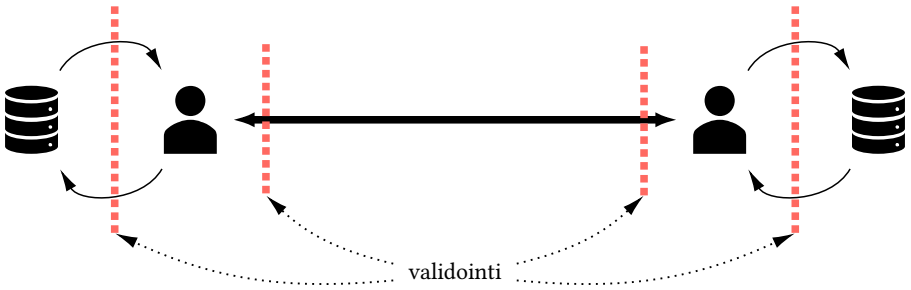
ylimääräisiä solmuja, polkuja ja viittausosoittimia ja niiden ylläpitoa, mikä on sekä virhealtista että työlästä.

14. Versionhallinta eroaa periaatteellisesti päivämäärien, nimien tai muiden johdannaisten tietojen avulla tehtävästä tiedon versioinnista.
15. Erilaiset käyttötapaukset (esimerkiksi projektityö vs. tiedon arkistointi) vaativat erilaisia versionhallinnan toteutuksia.

2.3 Validointi

Tässä kappaleessa käsitellään rakenteisen ja rakenteettoman tiedon validointia erilaisten tyyppien ja toimenpiteiden kautta. Termi ei itsessään ole eksakti, ja sillä on lukuisia käyttökohteesta riippuvia sivumerkityksiä. selvityksessä *validointia* käytetäänkin sateenvarjona tässä kappaleessa tarkemmin kuvattaville kohteille.

Siinä missä versionhallinta kytkee kaiken suunnitteluprosessin aikana käsitellyn tiedon yhteen, validoinnilla pyritään takaamaan että tietoon tehdyt muutokset ovat tavoitteellisen toiminnan päämäärän kannalta *varmoja* tai *oikeita*. Validoinnin konteksti määrittää sen, mitä varmuus tai oikeellisuus tarkoittaa ja miten sitä mitataan sekä hallitaan.



Kuva 16: Validointirajapinnat kahden osapuolen omassa työssä ja osapuolten välisessä tiedonvaihdossa

Validointi käsittää seuraavat yleiset aspektit, joita havainnollistetaan kuvassa 16:

1. Järjestelmästä tai toiselta osapuolelta vastaanotettua tietoa on mahdollista tulkita (tekninen validointi).

OTSO.dev
consult@otso.dev
+358 50 544 9174

2. Vastaanotettu tieto on rakenteeltaan ja sisällötään tietyt sovitut standardit täyttävää (tekninen validointi).
3. Tallennettava tieto on rakenteeltaan ja sisällötään tietyt sovitut standardit täyttävää (tekninen validointi).
4. Tiedon tuottanut taho on varmennettavissa (oikeudellinen validointi).
5. Tiedon väitetty lähettäjä on varmennettavissa (oikeudellinen validointi).
6. Voidaan varmistaa, että tieto on siirtynyt osapuolten välillä muuttumattomana (oikeudellinen validointi).
7. Vain tarkoituksenmukaisten tahojen on mahdollista saada sekä tulkita tietoa (oikeudellinen validointi).
8. Tallennettavan tiedon vastaanottavaan järjestelmään aiheuttamat muutokset voidaan rajata tarkoituksenmukaiselle tasolle (oikeudellinen validointi).

Yllä mainitut tarkistukset ovat edellytys tavoitteelliselle tiedonhallinnalle, mutta ne eivät sisällä varsinaista ohjausvaikutusta. Tätä ohjausvaikutusta käsitellään selvityksen puitteissa nimellä *laadullinen validointi*.

Kuva 16 esittää myös tärkeän yleisen työnkulkuihin liittyvän periaatteen: vaikka tieto olisi validoitu tallennettaessa tai lähetettäessä, on työnkulku tietoturvallinen ja luotettava ainoastaan, mikäli tieto validoidaan myös haettaessa tai vastaanotettaessa.

Lähtökohtaisesti käyttöön otettua tietoa ei koskaan tulisi kohdella automaattisesti validina vain siksi, että se tulee auktorisoidusta lähteestä – jokaisen työnkulkuun osallistuvan toimijan, järjestelmän ja työkalun tulee olettaa, että tietoa lähettänyt osapuoli on voinut tehdä virheen, joutua hyökkäyksen kohteeksi tai käsitellä tietoa vikaantuneella tietojärjestelmällä.

2.3.1 Tekninen validointi

Tekninen validointi kohdistuu kahteen vaiheeseen: tiedon tulkintaan ja tallennukseen.

Rakenteisen tiedon tapauksessa skeema – joko tietokantaan määriteltynä tai esimerkiksi erillisenä XSD-skeemana (kts. kappale 2.1.2) – huolehtii sekä tiedon luettavuudesta että tietomallin mukaisten rajojen noudattamisesta.

Skeemaa tukevista tietokantasovelluksista teknisen validoinnin toiminnallisuus löytyy sisäänrakennettuna; korruptoitunutta tai rakenteeltaan poikkeavaa tietoa ei siis yksinkertaisesti pysty viemään tietokantaan. Muissa ohjelmistoissa sekä monista ohjelmista koostuvissa työnkuluissa vastaava validointi voidaan toteuttaa joko rajapintakirjastojen tai erillisten ohjelmien (nk. *validaattorien*) avulla. Validoinnissa virheiden sijainti sekä tyyppi lähdeaineistossa pystytään tunnistamaan tarkasti, mikä tarjoaa mahdollisuuksia hallita sitä, miten niihin reagoidaan tallennus- tai tulkintaprosessin aikana.

Implisiittisen rakenteisen tiedon tapauksessa (kappale 2.1.2) usein ainoa keino varmentaa tiedon tekninen validiteetti on avata suljetun formaatin tiedosto kyseistä formaattia tukevalla ohjelmistolla tai ohjelmistovalmistajan tarjoamalla kirjastolla.

Puhtaasti rakenteettoman tiedon tapauksessa tekninen validointi ei ole käsitteenä mielekäs.

Teknisen validoinnin tärkeys korostuu tilanteissa, joissa tietoa keskitetään yhteen järjestelmään (esimerkiksi tietokantaan), josta sitä hyödyntää suuri joukko toimijoita. Puutteet validoinnissa johtavat järjestelmän ”saastumiseen” virheellisellä tiedolla, mikä voi pahimmillaan levitä myös muihin järjestelmiin.

Validointiin liittyy myös tietoturvakysymyksiä: mikäli rajapinnan tarkoituksena on vain sallia päivitetyn tiedon tallentaminen kohdejärjestelmään, sen tulisi sallia vain uutta vastaavien vanhojen kohteiden korvaaminen, ei muun järjestelmässä olevien tietojen muuttamista[22] (OWASP, 2019). Jossain tapauksissa myös muiden tietojen muokkaaminen saattaa olla aiheellista ylläpidollisista syistä johtuen, mutta tämä tulisi aina suorittaa erillisen, ei yleiseen käyttöön tarkoitetun rajapinnan kautta.

1. Sisällön luettavuuden tarkistaminen perustavalla tasolla: esimerkiksi XML-dokumentin täytyy sisältää kulmasulkeilla merkittyjä elementtejä (`<elementti>`) puuhierarkiaan järjestettyinä (kts. 2.1.2).
2. Sisällön korrektiuden tarkistaminen: tietojen tulisi vastata niille määritettyjä tyyppejä (esim. numeeriseksi määritelty arvo ei saa sisältää kirjaimia), ja rakenteita (esim. luokan sisällä saa olla vain määrätty määrä tiettyjä aliluokkia).
3. Sisällön johdonmukaisuuden tarkistaminen: tietojen tulee noudattaa skeemassa määriteltyjä rajoitteita (esim. korttelin pinta-ala ei saa olla kaava-aluetta suurempi, ja sen tulee sijaita kaava-alueen rajojen sisällä).

2.3.2 Oikeudellinen validointi

Oikeudellista validointia voidaan pitää osana teknistä validointia pääosin siitä syystä, että hyvin toteutettuna se on myös pitkälti automatisoitua.

Oikeudellisen validoinnin perustan muodostavat tunnistautumismenetelmät, kuten esimerkiksi Suomi.fi-alustan[23] kaltainen yhteinen SSO-tunnistautuminen.

Sen lisäksi, että tietoa vaihtavat osapuolet ja tietojärjestelmien käyttäjät voidaan tunnistaa luotettavasti, tarvitaan kaikkiin järjestelmiin myös tieto kyseisiin tunnuksiin liitetyistä kontekstisidonnaisista oikeuksista. Jotkin oikeudet ovat roolisidonnaisia (pysyviä), toiset taas tapauskohtaisia. Esimerkki tapauskohtaisesta oikeudesta on kaavaprosessin osalliselle sen ajaksi myönnetty lukuoikeus tiettyihin tietosisältöihin.

Oikeudelliseen validointiin kuuluu myös käyttöoikeuksien yhdistäminen niillä tehtyihin toimenpiteisiin – on olennaista, että henkilöyden varmistamisen lisäksi voidaan seurata, mitä tunnuksilla on kulloinkin tehty järjestelmissä.

Tyypillisesti tietojärjestelmien sisältämää tietoa pidetään lähtökohtaisesti luotettavana, sillä niihin tallennettu tieto on läpäissyt sekä teknisen että oikeudellisen validoinnin. Tietoon ja sen lähteeseen voidaan luottaa, kun sen kanssa kommunikoidaan suoraan rajapinnan kautta. Tilanteissa, joissa tietoa viedään rajapinnasta ulos esimerkiksi tiedostoihin tai muihin itsenäisiin aggregaatteihin, yhteys lähteeseen kuitenkin katkeaa.

[22] Sanitoinnilla (engl. *input sanitization*) tarkoitetaan tietoturvariskien ja mahdollisen ilkivallan minimointia erityisesti tilanteissa, joissa tallennusrajapinta tai tallennettava sisältö mahdollistavat käskyjen antamisen kohdejärjestelmässä (esim. SOAP tai SQL).

[23] `https://esuomi.fi/palveluntarjoajille/tunnistus/`

Tällöin sekä tiedon lähteestä että sen muuttumattomuudesta voidaan varmistua hyödyntämällä kappaleessa 2.2.1 kuvattua tiivistettä ja avaimia. Kun kaikilla tietoa vaihtavilla osapuolilla on toistensa julkiset avaimet, voidaan yksittäisten tietojen lähteen identiteetti ja tiedon aitous tarkistaa, vaikka yhteyttä itse tiedon lähteeseen ei olisi. Tiivisteet ovat olennaisessa osassa erityisesti muun kuin avoimen rakenteisen tiedon (kts. 2.1.2) aitouden tarkistamisessa.

Tietoon liitettyä digitaalista allekirjoitusta tulisi hyödyntää erityisesti silloin, kun jokin taho hyväksyy tai ottaa vastuun sisällöstä, esimerkiksi poliittisen päätöksenteon tai vastaavan auktorisointiprosessin kautta.

Osassa versionhallintaan käytettävistä ohjelmistoista (kts. 2.2.1) muutoksien aitouden varmistaminen on rakennettuna sisään. Esimerkiksi Git-ohjelmistossa jokainen muutos tunnistetaan tiivisteellä, jonka laskemiseen käytettävään sisältöön on upotettu edeltävien muutosten tiivisteet. Minkä tahansa version sisällön muokkaaminen johtaa siis siihen, että kaikkien sitä seuraavien muutosten tiivisteiden arvot muuttuvat – historiaan tehdyt muutokset voidaan siis tunnistaa helposti.

XML-formaatin kanssa voidaan erillisten tarkistusten sijaan hyödyntää niihin upotettavaa allekirjoitusta[24].

2.3.3 Laadullinen validointi

Kuten aiemmin mainittiin, yllä kuvatut validointitavat eivät luonteeltaan liity itse käsiteltävän tiedon sisältöön, ainoastaan sen käsittelyn korrektiuteen.

Laadullinen validointi sen sijaan on puhtaasti kvalitatiivinen toimenpide, jonka tehtävänä on varmistaa, että auktorisoitujen toimijoiden tekemät teknisesti korrektit muutokset tietoon ovat:

1. johdonmukaisia merkityssisällöltään, ja
2. vievät sisältöä kohti tavoitteissa määriteltyä tilaa.

Manuaalinen laadullinen validointi on kiinteä osa kaikkea suunnittelu- ja projektityötä, eikä sitä voida järkeistää (ja formalisoida) koskaan täysin. Käyttötarkoitukseen soveliaalla ja ilmaisuvoimaisella tietomallilla voidaan tällaista toimintaa kuitenkin tukea huomattavasti.

Laadullista validointia voidaan tukea muuttamalla tietosisältöä laajemmin rakenteiseen muotoon, jolloin:

1. siihen kohdistuvia ehtoja voidaan formalisoida ja siirtää näin teknisen validoinnin harteille,
2. validoinnin pohjana käytettävä tieto on yhteisesti hyödynnettävässä muodossa, ja
3. sen sisältöä voidaan tarkastella useasta eri näkökulmasta ennalta määriteltyjen näkymien kautta.

[24] XML Signature, kts. `https://www.w3.org/TR/xmldsig-core2/`

OTSO.dev
consult@otso.dev
+358 50 544 9174

Vaikka yleisenä käsityksenä on, että rakenteistaminen ei ole mahdollista laadulliselle kuvailevalle aineistolle, sitä voidaan kuitenkin tehdä joko formaalin ontologian puitteissa tai vapaamuotoisesti sisältöä annotoiden. Annotoinnilla tarkoitetaan sisällön merkitsemistä (engl. *tagging*) metatiedolla.

Formaalin ontologian puitteissa tehty annotointi on yksikäsitteinen ja tarkkarajainen, mutta väärin tehdyt tulkinnat merkinnän yhteydessä heikentävät aineiston laatua. Epämuodollinen annotointi käyttäjäkohtaisten vapaasti määriteltävien aihesanojen avulla (sosiaalisen median hashtag-tunnisteiden tapaan) ei puolestaan ole tarkkarajainen, mutta se tekee kuvailusta rikkaampaa laajemman sanaston ja sen tarjoamien nyanssien ansiosta.

Sisällön annotointiin käytetään tyypillisimmin semanttista webiä varten kehitettyjä standardeja (mm. OWL). Niiden etuna on laaja yhteensopivuus XML-kielen kanssa ja laaja tuki rajapinta- ja ohjelmistokirjastoilta.

Kuvailua voidaan tehdä tekstisisältöön lisättävän metatiedon lisäksi myös yleisesti monimuotoiselle aineistolle (multimedia): esimerkiksi kuviin ja videoihin voidaan kuvauspaikan ja päivämäärän lisäksi tallentaa niiden sisältämiä kohteita. Muodollisen ja epämuodollisen metatiedon yhteiskäytöstä on saatu lupaavia tuloksia jo aiemmin (Ciccarese et al., 2011; Braun, Schmidt ja Walter, 2007).

Manuaalisen annotoinnin ohessa teknologiset mahdollisuudet annotoida tietoa automaattisesti kasvavat kuitenkin jatkuvasti. Tällä hetkellä kuva- ja videomateriaalin sisällön luokittelussa ollaan niin pitkällä, että uutta keinotekoista materiaalia voidaan syntetisoida (luoda) hyvin vähäisen lähdeaineiston pohjalta (kts. esim. Wang et al., 2019[25]). Tekstin merkityssisällön ymmärtäminen on osoittautunut huomattavasti haastavammaksi tehtäväksi, mutta luonnollisten kielten käsittelyssä (engl. NLP: *Natural Language Processing*) on saavutettu viime vuosina merkittäviä harppauksia (kts. esim. GPT-2[26] ja BERT[27]).

16. Tekninen ja oikeudellinen validointi luovat olennaisen perustan kahden tai useamman osapuolen väliselle tiedonvaihdolle ja käsittelylle.
17. Täysin automaattista validointia voidaan tehdä vain tapauksissa joissa voidaan varmistua siitä että validointimenetelmä on täysin ennakoitava (deterministinen).
18. Laadullisen validoinnin tekniset menetelmät kehittyvät jatkuvasti, mikä tulisi ottaa huomioon toiminnan ohjauksessa ja suunnittelussa.

[25] `https://nvlabs.github.io/few-shot-vid2vid/`
[26] `https://openai.com/blog/better-language-models/`
[27] `https://github.com/google-research/bert`

OTSO.dev
consult@otso.dev
+358 50 544 9174

3 Maankäytön tiedonhallinnan vertailu

Tässä luvussa vertaillaan lyhyesti aiemmin määriteltyjä validoinnin ja versioinnin prosesseja empiiriseen tutkimusaineistoon.

Määritelmiä verrataan tietojenhallinnan toteutuksiin muutamassa kansainvälisessä kohteessa, tarkoituksena kartoittaa hyvien käytäntöjen sovellettavuutta Suomeen sekä oppia muiden maiden uudistuksissa kohtaamista haasteista.

Kansainvälinen vertailu on toteutettu haastattelututkimuksena. Otokseen on valittu maita, joissa on meneillään tai toteutettu tietomalliin pohjautuvia maankäytön suunnittelujärjestelmän uudistuksia. Haastateltaviksi on pyritty saamaan maanlaajuista suunnittelutietojärjestelmää hallinnoivia tahoja ja paikallisella tasolla (osavaltio, kunta tai kaupunki) sitä käyttäviä maankäytön suunnittelua toteuttavia tahoja. Kaikki tarkentavat haastattelut on toteutettu sähköpostin, puhelimen tai videoyhteyden avulla.

Tämän jälkeen kaavatietomalliin liittyvien uudistusten vaikutuksia tarkastellaan suhteessa keskeisiin suomalaisiin toimijoihin, tässä tapauksessa kaavoituksesta vastuussa oleviin kaupunkeihin ja kuntiin. Vaikka kaavojen valmistelua tekevät myös konsultit, rajattiin tarkastelu julkiseen puoleen, sillä valmiudet tiedon versionhallintaan ja validointiin ovat olennaisia sekä itse valmistelutyössä että konsulttiosapuolen ohjauksen ja laadunvarmistuksen osalta. Suomalaisten toimijoiden osalta on pyritty kartoittamaan tilannetta yleisluontoisesti, joten selvityksen resurssien puitteissa tutkimus on toteutettu lyhyenä kyselynä.

3.1 Vertailu kansainvälisellä tasolla

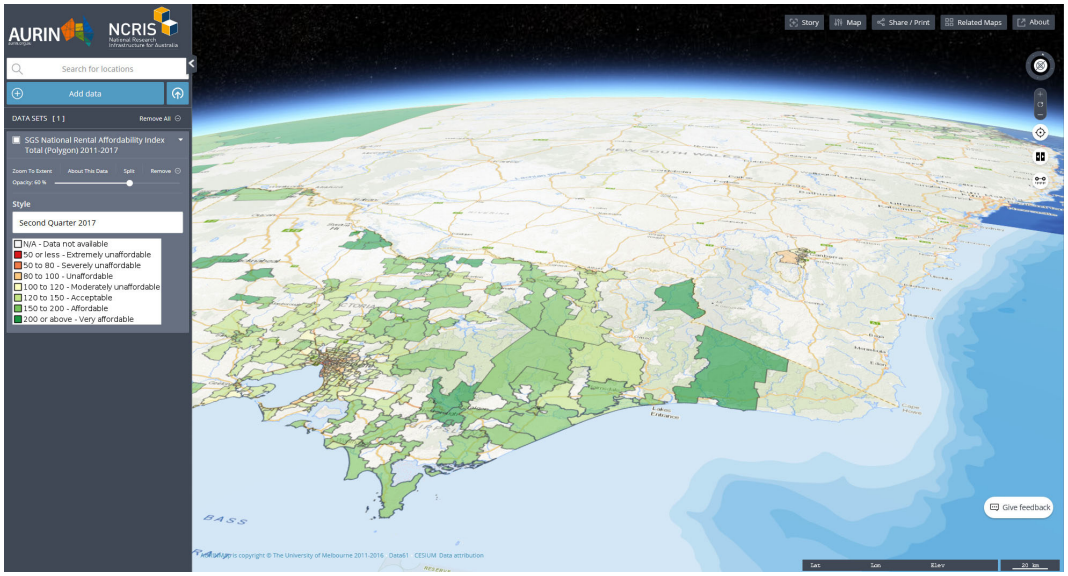
Kansainvälisen tason tarkasteluun valittiin alun perin Tanskan ja Norjan lisäksi väitöskirjatutkimuksen kontakteihin tukeutuen Hollanti ja Australia. Tarkasteluun tarvittavia kaavoitusta koskevia tietoja ei kuitenkaan saatu kahden jälkimmäisen maan tapauksessa käyttöön riittävän aikaisin, joten ne jouduttiin jättämään pois. Australiasta otettiin kuitenkin mukaan osittain Suomen paikkatietoalustaa vastaavan kansallisen AURIN-paikkatietojärjestelmän esittely.

3.1.1 Australia

Australiassa vuonna 2010 perustettu AURIN[28] on Australian liittovaltion hallituksen perustama toimielin, joka toimii opetusministeriön[29] alaisuudessa. Järjestelmää on rahoitettu perustamisesta lähtien noin 24 miljoonalla Australian dollarilla, ja vuotuisten 2 miljoonan toimintakulujen päälle on budjetoitu 2017–2022 välille ylimääräiset 7,4 miljoonaa dollaria tutkimuksellisten hyötyjen maksimoimiseksi.

[28] *Australian Urban Research Infrastructure Network*

[29] Rahoittava taho on *National Collaborative Research Infrastructure Strategy*, liittovaltion opetusministeriön strategisesti tärkeän tutkimuksen rahoitusverkosto.



Kuva 17: AURIN-järjestelmän julkinen käyttöliittymä

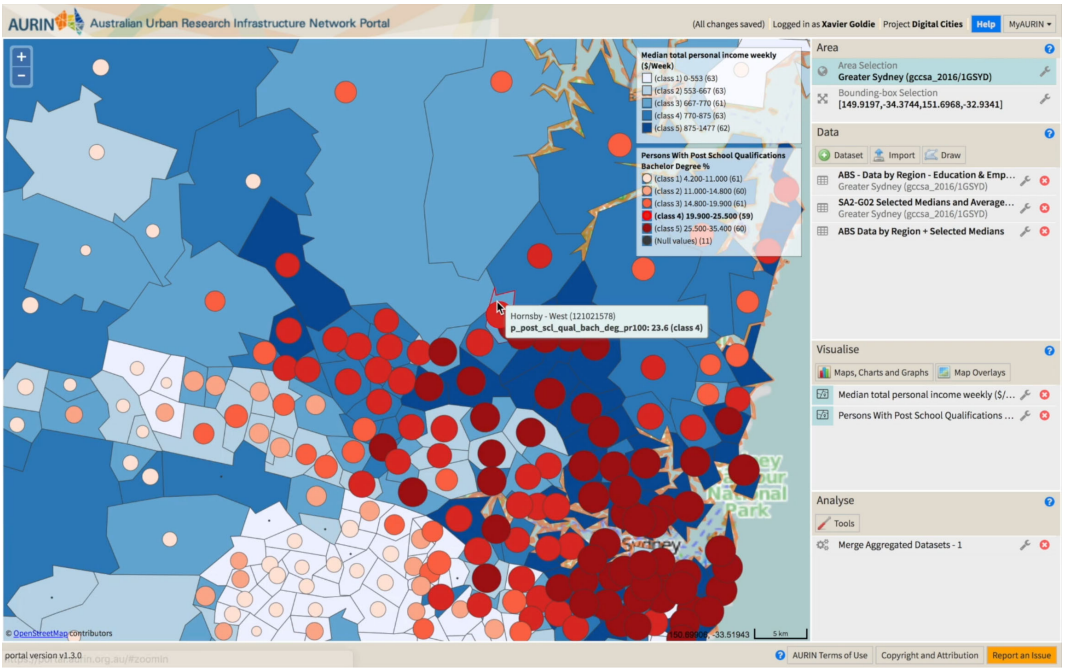
AURINin tarkoituksena on tuottaa tutkimuksen kautta tietoa, jota voidaan hyödyntää Australian kaupunkien kehittämiseen liittyvässä päätöksenteossa. Tämän selvityksen julkaisuhetkellä järjestelmässä on noin 15 000 käyttäjää ja 5 405 tietojoukkoa 112 toimittajalta. Tietojoukkojen aihepiiri on laaja ja sisältää mm. kiinteistö- ja asumistietojen lisäksi liikenteeseen, energiaan, vesihuoltoon, terveydenhuoltoon, sosioekonomisiin tekijöihin, äänestämiseen, demografiaan ja ympäristöön liittyviä tietoja. Rajattuun osaan tiedoista pääsee käsiksi mm. julkisen selainkäyttöliittymän kautta (kuva 17), kriittisemmäksi luokiteltuihin tietoihin pääsy on rajattu erilaisiin käyttöoikeusluokkiin mm. julkisen, yksityisen sekä tutkimussektorin toimijoille.

AURINissa merkillepantavaa ovat sen tutkimuksellinen painopiste sekä tiedon tuotantotapa. Järjestelmän ylläpitoa ja kehitystä ohjataan Melbournen yliopistolla, ja sen kehitystä ohjataan merkittävällä panostuksella tutkimukseen. Tieto ei ole puhtaasti julkishallinnon tuottamaa, vaan toimittajien joukko sisältää myös yksityisen sektorin toimijoita. Järjestelmä avustaa toimittajia tiedon laadunvarmistuksessa, metatietokuvauksissa sekä lisensoinnissa. Oletuksena aineistot pyritään saattamaan avoimiksi (Creative Commons 4.0 -lisenssillä[30]), mutta myös muita rajatumpia lisenssejä hyödynnetään tilannekohtaisesti.

