Otso Helenius
Aalto-yliopiston rakennetun ympäristön laitos
KIRA-digi loppuraportti

Maankäytön suunnittelu-prosessien revisiointi

Revisioning Land-Use Planning Processes

Loppuraportti Helsingin Kaupunkiympäristön toimialan (KYMP) ja Ympäristöministeriön käyttöön

Aalto University

Helsinki

KIRA-digi

Sisällys

1. Johdanto ... 3
2. Kokeiluhankkeen kuvaus ... 3
2.1. Taustatutkimus ... 3
2.2. Määrittely ... 5
2.2.1. Hypoteesi ... 5
2.2.2. Spesifikaatio ... 12
2.2.3. Toteutus ... 15
2.3. Toteuma ... 16
2.3.1. Hyödynnettävyys ... 18
2.3.2. Viestintä ja tulosten jakaminen ... 18
3. Kehittämistarpeet ... 19
3.1. Potentiaaliset hyödyntämiskohteet ... 19
3.1.1. Suunnittelun sisällöllinen analyysi ... 19
3.1.2. Prosessin toteutuman analyysi ... 19
3.1.3. Reflektio ja prosessioppiminen ... 19
3.1.4. Ulkoisen tiedon sanitointi ... 20
3.1.5. Suunnittelun joukkoistaminen ... 20
3.1.6. Kaavan valmistelu ja vuorovaikutus ... 20
3.1.7. Hyödyntäminen yleiskaavatasolla ... 21
3.1.8. Kaavatasojen välinen linkitys ... 21
3.1.9. Työryhmätyö ja visiointi ... 21
3.2. Jatkokehitys ... 21
4. Lähteet ... 21

1. Johdanto

Maankäytön suunnittelulla on laajoja ja kauaskantoisia konkreettisia vaikutuksia rakennettuun ympäristöön erityisesti kunnissa tapahtuvan kaavoituksen tasolla. Julkinen keskustelu ja poliittinen päätöksenteko keskittyvät suurilta osin eri vaiheissa esiteltävien suunnitelmien laadun, validiteetin ja kattavuuden arviointiin. Itse ongelmanratkaisun ja tiedonjalostuksen prosessi, jonka seurauksena suunnitelmia syntyy, jää kuitenkin usein vähälle huomiolle.

Tämä yhteistyössä Helsingin Kaupunkiympäristön (KYMP) kanssa toteutettu kokeiluhanke rakentuu tutkija Otso Heleniuksen Aalto-yliopistossa tekeillä olevan väitöskirjan pohjalle. Väitöskirjan aiheena on spatiaalisessa suunnittelussa käytettävän teknologian kehittäminen. Kokeiluhankkeessa on kehitetty useassa kaupungissa kerättyyn empiiriseen aineistoon pohjautuen hypoteesi työkalusta ja työtavasta, jonka tarkoituksena on yhtenäistää, keventää ja johdonmukaistaa kaavaprosessien tiedonhallintaa. Kokeiluhankkeen tavoite on täten linjassa KYMP:n toteuttamien Kaavapino- ja Asemakaavat yhteisenä tietovarantona -kehityshankkeiden kanssa. Kyseisten hankkeiden kanssa on tehty yhteistyötä ennen hanketta sekä sen aikana, ja yhteistyö jatkuu myös hankkeen loputtua.

Tässä loppuraportissa kerrataan hankkeen tavoite ja kuvataan toteutus, tulokset ja poikkeamat suhteessa alkuperäiseen tavoitteeseen. Lopuksi käydään läpi havaitut jatkokehittämistarpeet.