AURIN tarjoaa aineistoja seuraavien käyttäjärajapintojen kautta:

- AURIN Map, avoin selainkäyttöliittymä rajatulla aineistolla,
- AURIN Portal, suljettu selainpohjainen analytiikka- ja datavisualisointityökalu käyttöoikeuden mukaan rajatuilla aineistoilla (kts. kuva 18),
- ENVISION: The Greyfield Precinct Iidentification E-Tool, monikriteerianalyysiin pohjautuva työkalu täydennysrakentamisalueiden tunnistamiseen rajatulla käyttöoikeudella,
- The Online WhatIf? PSS, avoimen lähdekoodin kaupunkisuunnittelutyökalu,
- AURIN API-ohjelmistorajapinta Python (sis. Jupyter) sekä R -ohjelmointikielille, sekä

[30] `https://creativecommons.org/licenses/?lang=fi`



Kuva 18: AURIN-järjestelmän suljettu selainpohjainen analytiikkatyökalu

- WFS ja WMS-rajapinnat GIS-ohjelmistoille,

AURIN ei tue Australian valtiollisen tason MyGov- tai Digital iD -SSO-tunnistautumistapoja, eikä myöskään osavaltiotasoisia sähköisen tunnistautumisen menetelmiä. Järjestelmällä on oma sisäinen käyttäjärekisterinsä, johon käyttöoikeuksia haetaan sähköisellä lupakaavakkeella ja myönnetään manuaalisesti.

Valitettavasti selvitystä tehdessä ei saatu käyttöön tarkempia ajantasaisia määrityksiä AURIN-järjestelmään toteutetusta versionhallinnasta ja validoinnista.

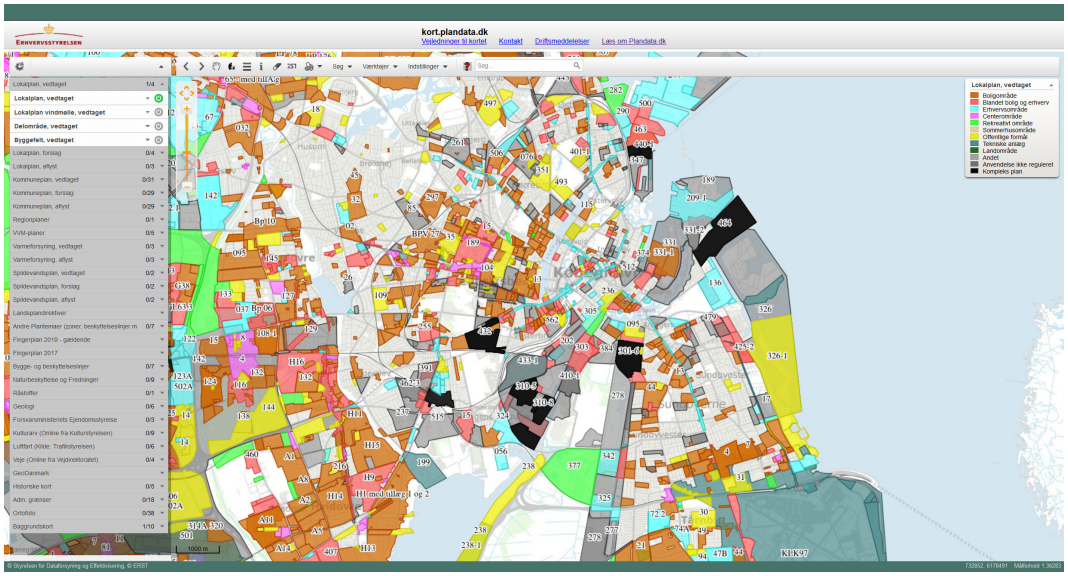
Merkittävä osa järjestelmän kehityksessä tuotetuista ohjelmistoista ja kirjastoista on julkaistu avoimena lähdekoodina[31].

3.1.2 Tanska

Tanskassa on veroviranomaisten ja elinkeinoviraston toimesta toteutettu kansallinen PlanData-alusta[32], jonne kuntien tuottamia asemakaavoja sekä muita maankäyttöä koskevia päätöstietoja tallennetaan keskitettyä hyödyntämistä sekä kuntien ja valtionhallinnon tason tiedonvaihdon parantamista varten. Merkillepantavaa sekä kaavaprosessien että -tietomallien osalta on Tanskan vahva painottuminen verkostomaiseen osapuolten väliseen ohjaukseen ja suunnittelun alisteinen suhde elinkeino- ja kasvutavoitteisiin (Lehtovuori et al., 2019), minkä voidaan olettaa vaikuttaneen myös tietomallikehityksen perspektiiviin.

[31] `https://github.com/AURIN`
[32] `http://kort.plandata.dk/`

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 19: Kansallisen PlanData-järjestelmän julkinen käyttöliittymä

PlanData-järjestelmä on ollut käytössä vuodesta 2018 ja sitä edeltänyt PlansystemDK-järjestelmä 2006–2018. Maankäyttöä koskevia tietorakenteita puolestaan on kehitetty jo vuodesta 2001. Skeemoja on käytössä useampia rinnakkain, nykyisellään vanhempi *PlanDK2*, sen päivitetty *PlanDK2+*-versio, sekä uudempi *PlanDK3*, ja ne palvelevat rajattuja sektoreita painottuen erityisesti verotuksen ja kiinteistörekisterin kaltaisiin tarpeisiin. Esimerkiksi asemakaavojen PlanDK2+-skeema ei kata koko asemakaavoituksen kentän ontologiaa ja tietotarpeita.

Tanskan kaavoitusta koskeva laki velvoittaa kunnat lähettämään hyväksytyt maankäyttösuunnitelmansa PlanData-järjestelmään. Itse järjestelmän skeemojen mukaan tallennettavalla rakenteiseella tiedolla ei esimerkiksi maankäytön suunnittelun osalta ole tällä hetkellä lainvoimaa, joten kaava-alueen rajauksen, muiden objektien ja niiden attribuuttien lisäksi tulee aina lähettää lainvoimainen PDF-muotoinen kaavadokumentti.

Tanskassa on tiedostettu että kyseinen ratkaisu ei ole pitkällä aikavälillä kestävä, ja tämän vuoksi meneillään on kehitystyö ”täysin digitaalisen” – toisin sanoen rakenteisen – kaavasisällön kehittämiseksi.[33] Kehitystyö on perua jo 2012 pilottiprojektina aloitetulle maankäytön suunnittelutiedon rakenteistamistyölle. Tanskan Ympäristöministeriön näkemyksen mukaan kaavoitusprosessit vaativat huomattavan paljon resursseja, joten toimintaa tulisi rationalisoida ja yhtenäistää. Lisäksi Aalborgin yliopiston 2010 teettämässä tutkimuksessa oli havaittu, että tuolloin voimassa olleissa asemakaavoissa oli kussakin enimmillään 7 ongelmallisena pidettävää määräystä, joiden sisältö oli joko lainvastaista, epätarkkaa, toisteista tai vaikeaselkoista. Tätä pidettiin kaavojen sisällön velvoittavuus huomioon ottaen merkittävänä riskinä. (Økonomistyrelsen, 2012)

Kehitystyön päämääränä Tanskassa on seuraavien etujen realisointi:

- parempi päätöstiedon käytettävyys ja saatavuus kansalaisille ja yrityksille yksityi-

[33] Tämänhetkisen kehitystyön tuloksia ei ollut selvityksen aikaan vielä saatavilla.

OTSO.dev
consult@otso.dev
+358 50 544 9174

seen käyttöön,

- kuntarajat ylittävien analyysien (esimerkiksi kaavataloudelliset seikat) tekemisen helpottuminen,
- yhtenäisen tietomallin mahdollistama prosessien suoraviivaistaminen ja tehostaminen, sekä
- kaavojen sisällöllisen laadun parantaminen ja validointi.

Tanskan käytössä olevassa järjestelmässä kehitystyön fokus on ollut lähes puhtaasti kansallisen puolen uudistamisessa ja helposti kvantifioitavan tiedon määrittelyssä. Selvityksen puitteissa käydyissä keskusteluissa kävi ilmi, että kuntien tasolla järjestelmän tuomat muutokset eivät varsinaisesti ole johtaneet niiden sisäisten suunnitteluprosessien ja tiedonhallinnan merkittäviin uudistuksiin. Kuntien tasolla merkittävimmät uudistukset ovat rajoittuneet toimiin, joilla tiedonsiirtoa kansallisen tason suuntaan voidaan tehostaa: kehitystä on esimerkiksi tehty sen eteen, että kerrosalan ja -korkeuden sekä rakennustehokkuuden kaltaisia tietoja voitaisiin raportoida järjestelmään vaivattomammin. Kunnat ylläpitävät PlanDatasta riippumattomia omia paikkatietopohjaisia järjestelmiään joissa suunnittelut- ja kaavatieto on säilytetty kansallista tasoa rikkaammassa muodossa.

Kunnan lähettäessä tietoa PlanDataan, lähetetään lainvoimaisen PDF-tiedoston lisäksi vaaditut rekisteritiedot joko XML-muodossa tai ShapeFile- tai MapInfo TAB -formaateissa, jotka ovat tyypillisiä GIS-ohjelmistojen tallennusmuotoja. Tietoa voidaan lähettää joko selainkäyttöliittymän tai SOAP-rajapinnan[34] kautta.

Lähetettävän tiedon oikeudellinen validointi on toteutettu keskitetysti kansallisen tason FBRS-järjestelmän[35] tunnistautumisen varaan, johon on liitetty kuntien edustajille myönnetyt kirjoitus- ja lukuoikeudet.

Lähetetyt XML- ja GIS-tiedostot validoidaan syntaktisesti tyypillisimpien ongelmien (risteävät alueet, virheelliset lukuarvot, korruptoitunut tieto) varalta. Järjestelmä tallentaa tehdyistä toimista lokitiedot ja raportoi toimenpiteiden onnistumisesta ja havaituista virheistä käyttäjille sähköpostin välityksellä. Muuta validointia tiedoille ei tehdä, sillä semanttisen sisällön oikeellisuuden katsotaan olevan kunnan vastuulla sekä sisällöntuotannon että hyväksynnän osalta, ja se tehdään Suomen hyväksymisprosessin tapaan täysin manuaalisesti.

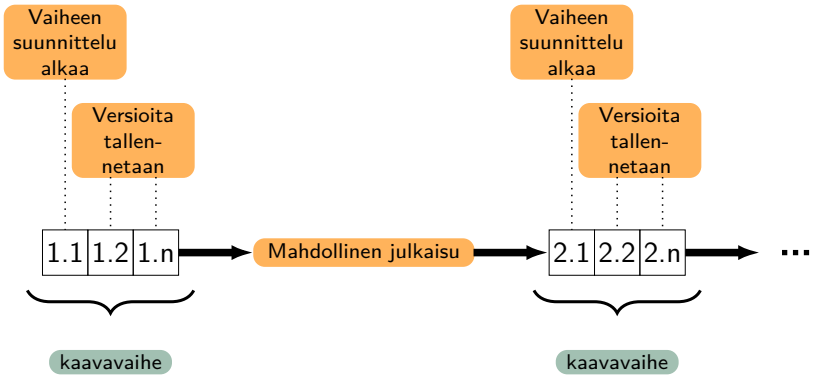
Kansallisen tason laadullisen validoinnin kehityksen puutetta voidaan tyypillisen järjestelmäkehityksen teknisen fokuksen ohella pitää osittain lainsäädännön seurauksena, sillä Tanskassa ”kaavan sisältöratkaisujen katsotaan olevan puhtaasti demokraattisesti valitun kunnallishallinnon asia”, ei kansallinen kysymys. Voidaan toisaalta myös todeta, että rakenteinen kvalitatiivinen sisältö ja tarkka muutoshistoria edesauttaisivat erityisesti kansallisella tasolla käsiteltäviä oikeudellisia kysymyksiä, kuten ”onko kunnallinen viranomainen noudattanut kaavalain säännöksiä, esimerkiksi menettelyn tai kaavahierarkian osalta, tai onko sillä ollut toimivalta tehdä kyseinen ratkaisu”. (Lehtovuori et al., 2019)

Kunnallisella tasolla kaavatiedon versionhallinta on toteutettu Suomen tapaan kuntien paikalliseen käyttöön hankkimilla CMS- ja Groupware-ohjelmistoilla – käytännöt siis vaih-

[34] engl. *Simple Object Access Protocol*, XML-muotoisia viestejä hyödyntävä protkolla, jota voidaan käyttää autentikointiin, tiedonsiirtoon, erilaisten prosessien käynnistämiseen jne.

[35] *FællesBrugerRettighedsSystemet*, suom. ”keskitetty käyttäjäoikeuksien hallintajärjestelmä”

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 20: PlanData-järjestelmän versionhallinta

televat paljon. Kansallisella tasolla sisällön versionhallinta on toteutettu kuvan 20 mukaisesti. Niissä kohdin prosessia, joissa kunta julkaisee kaavamateriaalia kommentoitavaksi tai hyväksyttäväksi (luonnos, ehdotus, hyväksyntä), se lähetetään järjestelmään validoitavaksi. Hyväksytyn validoinnin jälkeen meneillään olevan vaiheen alle luodaan uusi versio juoksevalla numeroinnilla ja uusin versio määritellään aktiiviseksi. Versiointia jatketaan vastaavalla menetelmällä myös jo hyväksyttyjen kaavojen kohdalla, mikäli aineistoon tulee täydennyksiä tai korjauksia.

Tanskan järjestelmä on erityisesti tietomallien osalta edustava läpileikkaus järjestelmäkehityksen muutoksesta 90-luvulta lähtien. Tietomallimäärittely on XSD-muotoinen ja jaettu 151 skeematiedostoon, joista kaikki kolme tietomalliversiota on koostettu. Skeemassa hyödynnetään sekä itse määriteltyjä että GML-tietotyyppejä.

Kappaleissa 1.3 sekä 2.1.1 kuvatut ongelmat näkyvät hyvin skeemoissa: edeltävien tietojärjestelmien rakenteita on kuvattu uusiin järjestelmiin sellaisinaan koordinoitujen laajojen uudistusten minimoimiseksi, mikä on johtanut järjestelmän teknisen velan kasvuun. Tehdyissä ratkaisuissa näkyvät rakennettavia järjestelmiä ja niillä käsiteltävää tietoa koskevien elinkaarimallien ja resilienssin puute.

```
<simpleType name="DateType">
  <annotation>
    <documentation>Dato YYYYMMDD mindato 18000101 maxdato 24991231</
    documentation>
  </annotation>
  <restriction base="integer">
    <pattern value="(((18|19|20|21|22|23|24)[0-9]{2})
    (((01|03|05|07|08|10|12)(0[1-9]|1[0-9]|2[0-9]|3[0-1]))
    |((04|06|09|11)(0[1-9]|1[0-9]|2[0-9]|30))|((02)
    (0[1-9]|1[0-9]|2[0-9]))))|00000000"/>
  </restriction>
</simpleType>
```

Listaus 2: Ote `PLANDK2_DateType.xsd`-skeemasta

Konkreettisen esimerkin edellä mainituista ongelmista antavat päivämäärien sekä versionhallinnan sovellusprofiilien toteutus asemakaavoissa. Päivämäärä (listaus 2), jota käytetään osana versionhallinnan tietoja on määritelty kokonaislukuna ja sille annetut rajat ovat melko summittaisesti määriteltyjä (vuoden 1800 alusta vuoden 2499 loppuun).

OTSO.dev
consult@otso.dev
+358 50 544 9174

Päivämäärää merkitsevän luvun oikeellisuus tarkistetaan tarpeettoman kömpelöllä ja pitkällä säännöllisellä lausekkeella (nk. *regex*), ja puuttuvaa päivämäärätietoa merkitään merkkijonotyyppisesti kahdeksalla nollalla, vaikka tietotyyppi on luku[36]. Päivämäärän määrittelyyn olisi ollut tarjolla jo vuonna 1988 julkaistu ISO 8601 -mukainen standardoitu merkkijonoesitys, jota tukee valtava määrä päivämäärien käsittelyyn suunniteltuja kirjastoja. Lisäksi semanttisesti oikeaoppinen tapa merkitä puuttuvaa päivämäärätietoa W3 XML Scheman mukaan olisi esimerkiksi `<DateType xsi:nil = "true"/>`[37].

```
<xs:simpleType name="VersionnrType">
    <xs:annotation>
        <xs:documentation>versionsnummer</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="9999999"/>
    </xs:restriction>
</xs:simpleType>
```

Listaus 3: Ote `PLANDK2_VersionnrType.xsd`-skeemasta

Listauksessa 3 on esitetty versionumeroinnin elementtityyppi. Myös siinä rajat ($0 \leq v \leq 9999999$) on valittu melko summittaisesti. Tarkoituksena lienee ollut vain varmistaa että versionumero on positiivinen luku.

```
<element name="SFORAKTUEL" type="pdk:BooleanType" minOccurs="0"/>
<element name="PLANID" type="pdk:PlanIdentifierType" minOccurs="1"/>
<element name="DATOATTR" type="pdk:DateType" minOccurs="0"/>
<element name="DATOGEOM" type="pdk:DateType" minOccurs="0"/>
<element name="DATOOPDT" type="dateTime" minOccurs="0"/>
<element name="VERSIONSNR" type="pdk:VersionnrType" minOccurs="0"/>
<element name="SUBVERSION" type="pdk:VersionnrType" minOccurs="0"/>
```

Listaus 4: Ote `PLANDK2_LokalPlanDelPlusType.xsd`-skeemasta

Rakenteisen kaavatiedoston juuressa on yksi gml:AbstractFeatureCollectionType-tyypillä laajennettu PlanType-elementti, joka voi sisältää esimerkiksi lukuisia LokalPlanPlusType- (kaava-alue) sekä LokalPlanDelPlusType (kaavan osa-alue) -elementtejä. Päivämääriä ja versioita käytetään vapaaehtoisena osana näiden elementtityyppien tietoja (kts. listaus 4)[38] suunnitelman tunnisteen sekä julkaisutilan ohessa. Pysyvää alueen tunnistetta edustaa PlanIdentifierType-tyypin elementti, jonka järjestelmä luo automaattisesti jokaiselle instanssille.

Tätä mallia ei voida kutsua versiohallituksi, sillä yksikään näillä elementeillä kuvattu objekti ei viittaa sitä edeltäneeseen ja versioiden välinen polku joudutaan päättelemään vain näiden elementtien arvojen kautta. Järjestelmän mukainen malli toimii sen oletuksen va-

[36] Ratkaisu näyttäisi viittaavan siihen, että päivämäärät on tallennettu järjestelmään alun perin merkkijonoina. Vuosi-kuukausi-päivä-formaatti kokonaislukuun talletettuna mahdollistaa helpon järjestysvertailun ($pvm_a > pvm_b$) mutta tekee muista vertailuista hyvin vaikeita toteuttaa.

[37] Oikeaoppinen metodi sisältää koneluettavassa muodossa tiedon siitä että arvo puuttuu, kun taas arvo 00000000 on koneluettavuuden näkökulmasta päivämäärä aivan kuten esimerkiksi 20191212. Puuttuvan arvon ilmoittaminen arvona vaatii tällöin erillisen teknisesti hauraan validointimetodin rakentamista järjestelmään.

[38] Elementtien nimet vastaavat suoraan tallennukseen käytettävän relaatiotietokannan taulujen sarakkeiden nimiä.

rassa, että esimerkiksi versio 1.2 edeltää versiota 1.3, mutta tämän tarkistamiseen joudutaan käyttämään esimerkiksi päivämääriä ($pvm_{1.2} \leq pvm_{1.3}$).

Versionumeroiden päivittämisestä, päivämäärien oikeellisuudesta ja näiden keskinäisestä johdonmukaisuudesta (esimerkiksi siitä että samassa versiossa muokatuilla osa-alueilla on identtiset versionumerot ja päiväykset) joudutaan huolehtimaan kuitenkin käsin. Kaikki versiointiin ja aikaan liittyvät elementit ovat vapaaehtoisia, joten tämä pakottaa myös linjaamaan, missä tilanteissa ne tulisi sisällyttää mukaan alueobjekteihin, ja minkä niistä tulee esiintyä samanaikaisesti.[39]

Osa koodilistojen virkaa toimittavista rakenteista on toteutettu yksinkertaisina lueteltuina tyyppeinä (nk. *enumeration*) jotka eivät sisällä sallittujen numeroarvojen lisäksi mitään kuvailevaa tai linkittävää tietoa, eivätkä täten varsinaisesti täytä koodilistan määritelmää.

Kos-
ka sekä PlanDK2+- että PlanDK3-skeemat koostetaan suurelta osin PlanDK2-tyypeistä, tässä kappaleessa kuvaillut ongelmat koskevat kaikkia versioita.

Kuten edellä mainittiin, PlanData hyväksyy myös ShapeFile- sekä MapInfo TAB -formaattien mukaista tietoa, joka muutetaan skeeman mukaiseksi validoinnin jälkeen. Molemmat ovat ohjelmistovalmistajien (Esri sekä MapInfo) omia suljettuja formaatteja, jotka ovat vuosikymmenien aikana saavuttaneet alalla *de facto*-standardin aseman (Bentleyn DGN ja Autodeskin DWG -formaattien tavoin). Molempia kuitenkin vaivaa formaatin iän mukanaan tuoma tekninen velka, josta irti pääseminen on haastavaa hyvin laajan ja kirjavan käyttäjäkunnan vuoksi.

Sen lisäksi, että molemmat formaatit sisältävät tiedon tallentamisen ja siirron kannalta hyvin rajoittavia historiallisia ratkaisuja[40], niitä hallinnoi yksi kaupallinen taho eikä formaatin tekninen spesifikaatio ole standardoitu saati avoin. Kyseisten formaattien tukeminen sekä ylläpitää organisaatioiden sitoutumista toimittajaloukkuihin että hidastaa avoimien rajapintojen ja standardien käyttöönottoa merkittävästi. Vertailuna voidaan todeta PlanDK-skeemojen olevan puutteineenkin huomattavasti joustavampi ja tehokkaampi tapa siirtää tietoa valtiollisen ja kunnallisen tason välillä, erityisesti siitä syystä, että skeema on avoin ja sen puutteita voidaan korjata hallitusti. PlanDK-skeemat eivät nykyisellään täytä INSPIRE-velvoitteita.

3.1.3 Norja

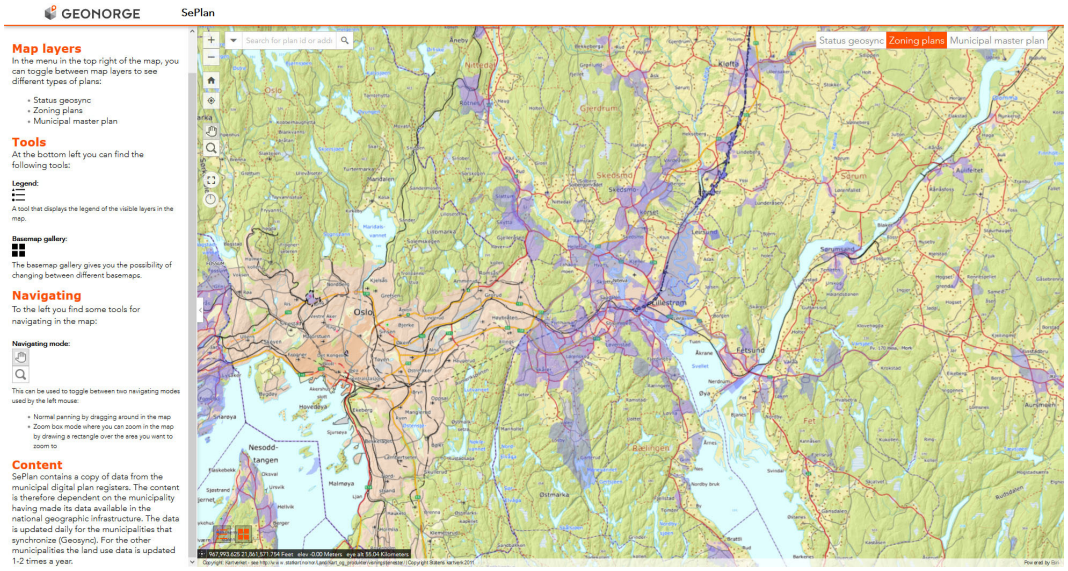
Norjassa on käytössä *sosi*[41] *Plan*-niminen kaavatietomalli, joka on osa laajempaa SOSI-tietomalli- ja rekisterijärjestelmää. Taustalla olevaa tietomallia on kehitetty pitkäjänteisesti kansallisella tasolla: ensimmäinen versio on julkaistu jo vuonna 1987; nykyisellään

[39] Vaikka toteutumista ei valvota skeeman avulla, voidaan implisiittisesti päätellä että esimerkiksi VERSIONSNR ja SUBVERSION-elementtien tulee aina esiintyä yhdessä muodostaakseen ”kokonaisen versionumeron”.

[40] Esimerkkeinä topologiatuen puute, attribuuttien rajoittaminen 10 merkkiin, 4000 tavun raja tietueen koolle, 250 tietueen ja 2 gigatavun kokorajoite per tiedosto jne.

[41] *Samordnet Opplegg for Stedfestet Informasjon*, suom. ”koordinoitu lähestymistapa spatiaaliseen tietoon”

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 21: Kansallisen SePlan-järjestelmän julkinen käyttöliittymä

käytössä ovat mallin neljäs ja viides versio. Kehityksen ohella vuodesta 1992 lähtien julkisen sektorin toimijoita on ohjattu yhteen mm. vapaaehtoisen *Geovekst*-ohjelman kautta motolla ”anna vähän, saa paljon”.

Tietomallin ja sen varaan rakennetun kansallisen rekisterin ja palvelujen koordinoinnista ja ylläpidosta vastaavat Kunta- ja uudistusministeriö[42] ja sen alaisuudessa toimiva kartoitusviranomainen[43].

SOSI-järjestelmän rekisteri kattaa sisällöltään huomattavasti maankäytön suunnittelua laajemman kentän, eikä sen määrittely rajoitu geospatiaaliseen tietoon. Kansallisesti harmonisoidun kiinteistörekisterin, geologisen ja hydrografisen tiedon, maankäytön suunnittelutiedon ja hallinnollisten tietojen lisäksi rekisterissä tarjolla on runsaasti laajaa temaattista tietoa.[44] Tiedon laajuuden, rajapintojen ja teknologisen tason osalta järjestelmä rinnastuu ennemmin AURINiin kuin PlanDataan.