2. Kokeiluhankkeen kuvaus

2.1. Taustatutkimus

Hankkeessa luotu hypoteesi pohjautuu useassa suomalaisessa kaupungissa (Helsinki, Espoo, Vantaa, Jyväskylä, Lahti), sekä Australiassa ja Hollannissa toteutettuihin suunnitteluprosesseja ja tiedonhallintaa koskeviin haastatteluihin sekä täydentävän tutkimusaineiston keruuseen. Aineiston analyysissa on tunnistettu riskirakenteita, jotka altistavat kaavaprosessit virheille, heikentävät niiden ennakoitavuutta ja lisäävät kuormittavuutta. Yhteisenä nimittäjänä näille ongelmille on tiedonhallinnan epäjatkuvuus. Tietoa tuodaan prosessiin, hallitaan, muokataan sekä tuotetaan lisää useissa eri muodoissa, ja valtaosa näistä vaiheista vaatii teknologisen tuen puuttuessa manuaalista tulkintaa tai konversiota.

Tutkimusaineistosta johdetut olennaisimmat prosessia koskevat riskit ovat:

- Suuri osa prosessiin tuotavasta sekä sen aikana tuotetusta tiedosta ei ole strukturoitua (rakenteellista, ts. tietyn skeeman mukaista).
- Kyseiseen tietoon tehdyt muutokset (ml. aineistosta johdettu uusi tieto) ei versioidu automaattisesti (sisällä viitettä lähteenä olleisiin tietojoukkoihin, etenkään viittauksen kohteena oleviin joukkojen osiin).
- Suunnitteluaineiston osien keskinäisen johdonmukaisuuden varmistaminen ja virheiden löytäminen on yllä mainittujen syiden vuoksi haastavaa.

- Tavoitteet ja reunaehdot parhaiten täyttävien ratkaisujen iteratiivinen kehittäminen tapahtuu pitkälti näkymättömissä kaavavaiheiden sisällä. Kaavavaiheesta toiseen siirryttäessä suunnitelman iteratiivinen liikkumavara suhteessa suunnitteluympäristön muutoksiin kapenee. Tämä rajoite ohjaa kaavojen nopeaan läpivientiin suunnittelussa tehtyjen oletusten ja toteutusvaiheen välisen divergenssin minimoimiseksi. Samalla se kuitenkin myös kasvattaa riskiä divergenssille mahdollisten kiireestä johtuvien virheiden ja puutteiden kautta.

Yllä mainitut riskit eivät ole vain teknisiä tuotantotehokkuuteen vaikuttavia tekijöitä, vaan niillä on vaikutusta myös prosessin laadulliseen puoleen. Tiedon monimuotoisuus ja sen hallintaa tukevien työkalujen puute korostaa subjektiivisen tulkinnan roolia, mikä voi johtaa tiedostamattomiin vinoumiin merkittävien tietomäärien käsittelyssä. Manuaalisen työn suuri määrä taas voi pakottaa hylkäämään vaihtoehtoisten suunnitteluratkaisujen tutkimisen tai rinnakkaisen edistämisen aika- tai muiden resurssirajoitteiden vuoksi. On olennaista huomioida, että edellä mainitut prosessiriskit eivät ole vain kaavoitustyön ja -organisaatioiden erityispiirteitä, vaan ne koskevat yhtä lailla konsultteja, viranomaisia sekä muita tahoja.

Tutkimuksessa on saatu selville, että osa kaavoitusprosessin lineaarisuudesta ja iteraation välttämisestä on seurausta ulkoisista syistä, kuten tilattavien selvitysten korkeista kustannuksista tai aikatauluriippuvuudesta, sekä muiden rinnakkaisten prosessien ja osallisten asettamista rajoitteista. Kokeiluhankkeen vaikutusmahdollisuuksien ulkopuolella olevat kaavaprosessien laatuun vaikuttavat tekijät on kuitenkin otettava huomioon kehitystyöhön vaikuttavina sitä rajaavina ei-toiminnallisina vaatimuksina (engl. *non-functional requirements*).

Tutkimusaineistosta johdetut olennaisimmat työskentelyä koskevat riskit ovat:

- Työkalut eivät usein ole yhteentoimivia prosessin kontekstissa muutamaa poikkeusta lukuun ottamatta (esimerkiksi paikkatietoaineistoon viittaaminen suoraan Microstationissa).
- Kaavoituksen ydintyön suorittamiseen ei ole sen monimuotoisuuden ja sisällöllisen laajuuden vuoksi olemassa valmiita tarkoitukseen kehitettyjä COTS (engl. *Commercial Off-The-Shelf*) -ratkaisuja, mikä on ohjannut sekä yksilö- että ryhmätasolla luomaan hyvin vaihtelevia ammatilliseen kokemukseen pohjautuvia työkaluvalintoja ja niiden ympärille rakentuneita työkäytäntöjä.
- Työkaluiksi on monissa tapauksissa valikoitunut ohjelmistoja tai välineitä (esim. kynä ja skissipaperi), jotka mahdollistavat suunnitelman sisällön jalostamisen ja sillä kommunikoinnin joustavasti ilman esimerkiksi käsiteltävän tiedon skeemasta tai ohjelmiston pakottamasta käyttötapauksesta (engl. *use case*) johtuvia rajoitteita.
- Prosessissa on väärällä tavalla kognitiivisesti kuormittavia tehtäviä: ydintyöhön keskittymisen sijaan prosessissa joudutaan käyttämään aikaa manuaaliseen tiedon etsimiseen, konversioon, tulkitsemiseen ja validointiin. Nämä ongelmat ovat seurausta edellä mainituista valinnoista.

- Prosessiin osallistuvilla ei ole mielekästä insentiiviä tuottaa resurssien puitteissa strukturoitua ja linkitettyä tietoa, sillä manuaalisesti toteutettuna se ei tarjoa välittömiä hyötyjä ja heikentää tuottavuutta.
- Prosessin lineaarinen rakenne ohjaa keskittymään tuloksiin ja hyödyntämään tuotannossa *ad-hoc* -metodeja, jotka – kuten edellä mainittiin – nojaavat kokemukseen. Kokemuksellisen osaamisen tulkinta eksplisiittiseen muotoon on usein haastavaa tai mahdotonta jopa itse sen hyödyntäjälle.
- Edellä mainittujen seikkojen vuoksi parhaita käytäntöjä on vaikeaa tai mahdotonta verrata ja siirtää yksilöltä tai ryhmältä toiselle, joten organisaatiotason hyöty jää saamatta.