Mielenkiintoisen siitä tekee se, että järjestelmän tarjoama yhtenäinen rajapinta on oikeastaan vain näkymä kuntien tuottamaan tietoon – ei siis tietovaranto itsessään (Ramboll Finland Oy ja Ubigu Oy, 2018). Kun alla puhutaan ”järjestelmästä” – tarkoitetaan siis varsinaista jokaisesta kunnasta löytyvää SOSI-tietoa säilövää järjestelmää.

Järjestelmä on alusta alkaen suunniteltu laajennettavaksi, jotta sitä voidaan hyödyntää myös esimerkiksi tilastotiedon tallennukseen ja siirtämiseen. Dokumentaation, spesifikaatioiden ja toteutuksen osalta voidaan nähdä, että kehitystyö on ollut pitkäjänteistä, kauaskatsovaa ja AURINin tapaan hyvin resursoitua. Kehityksessä on paneuduttu johdonmukaisesti standardointiin: UML-mallien ylläpito ja kehitys on versiohallittu Subversion-työkalulla (kts. kappale 2.2.1), järjestelmässä ylläpidetään laajaa 57 aihealueen taksonomiaa (General Feature Catalog, ISO 19110) ja metatiedot on kuvattu ISO 19115 mukaan.

[42] KMD; *Kommunal- og moderniseringsdepartementet*
[43] *Statens kartverk*
[44] XSD-skeemoja löytyy 152 kategorialle eri viranomaisten rekisteritarpeisiin.

OTSO.dev
consult@otso.dev
+358 50 544 9174

```
1 .KURVE 382:
2 ..OBJTYPE Hjelpekurve
3 ..HØYDE 50.00
4 ..OPPDATERINGSDATO 19930101
5 ..KVALITET 64 5000
6 ..MEDIUM T
7 ..NØ
```

Listaus 5: Ote sosi dot dot -muotoisesta paikkatiedosta

Tiedon tallentamiseen on valittu aikanaan nk. *Backus–Naur* (bnf) -muotoinen 1980-luvun alkupuolella kehitetty kieli (”sosi dot dot”; Bie ja Stormark, 1980), mikä tekee ratkaisusta sekä hyvin uniikin että ymmärrettävän, sillä esimerkiksi nykyisellään laajalti käytetty xsd-kieli xml-skeemojen kuvailuun tuli saataville standardina vasta vuonna 2001. Merkittävän tästä valinnasta tekee se, että kyseisen aikakauden konventioiden vastaisesti kielellä tallennettu tieto on ollut ”rakenteista” (kts. listaus 5) ja täten suhteellisen helposti siirrettävissä ja tulkittavissa muuhun muotoon myöhempiä mahdollisia tarpeita varten.

Vaikka kaikki tieto on järjestelmässä sisäisesti edelleen dot dot -formaatissa, on tarjolla mm. Geoserver, FME, wms, https, wfs, wmts ja rest -rajapinnat, joiden kautta tietoa voidaan vaihtaa dot dot -formaatin lisäksi owl-muotoisena semanttisena sisältönä, Excel-formaatissa, sekä gml- ja xml-skeemojen mukaisena. Autentikointi tiedonsiirtoa varten tapahtuu kansallisen sso[45]-rajapinnan kautta.

Oman kansallisen formaatin (sekä tulevaisuudessa myös xml/gml-muotoisen sisällön) validointiin on tarjolla ilmainen sosi-Control-validaattori sekä sosi-Vis-tarkasteluohjelma. Mallit, niiden tyypitykset, luokat, koodilistat ja validointiehdot on hyvin dokumentoitu järjestelmän jokaiselle versiolle Geonorge-palvelussa[46].

Tekninen validointi ei nojaa puhtaasti xsd-skeemoihin, sillä monessa kohtaa dokumentaatiota kerrotaan elementtien sisältöä tai attribuuttien arvoja koskevista rajoitteista, jotka eivät kuitenkaan sisälly itse skeeman määrittelyyn. Validointi ulottuu sallittujen elementtien tyypeistä ja määristä sekä keskinäisestä hierarkiasta yleisimpiin automaattitarkistuksiin; numeerisista arvoista varmistetaan että sisältönä on numeroita, koodilistoista taas että sallittu arvo löytyy listalta. Suurin osa tarkistuksista tapahtuu kuitenkin rekisteriin (tietokantaan) tietoja tallennettaessa: tällöin arvot tarkistetaan sosi:n sisäisen vastaavan tyypin (SOSI_navn) rajoitteiden kautta.

Norjan mallissa asemakaava rakentuu ReguleringsPlan-luokan varaan, jolla on riippuvuus yleisten jaettujen ominaisuuksien Rp Felles-luokkaan, sekä tilanteesta riippuen joko vuoden 1985 tai 2008 lain mukaiseen Rp PBL1985 tai Rp PBL-luokkaan.

Kaikki suunnitelmat toteutetaan yhteisen Plan-sovellusskeeman kautta. Se sisältää riippuvuudet kaikkiin eri tasoisiin suunnitelmiin: kansalliset maankäyttöä koskevat säädökset (*nasjonale planbestemmelser*), seutukaavat (*regionalplaner*), kunta- ja kuntaosakaavat (*kommune[del]planer*), asemakaavat (*reguleringsplaner*) sekä rakennusluvat (*bebyggelseplaner*). Kaikki suunnitelmatyypit ovat rinnakkaisia, eivät hierarkkisia luokkia.

[45] engl. *Single Sign-On*, kertakirjautuminen on tapa toteuttaa kirjautuminen useaan palveluun yhden käyttäjätunnuksen autentikoinnin kautta.
[46] `https://register.geonorge.no/`

OTSO.dev
consult@otso.dev
+358 50 544 9174

Kyseisen rakenteen ansiosta ”suunnitelmia voidaan muuttaa joustavasti joko päivittämällä koko suunnitelma tai sen eri teema- tai osa-alueita, vaikka vain sen sosiaalista osiota tai erillisiä politiikkoja, kuten asunto-, elinkeino- tai ympäristöpolitiikkaa koskevia linjauksia.” (Lehtovuori et al., 2019)

Asemakaavaan kuuluvan alueen (RpOmråde) lisäksi asemakaavan tietomallissa on erillisiä siihen liittyviä juridisia luokkia, kuten alueiden rajaa edustava polku RpJuridiskLinje, tai tiettyä pistesijaintia ilmaiseva RpJuridiskPunkt. Näihin luokkiin liittyy RpPåskriftluokka, johon voidaan sisällyttää tekstimuotoinen kaavamääräys tai vastaava velvoite. (Kommunal- og moderniseringsdepartementet, 2019)

Tanskan tapaan Norjakaan ei ole voinut välttyä täysin pitkään kehitetyn järjestelmän ja rakenteisen tiedon murroksen mukanaan tuomasta teknisestä velasta.

```
<complexType name="NasjonalArealplanIdType">
  <sequence>
  <element minOccurs="0" name="kommunenummer" type="gml:CodeType">
    <annotation>
      <appinfo>
        <defaultCodeSpace>
          http://skjema.geonorge.no/SOSI/produktspesifikasjon/
    Reguleringsplan/20190401/Kommunenummer.xml
        </defaultCodeSpace>
        <taggedValue tag="SOSI_navn">KOMM</taggedValue>
      </appinfo>
    </annotation>
  </element>
  ...
```

Listaus 6: Esimerkki tavasta, jolla asemakaavan skeemassa hyödynnetään ulkoista XML-koodilistaa joka sisältää avain-arvo-pareja kaavan kunnan tallentamiseen.

Esimerkiksi asemakaavoja kuvaavaan ReguleringsPlan-skeemaan on joissain tapauksissa upotettu Tanskan PlanDK:n tapaan kiinteästi lueteltuina tyyppeinä koodilistojen virkaa toimittavia rakenteita, joilla ei ole esimerkiksi omaa nimiavaruutta, viitettä ulkoiseen koodilistaan tai muuhun koneluettavaan sisältöön Toisaalla taas koodilistoja käytetään ”oikeaoppisesti” esimerkiksi GML:n tarjoamien rakenteiden kautta (kuva 6). Lisäksi kaikessa XML-serialisoidussa tiedossa joudutaan taggedValue-elementtien kautta ylläpitämään viittauksia järjestelmän sisäisiin dot dot -tyyppeihin (kuvissa 6 ja 7 KOMM sekä VERSJONID).

```
<complexType name="IdentifikasjonType">
  <sequence>
    ...
    <element minOccurs="0" name="versjonId" type="string">
      <annotation>
        <appinfo>
          <taggedValue tag="SOSI_navn">VERSJONID</taggedValue>
        </appinfo>
      </annotation>
    </element>
  </sequence>
</complexType>
```

Listaus 7: IdentifikasjonType-tyypin versionhallinnan toteutus

Kaavojen välillä ei ole nykyisessä mallin versiossa 4.5.2 varsinaista aitoa versionhallin-

OTSO.dev
consult@otso.dev
+358 50 544 9174

taa; kaavojen tiloja ilmaistaan attribuuteilla koodilistojen arvojen (esimerkiksi *voimassa* tai *kumoutunut*) sekä tallennettujen päivämäärien avulla.

Kaavoja sekä niiden osien (esimerkiksi RpOmrådeType-tyyppisiä kaava-alueita) versiointi on toteutettu IdentifikasjonPropertyType-tyypillä, joka viittaa IdentifikasjonType-tyyppiseen elementtiin. Elementti ei kuitenkaan ole pakollinen kaikille kaava-alueille, ja se sisältää tunnisteena ainoastaan ISO 8601 -muotoisen aikaleiman, jota käytetään saman elementin eri versioiden tunnistamiseen. Elinkaaren aikainen tunniste löytyy samaisesta IdentifikasjonType-tyypistä, kuten myös paikallinen nimiavaruus, jonka puitteissa tunnisteen täytyy olla uniikki.

SOSI-mallin versiosta 5.0 kuitenkin löytyy luonnosasteella oleva Plan-luokkaa vastaava yleinen ArealPlan-luokka[47], jonka avulla versionhallinta voidaan toteuttaa viittaamalla yhteen tai useampaan saman luokan järjestyksessä edeltävään objektiin.

Tanskan ja Suomen tapaan myös Norjassa kunnat tuottavat suunnittelutietoa vaihtelevilla työkaluilla ja prosesseilla. Jokaista kaavavaihetta varten tuotetaan tietomallimuotoinen esitys kommentoitavaksi, mutta itse suunnitteluorganisaatioiden sisällä tieto on edelleen hyvin heterogeenista, eikä suunnitteluprosessin sisäiseen versionhallintaan ole vielä kehitetty ratkaisua – tyypillinen tapa on hyödyntää projektikansioita sekä tiedostojen nimiin pohjautuvaa versiointia. Norjalaisten oman arvion mukaan noin 75 % kaavamääräyksiä koskevasta tietomallimuotoisiin suunnitelmiin päätyvästä sisällöstä on tällä hetkellä rakenteista.

Oikeudellinen validointi Norjan järjestelmässä rajoittuu tällä hetkellä SSO-alustan kautta myönnettyihin käyttöoikeuksiin ja lokitietoihin.

Norjan karttaviranomainen julkaisee runsaasti kehitettyjä kirjastoja avoimena lähdekoodina.[48].

3.2 Vertailu kansallisella tasolla

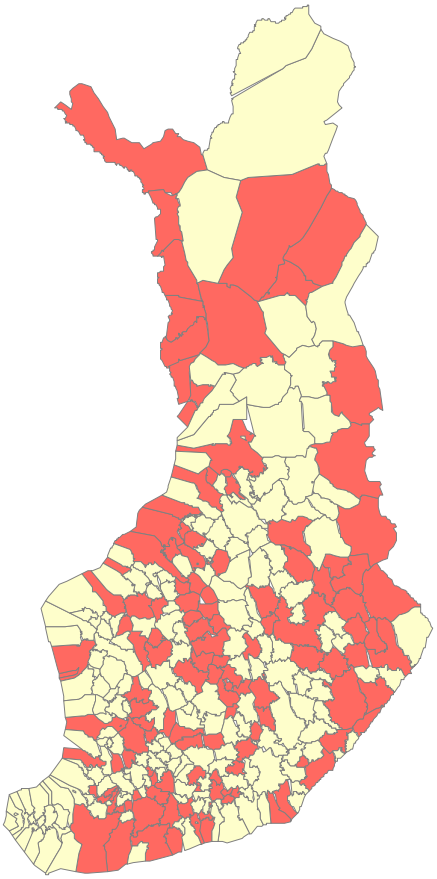
Suomen yli 300 kuntaa muodostavat hyvin heterogeenisen joukon, joten edustavien tulosten saaminen on sekä otoksen koostamisen että kysymyksenasettelun kannalta hyvin haastavaa. Kyselyssä tarkoituksena oli saada kartoitettua yleisellä tasolla seuraavia tekijöitä:

1. kaavoituksen henkilö- ja teknisten resurssien määrä ja laatu,
2. kaavaprosessin sisäisen tiedonhallinnan menetelmät,
3. mahdolliset kehitysintressit,
4. tietomallien käyttö ja nykyinen tuntemus sekä
5. asenne MRL-uudistuksen mukanaan tuomia muutoksia kohtaan.

Koska selvitystä ei lähtökohtaisesti rakennettu tilastollisesti validiksi, ei otoksen koolle asetettu alarajaa ja kyselyä pyrittiin lähettämään mahdollisimman moneen kuntaan. Otokseen otettiin aluksi mukaan kunnat joista oli jo kerätty vastaavaa tutkimusmateriaalia

[47] Toteutus liittyy SOSI:n INSPIRE-yhteentoimivuuden edistämiseen.
[48] `https://github.com/kartverket`

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 22: Kyselyn otantaan valikoituneet kunnat merkittynä punaisella

tutkimuksen puitteissa (ja joihin voitiin ottaa suoraan yhteyttä tarkentavien kysymysten osalta), jonka jälkeen loput valikoitiin mukaan satunnaisotannalla (kuva 22).

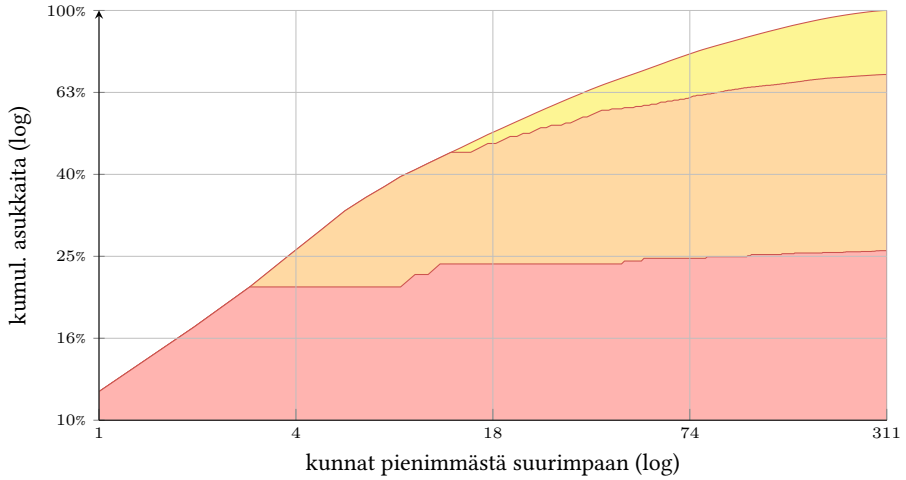
Jokaisesta kunnasta otettiin yhteyttä maankäytön suunnittelun päällikköön, kaavoittajiin, teknisen toimen vastuuhenkilöihin tai muihin kaavoituksen yhteydessä työskenteleviin, kunhan yhteystiedot olivat selvitettävissä kohtuullisella vaivalla. Selvitystyön aikana kyselyitä ehdittiin lähettää 129 kuntaan (kts. kuva 22, kattavuus noin 41,8 % kunnista ja 69,7 % maan asukkaista), ja vastauksia ehdittiin saada 27 kpl (8,7 % kunnista, 32,8 % asukkaista). Kyselyn vastausprosentti oli 20,9 %. Vastaukset analysoitiin kvalitatiiviseen analyysiin tarkoitetulla Atlas.TI-ohjelmistolla[49].

Vastausten analysoinnissa käytettiin aluksi lähtökohtana Tilastokeskuksen tilastollista kuntaryhmitystä[50] kaupunkimaisiin, taajaan asuttuihin ja maaseutumaisiin kuntiin. Kaupunki–maaseutu-jako ei kuitenkaan paljastanut aineistosta mitään selkeää trendiä tai jakaumaa. Kaikista selvimmin kuntien välisiä eroja näytti vapaana muuttujana selittävän yksinkertaisesti niiden asukaluku.

Kuvassa 23 on esitetty kyselyn kattavuus suhteessa kuntien kokoon. Kuvassa on järjestetty Suomen kunnat laskevaan suuruusjärjestykseen, ja esitetty niiden kumulatiivinen

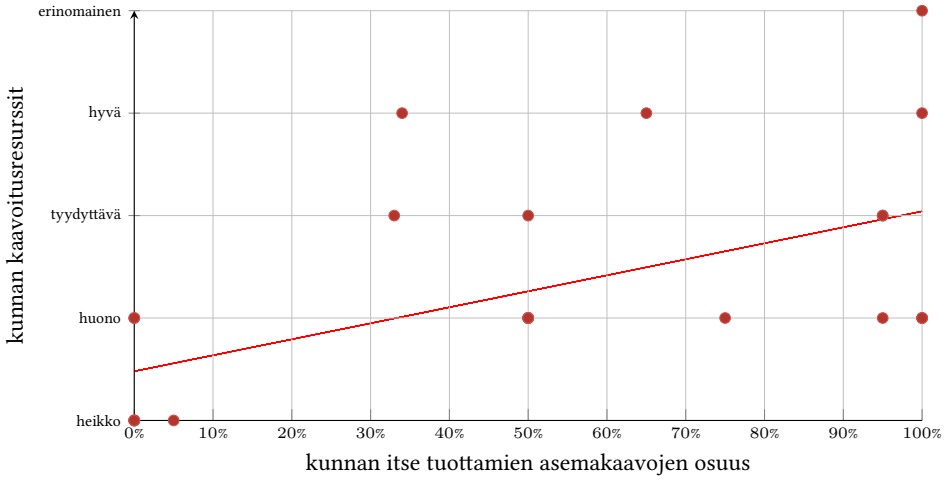
[49] `https://atlasti.com/`
[50] `http://www.stat.fi/meta/kas/til_kuntaryhmit.html`

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 23: Suomen kuntien, lähetettyjen kyselyjen ja saatujen vastausten jakauma kumulatiivisella asukasluvulla mitattuna. Punaiset pystyviivat edustavat korijaottelua (kiinteät viivat kolmen yhteenvetoon käytetyn korin rajoja).

asukasjakauma (keltainen käyrä). Kyselyn kumulatiivinen kattavuus asukkaiden osalta on esitetty oranssilla ja saatujen vastausten kattavuus punaisella.

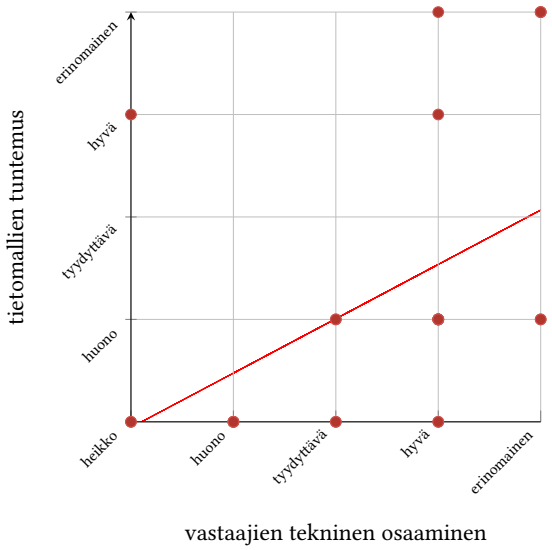


Kuva 24: Kyselyyn vastanneiden kuntien kaavoitusresurssien määrä suhteessa itse valmisteltuihin kaavoihin

Alle 10 000 asukkaan kunnissa kokonaan konsulteille ulkoistetun valmistelutyön osuus oli merkittävä – yli 60 % kertoi olevansa täysin ulkoistuksen varassa. 10 000 asukkaasta ylöspäin vastausjoukossa oli tasaisesti kuntia, joiden oma valmistelutyö kattoi yli 95 % kaikista kaavoista. Kaikkien vastaajien keskiarvo omalle valmistelutyölle oli 48 %.

Valtaosa vastaajista halusi säilyttää työnjaon ennallaan. Yleisin syy ulkoistamisen kasvattamiseen oli raha. Yleisin syy lisätä oman työn osuutta vuorostaan osaamisen keskittäminen ja aktiivisemman roolin hakeminen. Kuten kuvasta 24 nähdään, kaavoituksen resursointi korreloi selvästi valmistelutyön ulkoistuksen kanssa.

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 25: Kyselyyn vastanneiden kuntien tekninen osaaminen suhteessa tietomallien tuntemukseen

Suuria kuntia sekä kahta poikkeusta lukuun ottamatta, yksikään kunta ei tuottanut (tai tiennyt tuottiko) virallista kaava-aineistoa tietomallimuotoisena tai INSPIRE-yhteensopivana. Samoin kyseistä joukkoa lukuunottamatta yhdessäkään kaavatietomallin käsite ei ollut tuttu tai selkeä, eikä niistä tai niiden käytöstä oltu keskusteltu lainkaan kunnan sisällä.

Kuva 25 esittää kuntien teknisen osaamisen itse arvioitua tasoa suhteessa annettuihin vastauksiin tietomalleista ja tietomallipohjaisesta kaavoituksesta. Tulos oli odotettava, ja kuvaa hyvin sitä, kuinka tietomallipohjainen suunnittelu mielletään tekniseksi ja varsinaiseen suunnittelutyöhön liittymättömäksi asiaksi.

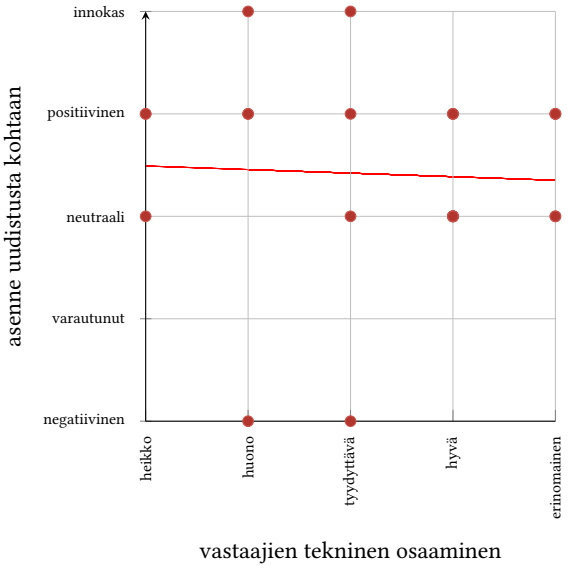
Suurimmissa kunnissa, joissa kaavoja valmisteltiin itse runsaissa määrin, tietomallimuotoista kaava-aineistoa tuotettiin rutinoituneesti (kaikki vastaajat käyttivät KuntaGML:stä jalostettua tietomallia).

Näissä kunnissa suuri koko on mahdollistanut omien pysyvien kehitysresurssien allokoinnin ja henkilöstön palkkauksen, mikä tekee merkittävän eron moniin pieniin kuntiin, joissa haluja kehittyä olisi, mutta resurssit eivät salli sitä. Pienten kuntien keskuudessa oli osittain omatoimisesti kartoitettu avoimen lähdekoodin työkalujen käyttöönottoa ratkaisuna resurssien puutteeseen.

Yhtä lukuunottamatta kaikissa alle 50 000 asukkaan kunnissa resurssit olivat joko keskinkertaisella tai huonolla tasolla. Monissa aliresursoiduissa ja ulkoistukseen luottavissa kunnissa ei ollut varsinaista sisäistä kaavoitusosaamista jäljellä. Kyseisissä vastauksissa korostuivat huoli jokapäiväisistä tehtävistä selviytymisestä, eikä toiminnan kehittämistä pidetty lainkaan relevanttina kysymyksenä.

Kunnat pyrkivät keskimääräisesti operoimaan ja kehittämään toimintaansa MRL:n uudistamisen aikataulusta riippumatta. Kehitysmyönteisimmät kunnat (11 % vastaajista) suhtautuivat kaavatietomalliin ja kokonaisuudistukseen positiivisesti, keskimääräinen suhtautuminen oli neutraalia.

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 26: Kyselyyn vastanneiden kuntien tekninen osaaminen suhteessa kaavoitusjärjestelmän uudistukseen

Kuvasta 26 nähdään, että tietomallimuotoisen kaavoituksen edistäminen osana MRL-uudistusta herättää itse asiassa vähemmän teknisesti orientoituneissa vastaajissa enemmän sekä voimakkaita positiivisia että negatiivisia tunteita. Näissä kunnissa uudistusta pidettiin joko uhkana esimerkiksi omalle toiminnalle tai pelastuksena kunnan tilanteeseen.

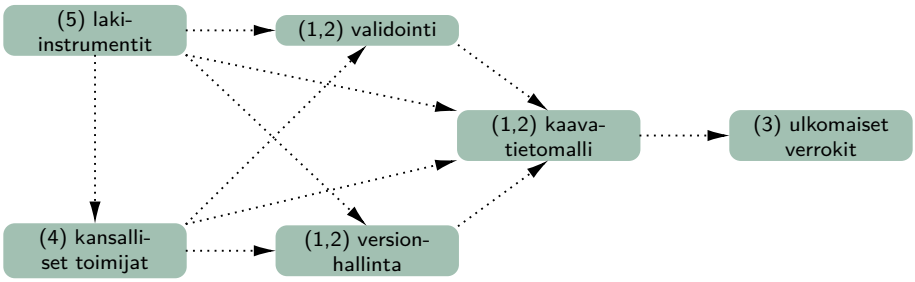
Kuntien käsitys versionhallinnan ja validoinnin sisällöstä oli suurilta osin hyvin yksinkertainen. Yleisin menetelmä validoinnille oli sekä kvalitatiivisen että kvantitatiivisen tiedon osalta ”useiden silmäparien käyttö”, joskin joissain kunnissa paikkatietomuotoista sekä rekisteriaineistoa hyödyntämällä voitiin varmentaa rajatusti joitain lukuarvoja.