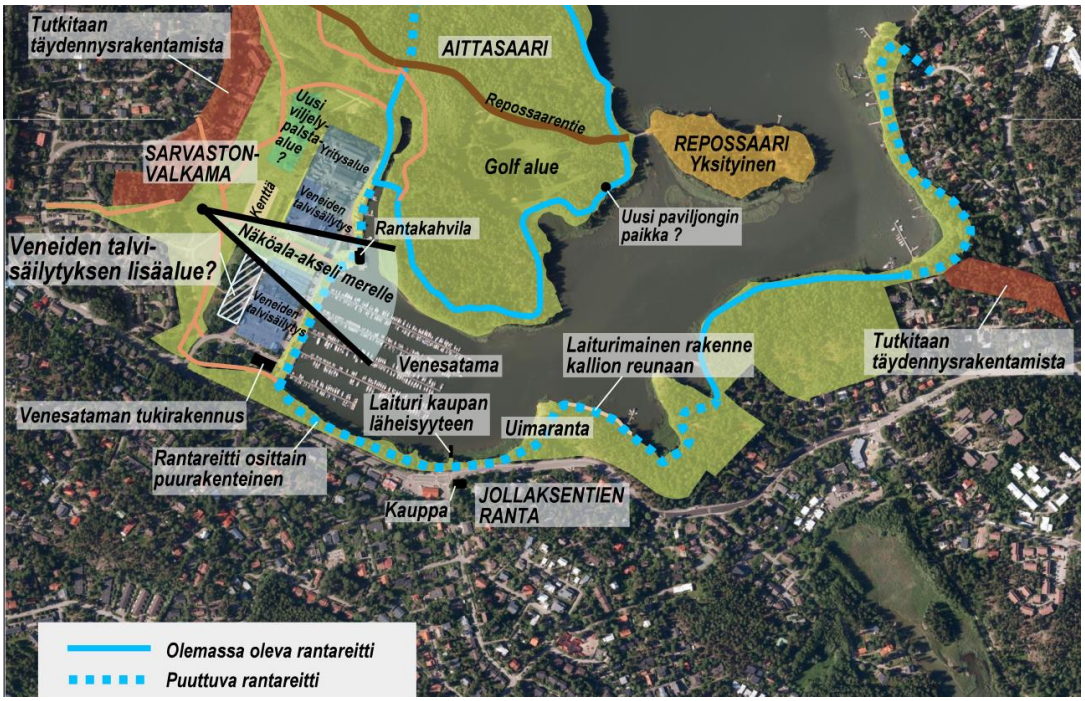
Kaavaprosessien käytännön operatiivisen tason tarkastelu ja niiden toiminnan selvittäminen on olennaista, sillä vasta operatiivisella tasolla ylempi ohjaava informaatio, reunaehdot ja tavoitteet muuttuvat konkreettisiksi suunnitelmiksi. Operatiivisella tasolla tehdyt valinnat propagoituvat läpi kaavaprosessin ja vaikuttavat epäsuorasti myös rinnakkaisiin ja ylemmän tason prosesseihin. Maankäytön suunnittelua ei siis voida tarkastella puhtaan hierarkkisena vaan pikemminkin kaksisuuntaisena prosessina.

2.2. Määrittely

2.2.1. Hypoteesi

Tutkimuksen pohjalta kokeiluhankkeen kehityskohteiksi valittiin kaksi prosessin ongelma-aluetta: 1) strukturoimaton tieto ja 2) tietoon kohdistuvat muutokset. Ottaen huomioon suunnitteluprosessin riippuvuuksista johtuvat rajoitteet sen rakenteen muokkaamisessa sekä joidenkin ohjelmistojen tiiviin integraation prosessiin (mm. kaavatietomallin mukainen kaavapiirtäminen, asianhallinta- ja projektinhallintajärjestelmät), todettiin suurimman liikkumavaran ja optimointipotentiaalin olevan kaavan sisällön jalostamisessa ja siihen liittyvässä yksilö- ja ryhmätason tiedonhallinnassa. Tällä tavoin oletettiin myös pystyttävän tehokkaimmalla tavalla edesauttamaan MRL §5 suunnittelua koskevien tavoitteiden täyttymistä ja ylläpitämistä.

Kuten edellä todettiin, varsinaisia kaavoituksen sisällöntuotannon erityistarpeisiin sopivia COTS-ratkaisuja ei ole. Sisällöntuotannolla tarkoitetaan tässä yhteydessä kaavakartan ja -selostuksen tuotantoon tähtäävää toimintaa. Tähän liittyviä työtehtäviä ovat mm. kaava-aluetta koskevien ideoiden hahmottelu kartalle ja niihin liittyvien attribuuttien tai muiden tietojen liittäminen karttamuotoiseen kuvaukseen (nk. *suunnittelupiirtäminen*, ks. kuva 1) sekä suunnittelutilannetta selostavan tai ratkaisuja argumentoivan tekstin, kaavioiden ja muun materiaalin luonnostelu. Näitä tehtäviä kuvataan tutkimuskirjallisuudessa yleisnimikkeellä *sketch planning*.



Kuva 1. Esimerkki asemakaavaprojektin (Laajasalon itäranta) OAS-vaiheen suunnittelun lähtökohtia varten tehdystä tavoitteita kuvailevasta kartasta.

Nämä tehtävät toistuvat monesti lähes läpi koko kaavaprosessin ja muodostavat tärkeän usein toistuvan tehtävän, jota hyödynnetään tulkittaessa lähtötietoja helpommin hyödynnettävään muotoon, viestittäessä ja neuvoteltaessa erilaisista ideoista muiden suunnitteluun osallistuvien tahojen kanssa esimerkiksi kokouksissa, ja kehiteltäessä reunaehtoihin sopivia ratkaisumalleja.

Kokeiluhankkeen kehityskohteena ovat siis yhtä lailla itse prosessi kuin siinä käytettävät työkalutkin. Hankkeen tarjoamassa ratkaisumallissa olennaista on se, että se ei pyri arvottamaan prosessissa tehtäviä ratkaisuja tai ohjaamaan niiden tekemistä, vaan pyrkii tarjoamaan tukea agnostisesti. Ratkaisumallin tehtävänä on ainoastaan pyrkiä varmistamaan prosessin maksimaalinen sisäinen johdonmukaisuus siten, että kaikki relevantti tieto

- säilyy läpi prosessin,
- on käytettävissä, kun sitä tarvitaan,
- on myös muiden kuin sen tuottaneen tahon ymmärrettävissä,
- on sidottu tiettyyn ajanhetkeen ja tekijään, sekä
- on jäljitettävissä yhteen tai useampaan aiempaan tietojoukkoon.

Kuten edellä on mainittu, tutkimuksessa ilmi tulleet tiedonhallinnan haasteet eivät ole kaavoituksen erityispiirteitä, vaan ne koskevat kaikkia suunnittelua ja tiedonhallintaa tekeviä organisaatioita. Tämän vuoksi ratkaisumallin spesifikaatiota luodessa katsottiin aiheelliseksi tarkastella aihetta laajemmin.

Tiedonhallinta nojaa alasta riippumatta nykyisellään suurelta osin joko COTS- tai asiakasta varten toteutettuihin ohjelmistoihin, joten pääasialliseksi kysymykseksi nousee, miten tiedonhallinnan vaatimuksia (engl. *requirements*) on kyseisten ohjelmistojen tuotannossa tulkittu ja käännetty formaaliin muotoon (lähdekoodiksi). Hankkeessa päädyttiin tekemään kaava- ja ohjelmistotuotannon prosessien suora vertailu, sillä aiempi kokemus ja ohjelmistotuotannon tutkimuskirjallisuuden katsaus antoivat aihetta olettaa kyseisten prosessien välillä olevan paljon yhtäläisyyksiä.

Hypoteesin aikana johdetut olennaisimmat yhtäläisyydet kaava- ja ohjelmistotuotannon välillä ovat:

- Tavoitteet ja reunaehdot on ilmaistu luonnollisella kielellä. Täten niitä ei voida yksikäsitteisesti kääntää valmiiksi kaavaksi tai ohjelmistoksi, vaan prosessiin sisältyy väistämättä tulkintaan liittyvää epävarmuutta.
- Myöskään kaavan tai ohjelmiston formaalin sisällön tuottaminen ei ole puhtaasti rationaalista toimintaa, vaan sisältää runsaasti kokemukselliseen ammattiosaamiseen, käytäntöihin ja priorisointiin pohjautuvia valintoja.
- Tuotantoon osallistuu usein suuri joukko eri alojen asiantuntijoita, joiden tuottama sisältö ei usein ole yhteismitallista.
- Poikkeavista osaamisalueista ja intresseistä johtuen osallisten välille on usein mahdotonta muodostaa yhtenäistä täysin ristiriidatonta kuvaa kehityksen kohteen tarkasta tavoitteesta, toteuttamistavasta ja sisällöstä.
- Tavoitteet ja reunaehdot voivat muuttua ulkoisten tai sisäisten syiden vuoksi kesken projektin. Mahdollisia syitä ovat mm. kohdeympäristön muutokset, aikataulu- tai budjettirajoitteet, tekniset esteet tietyn tavoitteen ja reunaehtojen yhteensovittamisessa, tai edellä mainitut osallisten väliset tulkintaerot.