Versionhallinta oli lähes kaikissa tapauksissa kunnan koosta riippumatta toteutettu joko verkkolevyllä ja kansiorakenteella, tai se ei varsinaisesti kuulunut aktiivisena osana prosessiin.

Ainoastaan suurimmilla sekä yhdellä kehitysmyönteisellä kunnalla oli käytössä versiohallittuja CMS- ja Groupware-ohjelmistoja projektipankkeina sekä projektinhallinnan käyttöön. Sisäisen tietomallivarannon päivitykset kohdistettiin suoraan muutoskokonaisuuksiin, joiden järjestystä ylläpidettiin pääosin päivämääräviittauksilla.

OTSO.dev
consult@otso.dev
+358 50 544 9174

4 Tulokset



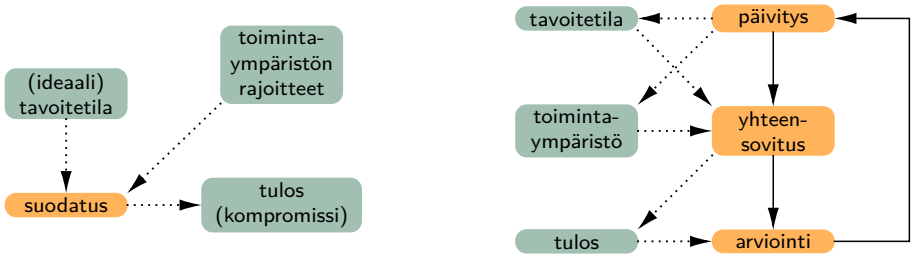
Kuva 27: Tutkimuksista johdettu käsitteellinen kartta tulosten keskinäisistä riippuvuussuhteista

Tässä luvussa vastataan selvitykselle asetettuihin tutkimuskysymyksiin yhteenvedon muodossa. Ennen vastauksiin siirtymistä on kuitenkin aiheellista kerrata selvityksen perspektiviä, joka on esitetty kaaviona kuvassa 27.

Tutkimuskysymykseen 1 (”Mitä merkittäviä hyötyjä ja haasteita liittyy kansallisen kaavatiedon versionhallinnan ja validoinnin käyttöönottoon ja toteuttamiseen?”) vastaaminen vaatii aluksi sen määrittelyä, millaisena versionhallinta ja validointi *tulisi toteuttaa*. Niiden toteutus vuorostaan on riippuvainen tutkimuskysymyksen 2 (”Miten kaavatietomallin rakenne vaikuttaa tiedon versionnhallintaan ja validointiin?”) vastauksesta.

Kysymys 3 (”Miten maankäyttöpäätöstietojen versionhallinta ja validointi on toteutettu kansainvälisissä esimerkeissä?”) informoi tietomallin, versionhallinnan ja validoinnin määrittelyä, mutta koska tässä tapauksessa opin hakeminen muista maista ei ole ollut vastavuoroista, on assosiaatio yksisuuntainen. Kysymykseen 4 (”Mitkä ovat keskeisten kansallisten toimijoiden valmiudet versionhallintaan ja validointiin?”) vastaaminen informoi tavoitetilan toteuttamiseen liittyvistä haasteista, muttei kuitenkaan voi toimia sitä määrittävänä tekijänä, sillä tavoitetila muodostuu kaavatietomallin tutkimukseen pohjautuvista suosituksista.

Lopuksi, kysymys 5 (”Mitä tulee ottaa huomioon maankäyttöpäätöstietojen versioinnista ja validoinnista säädettäessä ja ohjeistaessa?”) on alisteinen kaikille muille tekijöille, sillä selvityksen tehtävänä on informoida sen ulkopuolista uudistusprosessia toimivan versionhallinnan ja validoinnin edellytyksistä.



Kuva 28: Vasemmalla on malli jossa vakiintuneiden käytäntöjen muuttamista karsastetaan, oikealla malli, jossa ympäristöön ja tavoitteisiin tehtäviä muutoksia yhteensovitetaan ja muokataan iteratiivisesti lopputuloksen laadun maksimoimiseksi.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Tämän selvityksen puitteissa määritellään tavoitetilaa, mutta määrittely on pyritty tekemään hakemalla tasapainoa tavoitteiden vaatiman muutoksen ja toimintaympäristön odotetun mukautumiskyvyn välille (kuva 28). Tarkastelu ei kuitenkaan ole millään muotoa kattava, ja selvityksen tuloksia ja niistä johdettuja suosituksia tuleekin tulkita ja hyödyntää vain osana laajempaa systeemistä tarkastelua (kts. kappale 1.3).

Seuraavaksi käymme tulokset lyhyesti läpi kuvassa 27 kuvatussa riippuvuusjärjestyksessä (säädökset sisältyvät suosituksiin).

4.1 Tutkimuskysymys 3: Kansainväliset verrokit

Selvityksessä tarkasteltuihin kansainvälisiin verrokkeihin tulisi suhtautua rohkaisevana esimerkkinä siitä, mitä riittävällä resursoinnilla ja pitkäjänteisellä kehitystyöllä voidaan saada aikaan.

Sekä Norjan että Tanskan kohdalla kansallinen rekisteri ja sen palveluekosysteemi on onnistuttu vakiinnuttamaan käyttöön valtiollisen ja kunnallisen tason roolien muutoksen ohessa. Kehityksessä on täten voitu siirtyä yhteentoimivuuden kasvattamiseen (mm. INSPIRE), toiminnan tehostamiseen ja laadun parantamiseen.

Australia on erityisen vaikuttava esimerkin onnistuneesta sektorit ylittävien siltojen rakentamisesta, sillä siellä kansallisen tason järjestelmän toteutusta ja onnistunutta käyttöönottoa voidaan maan koon sekä osavaltiomallin vuoksi pitää vielä huomattavasti Pohjoismaita haastavampana. Positiivista on myös se, että julkishallinnon lisäksi on yksityissektori on saatu osallistettua tuotetun tiedon jakamiseen, ja perinteiseen *public-private partnership* -mallin sijaan tutkimussektorin yhteiskunnallinen arvo on tunnistettu antamalla sille järjestelmässä keskeinen johtava rooli.

Sekä Norjan että Tanskan järjestelmissä havaittiin kuitenkin jo tämän selvityksen kevyen tarkastelun perusteella joitakin johdannossa (kappale 1.3) kuvattujen kehitystyön riskien ilmentymiä.

Tanskassa havaitut ongelmat ovat pitkälti seurausta Norjaa kapeammasta sektorikohtaisesta näkökulmasta järjestelmän vaatimusmäärittelyssä ja kerralla tehtävän laajan toteutuksen ohjauksen haasteista. Tämän seurauksena Tanskan tietomallimäärittely on vielä tällä hetkellä puutteellinen esimerkiksi asemakaavoituksen (ja laajemmin maankäytön suunnittelun) tarpeiden osalta, ja mallin rakenteesta sekä työnkulusta löytyy määrittelynaikaisia lyhytnäköisiä ratkaisuja (kts. kappale 3.1) jotka tulisi korjata kustannussäästöjen, käytön sekä jatkokehityksen helpottamisen vuoksi. Järjestelmän kehitystä ja sen ympärille rakennettavien verkostojen muodostumista haittaa tällä hetkellä osaltaan läpinäkymättömyys, mikä on seurausta hajanaisesta ja osittain puutteellisesta dokumentaatiosta ja kehitystyön sulkeutuneisuudesta.

Norjan kohdalla kehitystyö on jo 90-luvulla ylittänyt sektorirajat ja (hallinnollisesti ja teknillisesti) holistiseen malliin on pyritty johdonmukaisesti. Ongelmat näyttävät pääosin liittyvän esimerkiksi lakiuudistusten ja muuttuvien säädösten aiheuttamiin muutostarpeisiin, jotka ovat johtaneet mallien rakenteiden ja niihin upotettujen koodilistojen kohdalla tapauskohtaisiin ratkaisuihin. Myös nämä rakenteet tulevat pidemmällä aikavälillä vaatimaan vuorovaikutteista – ei ainoastaan alisteista – kehitystä suhteessa lainsäädäntöön.

Kaikkien kolmen maan kansallisten järjestelmien tekninen validointi on pääosin toteutettu ennakko-odotusten mukaisesti: tavoitteena on varmistaa vain se, että kaavojen sisällöt siirtyvät sekä lähettäjän että vastaanottajan järjestelmien välillä yhteisesti ymmärretyssä muodossa. Teknisessä validoinnissa on hyödynnetty tyypillisiä XML- ja relaatiotietokantaskeemojen tarjoamia mahdollisuuksia koskien rakenteiden, tietotyyppien sekä arvojen tarkistamista.

Oikeudellisen validoinnin voidaan katsoa olevan puutteellista kaikissa tapauksissa, erityisesti Australian kohdalla (puuttuva SSO-autentikaatio). Sekä Tanskan että Norjan mallissa oikeudellinen validointi rajoittuu tiedon julkaisuun ja siirtämiseen liittyvään autentikaatioon. Prosessin aikana itse sisällön osiin kohdistuvia muutoksia ja kansallisesti tallennettuja versioita ei allekirjoiteta sähköisesti kummassakaan maassa, sillä vastuunjaon mukaisesti oikeudellisen validoinnin katsotaan kuuluvan sisällöllisen validoinnin ohella kunnan sisäisiin tehtäviin.

Myös versionhallinta on yleisesti ottaen toteutettu Tanskan ja Norjan järjestelmissä ennakko-odotusten mukaisena, kohteita versioidaan mm. juoksevalla numeroinnilla sekä päivämäärillä, ja ne identifioidaan pysyvillä elinkaarenaikaisilla tunnisteilla. Toteutuksissa on kuitenkin parannettavaa, sillä nyt ne soveltuvat varsin yksinkertaisiin käyttötapauksiin ja niiden laajennettavuus nykymuodossaan on heikko.

4.2 Tutkimuskysymykset 1,2,4: Vaikutukset, hyödyt, haasteet ja valmiudet

4.2.1 Tietomallin rakenteen vaikutukset versionhallintaan ja validointiin

Tietomallin rakenne vaikuttaa suoraan siihen, millä tasolla versionhallintaa on kannattavaa harjoittaa. Yksinkertainen malli, jossa tallennetaan ainoastaan suunnitelman tila tietyllä ajanhetkellä, ei tarjoa varsinaista versionhallinnan tukea, sillä muutoksista voidaan ainoastaan päätellä yksilöllisten tunnisteiden pohjalta, onko jokin alue vain muuttanut muotoaan, luotu kyseisessä versiossa tai poistunut siinä.

Tämänhetkisessä TUMA-luonnosmallissa luokka rak:KohteenMuutostapahtuma sallii muutoskokonaisuuksien (versioiden) sisällä yksittäisten maankäyttökohteiden elinkaaren seuraamisen. Mallin käyttötapauksia ja joustavuutta laajentaisi kuitenkin vielä merkittävästi se, mikäli versiohistoriassa kyettäisiin kuvaamaan maankäyttökohteen jakaminen useampaan tai käänteisesti niiden yhdistäminen yhdeksi kohteeksi.

Minimissään kohteiden ja muutosten elinkaaren säilyttäminen on merkittävässä roolissa myös validoinnin osalta: viittausten avulla voidaan tarkistaa automaattisesti, että kansalliselle alustalle tai muuhun vastaavaan järjestelmään ei ikinä hyväksytä inhimillisen virheen vuoksi esimerkiksi sisällöltään nykyistä voimassa olevaa versiota edeltänyttä sisältöä.

Mallin rakenteella on myös muita merkittäviä vaikutuksia validointiin: erityisesti lähtötietojen osalta olisi tärkeää pystyä seuraamaan niiden elinkaarta (milloin lähtötieto on sisällytetty kaavaprosessiin sekä milloin sen sisältö on päivittynyt).

Tilanne, jossa kaikki kaavaprosessiin tuotava ja kaavatietomallin yhteydessä käytettävä lähtötieto on versiohallittua ja sitä voidaan hyödyntää ulkoisten viittausten kautta, on vielä

kaukainen. Mikäli kaavatietomalli tukee viittausta ulkoisiin (esimerkiksi tiedostomuotoisiin) aineistoihin yhdessä niistä lasketuille tiivisteille, voidaan yksikäsitteisesti varmistaa kaavatietomallin spesifin version viittaavan juuri tiettyihin lähtöaineistoihin. Tiivisteiden (ja niitä vastaavien tiedostojen) vaihtaminen tietomalliesityksessä voidaan estää liittämällä tietomalliin vaatimus sen sähköisestä allekirjoittamisesta (XML Signature).

Tietomallin sisäisellä luokkahierarkialla ja tunnisteiden toteutuksella on merkitystä myös sen osalta, kuinka helposti yhden kaavasuunnitelman sisältämiä kohteita voidaan myöhemmin hyödyntää itsenäisesti. Jotta kaavan sisältämät atomiset (pienimmät itsenäiset) kohteet voidaan irrottaa kaavasta muualla käytettäväksi, täytyy niiden sisältää esimerkiksi attribuuttiviittausten avulla kaikki niihin kohdistuvat yleisemmät arvot ja rajoitteet. Toisin sanoen, esimerkiksi yksittäisen kaavayksikön tulee tietää koko kaava-alueeseen kohdistetusta rajoitteesta, sen ei tulisi selvitä ainoastaan epäsuoran viittauksen kautta (yksikkö on hierarkkisesti katsottuna kaava-alueen jäsen tai joukko-opillisesti katsottuna osajoukko).

4.2.2 Yleisiä kansallisen rekisterin käyttöönottoa puoltavia seikkoja

Keskittämällä kaikki tieto yhteen kansalliseen rekisteriin, voidaan sitä hakea sijainnista ja laajuudesta riippumatta aina yhdellä haulla ja tulosta käsitellä yhtenä joukkona. Rekisterien hajauttaminen kuntiin pakottaisi hakemaan kuntarajat leikkaavaa sisältöä useammasta järjestelmästä sekä yhdistämään ne hakukohtaisesti.

Oikein toteutettuna kansallinen rekisteri sisältää kaavojen tiedot muodossa, joka sallii kyselyjen tekemisen hallinnollisista rajoista riippumatta: pyydettäessä tietokannalta tietyn rajatun alueen sisään sijoittuvia kohteita on sille yhdentekevää, minkä kaavan, kaupunginosan tai kunnan alueelle ne kuuluvat. Kyseiset tiedot siirtyvät kyselyn tulokseen tällöin osana kohteisiin sisältyviä ja niiden sisältöä kuvailevia viittauksia. Tämän seurauksena hallinnolliset muutokset (esimerkiksi kuntarajojen siirto tai kuntien yhdistyminen) ei aiheuta mallin näkökulmasta merkittävää tiedon uudelleenjäsentelyä.

Rekisteri voitaisiin toki toteuttaa Norjan geosynkronontimallia vastaavasti (kts. 3.1.3), mutta tämä selvitys puoltaa Kuntapilotin näkemystä siitä, että satojen itsenäisten järjestelmien pystytystä, testausta sekä ylläpitoa ei voida näillä näkymin pitää realistisena ratkaisuna (Ramboll Finland Oy ja Ubigu Oy, 2018).

Toteutus vaatisi lisäksi monimutkaisempaa järjestelmärakennetta, sillä tiedon eri osien ollessa kuntakohtaisesti hajautuneita eri järjestelmiin, ei voida hyödyntää perinteisiä klusterointiratkaisuja (peilausta). Tämän vuoksi hakukyselyiden kohdentaminen oikeisiin järjestelmiin muodostuisi kriittisen tärkeäksi: kyselyt, joiden tarkoituksena on kohdistua yhteen, tehokkaan järjestelmän varassa toimivaan kuntaan, saattaisivat kohdistua myös toiseen, pieneen ja hyvin kevyellä järjestelmällä varustettuun kuntaan – kysely voisi tällöin toimia epäsuorana palvelunestohyökkäyksenä.

Keskitetty järjestelmä vaatii tiedon tuottajilta sekä hyödyntäjiltä ainoastaan rajapinnan ehdot täyttävää vuorovaikutusta, mikä on huomattava kevennys oman infrastruktuurin ylläpitoon verrattuna. Lisäksi keskitetyn järjestelmän skaalaus tietomäärien kasvaessa voidaan toteuttaa hajautettuja järjestelmiä tehokkaammin.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Yhteisen rekisterin hyötyinä voidaan sekä Tanskan että Norjan esimerkin mukaisesti pitää myös sitä, että mahdollisimman suuri määrä koneluettavaksi kuvattavaa tietoa voidaan upottaa tietomalliin ja saada muiden rinnakkaisten ja kaavoituksen jälkeisten prosessien käyttöön keskitetysti. Erityisesti epätarkoituksenmukainen manuaalinen raportointi jää tällöin pois, kun mahdollisimman suuri määrä raportoitavaa tietoa syntyy ohessa kun suunnitelmaa työstetään tietomallimuotoisena.

Kansallisen ja paikallisen tason vuorovaikutuksesta voidaan tehdä seuraavanlainen karkea arvio: mikäli lähdetään oletuksesta, että koko maassa hyväksytään vuosittain karkeasti noin 2 000 asemakaavaa (Rinkinen ja Kinnunen, 2017) joista tallennettaisiin lisäksi OAS, luonnos- sekä ehdotusvaiheet, puhutaan päivätasolla noin yhdestä kirjoitusoperaatiosta per tunti (8 000 kpl/a, 21,9/pvä).

Sen sijaan lukuoperaatioiden määrä tullee olemaan niihin suhteutettuna suuruusluokkaa 1 000:1 tai 10 000:1, erityisesti johtuen siitä, että versiohallittu ja sisällöltään validoitu rekisteri on rikas lähde aikasarja-analyyseille, korrelaation ja kausaalisten linkkien löytämiselle sekä erilaisille temaattisille aineistoille.

Rekisterin käyttäjäkuntaan voidaan kuvitella kuntien ja konsulttitahojen lisäksi kuuluvan muita viranomaisia, yrityksiä, kansalaisjärjestöjä ja yksittäisiä kansalaisia sekä yliopistoja ja tutkimusinstituutteja, joilla on käyttäjästä riippuen laajat tai rajatut käyttöoikeudet järjestelmään.

4.2.3 Kansallisen tason versionhallinta ja validointi

Teknisesti ja oikeudellisesti sisällöltään validoidun kansallisen rekisterin merkittävin hyöty on virheettömän sekä vastuukysymyksien osalta jäljitettävän tiedon keskittäminen yhden rajapinnan varaan. Osapuolten kykyä viestiä keskenään rajapinnan edellyttämällä tavalla voidaan pitää valtaosassa tapauksia riittävänä auditointina toiminnan luotettavuudelle.

Itse tietomallin rakenteella auditointi voidaan ulottaa myös sisäisen tiedonhallinnan johdonmukaisuuden validointiin – sen kautta voidaan esimerkiksi varmistua siitä, että osapuolen tuottama tieto on kohteiden versiohistorian osalta linkittynyttä kyseisen osapuolen saamaan tietoon.

Tilanteessa jossa teknistä tai oikeudellista validointia voidaan tehdä automatisoidusti, pitäisi lähtökohtaisesti pyrkiä siihen että se tehdään, ja tekemättä jättämiseen pitäisi löytyä erittäin hyvin perusteltu poikkeussyy, jollaista tämän selvityksen puitteissa ei ole keksitty.

Kaavoitusprosessin työnkulkuun kytkeytyy sen aikana useita toimijoita, työkaluja ja järjestelmiä. Prosessin onnistuneen lopputuloksen kannalta onkin kriittistä, että osapuolet voivat luottaa toistensa tuottamaan tietoon sitä hyödyntäessään – työnkulkujen tulisi siis toteutua kappaleessa 2.3 kuvaillun mallin mukaan.

Versionhallinnan käyttö laajentaa validointikykyjä itse tiedonkäsittelyn prosessien sisältöön. Mikäli kansallinen alusta toteutettaisiin ilman versionhallintaa, olisi kyseessä puhdas ajantasakuva maankäytön suunnittelun tilanteesta.

Järjestelmän hyöty olisi kuitenkin kyseenalainen: kun tietoa haettaisiin rekisteristä tietyn prosessin käyttöön ja tällä välin rekisterin tiedot päivittyisivät, jäisi ainoa otos kyseisestä ajanhetkestä sen ladanneen osapuolen haltuun On tällöin epäselvää, miten kyseisen tiedon validiteetti voitaisiin varmentaa, ja miten toimittaisiin sellaisten osapuolten tapauksessa, jotka tarvitsisivat järjestelmän nykyhetkeä aiempaa tietoa mutta eivät pystyisi sitä enää lataamaan.

Ajallisen historian tallentaminen on validoinnin tapaan valinta, jonka pitäisi periaatteellisella tasolla olla oletus, ja siitä poikkeaminen perustella hyvin.

Mahdolliset kaavoista tehtävät valitukset ja oikeuskäsittelyt ovat yksi tärkeä syy niiden muutoshistorian tallentamiselle.

Toisen muodostavat MAL-sopimusten, maakuntakaavojen, strategisten suunnitelmien ja tulevaisuusvisioiden kaltaiset tarpeet, joissa tulevaisuutta halutaan ohjata ennusteisiin ja analyysiin pohjautuen.

Ylipäänsä kaikki ylikunnalliset – esimerkiksi temaattiset – tietotarpeet muodostavat mielekkäitä käyttötapauksia versiohallitulle tiedolle.

Kansallisen tason versionhallinnan tuen tärkeys korostuu myös kaikkien kaavaprosesseihin osallistuvien toimijoiden sisäisen toiminnan sekä keskinäisen vuorovaikutuksen ymmärtämisessä ja ohjaamisessa, ts. toiminnan laadullisessa validoinnissa (kts. 2.3.3).

Tarve laadullisen validoinnin ohjausvaikutukselle syntyy siitä, että kansalliseen rekisteriin tallennettavan tiedon hyödyllisyyden aste ei riipu ainoastaan sen teknisestä korrektiudesta: rekisteri olisi hyödytön, mikäli hypoteettisessa tilanteessa tietoa tuottavat tahot syöttäisivät sinne teknisesti validia mutta sisällöltään mielivaltaista tietoa.[51]

Yhteenvetona voidaan todeta, että yhteinen tietomalli sekä kansallinen rekisteri muodostavat keskinäisen riippuvuussuhteen, jossa tietomalliin kumuloituvan tiedon arvo on suoraan verrannollinen yksittäisten toimijoiden kykyyn tuottaa sitä, ja tuotannon toimivuus vuorostaan on riippuvainen sille asetetuista yhteisistä vaatimuksista. Tämän vuorovaikutuksen sekä sen ympärille rakentuvan ekosysteemin arvon maksimoinnissa tiedon kattavalla validoinnilla ja versionhallinnalla on merkittävä rooli.

4.3 Nelikenttätarkastelu

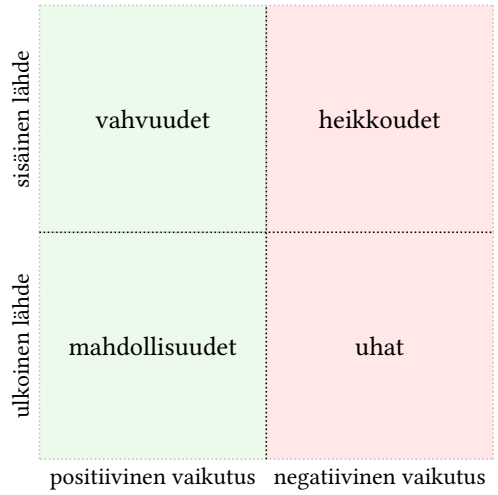
Lopuksi aiheeseen liittyviä tekijöitä tarkastellaan vielä systeemisesti eri näkökulmista yksinkertaista SWOT-matriisia[52] hyödyntäen (kuva 29).

4.3.1 Mahdollisuudet

Kaavoitusprosessin laatu Toimintaympäristön keskeisenä osana tietomallien potentiaali versionhallinnan ja validoinnin parantamiseen on merkittävä. Ilmaisukyvyltään riittävän rikas ja joustava tietomalli tekee sisällön versionhallinnasta mielekästä hyvin hienostuneella (yksityiskohtaisella) tasolla maankäytön suunnitteluprosessien sisällä. Mahdolli-

[51] engl. GIGO: *Garbage In, Garbage Out*
[52] engl. *Strengths, Weaknesses, Opportunities, Threats*

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 29: SWOT-matriisi kuvaa sisäisiä ja ulkoisia lopputulokseen vaikuttavia positiivisia ja negatiivisia tekijöitä.

suus työstää ratkaisuehdotuksia läpinäkyvästi rinnakkain, yhteensovittaa niitä joustavasti ja jalostaa nopealla syklillä eri versioista havainnollista materiaalia keskustelun pohjaksi ovat merkittäviä käytännön etuja.

Valmistelunaikainen ja kansallisella tasolla julkaistava sisältö ovat loogisesti osa samaa kaavoitusprosessin jatkumoa. Tarkoituksenmukainen mallin rakenne sulkee näiden tasojen välisen kuilun, vähentäen epätarkoituksenmukaista tiedon muuntamista (tulkintaa) kontekstien ja järjestelmien välillä. Esimerkiksi kansalliselta tasolta poimittu rakenteinen lähtöaineisto voidaan parhaassa tapauksessa viedä saumattomasti osaksi valmisteluaineistoa, ja syntynyt suunnitelma vuorostaan takaisin osaksi kansallista varantoa saman skeeman mukaisena.

Prosessin kaikilla tasoilla käyttökelpoinen yhtenäinen tietomalli tekee suunnitteluprosessin sisällöstä järjestelmäagnostista. Kun yhteisistä kohteista, niiden sisällöstä ja välisistä suhteista on sovittu tietomallin kautta, voivat osapuolet hyödyntää työhönsä haluamiaan työkaluja ainoastaan sillä rajoituksella, että niiden täytyy kyetä lukemaan ja kirjoittamaan tietomallin määrittelyn mukaista sisältöä.