Ohjelmistotuotannossa on menneinä vuosikymmeninä noudatettu *separation of concerns*-mallia (Boehm 2003) ja keskitytty ohjelmien spesifikaatioiden formaaliin toteutukseen (ohjelmointiin), mutta itse spesifikaatioiden luominen ja niiden pohjana olevien vaatimusten tunnistaminen (engl. *requirements elicitation*), analysointi ja validointi on ollut usein puutteellista. Ohjelmoijien tehtävän on perinteisesti katsottu olevan ainoastaan valmiiden vaatimusten arvoneutraali kääntäminen lähdekoodiksi. Tämän seurauksena etenkin suurista ohjelmistoprojekteista jopa puolet on epäonnistunut tavoitteiden saavuttamisessa, ylittänyt budjetin tai aikataulun tai keskeytetty epäonnistuneina (kts. esim. Standish Groupin CHAOS-raportit).

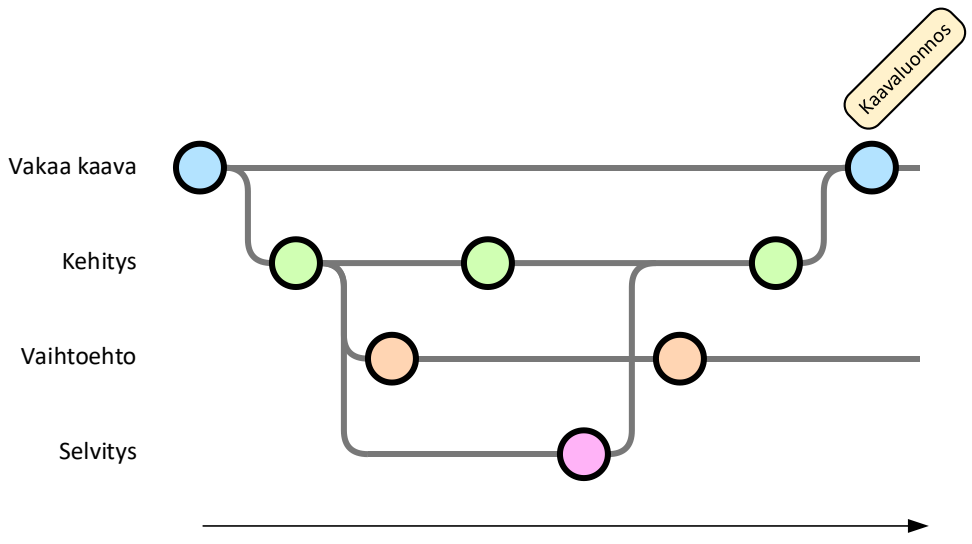
Rinnakkainen ja projektien onnistumisen kannalta yhtä merkittävä ongelma on ollut rakennettavien ohjelmien rakenteen kompleksisuuden ja lähdekoodin määrän kasvu. Kehittäjän yhteen komponenttiin tekemä muutos saattaa riippuvuuksien kautta vaikuttaa lukuisiin ohjelman muihin komponentteihin. Usean kehittäjän rinnakkain tekemät muutokset voivat aiheuttaa ennakoimattomia ja vaikeasti jäljitettäviä virheitä, ja näin hidastaa kehitystyötä. Näiden

kahden ongelman ratkaisuun on luotu useita eri kehitysmalleja, jotka keskittyvät joko tiedonhallintaan (lähdekoodi ja sitä koskevat muutokset) tai tietoa käsittelevien tahojen yhteistoiminnan parantamiseen.

Yksi merkittävimpiä koko ohjelmistotuotannon alaan vaikuttavia keksintöjä on ollut *versionhallinta* (myös *revisiointi*), erityisesti myöhemmin kehitetty *hajautettu* versionhallinta. Nykyään voidaan yleistäen sanoa käytännössä kaiken ohjelmistotuotannon toimivan versionhallinnan varassa. Yksi laajimmin käytetyistä ja tunnetuimmista versionhallinnan työkaluista on Linus Torvaldsin perustama Git-projekti. Versionhallinnalla tämän projektin kontekstissa tarkoitetaan ohjelmistoa, joka mahdollistaa

- tietojoukkoon tehtyjen muutosten säilömisen pysyvästi graafimuotoiseen kantaan (engl. *repository*). Talletettua muutosta kutsutaan yleensä termillä *commit*,
- mihin tahansa aiempaan muutokseen palaamisen,
- kahden muutoksen sisällön välisten erojen vertaamisen (engl. *diff*),
- mahdollistaa useamman rinnakkaisen version haarauttamisen (engl. *branching*) tietystä muutoksesta,
- rinnakkaisten muutosten yhdistämisen takaisin yhdeksi (engl. *merging*), tai
- kannan monistamisen triviaalisti (engl. *cloning*).

On huomattavaa, että versionhallinnan termeille ei ole vakiintuneita suomenkielisiä käännöksiä. Tässä dokumentissa käytetään edellä mainittuja termejä tilanteesta riippuen kontekstiin soveltuen.



Kuva 2. Mock-up-esimerkki mahdollisesta versionhallinnasta kaavoituksen kontekstissa.

Kuvassa 2 on esitetty kuvitteellinen skenaario kaavoitusprosessin versionhallinnasta:

1) Suunnittelun alussa kaikki tieto on ”Vakaa kaava”-nimisen haaran ensimmäisessä (vasemmanpuoleisin) commitissa (sininen merkki).

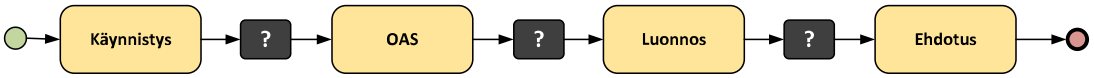
2) Tästä monistetaan uusi ”Kehitys”-niminen haara (*branch*), jossa suunnitelman sisältöä muokataan.
3) Suunnittelun aikana ilmenee myös tarve tutkia vaihtoehtoista suunnitteluratkaisua, joten tälle luodaan oma ”Vaihtoehto”-niminen haaransa, ja Kehitys-haaran viimeisin muutos kopioidaan kyseisen haaran muutosten pohjaksi.
4) Kehitys-haaran ensimmäiseen versioon liittyen on myös tilattu selvitys, jonka sisältö lisätään kyseiseen versioon ja talletetaan selvyyden vuoksi omaan haaraansa.
5) Vaihtoehto-haaran ratkaisu todetaan mahdottomaksi, eikä siinä kehitettyä ratkaisua enää viedä kohti vakaata kaavaa.
6) Selvityksen tuoma lisätieto yhdistetään Kehitys-haarassa päivittyneeseen uuteen muutokseen, ja Selvitys-haara suljetaan (*merge*).
7) Lopuksi todetaan Kehitys-haaran viimeisimmän version olevan valmis ja se yhdistetään kaavan vakaaseen versioon. Kyseinen versio merkitään nk. tagilla merkitsemään, että kyseessä on valmis kaavaluonnos.