Siinä missä tietomallien tuotanto ja käsittely nojaa nyt vahvasti yksittäisiin monoliittisiin ohjelmistoihin (mm. Microstation, QGIS, jne.), voidaan käyttötapauksesta riippuen sisältöä tuottaa ja käsitellä niiden lisäksi myös selainteknologioihin (CesiumJS[53], Leaflet[54], MapBox[55], jne.) pohjautuvilla työkaluilla, data-analyysiin soveltuvilla ympäristöillä (R[56], Python & Geopandas[57] jne.), tietokantatyökaluilla (Neo4j[58], AgensGraph[59] jne.) sekä prosessianalyysityökaluilla (Camunda stack[60]). Konkreettinen esimerkki on kunta, joka ohjaa GIS-

[53] `https://cesium.com/cesiumjs/`
[54] `https://leafletjs.com/`
[55] `https://www.mapbox.com/`
[56] `https://www.r-project.org/`
[57] `http://geopandas.org/`
[58] `https://neo4j.com/`
[59] `https://bitnine.net/agensgraph/`
[60] `https://camunda.com/`

OTSO.dev
consult@otso.dev
+358 50 544 9174

työpöytäsovellusta käyttävän konsulttikaavoittajan valmistelutyötä kevyen selainpohjaisen työkalun avulla.

Onnistunut tietomallimäärittely yhdistettynä suunnitteluhistoriaa ylläpitävään versionhallinnan ratkaisuun johtaa parhaimmillaan myös kaavoituksen lähtötietojen, suunnitteluratkaisujen ja päätöksenteon laadulliseen loikkaan. Tietomallimuotoisen työnaikaisen aineiston tekninen validointi ylläpitää sen johdonmukaisuutta, vapauttaen suunnittelijoiden kognitiivisia resursseja sisällöllisiin asioihin.

Vielä huomattavasti tätä merkittävämpi ero muodostuu käytettäessä tietomallia joka tukee semanttista (rakenteista ja kuvailevaa) tietoa suunnitelman sisällöstä. Esimerkiksi rakenteiset kaavamääräykset mahdollistavat määräysten keskinäisen ristiriidattomuuden varmistamisen teknisen validoinnin avulla.

Jatkuva vuorovaikutus käsitteistön sekä aineistojen välisten viittausten kanssa ohjaa tekemään näkyväksi suunnitelmaa koskevia johtopäätöksiä sekä ratkaisuja, mikä osaltaan parantaa suunnitteluargumentaatiota. Tietomallin laajentaminen myös muille kaavatasoille avaa lisäksi mahdollisuuden tasojen välisen ohjausvaikutuksen rakenteiselle määritykselle ja varmentamiselle.

Versiohallitun maankäytön suunnitteluprosessin aikana jatkuvasti syntyvä rakenteinen tieto tarjoaa organisaatioille ajantasaisen, tiheästi päivittyvän tilannekuvan suunnitelmien etenemisestä. Sen avulla voidaan tietomallin ja siihen liittyvän muun tiedon laajuudesta riippuen analysoida ja tunnistaa esimerkiksi prosessia koskevia pullonkauloja, tunnistaa syitä jotka johtavat tyypillisimmin poikkeamispäätöksiin, ja seurata esimerkiksi strategisen tason ohjauksen toteutumista suunnittelussa.

Yhteistyö ja oppiminen Rakennetun ympäristön toimintahaasteisiin vastaava kehitystyö tapahtuu nykyään usein kuntien itse rahoittamissa hankkeissa. Niistä useat ovat pilottiluonteisia projekteja, joiden budjetointi ei riitä tulosten jalostamiseen ja käyttöönottoon. Kehitetyt ratkaisut ovat lisäksi sidottuja kuntien omiin toimintaympäristöihin, jolloin tulosten siirtäminen toisaalle on käytännössä mahdotonta tai hyödytöntä. Vaikka jakaminen olisi mahdollista, ei siihen ole insentiivejä, sillä itse maksettua sisältöä ei yleensä haluta luovuttaa muille ilmaiseksi.

Kansallisen tietomallin, rekisterin sekä rajapintojen muodostama toimintaympäristö edesauttaa nykyisen pirstoutuneen kentän siilojen purkamisessa sekä kaikkia hyödyttävän yhteistyön rakentamisessa. Yhteisen toimintaympäristön puitteissa kehitystyötä voidaan tehdä WORE/COPE-mallien[61] mukaisesti, jolloin kaikki pääsevät hyötymään kerran tehdystä työstä.

Yksittäisen itsenäisen organisaation tapauksessa sisäisistä ongelmista kertominen nähdään usein puhtaasti negatiivisena ja leimaavana asiana.

Harmonisoidussa toimintaympäristössä organisaatioilla on enemmän insentiivejä jakaa ongelmiaan, havaintojaan ja ideoitaan toisille, sillä jaetun tietopohjan ansiosta on todennäköistä että ne ovat relevantteja myös muille osapuolille. Myös kehitysideoiden jaka-

[61] engl. *Write Once, Run Everywhere* sekä *Create Once, Publish Everwyhere*, ohjelmisto- ja julkaisualan käsitteitä, jotka kuvaavat sisältöä jota voidaan luomisen jälkeen hyödyntää monilla eri alustoilla (niiden rajoitteisiin automaattisesti mukautuen).

OTSO.dev
consult@otso.dev
+358 50 544 9174

minen muuttuu kannattavaksi, sillä niiden realisointiin löytyy tällöin todennäköisemmin intressejä (ja resursseja) laajemmalta joukolta.

Toimintaympäristön jaettu käsitteistö, mallit ja teknologia tekevät parhaimmillaan maankäytön suunnittelun ja hallinnan kokonaisuudesta kaikille osapuolille ymmärrettävämpää ja läpinäkyvämpää. Sen kautta kertynyt kokemus ja parantunut ymmärrys koko suunnittelujärjestelmän rakenteesta ja toiminnasta avaavat mahdollisuuksia käydä tärkeitä keskusteluja esimerkiksi kaavatasojen määrästä ja nykyisen hierarkian tarkoituksenmukaisuudesta.[62]

Liiketoiminta Toimintaympäristön yhteiset rajapinnat tekevät mahdolliseksi avoimen ja suljetun lähdekoodin ratkaisujen kehittämisen sekä yksittäisten toimijoiden tarpeisiin että kaikkien osapuolten käyttöön. Rajapinnat vähentävät myös toimittajaloukkujen syntymisen riskejä ja tervehdyttävät täten kilpailua.

Yhteisen rajapinnan ansiosta tieto siirtyy niin eri osapuolten kuin yhden osapuolen omienkin järjestelmien välillä riippumatta siitä, missä muodossa ja millä teknisillä ratkaisuilla itse sisältö on tallennettu. Tämän ansiosta esimerkiksi tiedonhallintajärjestelmän vaihtaminen (*migraatio*) voidaan toteuttaa ajamalla uutta järjestelmää vanhan rinnalla ja peilaamalla kaikki toiminnan aikana syntyvä tieto välittömästi siihen. Näin toimintaa voidaan vähitellen siirtää vanhasta järjestelmästä uuteen, ja siirtymän valmistuttua vanha järjestelmä ajaa heti alas.

Koko toimintaympäristön uudistuminen luo merkittävissä määrin uusia liiketoimintamahdollisuuksia, sillä kilpailu ei enää kohdistu vain toistuviin teknisen tiedonkäsittelyn toimeksiantoihin, kertakäyttöisiin selvityksiin ja asiakkaiden sitouttamiseen suljettuihin tiedon- ja prosessinhallinnan alustoihin. Yhtenäinen toimintaympäristö altistaa kilpailulle myös tällä hetkellä ”kalustukseen kuuluvat” toimisto-ohjelmistot sekä suunnittelu- ja visualisointityökalut.

Rakenteinen tieto ja rajapinnat antavat toimijoille mahdollisuuden kehittää sekä usean aihealueen tietoa luovasti yhdistäviä palveluja, että maankäytön suunnittelun erityistarpeita paremmin palvelevia työkaluja. Erityisen positiivisena voidaan pitää sitä, että helposti hyödynnettävä ja standardoitu tieto tekee alasta saavutettavamman uusille toimijoille. Niiden osallistumisen myötä syntyy lisää mahdollisuuksia vähentää manuaalista työtä, päällekäisiä tehtäviä sekä virheitä, ja parantaa toimijoiden välistä kommunikaatiota. Täten voidaan auttaa erityisesti tiukkojen taloudellisten reunaehtojen puitteissa toimivaa julkishallintoa vapauttamaan resursseja ja ohjaamaan niitä tarpeellisempiin kohteisiin.

4.3.2 Uhat

Muutoksen suuruus ja nopeus Kuntien toiminta ei nykyisellään ole rakentunut jatkuvan oppimisen ja uudistumisen varaan, joten uudistukset johtavat niissä erillisten muutosprosessien käynnistämiseen. Erityisesti mittakaavaltaan suuret tai pitkään kestävät muutosprosessit rasittavat kuntien operatiivista toimintaa merkittävästi.

[62] Huomautuksena lisättäköön, että keskustelun lähtökohtana ei voi olla tasojen määrän minimointi itseisarvona esimerkiksi toiminnan ”virtaviivaistamisen” varjolla, muttei myöskään säilyttäminen niistä riippuvaisten organisaatioiden aseman turvaamiseksi.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Tarpeellisten maankäytön suunnittelujärjestelmän uudistusten toteuttamiseen varattu aikaikkuna on suhteellisen lyhyt. Mikäli toteutettavien muutosten määrää karsitaan, kappaleessa 1.3 kuvatut tavoitteet vesittyvät herkästi.

Koska myöskään valtion ja ministeriöiden tasolla MRL-kokonaisuudistuksen kaltaiset muutosprosessit eivät etene asteittain pienissä osissa vaan suurina monoliittisina kokonaisuuksina, on tämän aikaikkunan sulkeuduttua epävarmaa, milloin tilannetta päästään seuraavan kerran kehittämään johdonmukaisesti koko valtakunnan mittakaavassa.

Käytössä olevan rekisterin jatkuvuudelle ja siitä saatavalle hyödylle uhan muodostaa tietojen päivittämiseen kohdistetut vaatimukset: vaikka laatutaso tulee pitää korkeana, saattaa sen täyttämisestä muodostua pullonkaula kaavaprosessien edistämiselle. Ylivoimaisiksi koetut ehdot johtavat herkästi vuorovaikutuksen kriisiytymiseen, sekä kansallisen alustan tarkoituksenmukaisuuden kyseenalaistamiseen.

Näkökulmien kapeus Uudistuksessa mukana olevien lukuisten intressiryhmien prioriteetit sekä näkökulmat ovat monilta osin eriäviä. Kriittiseksi riskitekijäksi muutosten toteuttamisen kannalta muodostuu tilanne, jossa:

1. kansallisen tason näkökulmasta itse rekisteriä pidetään itseisarvona, ja kuntien rooli katsotaan alisteiseksi ja velvoittavaksi, ja
2. kuntien näkökulmasta kansallista rekisteriä pidetään vain yhtenä toissijaisena velvoitteena muiden joukossa, kunnan itsenäisten järjestelmien ollessa etusijalla.

Lisäksi toisen uhan muodostaa kappaleessa 1.3.2 kuvattu järjestelmäntuottajien ja asiakkaana toimivien organisaatioiden välinen suhde. Vaatimusmäärittelyssä saatetaan jo alusta lähtien sitoutua ratkaisumalleihin jotka ovat alalla vallitsevia konventioita, ja itse kehitystyön ajatellaan olevan vain vakiintuneiden ratkaisujen implementointia.

Tämän riskin toteutumista lisää toimijoiden välillä ja toimijaorganisaatioiden sisällä vallitseva teknologinen kuilu (kappale 3.2; Sitowise, 2019; Taskinen, 2019a): onnistunutta tavoitetilaa ei voida saavuttaa mikäli teknisiä ratkaisuja viedään eteenpäin rationalisointiin vedoten samalla, kun osa toimijoista on huolissaan oman työnkuvansa puolesta sekä kontrollin menettämisestä työhönsä teknisen osaamisen puuttuessa.

Mahdollisena uhkana on myös tiedolla johtamiseen ja kvantifioituun tietoon liittyvä hybris: erityisesti tekoälyyn liittyvässä keskustelussa on julkisuudessa noussut viime vuosina toistuvasti esiin usko koneluettavan tiedon ja algoritmien objektiivisuuteen, sekä niiden tekemien päätösten absoluuttiseen paremmuuteen suhteessa ihmisiin.

Kaikki automatisoitu tietojenkäsittely on kuitenkin väistämättä aina sidoksissa ihmisen kognitioon ja vinoumiin, niin tietomalleissa (mallin pohjana olevan ontologian rakenne), tallennettavassa tiedossa (tallennettavaksi valitut kohteet, tallennustarkkuus sekä -muoto), kuin algoritmeissa (tiedon käsittelyyn valittu menetelmä). (Strickland, 2019)

4.3.3 Vahvuudet

Toimintaympäristön vahvuuksina voidaan pitää hyvää paikkatieto-osaamista, jo pitkään meneillään ollutta kehitystä julkisten toimijoiden tietojen avaamiseksi, kansallisella ta-

OTSO.dev
consult@otso.dev
+358 50 544 9174

solla jo toiminnassa olevia rajapinnoin yhdistettyjä rekistereitä ja palveluita (monet viranomaispalvelut, sekä Suomi.fi -tunnistautuminen), sekä meneillään olevaa kansallista sanasto- ja tietomallityötä.

Positiivisena voidaan myös pitää meneillään olevaa avointa keskustelua suunnittelujärjestelmän muutostarpeista ja toteutusmuodosta – positiivisten lisäksi erityisesti negatiivisista kommenteista tulisi keskustella laajalti, sillä niiden huomioiminen suunnittelussa ja toteutuksessa sekä vähentää muutosvastarintaa, että parantaa järjestelmän laatua.

Parhaimmillaan julkisen puolen toimijoiden valmiudet (sekä resurssit että toimintakulttuuri) versionhallinnan ja validoinnin käyttöönottoon ovat hyvät. Tämän aktiivisen joukon koko on suhteellisen pieni, mutta suuremman muutoksen saamiseksi sen tarvitsee ainoastaan toimia etujoukkona (engl. *early adopters*; E. M. Rogers ja Shoemaker, 1971) sillä oletuksella, että muutoksen leviämistä tuetaan esimerkiksi Kuntaliiton ja Ympäristöministeriön kaltaisten tahojen avulla pitkäjänteisesti.

4.3.4 Heikkoudet

Alan toimijoiden ammattiroolit ovat vielä pitkälti siiloutuneita, ja niiden työnkuvat ovat usein melko hitaasti uudistuvia – jatkuva uudelleenkouluttautuminen ja omien osaamisalueiden päivittäminen eivät yksinkertaisesti vielä ole arkipäivää, eikä organisaatioiden toimintaympäristö tarjoa sellaiseen johdonmukaista tukea.

Historiallisista syistä hajanaisena alkanut tietojärjestelmien kehitys on johtanut järjestelmien merkittävään fragmentaatioon, eikä niiden yhtenäistämisen esteenä ole puhtaasti pelkästään migraation tekninen toteutus. Haasteena on muuttaa toimintakulttuuria siten, että järjestelmiä ei enää tilata vain omaan käyttöön puhtaasti itse tehdyn tarvemäärittelyn pohjalta, eivätkä nykyiset upotetut kustannukset (engl. *sunken cost fallacy*) muodosta päätöksentekoon estettä toimintaa heikosti palvelevien ratkaisujen hylkäämiselle.

Tätä voidaan pitää huomattavan vaikeana, sillä monien toimijoiden sisäiset prosessit on sidottu järjestelmiin, joiden vaihtaminen vuorostaan on sopimuksellisten sitoumusten sekä teknisten riippuvuuksien vuoksi erittäin haastavaa.

Monien julkisten toimijoiden taloudellinen tilanne ylipäätänsä muodostaa merkittävän heikkouden uudistusten toteuttamiselle. Merkittävä osa Suomen kunnista on tilanteessa, jossa omat resurssit, voimavarat sekä sisäinen osaaminen eivät riitä toiminnan ja teknologian uudistamiseen ja vakiinnuttamiseen käyttöön.

Yllä mainittuihin ongelmiin on monesti esitetty ratkaisuksi avoimen koodin (engl. FOSS[63]) järjestelmiä tai tuotteita. Avoimen koodin ratkaisujen laatu ei nykyään muodosta ongelmaa käyttöönotolle – suurin osa kriittisistä ja kaupallisista järjestelmistä on rakennettu avoimien komponenttien varaan – mutta ohjelmiston ilmaisuudesta huolimatta sen sovittaminen käyttötapauksiin, käyttöönotto sekä ylläpito eivät ole ilmaisia.

Monien potentiaalisten järjestelmien kohdalla (esimerkiksi Oskari-alusta[64] resursointi on melko pientä, eikä Suomessa ole vielä vakiintunutta toimintakulttuuria ja mekanismeja useiden toimijoiden resurssien keskittämiseen (engl. *pooling*) tiettyihin kehityskohteisiin.

[63] engl. Free and Open-Source Software
[64] `https://verkosto.oskari.org/`

OTSO.dev
consult@otso.dev
+358 50 544 9174

5 Suositukset

Näissä suosituksissa hahmotellaan sekä askelia eteenpäin, että pidemmälle vietyjä ideoita siitä, mikä kaavatietomallin rooli maankäytön suunnittelussa voi versionhallinnan ja validoinnin kanssa hyödynnettynä olla.

5.1 Suositus: Yleinen spatiaalinen sisältöluokka

Selvityksessä on tunnistettu tarve yleiselle avaruudellista (spatiaalista) sisältöä kuvaavalle luokalle, joka kattaa sekä suunnittelusisällön fyysiset että aineettomat kohteet. Koska emme tiedä, millä aikataululla kaavoituksessa tullaan mahdollisesti siirtymään täysin kolmiulotteiseen sisältöön – saati minkä muun tyyppistä spatiaalista sisältöä tulevaisuuden kaavoihin tarvitaan – on olennaista että tietomallin määrittely ei epätarkoituksenmukaisesti sulje pois tai hankaloita mahdollisia kehityspolkuja.

Tämän vuoksi spatiaalisen luokan tulee minimissään tukea yleisimpiä (esimerkiksi GML-standardissa määriteltyjä) geometrisia primitiivejä, ja se tulisi toteuttaa laajennettavana säilönä tulevia käyttötarpeita silmälläpitäen. Mahdollisia tulevaisuudessa hyödynnettäviä muotoja voivat olla esimerkiksi:

1. pistepilviaineistot,
2. vapaamuotoiset parametriset kaaret ja pinnat[65], tai
3. volumetrinen aineisto[66].

Tietomallissa ei tulisi tehdä tiukkaa poissulkevaa jakoa erilaisiin tyyppeihin: esimerkiksi kaksi- ja kolmiulotteista tonttia ei tulisi kohdella luokkamäärittelyssä lähtökohtaisesti täysin erilaisina kohteina.

Sen sijaan kaikkea spatiaalista aineistoa tulisi käsitellä joukko-opillisina kokonaisuuksina: esimerkiksi alun perin vain kaksiulotteisena piirrettyä kohdetta tulisi pitää leikkauksena (osajoukkona) vielä määrittelemättömästä kolmiulotteisesta kohteesta, joka voidaan luoda tarpeen mukaan myöhemmin[67]. Määriteltäessä tontin kolmiulotteista muotoa luokan kohteen määrittelyä laajennetaan, eikä sitä tarvitse korvata kokonaan uudella eri tyyppisellä kohteella.

Tällöin kohteita voidaan käsitellä kontekstisidonnaisesti nk. *duck typingin* avulla, ts. kohteen tyyppi päätellään sen ominaisuuksien kautta. Tätä periaatetta noudattaen rikkaammin kuvailtu kohde voi käyttötapauksesta riippuen toimia toisen kohteen sijaisena taaksepäin yhteensopivalla tavalla.

[65] NURBS; *Non-uniform rational B-spline*, lineaariseen interpolointiin pohjautuva menetelmä monimutkaisten kaarevien polkujen ja pintojen esittämiseen

[66] engl. Voxel; *Volumetric Pixel*, yhtenäisistä osista koottuja tilavuuskappaleita. Aineisto voi käsittää esimerkiksi volumetrisia luonnoksia (nk. *grease pencil sketch*). Käsite kattaa myös nk. massakappaleet (engl. *constructive solid geometry*), joita käytetään mm. rakennusalan BIM-malleissa. Voidaan ajatella kattavan myös muita matemaattisesti määriteltyjä muotoja kuten avaruuskulmat.

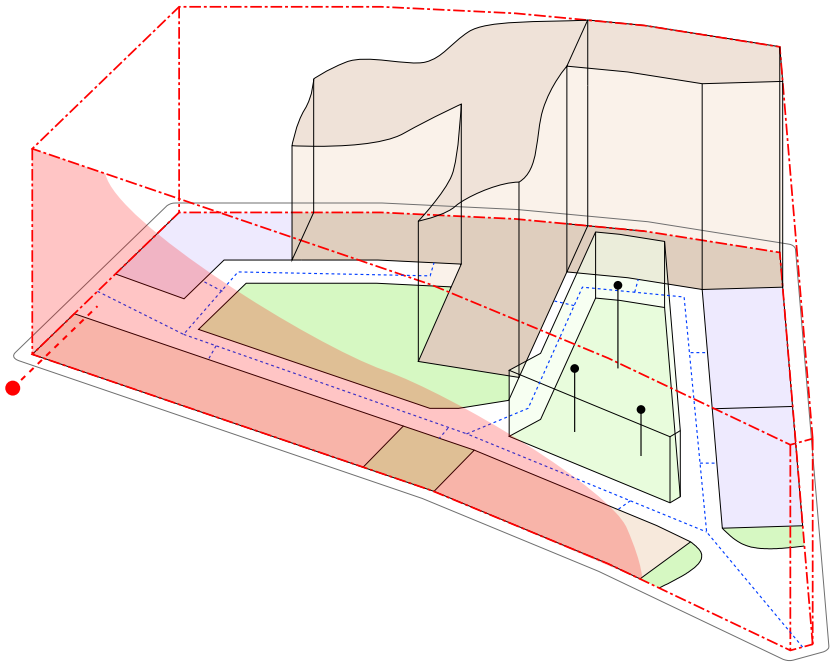
[67] Yleistettynä kolmiulotteiset volyymit sisältävät äärettömän määrän eri muotoisia kaksiulotteisia leikkauksia, ne taas erilaisia polkuja ja polut pisteitä. Tästä johdettuna seuraa myös mm. että jokainen volyymi koostuu äärettömästä määrästä pisteitä ja sisältää äärettömän määrän polkuja.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Esimerkiksi kolmiulotteisen kaavakohteen vaipan avulla määritelty rakennusoikeus kuvautuu kaksiulotteiselle kohteelle numeerisena arvona (m^3), ja kolmiulotteisen vaipan muoto projisoituu maan pinnalle kaksiulotteiseksi kohteen piiriksi. Vain kaksiulotteista aineistoa tukevalle ohjelmistolle sisältö näkyy kyseisessä redusoidussa muodossa, ei puuttuvana (tukemattomana) kohteena.

Kuvausten määrä ja laatu riippuu kohteesta: esimerkiksi pistemäisen kohteen tapauksessa ei ole mielekästä puhua pohjan pinta-alasta. Duck typingin periaatteena onkin kohdella kohteita yhtäläisinä kun ne kuvailevat ominaisuuksiaan yhtäläisesti: lähtökohtaisesti esimerkiksi kaikkia pohjan pinta-alan raportoivia kohteita voidaan tarkastella kaksiulotteisina kohteina kartalla.

Joukko-opillisen rakenteen vuoksi jokainen spatiaalisen luokan kohde voi sisältää $[0, n], n \geq 0$ spatiaalisen luokan kohdetta (osajoukkoa)[68]. Jokaisessa kaavassa suurimman yhtenäisen kohteen muodostaa kaava-alueen vaippa, joka sulkee sisäänsä kaikki muut volyymit, pinnat, polut ja pisteet. Tämän rakenteen ansiosta voidaan aina yksikäsitteisesti määritellä kohteiden hierarkkinen tai rinnakkainen suhde toisiinsa sekä topologisesti että metrisesti.



Kuva 30: Hypoteettinen rakenteinen kaavaluonnos joka koostuu monimuotoisesta spatiaalisesta sisällöstä

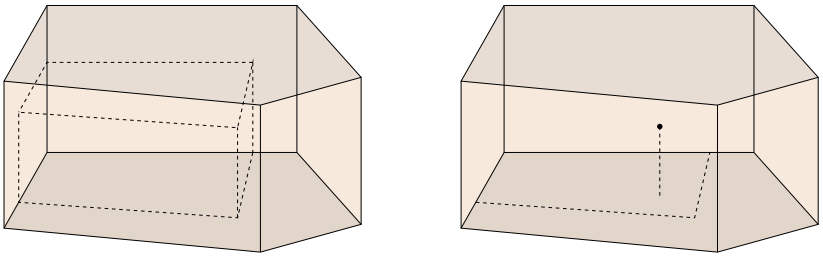
Kuvassa 30 on esimerkki monimuotoista spatiaalista aineistoa sisältävästä kaavaluonnoksesta. Kaava-alueen vaippa on merkitty punaisella pistekatkoviivalla. Sen osajoukkoina on kaksi rakentamista ja yksi viherrakentamista säätelevää volyymia, niiden pohjien alueet sekä muita volyymeiltaan määrittelemättömiä alueita. Kuvassa on myös esitetty sinisellä katkoviivalla kulkuyhteydet topologisena verkkona, sekä yksi kaavan ulkopuolelle si-

[68] Ainoana rajoitteena on, että osajoukkojen ulottuvuus ei ole itse kohdetta suurempi, esimerkiksi pistemäinen kohde ei voi sisältää alueita, mutta alueelle voidaan määritellä osajoukoiksi sen pinnalla sijaitsevia pisteitä.

OTSO.dev
consult@otso.dev
+358 50 544 9174

joittuva pistemäinen kohde, jonka huomioitava vaikutus projisoituu osajoukkona kaava-alueen yhdelle pinnalle.

Kaikki kuvan kohteet voidaan jakaa edelleen pienempiin käsiteltäviin osajoukkoihin tarpeen mukaan lukuunottamatta yksittäisiä pistemäisiä kohteita jotka ovat atomisia (eivät jaettavissa; yksittäisellä pisteellä ei ole aitoja osajoukkoja).



Kuva 31: Esimerkki rakennusalan rakennusoikeuden ylä- sekä alarajojen ja sijainnin ehtojen määrittämisestä spatiaalisesti

Osajoukkoja voidaan myös käyttää erinäisten rajoitteiden ilmaisuun (kts. kuva 31). Esimerkiksi rakennusalan rajoja kuvaavan volyymin sisään voidaan liittää osajoukko, joka ilmaisee käytettävän rakennusoikeuden alarajan sekä rakennusvolyymin, jonka minimissään tulee täyttyä. Minimi voidaan myös ilmaista numeerisena attribuuttina, ja kiinnittää esimerkiksi pohjan alaan tai yksittäiseen pisteeseen jonka täytyy sijaita käytetyn rakennusoikeuden sisällä.

Yhdenmukaisen rakenteen lisäksi johdonmukaisessa toteutuksessa olennaista on se, että spatiaalisiin kohteisiin ei kuulu semanttista sisältöä – kyseinen tehtävä kuuluu suosituksessa 5.2 kuvatulle luokalle. Tällä tarkoitetaan esimerkiksi sitä, että itse spatiaalinen volyymi sisältää vain attribuutin *maksimitilavuus*, jolle annetaan tilannekohtaisia merkityksiä erillisten semanttisten viittausten kautta.

5.2 Suositus: Yleinen semanttinen sisältöluokka

Suositukseen 5.1 viitaten selvityksessä on tunnistettu tarve yleiselle spatiaalisen sisällön merkitystä kuvaavalle luokalle.

Luokan kohteiden tehtävänä on toimia säilönä kaavamääräyksille, lähtötiedoille, mielipiteille, lausunnoille, kaavaselostuksen argumentaatiolle sekä muulle kaavoitusta ohjaavalle ja kaavan rakennetta selittävälle sisällölle.

Yksinkertaisimmassa tapauksessa luokka edustaa yksikäsitteistä numeerista rajoitetta, joka kohdistuu tiettyyn spatiaaliseen kohteeseen. Esimerkiksi tietyn kohteen rakennusoikeutta voidaan säädellä tekstillä ”Kohteeseen sisältyvästä rakennusoikeudesta tulee käyttää vähintään ro_{min} m^3 ja enintään ro_{max} m^3”, jossa sana ”Kohteeseen” viittaa tietyn spatiaalisen kohteen versiotunnisteeseen, ”rakennusoikeudesta” sanastossa kuvailtuun rakennusoikeuden ontologiseen käsitteeseen ja loppu lauseesta ehtoon $ro_{min} \leq kohde_{ro} \leq ro_{max}$. Ihmisluettava tekstin esitys sisältää siis rakenteisena (esimerkiksi RDF/XML-kielellä) ilmaistun formaalin koneluettavan ehdon.

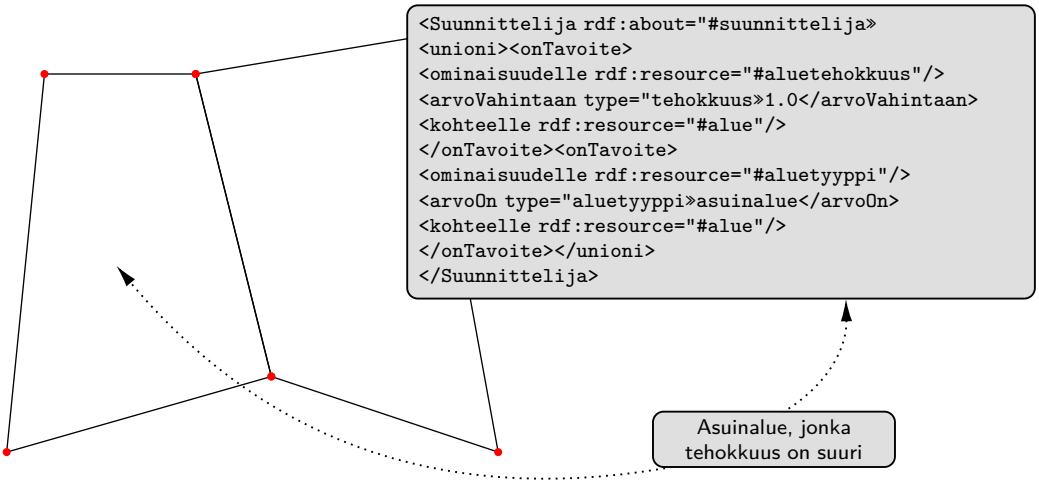
OTSO.dev
consult@otso.dev
+358 50 544 9174

Yllä kuvatun tapaisten yksikäsitteisten ehtojen lisäksi on olennaista, että myös kvalitatiivista kuvailevaa sisältöä voidaan jossain määrin tallentaa koneluettavassa muodossa. Yksinkertaisimmillaan se voidaan toteuttaa linkittämällä tiettyjä ihmisluettavan esitysmuodon termejä koodilistoihin, sanastoihin[69], vapaamuotoisiin relevanttien aihealueiden ontologioihin[70] sekä ulkoisten lähteenä toimivien aineistojen versiotunnisteisiin. Tällä tavoin tallennettua aineistoa voidaan myös analysoida kieliteknologisilla työkaluilla (engl. NLP; *Natural Language Processing*).

Luokkaan tulisi potentiaalisesti voida sisällyttää tekstisisällön oheen myös esimerkiksi:

1. tiettyihin projektioihin liitettyjä rasteri- tai vektorimuotoisia piirroksia (tietyn näkymän päälle sijoitettava graafinen aineisto, esimerkiksi karttaprojektion päälle luonnosteltu kortterelirakenne),
2. georeferoituja, tiettyyn sijaintiin, kameran asentoon ja polttoväliin yhdistettyjä valokuvia (mahdollistaa kolmiulotteisen kaavaympäristön mallin ja vastaavasta paikasta otetun valokuvan vertaamisen samassa näkymässä),
3. georeferoitua videoaineistoa, tai
4. ääntä (esimerkiksi annotoidut keskustelutilaisuudet).

5.3 Suositus: Ilmaisuvoimainen tietomalli



Kuva 32: Esimerkki RDF/XML-muotoisesta rakenteisesta kaavamääräyksestä ihmis- ja koneluettavassa esitysmuodossa, jotka viittaavat kohteena olevaan spatiaaliseen kaavaobjektiin.

Suositusten 5.1 sekä 5.2 hahmottelemiin rakenteisiin viitaten, kaavatietomallin tulisi olla mahdollisimman ilmaisuvoimainen ja muutosjoustava.

Tietomallin rakenteen tulee pohjautua abstraktiotasoon, joka on yhtä askelta nykyisen lainsäädännön ja toimintaympäristön vakiintuneita luokkia, käsitteitä ja keskinäsiä suhteita ylempänä. Tavoitteena on määritellä mallille pysyvä rakenne jota ei jouduta päivittä-

[69]Esimerkiksi Yhteentoimiva Suomi -sanasto; `https://yhteentoimiva.suomi.fi/fi/sanastot`
[70]Kts. esimerkkinä vapaamuotoisesta ontologiasta FOAF (`http://www.foaf-project.org/`)

OTSO.dev
consult@otso.dev
+358 50 544 9174

mään toimintaympäristön muuttuessa, vaan tarvittava mukauttaminen voidaan toteuttaa mallista irrallisten tilannekohtaisten rajoitteiden (*constraint*) kautta.

Abstrahointi avaa mahdollisuuden laajentaa saman tietomallin käyttöä sekä eriasteisen tiedon (luonnossuunnittelusisältö, lainvoiman saanut sisältö ym.) tallentamiseen, että eri käyttökohteisiin (asemakaava, yleiskaava, maakuntakaava ym.).

Tämän mahdollistamiseksi mallissa tulisi pyrkiä minimoimaan esimerkiksi vain yhtä käyttötarkoitusta varten varattujen luokkien määrä[71], sekä niiden välille luodut kiinteät (esimerkiksi hierarkkiset) riippuvuussuhteet.

Tarkoituksenmukaisen abstraktiotason avulla rakenteesta saadaan laajennettava ja samalla taaksepäin yhteensopiva. Sopivaa ratkaisua tulisi minimissään hakea ainakin helposti ennakoitavien tulevaisuuden käyttötarpeiden kautta.

Kiinteän ja dynaamisen rakenteen eriytys Mallin muuttuviin osiin sisältyvien rajoitteiden (esimerkiksi käyttötarkoitusalueeseen liittyvät ominaisuudet ja sen hierarkkien suhde muihin kaavan alueisiin) määrittämiseen tulisi hyödyntää pelkkien XSD:n sallimien rajoitteiden sijaan esimerkiksi Schematron-kieltä.[72]

```
<SpatiaalinenKohde>
  <Tyyppi codeURI = "https://suomi.fi/codelist/Korttelialue:1">
    Korttelialue</Tyyppi>
    <SpatiaalinenKohde>
    <Tyyppi codeURI = "https://suomi.fi/codelist/Rakennusala:5">
    Rakennusala</Tyyppi>
    </SpatiaalinenKohde>
</SpatiaalinenKohde>
```

Listaus 8: Hyvin yksinkertainen ja abstrakti esimerkki saman luokan käytöstä. Rakennusala on hierarkkisessa suhteessa (sisältyy) korttelialueeseen, rajoitteen validointi tapahtuu koodilistan tyyppien ja ulkoisen luokkien suhteita ja niihin liittyviä rajoitteita kuvaavan rakenteen kautta.

Listauksessa 8 on yksinkertainen esimerkki tietomallirakenteesta, jossa ainoa skeemasta löytyvä kiinteä alueita kuvaava luokka on SpatiaalinenKohde. Tiettyjen alueiden esiintyminen suunnitelmassa on toteutettu tyyppielementillä, joka viittaa ulkoiseen voimassaolevia aluetyyppejä kuvaavaan koodilistaan.

Tämän tyyppisen toteutuksen seurauksena pysyvä tiedon tallennukseen käytettävä XSD-skeema on suhteellisen yksinkertainen. Alueiden sallittujen tyyppien sekä keskinäisen hierarkian tarkistamiseen käytetään tällöin tietyllä ajanhetkellä voimassa olevaa (esimerkiksi juuri Schematronilla kirjoitettua) säännöstöä. Säännöstö on siis kiinteästä skeemasta irrallinen oma skeemansa, jossa *korttelialueen* ja *rakennusalan* kaltaisia käsitteitä ja niiden välistä hierarkiaa voidaan päivittää versiohallitusti puuttumatta varsinaisen tietomallin skeemaan.

[71] Erillisiä luokkia ei tulisi perustaa nyt käytössä oleviin käsitteellisiin jakoihin pohjautuen. Esimerkiksi korttelі, käyttötarkoitusalue, yleinen alue, rakennusala ja tontti ovat pohjimmiltaan saman spatiaalisen tyypin erilaisia ilmentymiä, eivät eri luokkia.

[72] Schematron on ISO-standardoitu XML-pohjainen kieli XML-sisällön validointiin tiettyjen sääntöjen puitteissa. Se on huomattavasti XSD:tä ilmaisuvoimaisempi, muttei kuitenkaan kaikenkattava. Tämän vuoksi Schematronin ohella tulisi tarkastella muita helposti implementoitavia vaihtoehtoja tehtävään.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Yksinkertaisen pysyvän mallin ansiosta tiedonhallinnan järjestelmien puolella ei jouduta ympäristön muuttuessa tekemään raskaita sisällön kuvauksia vanhan fyysisen skeeman rakenteesta uuteen[73].

Lisäksi skeeman varaan rakennetuilta ohjelmistoilta ei suosituksen mukaisen toteutuksen tapauksessa vaadita välittömiä päivityksiä mallin sisällöllisen rakenteen muuttuessa, sillä niiden näkökulmasta rakenne säilyy loogisesti identtisenä.

Koodilistat Edellä mainitusta syystä malliin ei tulisi myöskään upottaa kiinteitä koodilistoja (kuten on tehty KuntaGML-, PlanDK- sekä SOSI-malleissa, kts. listaus 9), vaan ne tulisi säilyttää muualla, ja niiden käytön tulisi tapahtua viittausten kautta. Toteutukseen voidaan ottaa esimerkkiä UMCLVV/GeneriCode[74]-mallista.

```
<xs:simpleType name="RakennusluokkaType">
    <xs:restriction base="xs:string">
    <xs:enumeration value="0110 omakotitalot"/>
    <xs:enumeration value="0111 paritalot"/>
    <xs:enumeration value="0112 rivitalot"/>
    <xs:enumeration value="0120 pienkerrostalot"/>
    ...
```

Listaus 9: Ote KuntaGML -skeeman luetellusta tyypistä joka toimittaa koodilistan virkaa

Koodilistojen tulisi olla skeeman toteutuksessa ulkoistettuja samasta syystä; listojen hyödyllisyys käytössä on riippuvainen kyvystä päivittää niitä muuttuvien tarpeiden mukaan (Kumar, 2007).

Rakenteiset kaavamääräykset Kaavamääräysten kohdalla joustoa mahdollisten tulevien tarpeiden ja mahdollisuuksien hyödyntämiselle tulisi lisätä luopumalla tietomalliluonnoksissa luokkien avulla toteutetuista luetelluista tyypeistä (*enumeration*) ja rajoitteista (*constraint*).

Samalla tulee luopua nykyisen kaltaisen täysin rakenteettoman kaavamääräystekstin tukemisesta ”varaventtiilinä”, sillä vanhentuneiksi ilmoitettujen mutta edelleen tuettavien ominaisuuksien (engl. *deprecated features*) tarjomainen ylläpitää ja saattaa jopa kasvattaa niihin kohdistuvia riippuvuuksia.

Kaavamääräysten rakenteistaminen ei rajoita niiden ilmaisuvoimaa, mutta tekee sisällöstä yksikäsitteistä ja ohjaa kirjoittamaan sisällön selkeässä ja johdonmukaisessa muodossa.

Rakenteinen muoto mahdollistaa kaavamääräyksen kirjoittamisen toteavan ehdon sijaan sääntömuotoisena.[75] Tällöin voidaan esimerkiksi muodostaa kaavamääräyksien keskinäisiä riippuvuussuhteita ja luoda määräyksiä joiden voimassaolo tai vaikutus riippuu toisista määräyksistä.

[73] Tästä operaatiosta käytetään nimitystä *schema migration* tai *database change management*

[74] engl. *UBL Methodology for Code List Value and Validation*, OASIS (*Organization for the Advancement of Structured Information Standards*) on standardeja kehittävä järjestö. UBL (engl. *Universal Business Language*) on XML-muotoinen organisaatiotoiminnan malli.

[75] Sääntöpohjaista kaavamääräysmallia tutkitaan ja kehitetään mm. Hollannissa.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Ehdolliset kaavamääräykset antavat kaavoittajalle uuden työkalun, jolla alueelle tehtävää suunnittelua voidaan ohjata kannustimin ja rajoittein jotka eivät kuitenkaan määrittele suunnitteluratkaisuja suoraan. Erittäin yksinkertaisena esimerkkinä kaavamääräys voi esimerkiksi sallia tietylle AK-alueelle yhden lisäkerroksen rakentamisen jokaista tietyn rajan ylittävää $500m^2$ virkistysalueen lisäystä kohden joka täyttää tietyt vaatimukset.

Ehtojen rakentaminen suorien rajoitteiden sijaan on kuitenkin haastavampaa, sillä sen onnistunut toteutus vaatii määräystä luovalta taholta taitoa muodostaa formaaleja lauserakenteita (esim. ”jos A niin B”). Lisäksi ehdolliset säännöt muodostavat mahdollisesti hyvinkin monimutkaisen nk. *hakuavaruuden* (yhdistelmä kaikkien sääntöjen kaikista mahdollisista tiloista), jonka mahdollistamat ratkaisuvaihtoehdot tulee pystyä varmentamaan jotta voidaan varmistua siitä, että kaava ei sisällä epätarkoituksenmukaisia ”porsaanreikiä”.

Kaavamääräyksiä koskeva laadullinen ja määrällinen sisältö voidaan kuvata esimerkiksi RDF/XML -muodossa, jolloin rakenne voi sisältää sekä ihmisluettavan osuuden (kuvaileva teksti), että siihen upotetun koneluettavan semanttisen rakenteen. Mahdollisia toteutustapoja voidaan hakea OWL[76]-standardin kautta, eikä toteutuksen välttämättä täydy olla XML-muotoinen, sillä määräyksen sisältö voidaan upottaa kaavamääräyselementtiin.

Muun kuin RDF/XML -pohjaisen ratkaisun käyttäminen vaatii kuitenkin yhden askeleen lisäämistä tekniseen validiontiin (kts. suositus 2.3.3), sillä kaavamääräyselementtien sisältö joudutaan tulkitsemaan kuvauskielen omalla parserilla sekä varmistamaan, että siinä käytetyt viittaukset elementtien tunnisteisiin ovat valideja.

Kaavamääräysten rakenteistamisen tulee ulottua myös mm. niiden kykyyn kuvailla spatiaalisia kohteita tarkemmin Norjan SOSI-mallista löytyvien juridisten kohteiden tapaan.

Esimerkiksi maankäyttöalueiden (tai volyymien) pintoihin, reunoihin ja pisteisiin tulee voida viitata kuvailevassa tekstissä, sillä määräys voi olla tarkoituksenmukaista kohdistaa esimerkiksi tiettyyn alueen sivuun (rakennuksen etäisyys tontin reunasta), pintaan (korttelijulkisivua koskeva melurajoite) tai pisteeseen (vähimmäisetäisyys tietystä kohteesta).

```
<gml:MultiSurface>
  <gml:surfaceMember>
    <gml:Polygon>
      <gml:exterior>
        <gml:Ring>
          <gml:curveMember>
            <gml:Curve>
              <gml:segments>
                <gml:LineStringSegment>
                  <gml:posList count="58">
                  429568.2033 7199568.5831
                  ...
                  429594.8506 7199577.126
                  </gml:posList>
                </gml:LineStringSegment>
              </gml:segments>
            </gml:Curve>
          </gml:curveMember>
        </gml:Ring>
      </gml:exterior>
    </gml:Polygon>
  </gml:surfaceMember>
```

[76]engl. *Web Ontology Language*

OTSO.dev
consult@otso.dev
+358 50 544 9174

```
23 </gml:MultiSurface>
```

Listaus 10: Ote Kuntapilotin skeeman mukaisen kaavan sisältämän alueen rakenteesta

Kuten listauksesta 10 nähdään, Kuntapilotissa luonnosteltu tietomallin sovellusskeema on hierarkialtaan syvä ja mahdollistaa moninaisten spatiaalisten kohteiden kuvailun. Suunnittelun kannalta mielekkäiden viittauksien mahdolistamiseksi tulee geometrialle luoda objektin pysyvästä tunnisteesta riippuva nimiavaruus, sekä mielekkäiksi katsotuille geometrian osille antaa nimiavaruuden sisäiset omat uniikit tunnisteet.

Jotta esimerkkilistauksessa voitaisiin viitata yhteen murtoviivasegmenttiin, tulee gml:posList-elementille asettaa rajoitteeksi kaksi arvoa jotka ovat gml:pointProperty-elementin mukaisia viittauksia muualla säilytettävään kohteen pistejoukkoon (näin vältytään siltä että vierekkäisten segmenttien yhteinen piste kuvataan kahtena erillisenä koordinaatiparina viittauksen sijaan).

Vaikka suosituksessa esiteltyihin sääntöpohjaisiin kaavoihin ei siirryttäisi lainkaan, mahdollistaa kuvailevan tekstisisällön yhdistäminen kvantitatiivisiin arvoihin (esimerkiksi kerrosalaa koskeva rajoite) niitä koskevan teknisen validoinnin.[77]

Aikasarjasisältö Selvityksen puitteissa on tunnistettu mahdollinen tarve tukea aikaan sidottua sisältöä kaavamallissa.

Mahdollisia käyttötapauksia ovat esimerkiksi alueen toteutukseen liittyvän vaiheistuksen tallentaminen osaksi kaavan sisältöä, ja tuki kaavaan liitettävälle rakenteiselle selvitysaineistolle. Ajan funktiona muuttuvan spatiaalisen sisällön tukemiseen on saatavilla mm. tuore OGC:n hyväksymä nk. *Moving Features* -standardi[78].

Luonnossuunnittelu ja lausunnot Selvityksen puitteissa on tunnistettu potentiaalisia käyttökohteita laajakirjoisemman suunnittelusisällön integroimiseksi tietomalliin.

2018–2019 välillä toteutetussa KIRA-digi -kokeiluhankkeessa havaittiin suunnittelijoiden hyödyntävän sekä kaavan suunnitteluratkaisuja kehittäessään, muistiinpanoja tehdessään että osapuolten välillä kommunikoidessaan metodeja, jotka ovat luonteeltaan hyvin lähellä tässä luvussa kuvattua jakoa semanttisiin ja spatiaalisiin kaavakohteisiin. (Helenius, 2019)

Suunnittelijat tuottavat erityisesti luonnossuunnitteluvaiheessa temaattisia tai kaaviokarttoja tietyistä ratkaisuista, joiden oheen sijoitetaan tekstimuotoista sisältöä nuoliviittauksin. Luonnossuunnittelun kaltaista kaaviokarttojen ja ratkaisuvaihtoehtojen tuottamista harjoittavat kaavoittajien lisäksi vaihtelevissa määrin myös mm. liikennesuunnittelijat.

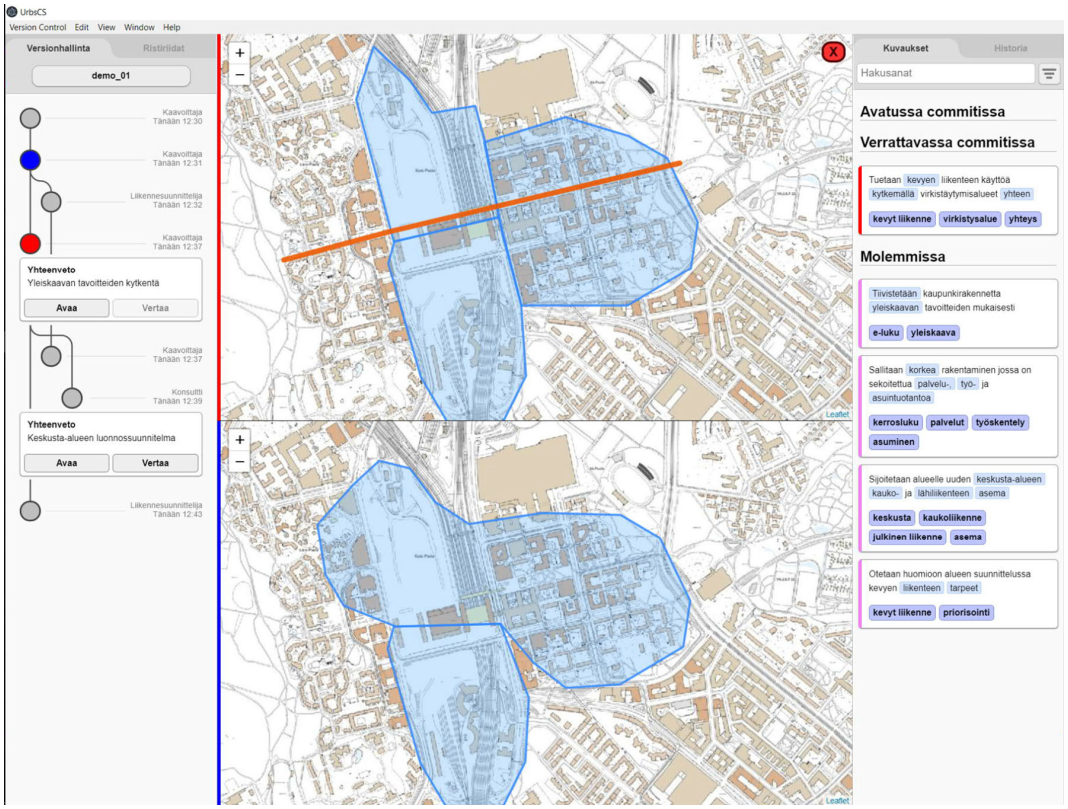
Kokeiluhankkeessa tuotettiin yksinkertainen ohjelmistoprototyyppi (kuva 33), jolla luonnosmaisia kaaviokarttoja on mahdollista luoda, verrata sekä yhdistää versionhallinnan avulla.

Alueisiin on prototyypissa mahdollista liittää kaavamääräyksiä muistuttavia tekstejä, joiden sisältö koostuu tekstin lisäksi sitä kuvailevista aihesanoista.

[77] Tarve rakenteiselle kaavamääräyssisällölle on noussut esiin myös Kuntapilotin loppuraportissa (Sitowise et al., 2019).

[78] `https://www.opengeospatial.org/standards/movingfeatures`

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 33: KIRA-digi -kokeiluhankkeen versiohallittuun luonnossuunitteluun kehitetty prototyyppiohjelmisto

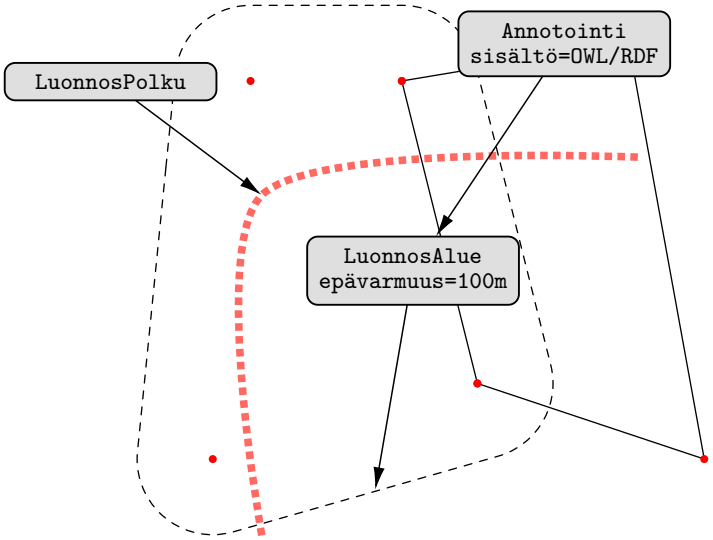
Tämä alun perin luonnossuunnittelun tarpeisiin kehitetty työkalu voidaan sopivilla tietomallin muutoksilla laajentaa tukemaan myös esimerkiksi kannanottojen, mielipiteiden, muistutusten, lausuntojen keräämistä ja liittämistä tietyn kaavaversion tiettyihin kohteisiin.

Tyypillisesti näiden aineistojen tuottaminen ja käsittely tapahtuu irrallaan itse suunnitteluprosessin sisäisestä aineistosta, jonka pohjalta ne ovat syntyneet ja johon niissä viitataan.

Mahdollisuus annotoida esimerkiksi muistutuksen sisältöä viittaamalla suoraan tietyn version sisältämiin kohteisiin kytkee sen semanttisesti osaksi työnaikaista aineistoa ja mahdollistaa sen johdonmukaisen huomioimisen suunnitelman laadullisessa validoinnissa.

Kuvassa 34 on esitetty mahdollinen spatiaalisen ja semanttisen luokan laajennus luonnosmaiselle kaavasisällön kommentoinnille: eksaktin alueen sijaan kaavaan voidaan lisätä myös varmuusasteeltaan eri laatuisia kohteita, sekä niitä kuvaavia rakenteisia kommentteja tai sisällön kuvauksia.

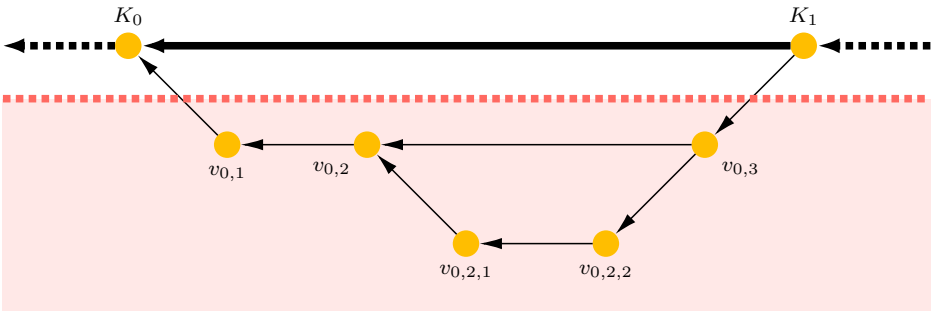
OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 34: Esimerkki tietomallirakenteesta, joka sallii epätarkkojen luonnosmaisten kohteiden käsittelyn sekä suunnittelun tavoitteiden, reunaehtojen tai muun sisällöltään vapaan kuvailevan sisällön liittämisen spatiaalisiin kohteisiin.

5.4 Suositus: Integroitu versionhallinta

Kaavaprosessien sisältämän tiedon elinkaarta sekä suunnitelman muutoksia tulee hallita integroidusti. Tämä tavoite vastaa mm. TUMA-hankkeen ”Ota kantaa” -kyselyssä esitettyihin toiveisiin: 87% alan ammattilaisista pitivät tärkeänä, että *Maankäytön suunnitteluun ja rakentamiseen liittyvä tieto on ajantasaista, ymmärrettävää ja helposti saatavilla*, ja 67% että ”Viranomaisten ja muiden toimijoiden välinen vuorovaikutus on sujuvaa”.



Kuva 35: Yksinkertainen havainnekuva läpi suunnitteluprosessin (sekä sen jälkeen) jatkuvasta versionhallinnasta. K_0 ja K_1 edustavat kansalliseen rekisteriin tallennettuja julkisia versioita, punaisen viivan alapuoliset suunnitteluorganisaatiossa jalostettuja ja rajatulle toimijoiden joukolle näkyviä versioita.

Integroidulla versionhallinnalla tarkoitetaan sitä, että jokainen kaavoituksen aikana syntyvä muutoskokonaisuus (yhteen tai useampaan tietoon yhtäaikaisesti kohdistunut muutos) täyttää seuraavat ehdot:

1. Muutoksesta on mahdollista päätellä sen ajallinen järjestys suhteessa kaikkeen muuhun prosessissa jo olevaan tietoon (muutokset muodostavat yhtenäisen ketjun, kts. kappale 2.2).

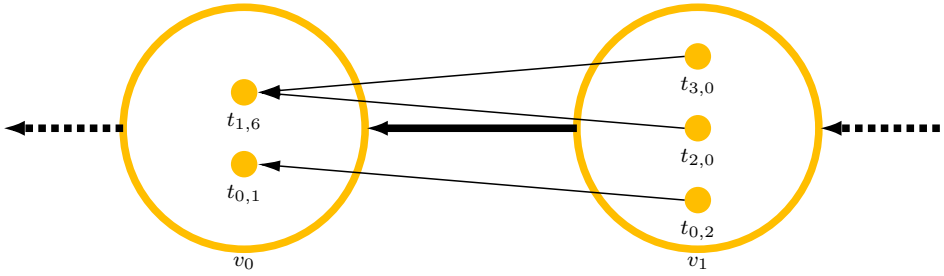
OTSO.dev
consult@otso.dev
+358 50 544 9174

2. Ajallisen järjestyksen lisäksi jokainen muutoskokonaisuuden sisältämä tieto sisältää viittauksen yhteen tai useampaan jonkin edeltävän muutoskokonaisuuden tietoon.
3. Tiedosta on mahdollista päätellä sen tuottanut taho sekä ne tahot, joilla on siihen käyttöoikeus; sama ehto pätee itse muutoskokonaisuuteen.
4. Yllä mainitut päätelmät ovat ristiriidattomia kaikkien prosessin toimijoiden kesken.

Esimerkkinä toteutuksesta voidaan käyttää kahden toimijan – kunnan ja kansallisen rekisterin – välistä vuorovaikutusta.

Kansalliseen rekisteriin on mielekästä viedä suunnitelman versioita (muutoskokonaisuuksia) ainoastaan tiettyjen (sisällön laatua ja kattavuutta koskevien) ehtojen täyttyessä, joten moni kunnan kaavoitusorganisaation tuottama työnaikainen versio jää kunnan omaan järjestelmään. Integroidun versionhallinnan ansiosta kaikki versiot ovat kuitenkin osa yhtä yhtenäistä muutosketjua kuvan 35 esimerkin mukaisesti. Kansallisella tasolla näkyvien versioiden voidaan tällöin ajatella olevan ”tiiviste” organisaation sisäisestä rikkaammasta ja tiheämmästä muutoshistoriasta.[79]

Suunnitelmassa mukana kulkevien tietojen elinkaaren hallinta tulee toteuttaa puhtaasti kohteiden välisillä viittauksilla – ei esimerkiksi aikaleiman kaltaisilla sekundäärisinä pidettävien tietojen avulla.[80] Viittaukset kertovat siis sekä järjestyksen (”Mitä tietoja käytettiin lähteinä tämän tiedon tuottamisessa?”) että ajallisen etäisyyden (”Kuinka vanhoja tämän tiedon lähteet ovat?”).



Kuva 36: Esimerkki versiohistorian rakenteesta. v_1 ja v_0 edustavat muutoskokonaisuuksia (esimerkiksi kaavan julkaistuja versioita, työnaikaisia versioita tms.) joiden sisällä on eriteltynä suunnitelman sisältöä. Muutoskokonaisuuksien välisen versioviittauksen lisäksi viittauksia ylläpidetään tietojen osalta.

On tärkeää tehdä ero itse *muutoskokonaisuuksien* sekä niiden sisältämien *tietojen* versioiden välille (kuva 36. Jokaiseen tietoon kohdistuvasta muutoksesta voidaan tehdä oma muutoskokonaisuus, mutta tämä on harvoin tarkoituksenmukaista. Sen sijaan muutoskokonaisuuden tulisi sisältää yksi mielekäs suunnitelman sisältöön kohdistuva muokkaus.[81]

Elinkaaresta puhuttaessa tarkoitetaan aikaväliä, jolloin sisällöltään itsenäiseksi kokonaisuudeksi miellettävä tietosisältö syntyy (tuodaan osaksi prosessia) ja poistuu (joko irroit-

[79] Tilanne vastaa karkeasti `git --squash` -tyyppistä muutosten yhdistämistä kansallisen rekisterin näkökulmasta.

[80] Kuten kuvasta 35 nähdään, järjestys on aina yksikäsitteinen saman lineaarisen polun sisällä: esimerkiksi versio $v_{0,3}$:n on oltava vanhempi kuin versio K_0 mutta nuorempi kuin version K_1.

[81] sellaisen voisi muodostaa esimerkiksi suunnittelijan luonnostelema karkea ratkaisuvaihtoehto, joka sisältää tietyn tavoitteen saavuttamiseksi tehtyjä muutoksia tielinjaukseen, korttelin rakennusten sijaintiin ja kaavamääräyksiin

tamalla prosessista tai sulauttamalla osaksi muuta tietoa).

Tietokokonaisuuden itsenäisyyden määrittely ei ole yksiselitteistä vaan tilanne- ja näkökulmasidonnaista. Tämän vuoksi versionhallinnan tehtävänä on tallentaa muutokset muodossa, joka sallii tarkastelun ja työskentelyn vaihtelevista perspektiiveistä käsin.

Työnkulku Tässä ehdotettu alustava työnkulku pohjautuu ohjelmistotuotannon alalla käytettyihin versionhallinnan käytäntöihin, sekä aiempaan kaavoitustyöstä tehtyyn tutkimukseen. Sen käyttöönotto vaatii luonnollisesti yleistä toimintakulttuurin muutosta sekä tarvittavien rajapintojen käyttöönottoa.

Kaavaprosessin ensimmäinen versio luodaan keräämällä kansallisen rekisterin sekä muiden lähtötietoaineistojen ajantasaisista tiedoista kokoelma, jonka pohjalta kunta luo suunnitelman tietovarannon (*repository*) ja tallentavat ensimmäisen version.

Toteutuksen on tarkoituksenmukaista noudattaa hajautettua mallia (kappale 2.2.1): sisällöntuotantoon osallistuvat toimijat peilaavat alkuperäisen tietovarannon itselleen (*clone*-operaatio), ja työskentelevät ensisijaisesti niiden varassa. Tarpeellisissa kohdin prosessia toimijat yhdistävät toistensa varannoista sisältöä omaansa (*pull*-operaatio), tai vaihtoehtoisesti esittävät toiselle osapuolelle pyynnön oman sisältönsä yhdistämisestä heidän varantoonsa (*push*-operaatio).

Konkreettisia esimerkkejä työtavasta ovat esimerkiksi kunnan valvoma ja konsultin tuottama kaavan valmistelutyö, tai saman kaavoitusorganisaation sisällä toimivan kaavoittajan ja liikennesuunnittelijan yhteistyö.

Konsulttikaavoittajan kehittäessä suunnitelmaa kunnalta peilatussa tietovarannossa, voi kunta sekä seurata että ohjata työtä joustavasti, sekä yhdistää kaavavaiheita varten viimeisteltyjä ja hyväksyttyjä versioita omaan tietovarantoonsa (josta ne yhdistetään kansalliseen rekisteriin).

Kaavoittajan ja liikennesuunnittelijan työskentely voi tapahtua saman tietovarannon sisällä kahdessa tai useammassa haarassa; kummallakin osapuolella on tällöin jatkuvasti päivittyvä ajantasakuva suunnitelman sisällön kehittymisestä. Kumpikin osapuoli voi tällöin kommentoida (annotoimalla) toisen työtä, sekä kokeilla sen yhteensovitusta omaan sisältöönsä toisen osapuolen työtä häiritsemättä.

Oikeus- ja vastuukysymykset Koska potentiaalinen osallisten joukko on hyvin laaja, ei yllä kuvattu työnkulku luonnollisesti toimisi, mikäli kaikki tuotettu tieto olisi peilauksen jälkeen avoimesti kenen tahansa tulkittavissa. Lisäksi oikeudellisten vastuukysymysten ja suunnitteluetiikan vuoksi on olennaista, että tehdyt muutokset suunnitelman sisältöön eivät ole anonyymejä, vaan niitä edustaa aina jokin taho.

Vaihtelevat lukuoikeudet voidaan toteuttaa salaamalla muutoskokonaisuus käyttäen tietyn halutun käyttäjäjoukon julkisia avaimia. Mikäli tarvitaan hienostuneempaa jakoa, voidaan muutoskokonaisuuden sisällä olevia tietoja salata samalla menetelmällä. Lopuksi itse muutoskokonaisuus allekirjoitetaan sen tuottaneen tahon omalla avaimella.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Tiedon tuottamiseen ja yhdistämiseen tarvitaan sähköinen allekirjoitus jolla tietoa tuottanut taho ottaa vastuun sisällöstä, sekä mekanismi jolla estetään toimijoita poistamasta vapaasti toisten osapuolien allekirjoittamaa sisältöä ilman erikseen myönnettyjä oikeuksia. Tietoa vastaanottava taho voi tällöin varmistua sekä siitä, että tieto on oikeasta lähteestä, että siitä että sitä käsitellyt taho ei ole ylittänyt toimivaltuuksiaan.

Tekniset ratkaisut ja käyttäjärajapinta

Yllä kuvattu integraatio sekä versiohallitun tiedon hyödyntäminen ja tuottaminen vaativat kaikilta osallisilta muutoksia sekä toimintatapoihin että työkaluihin. Tässä esitetyt ratkaisut tekevät kuitenkin mahdolliseksi uuteen toimintamalliin siirtymisen asteittain. Ratkaisut pohjautuvat vakiintuneeseen, laajalti käytössä olevaan teknologiaan.

Kaavaprosesseissa käsiteltävää tietoa ei voida nykyisellään rakenteistaa täysin, joten on selvää, että versionhallinnan täytyy tukea ympäristöä jossa merkittävä osa suunnittelutiedosta on jaettu lukuisiin tiedostoihin. Tiedostopohjaiseen versionhallintaan soveltuva, toimintavarma ja laajalti tuettu avoimen lähdekoodin ratkaisu on esimerkiksi Git[82], jonka ohessa suositellaan käytettävän tukea suurikokoisille tiedostoille[83].

Lähtökohtana on, että kaikki prosessin lähtöhetkellä saatavilla oleva rakenteinen tieto sisällytetään kaavatietomalliin, joka tallennetaan yhteiseen tietovarantoon esimerkiksi XML-muodossa. Kyseinen tiedosto muodostaa rungon, johon viralliseen muotoon jalostuvaa kaavan sisältöä tallennetaan.

Muu sisältö – joka tullee vielä pitkään koostumaan lukuista eri formaattien tiedostoista – voidaan organisoida vielä nykyään yleisesti käytössä olevien projektikansioiden tapaan, jotta työnkulku muuttuu suunnittelijoiden näkökulmasta mahdollisimman vähäisissä määrin. Jokainen versiohallittu muutoskokonaisuus vastaa tällöin projektikansion sisältöä tietyllä ajanhetkellä, ja suurin muutos nykyiseen on se, että päivämäärillä tai versioilla nimettyjen tiedostojen tallentamisen tai CMS/groupware-ohjelmistoihin versioiden tallentamisen sijaan koko projektikansiosta tehdään muokkausten jälkeen uusi muutoskokonaisuus versionhallintatyökalulla.

TUMA-tietomallin tuki versionhallinnalle Nyt kehitteillä olevan TUMA-osahankkeen kaavatietomallissa kuvattua versionhallintarakennetta ei tällä hetkellä ole yleistetty kattamaan kaavamääräyksiä tai muita sisältöä kuvaavia luokkia.

Tämän selvityksen näkökulmasta myös niiden sisällöllisten muutoksen seuraaminen versiohistorian kautta on perusteltua, ja tietorakenteiden sekä toiminnallisuuden johdonmukaistamiseksi versionhallinta tulisikin toteuttaa samalla mekanismilla luokan tyypistä riippumatta.

Tämän selvityksen tuottamisen aikana tarkastellussa TUMA-tietomallin luonnoksessa tuma:Kaavakohde-luokalla on assosiaatio rak:KohteenMuutoshistoria-luokkaan, jonka yksittäisten kohteiden elinkaaren aikaista versioiden muutosketjua ylläpidetään. Erillistä

[82] `https://git-scm.com/`
[83] Esimerkiksi Git Large File Storage (`https://git-lfs.github.com/`)

OTSO.dev
consult@otso.dev
+358 50 544 9174

versiohistoriaa ylläpidetään lisäksi tuma:Kaavamuutos-luokan avulla, joka sisältää viittaukset uusiin, poistuviin ja muuttuviin kohteisiin, sekä viittaukset $[0, n], n \geq 0$ vastaavaan muutoskokonaisuuteen jotka se korvaa.

tuma:Kaavamuutos-luokka on hyödyllinen tarkasteltaessa muutoksen sisältöä yksittäisen suunnitelman objektien elinkaaren tasolla (millä välillä tietty objekti on ollut olemassa ja miten sen sisältö on muuttunut). Suunnitteluprosessin kannalta on kuitenkin hyödyllistä pystyä seuraamaan muutoksia myös yli yksittäisten kohteiden elinkaaren, esimerkiksi kun käyttötarkoitukseltaan aiemmin yhtenäinen alue jaetaan useammiksi käyttötarkoituksiltaan erilaisiksi alueiksi.

Kaavakohteiden elinkaaren toteutus Kaavasuunnitelman sisällön tulisi yksikäsitteisesti pystyä kuvaamaan sen osia koskevien muutosten rakenne sekä järjestyssuhde. Sisällöllä tarkoitetaan tässä tapauksessa pääasiallisesti rakenteellista, tietomallissa kuvattujen luokkien sisältämiä kohteita.

Jotta yksittäisen kohteen muuttumista voidaan seurata, tulee sillä olla perustamisesta lähtien pysyvästi voimassa oleva uniikki tunniste jolla se voidaan identifioida. Tunnisteen tulee olla kansallisesti uniikki, toisin sanoen yhdenkään kaavasuunnitelman versiosta ei tule löytyä kohteita samalla tunnisteella. Sama ehto voidaan täyttää myös siten, että tunnisteet ovat uniikkeja kaavaprosessin sisällä (toimijoiden kesken), ja kansalliselle tasolle vietäessä ne esiintyvät ainoastaan kaavaprosessin määrittämässä nimiavaruudessa.[84]

Jokainen kohteen yksittäinen versio (*instanssi*) tarvitsee versiokohtaisten pysyvän tunnisteen, sekä $[0, n], n \geq 0$ viittausta johonkin aiemmin olemassa olleeseen versioon. Versiotunniste toimii avaimena tietyn kohteen spesifille versiolle, kohteen oma tunniste vuorostaan kaikille versioille läpi sen elinkaaren.

Versionhallinnan tulee olla toteutettu kohdeagnostisesti: kaikkien suunnitelman ydinsisältöön kuuluvien kohteiden – minimissään spatiaalisten (kuten alueiden) ja semanttisten (kuten kaavamääräysten) – muutoksia tulee pystyä seuraamaan samalla mekanismilla.

Kohdeinstanssi, jolla on useita viittauksia edeltäneisiin kohteisiin, kuvastaa muutosta jossa useampi kohde on yhdistetty yhdeksi. Jossain tapauksissa useampi instanssi viittaa samaan edeltäjään: tämä vuorostaan vastaa tilannetta, jossa yksi kohde on jaettu useaan osaan.

Koska jokainen kaavasuunnitelman tallennettu versio on joukko kohteisiin kohdistuvia muutoksia (muutoskokonaisuus), joidenkin kohteiden viittaukset voivat osoittaa myös edeltävää versiota aiempiin muutoskokonaisuuksiin. Suunnitelman jokaiseen versioon tallennetaan siis ainoastaan delta (kts. luku 2.2).

Jatkotutkimusta tarvitaan sen määrittämiseksi, sallitaanko muutoskokonaisuuksissa teknisellä tasolla täysin uusien kohteiden luominen tai olemassaolevien poistaminen. Mikäli nämä operaatiot sallitaan, uudet kohdeinstanssit tunnistetaan esimerkiksi siitä, että ne eivät sisällä joko yhtäkään viitettä edeltäviin kohteisiin.

[84] Kaavoissa K_1 ja K_2 voi kummassakin olla uniikki kohde samalla tunnisteella 48b6b9a4-184e-11ea-8d71-362b9e155667, mutta kyseistä kohdetta käytetään aina vain kaavaan liitettynä, esimerkiksi k_1:48b6b9a4-184e-11ea-8d71-362b9e155667, jolloin sekaannusta ei synny.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Vaihtoehtoisesti suunnitelmaan voidaan aina sisällyttää pysyvä null-kohde (aina olemassaoleva kohde jonka kanssa ei voi vuorovaikuttaa suoraan eikä näin ollen myöskään poistaa sitä), johon juuri luodut kohdeinstanssit viittaavat. Null-kohde voidaan ajatella yksinkertaistettuna ”muistitauluksi”, jota käytetään viittausten kanssa apuna luotujen ja poistettujen kohteiden kirjanpitoon versioiden välillä.

Null-kohteen tarve on ilmeisempi kohdetta poistettaessa, sillä poistamisoperaation jälkeen tallennetussa versiossa kyseistä kohdetta ei enää ole, ja tieto sen poistosta pitäisi hakea vertaamalla kyseisen version sisältöä edeltäneeseen.

Koska tieto poistamisesta halutaan saada sisällytettyä myös versioon, jossa kyseistä kohdetta ei enää ole, voidaan kyseisestä versiosta tarkistaa null-kohteesta löytyvät viittaukset edellisiin versioihin ja saada tieto niiden poistosta näin selville.

Tällä metodilla toteutettuna jokaisen kohteen elinkaari muodostuu null-kohteesta lähteväksi ja siihen päättyväksi silmukaksi.

Tekninen toteutustapa täytyy valita sen perusteella, mikä topologinen rakenne on syntyvien versiohistorioita edustavien graafien rakenteen osalta tehokkain toteutettava, ja sisältöä koskevan analyysin luontevimmalla tavalla.

Jotta kohteiden yhdistäminen, jakaminen ja niiden olemuksen muokkaaminen palvelevat suunnittelun tarpeita, ja jotta suunnittelussa syntyvä versiohistoria on tulkittavissa johdonmukaisena ja selkeänä polkuna, täytyy niiden käytölle asettaa joitakin rajoitteita.

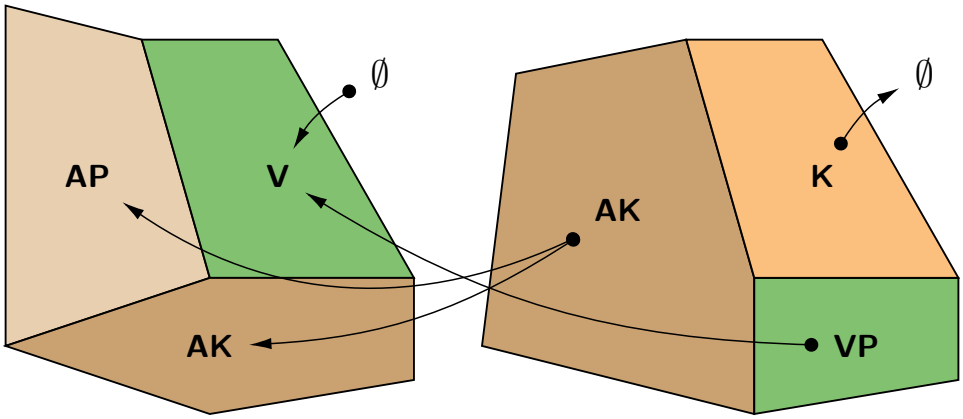
Kohdeinstanssin täytyy esimerkiksi aina viitata loogiselta käyttötarkoitukseltaan samankaltaiseen edeltävään instanssiin. Tämä johtuu siitä, että edeltävissä suosituksissa kuvatun jaon (semanttiset ja spatiaaliset tyypit) vuoksi jonkin kaavan kohteen muuttuminen esimerkiksi tavoitetta kuvaavasta tekstistä kyseisen tavoitteen mukaiseksi alueeksi tekee suunnitelman sisällön kehittymisen seuraamisesta haastavaa. Semanttisesti johdonmukainen tapa toteuttaa kyseinen muutos olisi kytkeä tavoite joko uuteen luotavaan tai muutettavaan alueeseen.

Tietyn kohdetyypin instanssien välisiä viittauksia voidaan ylläpitää automaattisesti, kun suunnittelussa suoritetaan seuraavia operaatioita:

1. instanssin sijaintia, muotoa tai muita attribuutteja muokataan (esimerkiksi aluetyyppi): tällöin uniikki pysyvä tunniste säilyy samana,
2. kun instanssi luodaan tiettyä versiota varten: syntyy viite null-kohteeseen,
3. kun vähintään kaksi kohdetta yhdistetään: syntyvälle uudelle kohteelle luodaan uusi pysyvä tunniste ja sen versioviittaus kohdistetaan kaikkiin edeltäjiin, sekä
4. kun kohde jaetaan kahteen tai useampaan uuteen kohteeseen: syntyville uusille kohteille luodaan pysyvät tunnisteet ja versioviittaukset kohdistetaan alkuperäiseen kohteeseen.

Elinkaaren hallinnan käyttäjärajapinta Vaikka lähtökohtaisesti suunnittelussa pyritään seuraamaan oikeusvaikutusten muutosta tietyllä alueella, se voidaan toteuttaa ilman kohteiden välisiä viittauksia tarkastelemalla, mitkä toisen version kohteet leikkaavat kyseisen alueen kanssa.

OTSO.dev
consult@otso.dev
+358 50 544 9174



Kuva 37: Esimerkki kahden suunnitelman version kohteiden välisistä viittauksista. Selkeyden vuoksi on esitetty ainoastaan aluekohteet, ei alueisiin kohdistuvien huomioiden tai kaavamääräyksien viittauksia.

Kohteiden välisten viittausten seurannalla vuorostaan mahdollistetaan esimerkiksi tiettyjen toimintojen kehittymisen seuraaminen versioiden välillä. Kaavassa voi esimerkiksi olla tavoitteena sijoittaa alueelle päiväkoti, ja tätä varten luodaan kolme rinnakkaista ratkaisuehdotusta joiden vaikutuksia arvioidaan ja verrataan keskenään. Kaikkien versioiden kohdalla on olennaista kyetä tunnistamaan, että kussakin ratkaisussa eri paikassa sijaitseva päiväkoti on semanttisesti sama kohde ja vastaa samaan suunnittelun päämäärään.

Kuten kuvan 37 esimerkistä voidaan päätellä, työn aikana tehtävän kohteiden jakamisen, yhdistämisen, poistamisen ja luomisen kautta viittaukset voivat muodostua työtavoista riippuen millaisiksi tahansa.

Suunnittelun sisällön kehittymistä kuvaava ketju voi katketa esimerkiksi tilanteessa, jossa suunnittelija poistaa työnaikaisesta versiosta jokaisen kohteen, ja luo tilalle uudet.

Ongelmaa voidaan koittaa väistää sallimalla muokkausoperaatioiksi ainoastaan kohteiden yhdistäminen ja jakaminen, jolloin kohteen poistaminen tapahtuu sulauttamalla se toiseen olemassaolevaan kohteeseen ja uuden luominen jakamalla olemassaoleva kohde. Ehdotettu ratkaisu ei kuitenkaan estä merkityssisällön hävittämistä kohteita yhdistämällä.

Siinä missä edellä kuvattiin keinoja kohteiden muutosketjun mahdollisimman katkeamattomalle seurannalle, kohteita käsittelevän käyttäjän työn kannalta merkitsevää on vain muokkauksen intuitiivisuus ja johdonmukaisuus. Toteutukseen ei ole saatavilla yksinkertaista valmista ratkaisua, ja se vaatiikin panostuksia mahdolliseen työkalukehitykseen sekä UX-suunnitteluun.[85]

Edellä kuvattujen haasteiden lisäksi kohdetyypin muokkausta koskevat operaatiot muuttuvat hieman monimutkaisemmiksi, kun otetaan huomioon spatiaalisten ja kuvailevien kohteiden välinen linkki. Muokattaessa toisen tyypin kohteita (esim. alueita), täytyy huomioida myös niihin viittaavien kohteiden (esim. määräysten) merkityksen muuttuminen operaation seurauksena.

[85] engl. *User eXperience*; sis. käyttöliittymäsuunnittelun lisäksi käyttäjävuorovaikutuksen suunnittelua.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Esimerkkinä voidaan käyttää tilannetta, jossa suunnittelija jakaa kaavamääräyksellä varustetun Y-alueekohteen AK- ja VP-aluekohteisiin. Tällöin toteutuksesta riippuen kaavamääräyksen linkki joko katkeaa, tai se kopioidaan sellaisenaan uusiin kohteisiin – kummassakin tapauksessa suunnittelijan täytyy huomioida syntynyt muutos kokonaisuutena, ei vain puhtaasti alueisiin tai kuvailevaan aineistoon kohdistuvana asiana.

Asiaan voidaan jossain määrin puuttua automaation (muutoksille asetettavien tarkistuslippujen tai visuaalisten vihjeiden avulla) kautta. Työnkulun ohjaukseen voidaan vaikuttaa erityisesti tapauksessa, jossa spatiaalisia kohteita kuvaileva sisältö on rakenteista. Tällöin voidaan huomioida ja nostaa automaattisesti esiin myös esimerkiksi aluekohteen tyypin suhteen ristiriidassa oleva määräys- tai kuvaileva sisältö.

OTSO.dev
consult@otso.dev
+358 50 544 9174

6 Säädökset ja yhteenveto

Tämän selvityksen puitteissa tehty sekä sitä edeltävä tutkimus (kappaleet 1.3.3; 3.2; Rinkinen, 2017; Sitowise, 2019; Taskinen, 2019a) kuvaavat melko yhtenäisesti tarvetta maankäytön suunnittelujärjestelmän uudistamiseen.

Onnistuakseen uudistuksen täytyy vastata kahteen järjestelmätuotannon alalla keskeiseen kysymykseen onnistuneesti: ”Vastaako uudistus oikeisiin tarpeisiin?”, sekä ”Vastaako uudistus tarpeisiin oikealla tavalla?”.

Alla on luonnosteltu 11.10.2019 julkaistuihin pykäläluonnoksiin liittyviä ehdotuksia, joiden tavoitteena on varmistaa, että tietomallimuotoisen kaavasisällön käyttö vakiintuu osaksi maankäytön suunnittelun tulevaa tilaa:

- ***Kaikella kaavasuunnittelussa käytetyllä tiedolla tulee olla uniikki, kaikkien toimijoiden jakama elinkaaren aikainen tunniste.***
- ***Suunnitteluun osallistuvien toimijoiden tulee siirtää tietoa vain yhteisten avoimien rajapintojen kautta.***
  Ensisijaisesti velvoite koskee kansallisen rekisterin kanssa vuorovaikutusta, sillä olennaisinta on kansallisesti keskitetyn tiedon saatavuus yhteismitallisessa avoimessa muodossa. Optimaalisessa tilanteessa samaa rajapintaratkaisua hyödyntävät kuitenkin keskenään myös kaikki toimijat (ml. osalliset rajapintaa hyödyntävien vuorovaikutustyökalujen kautta).
  Rajapintojen tulee velvoittaa, että siirtyvässä aineistossa on tieto sen tuottajasta, lähteestä, iästä, varmuusasteesta, käyttöoikeuksista sekä suhteesta muuhun prosessin tietoon.
  Säädös velvoittaa suoraan viranomaisia sekä muita osapuolia, jotka osallistuvat itse kaavasisällön tuottamiseen ja arviointiin. Muiden osallisten kohdalla viranomaisten velvollisuutena on viedä tieto (esimerkiksi kansalaisten mielipiteet) osaksi rakenteista tietomallisisältöä. Tähän liittyy velvoite tietomallisisällön oikeudellisen validoinnin toteuttamisesta (käyttöoikeudet) siten, että mm. yksityisyydensuoja toteutuu nykyistä prosessia vastaavasti.
- ***Virallisen lainvoiman saa kaavatietomallin sisältö (ei spesifi esitystapa).*** Lainvoimaa ei tulisi antaa tietosisällölle jota ei ole tarkastettu tai ymmärretty. Tämän vuoksi tulee säätää minimimäärä vakioituja näkökulmia[86] tietomallin sisältöön, joiden avulla suunnitelmasta muodostetaan mahdollisimman kattava ja vinoumista vapaa kuva.[87]
- ***Kaava astuu voimaan sen siirtyessä osaksi kansallista rekisteriä.*** Yhden kunnan alueelle sijoittuvalla kaavalla voi olla merkitystä muiden alueiden suunnitteluun tai muihin prosesseihin. Ajantasaisen lainvoimaisen tiedon tulee tällöin olla saatavilla yhdestä auktorisoidusta lähteestä jonka sisältämä tieto on muodoltaan ja varmuusasteeltaan yhdenmukaista.

[86] Näkökulmat voivat olla karttaesitysten lisäksi leikkauksia, kolmiulotteisia projektioita, kaaviokuvia, taulukoita, diagrammeja, reaaliaikaisia VR-näkymiä, simulaatioita jne.

[87] Vakioitujen esitystapojen määrittelyn tulee pohjautua monialaiseen tiedon visualisoinnin, käyttöliittymäsuunnittelun ja kognitiotieteiden tutkimukseen, ei alan vakiintuneisiin konventioihin.

6.1 Yhteenveto

Tässä selvityksessä on kuvailtu rakenteiseen kaavatietoon liittyvän versionhallinnan ja validoinnin eri ulottuvuuksia, ja luonnosteltu niihin liittyviä ratkaisuehdotuksia. Tässä selvityksen lopuksi painotetaan, että kuvailtuja ratkaisuja tai suosituksia ei tule tulkita toteuttamiskelpoisiksi sellaisinaan, vaan jokaisen osalla on tarpeen tehdä systeeminen tarkastelu yhteensopivuudesta uudistuvan järjestelmän muiden osien kanssa.

Osittain teknisestä luonteesta huolimatta selvityksen näkökulma on holistinen, ja kaikki annetut ratkaisuehdotukset on tarkoitettu edesauttamaan laadultaan korkeatasoisen ja tarkoituksenmukaisen maankäytön suunnittelujärjestelmän kehittämistä.

Sanasto

INSPIRE	engl. *Infrastructure for Spatial Information in the European Community*, Euroopan yhteinen paikkatietoinfrastruktuuri. 42, 46, 49, 52
NoSQL	engl. *Not only SQL*; joukko tietokantaratkaisuja, jotka eivät pohjaudu perinteiseen relaatiomalliin. 19–21
ACID	engl. *Atomicity, Consistency, Isolation, Durability*; ehtoja, jotka täyttävä tietokanta huolehtii säilytettävän tiedon ja sen käsittelyn johdonmukaisuudesta (suoritusnopeuden kustannuksella). 19, 20, 29
CMS	engl. *Content Management System*; ohjelmisto, jonka tehtävänä on tukea yhden tai useamman käyttäjän yhteisesti käsittelemän tiedon hallintaa. 23, 39, 50, 74
CSV	engl. *Comma Separated Value*, yksinkertainen tapa tallentaa taulukkoja tai listoja ihmisluettavaan muotoon siten, että sarakkeet erotetaan pilkulla (tai puolipilkulla). 21
GIS	engl. *Geographic Information System*, järjestelmä paikkatietosisällön tallentamiseen ja käsittelyyn. 37, 39, 57
GML	engl. *Geography Markup Language*, XML-muotoinen kieli paikkatiedon kuvaamiseen. 40, 44, 45, 62
JSON	engl. *JavaScript Object Notation*; XML-kielen kanssa kilpaileva kevyt, yksinkertainen ja ihmisluettava avoin formaatti. Soveltuu hyvin avain-arvo-parien sekä ohjelmistoissa sisäisesti käytettävien tietorakenteiden siirtoon. 21
MAL-sopimus	Valtion ja kaupunkiseutujen välinen sopimus yhdyskuntarakenteen ohjaukseen sekä maankäytön, asumisen ja liikenteen yhteensovittamiseen. 56
MRL	Maankäyttö- ja rakennuslaki. 5, 6, 46, 49, 50, 60
OGC	engl. *Open Geospatial Consortium*. 69
OWL	engl. *Web Ontology Language*; ryhmä tiedon sisällön kuvailukieliä. 34, 44, 68

PDF	engl. *Portable Document Format*; nykyään ISO-standardoitu tiedostoformaatti, joka tukee muotoillun tekstin lisäksi laajakirjoisen tiedon upottamista dokumentteihin. 11, 38, 39
RDF/XML	XML-muotoinen esitys metadatan kuvaukseen käytettävästä RDF (engl. *Resource Description Framework*) -kuvauskielestä. 64, 65, 68
REST	engl. *Representational state transfer*; rajapintaratkaisu. 21, 44
SOAP	engl. *Simple Object Access Protocol*. 21, 32, 39
SQL	engl. *Structured Query Language*, standardoitu kieli relaatiotietokantojen käyttöön. Kieli on niin yleinen, että relaatiokantoja kutsutaan tyypillisesti SQL-tietokannoiksi. 18, 21, 32
SSO	engl. *Single Sign-On*. 32, 37, 44, 46, 53
UML	engl. *Unified Modeling Language*; yleinen ohjelmistojärjestelmien suunnitteluun käytettävä formaali kieli. 17, 18, 20, 43
WFS	engl. *Web Feature Service*; rajapintaratkaisu mm. vektorimuotoisen paikkatiedon toimittamiseen. 21, 37, 44
WMS	engl. *Web Map Service*; rajapintaratkaisu rasterikarttojen toimittamiseen vapaasti rajatussa muodossa. 37, 44
WMTS	engl. *Web Map Tile Service*; rajapintaratkaisu rasterikarttojen toimittamiseen lohkoihin jaettuna. 44
XML	engl. *Extensible Markup Language*; ISO-standardoitu ihmisluettava rakenteisen tiedon kuvauskieli joka on laajennettavissa erilaisiin käyttötapauksiin. 18, 21, 22, 32–34, 39, 41, 44, 45, 53, 54, 66–68, 74
XSD	engl. *XML Schema Definition*; XML-kieleen pohjautuva kieli, jota käytetään XML-dokumenttien skeemojen määrittelyyn. 21, 31, 40, 43, 44, 66
annotointi	Aineiston sisällön kuvailua, tyypillisesti dokumentti-, kartta- tai kuva-aineistoon upotettavien sanallisten huomioiden tai metatiedon avulla. 11, 34, 65
asynkroninen	Ajallisesti toisesta osapuolesta riippumaton. 25, 27
attribuutti	Luokan, olion tai muun kohteen vaihtuvia ominaisuuksia kuvaava arvo. 17, 20, 22, 25, 30, 38, 42, 46
eksplisiittinen tieto	Kielellä tai muulla tavalla (esim. puhe, kaavio) esitetty tieto, jota voidaan siirtää ja tallentaa. Finto tietotermit. 14
formaali	Muodollinen; jonkin sovittujen sääntöjen avulla täsmällistetty. 7, 34

OTSO.dev
consult@otso.dev
+358 50 544 9174

GeoJson	JSON-formaattiin pohjautuva laajennos, joka on suunniteltu paikkatiedon tallentamiseen. 25, 26
hiljainen tieto	Subjektiivista, intuitiivista ja sanallistamatonta kokemuksen kautta kertynyttä tietoa jota on vaikeaa tai mahdotonta muuttaa eksplisiittiseksi tiedoksi. Finto tietotermit. 16
implisiittinen tieto	Julkilausumaton mutta asiayhteydestä pääteltävä tieto. Tieteen termipankki. 14
inferenssi	Uusien faktojen päättely jo todennetuista tai uusien olettamusten johteminen aiemmista olettamuksista päättelysääntöjen avulla. Tieteen termipankki. 14
metatieto	Tiedon sisältöä, rakennetta tai muita ominaisuuksia kuvailevaa tietoa. 34
prosessimuisti	Henkilöriippumaton kuvaus esimerkiksi suunnitteluprosessissa tehdyistä toimista eksplisiittisessä muodossa. 12
rakenteeton tieto	Tietosisältö jota ei ole tallennettu sen sisällön merkitystä johdonmukaisesti kuvaavalla rakenteella. Termin yleinen määritelmä (engl. *unstructured data*) ei ole ongelmaton, tässä selvityksessä rakenteettomaksi tiedoksi lasketaan sekä analoginen että digitaalinen tieto jonka merkitystä ei voida tulkita ilman ihmistä tai heuristiikkaa. Finto tietotermit. 11
semantiikka	Tämän selvityksen puitteissa semantiikalla tarkoitetaan jonkin tietokokonaisuuden sisältämää merkityssisältöä joko ihmis- tai koneluettavassa muodossa. 14
serialisointi	Tietosisällön muuntaminen muotoon, jossa kaikki sisältö on yksiulotteisessa muodossa peräkkäin (esim. tekstitiedosto, jolla on alku ja loppu, ja jokaiselle kirjaimelle voidaan määrittää etäisyys alusta tai lopusta laskien). 22
skeema	ontologinen, semanttinen ja/tai rakenteellinen kehys informaation tallentamiseen ja tulkintaan. 18, 29
tietomalli	Tiettyyn ontologiaan pohjautuva formaali malli tai ”kehys”, jonka rakenteen mukaiseksi tulkittua tietoa voidaan tallentaa ja käsitellä koneluettavassa muodossa. 5–7, 17, 18, 20, 33, 35, 37, 39, 40, 42, 43, 45, 46, 49–53, 56–58

tiiviste	Halutusta sisällöstä laskettu määrämittainen luku, jonka arvo on aina identtinen samalle syötteelle. Tiivisteen arvoa voidaan käyttää varmistamaan esimerkiksi, ovatko kaksi suurikokoista tietojoukkoa identtisiä, tai onko jokin tieto muuttunut edellisen tiivisteen laskennan jälkeen. 26, 27, 33, 54
TUMA	Tulevaisuuden Maankäyttöpäätökset -osahanke. 53, 71, 74
työnkulku	engl. *workflow*; suhteellisen samankaltaisena toistuva työmenetelmä joka voidaan kuvata esimerkiksi prosessikaavion muotoon. 12, 28, 29, 31, 52
versionhallinta	Tiedon muutoksien hallinta yhteisen viittausketjun kautta, ts. perättäisiä versioita ei tunnisteta esimerkiksi ainoastaan päiväyksen, vaan versiosta toiseen tehdyn viittauksen avulla. 5–7, 12, 20, 22–30, 33, 35, 37, 39–41, 43, 45, 46, 51, 53, 56, 58

Viitteet

Wang, Ting-Chun et al. ”Few-shot Video-to-Video Synthesis”. *arXiv:1910.12713 [cs]* (lokakuu 2019). arXiv: 1910.12713. URL: http://arxiv.org/abs/1910.12713.

West, Matthew. *Developing High Quality Data Models*. Elsevier, helmikuu 2011. ISBN: 978-0-12-375107-2.

Witten, Ian H., David Bainbridge ja David M. Nichols. ”Chapter 6 - Metadata: Elements of organization”. Teoksessa: *How to Build a Digital Library (Second Edition)*. Toim. Ian H. Witten, David Bainbridge ja David M. Nichols. Second Edition. The Morgan Kaufmann Series in Multimedia Information and Systems. Morgan Kaufmann, 2010, s. 285–341. ISBN: 978-0-12-374857-7. DOI: 10.1016/B978-0-12-374857-7.00006-2. URL: http://www.sciencedirect.com/science/article/pii/B9780123748577000062.

Becker, C. et al. ”Requirements: The Key to Sustainability”. *IEEE Software* 33.1 (tammikuu 2016), s. 56–65. ISSN: 0740-7459. DOI: 10.1109/MS.2015.158.

Berliner, Brian. ”CVS II: Parallelizing Software Development”. Teoksessa: *Proceedings of the USENIX Winter 1990 Technical Conference*. Vol. 341. 1990, s. 12.

Bie, Stein W. ja Einar Stormark. *Forslag til utvekslingsformat for digitale geodata (SOSI-formatet, versjon 1.0)*. Publikasjon (Norsk regnesentral: trykt utg.) 675. Norsk regnesentral, 1980. ISBN: 978-82-539-0151-0. URL: https://urn.nb.no/URN:NBN:no-nb_digibok_2015051908156.

Boehm, Barry. ”Value-based software engineering”. *ACM SIGSOFT Software Engineering Notes* 28.2 (2003), s. 3. ISSN: 01635948. DOI: 10.1145/638750.638775.

Braun, Simone, Andreas Schmidt ja Andreas Walter. ”Ontology Maturing: a Collaborative Web 2.0 Approach to Ontology Engineering” (2007), s. 10.

Chacon, Scott ja Ben Straub. *Pro Git*. 2. painos. Elokuu 2019. URL: https://github.com/progit/progit2/releases/tag/2.1.164.

Chaudhuri, Neil A., Jessica L. Glace ja Gregory A. Wilson. *Hierarchical vs Relational XML Schema Designs – A Study for The Environmental Council of States – Report ECO41T1*. LMI Government Consulting, kesäkuu 2006.

Ciccarese, Paolo et al. ”An open annotation ontology for science on web 3.0”. 2 (toukokuu 2011). ISSN: 2041-1480. DOI: 10.1186/2041-1480-2-S2-S4.

Dick, Jeremy, Elizabeth Hull ja Ken Jackson. *Requirements Engineering*. Springer International Publishing, 2017. ISBN: 978-3-319-61072-6. DOI: 10.1007/978-3-319-61073-3. URL: http://link.springer.com/10.1007/978-3-319-61073-3.

Emam, K. E. ja A. G. Koru. ”A Replicated Survey of IT Software Project Failures”. *IEEE Software* 25.5 (syyskuu 2008), s. 84–90. ISSN: 0740-7459. DOI: 10.1109/MS.2008.107.

Økonomistyrelsen. *Projektinitieringsdokument (PID) – Fuldt Digitale Lokalplaner – Initiativ 8.2 ”Digitalt overblik på planområdet”under Den Fællesoffentlige Digitaliseringsstrategi 2011–2015*. Heinäkuu 2012.

Fitzgerald, Brian. ”Software Crisis 2.0”. *Computer* 45.4 (huhtikuu 2012), s. 89–91. ISSN: 0018-9162, 1558-0814. DOI: 10.1109/MC.2012.147.

Greenberg, Jane. ”Understanding Metadata and Metadata Schemes”. *Cataloging & Classification Quarterly* 40.3-–4 (syyskuu 2005), s. 17–36. ISSN: 0163-9374. DOI: 10.1300/J104v40n03_02.

Helenius, Otso. *Maankäytön suunnitteluprosessien revisiointi – Loppuraportti Helsingin Kaupunkiympäristön toimialan (KYMP) ja Ympäristöministeriön käyttöön*. Maaliskuu 2019, s. 22.

ISO/IEC/IEEE. *ISO/IEC/IEEE 11179-1:2004(E) International Standard – Information technology — Metadata registries (MDR) – Part 1: Framework*. Syyskuu 2004, s. 1–32.

ISO/IEC/IEEE. *ISO/IEC/IEEE 29148:2018(E) International Standard – Systems and software engineering -– Life cycle processes -– Requirements engineering*. Lokakuu 2018, s. 1–104. DOI: 10.1109/IEEESTD.2018.8559686.

Jama, Teemu et al. *Ideoita kaavoituksen sisällön uudistamiseen – Kaavojen merkintöjen ja määräysten kehittäminen (KAMMI-hanke)*. Vol. 4/2018. Ympäristöministeriön raportteja. 2018. ISBN: 978-952-11-4779-1.

Kommunal- og moderniseringsdepartementet. *Nasjonal produktspesifikasjon for arealplan og digitalt planregister – Del 3.2 – SOSI Produktspesifikasjon*. Elokuu 2019, s. 67.

Kumar, Ram. *Using Code List Methodology for Value and Validation (from OASIS Code List Representation and UBL TCs) in OASIS CIQ Specifications V3.0 – A Case Study*. Kesäkuu 2007.

Lauesen, Soren. "Problem-Oriented Requirements in Practice – A Case Study". Teoksessa: *Requirements Engineering: Foundation for Software Quality*. Toim. Erik Kamsties, Jennifer Horkoff ja Fabiano Dalpiaz. Lecture Notes in Computer Science. Springer International Publishing, 2018, s. 3–19. ISBN: 978-3-319-77243-1.

Lehtovuori, Panu et al. *Tulevaisuuskatsaus eräiden maiden alueidenkäytön suunnittelujärjestelmiin*. Vol. 2019:24. Ympäristöministeriön julkaisuja. Ympäristöministeriö, 2019. ISBN: 978-952-361-031-6. URL: http://julkaisut.valtioneuvosto.fi/handle/10024/161779.

OWASP (lokakuu 2019). URL: https://github.com/OWASP/CheatSheetSeries.

Ontolog. "What Are Ontologies?" Teoksessa: *Ontology Summit 2012*. 2012.

O'Regan, Gerard. *Concise Guide to Software Engineering: From Fundamentals to Application Methods*. Undergraduate Topics in Computer Science. Springer International Publishing, 2017. ISBN: 978-3-319-57749-4. URL: http://www.springer.com/gp/book/9783319577494.

Potvin, Rachel ja Josh Levenberg. "Why Google Stores Billions of Lines of Code in a Single Repository". *Commun. ACM* 59.7 (kesäkuu 2016), s. 78–87. ISSN: 0001-0782. DOI: 10.1145/2854146.

Ramboll Finland Oy ja Ubigu Oy. *Maankäyttöpäätökset-hanke – Kaavojen digitoinnin selvitys*. Joulukuu 2018.

Rinkinen, Kristiina. *Alueidenkäytön ohjausjärjestelmän prosessit jä työnjako – Taustamuistio / päätelmiä 2017*. ProTo-hanke. 2017, s. 44.

Rinkinen, Kristiina ja Jani Kinnunen. *Asemakaavoituksen muutokset Suomen kasvuseuduilla – Kaavojen määrän, keston ja kaavoilla tuotetun kerrosalan vertailu vuosina 2004–-2005 ja 2014–-2015*. Vol. 20. Ympäristöministeriön raportteja. Elokuu 2017. ISBN: 978-952-11-4742-5. URL: http://urn.fi/URN:ISBN:978-952-11-4742-5.

Rogers, Bruce. "Why 84% Of Companies Fail At Digital Transformation". *Forbes* (tammikuu 2016).

Rogers, Everett M. ja F. Floyd Shoemaker. "Communication of Innovations; A Cross-Cultural Approach" (1971).

Sitowise. *Kuntapilotti: Ota Kantaa –kysely 2 – Kyselyn tulosten yhteenveto*. Kesäkuu 2019, s. 45.

Sitowise et al. *Kuntapilotti – loppuraportti 20.6.2019*. Kesäkuu 2019, s. 133.

OTSO.dev
consult@otso.dev
+358 50 544 9174

Strickland, Eliza. ”Racial Bias Found in Algorithms That Determine Health Care for Millions of Patients – Researchers argue for audit systems to catch cases of algorithmic bias”. *ieee Spectrum* (lokakuu 2019).

Taskinen, Satu. *Maankäyttöpäätökset-hanke – toteutussuunnitelma (luonnos).* Kesäkuu 2018, s. 84.

Taskinen, Satu. *Tulevaisuuden Maankäyttöpäätökset – Ota kantaa -kyselyn tulokset.* Toukokuu 2019.

Taskinen, Satu. *Tulevaisuuden Maankäyttöpäätökset – Tavoitetila 2030 & tiekartta.* Syyskuu 2019, s. 14.

Verbeek, Peter P. C. C. ”Cover story: Beyond Interaction: a short introduction to mediation theory”. *Interactions (ACM)* 22.3 (2015), s. 26–31. issn: 1072-5520. doi: 10.1145/2751314.

Verner, J., J. Sampson ja N. Cerpa. ”What factors lead to software project failure?” Teoksessa: *2008 Second International Conference on Research Challenges in Information Science*. Kesäkuu 2008, s. 71–80. doi: 10.1109/RCIS.2008.4632095.