Versionhallinnalla on lukuisia etuja:

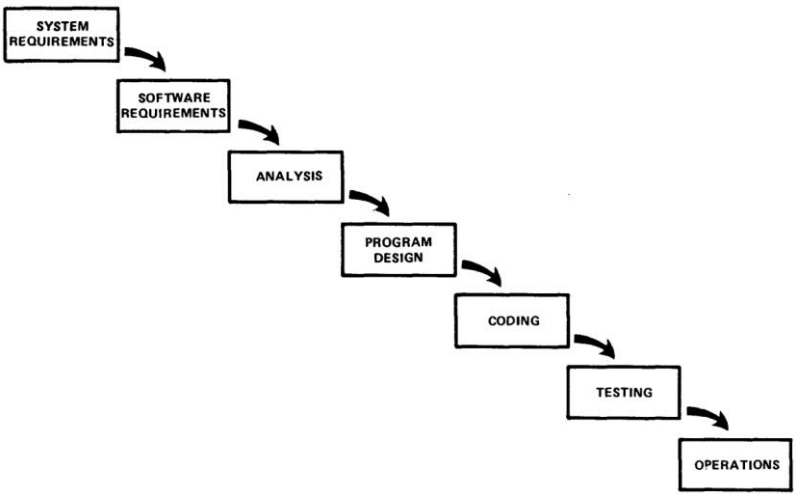
- Käyttäjien ei tarvitse huolehtia tiedostojen manuaalisesta versioinnista tai niiden hajautumisesta eri palveluihin.
- Versioidun tiedon menettäminen on riskinä triviaali.
- Suurempia kokonaisuuksia voidaan jakaa rinnakkain työstettäviksi pienemmiksi osiksi, kuitenkin säilyttäen mahdollisuus osien joustavaan synkronointiin ja yhdistämiseen (*branching* ja *merging* -toiminnallisuus).
- Ristiriitaisia tai päällekkäisiä muutoksia voidaan työstää toisistaan eristettyinä kunnes ne pystytään hallitulla tavalla yhdistämään.
- Versionhallinnan kautta tehty työ on itsedokumentoivaa muutosten sisällön, tekijän ja ajanhetken osalta.
- Versionhallinnassa tehty työ on läpinäkyvää ja mahdollistaa nopean tilannekuvan muodostamisen (*diff* sekä itse versiopuu metatietoineen).
- Edellä mainittujen ominaisuuksien seurauksena versionhallinta vähentää prosessin henkilöriippuvuuksia ja antaa lukuisia mahdollisuuksia tehokkaalle prosessin organisoinnille.

Prosessin organisointi taas liittyy edellä mainituista kahdesta kehitysmalleista toiseen (yhteistoiminnan parantaminen). Yksi suurimpia ohjelmistoprojektien epäonnistumiseen johtaneita syitä on ollut oletus siitä, että asiakkaan prosesseja voidaan parantaa ja tehostaa vain kuvaamalla ne pääpiirteittäin ”riittävällä” tarkkuudella (kts. Kuvat 3 ja 4) jättäen operatiivisen tason yksityiskohdat käsittelemättä, ja vaihtamalla operatiivisella tasolla käytettävä työkalu ”parempaan”. Metodi on kuitenkin mahdoton, sillä uusi työkalu väistämättäkin muuttaa prosessin rakennetta (Nuseibeh & Easterbrook, 2000). Vaikutukset voivat olla hankalasti ennustettavia, sillä operatiivinen taso kytkeytyy usein lukuisiin muihin prosesseihin ja työkaluihin. Tämä voi johtaa vaikeasti ennustettaviin ristiriitoihin ja epäyhteensopivuuksiin, ja vaikuttaa sekä prosessin kuvatun tason toimintaan että itse prosessin lopputuloksen laatuun.

Tästä syystä minkä tahansa prosessityökalun korvaaminen toisella onnistuneesti vaatii myös itse prosessin evaluointia, uudelleenkoostamista ja uuden prosessin verifiointia käytännössä.



Kuva 3. Perinteinen kaavoitusprosessi, jossa säädeltyjen virallisten osien välissä tapahtuvan sisällöntuotannon menetelmät jätetään usein kuvaamatta (engl. black-boxing) ja prosessin oletetaan etenevän suunnitelman osien lukkoon lyömisen kautta lineaarisesti kohti loppua.



Kuva 4. Perinteinen ohjelmistokehityksen vesiputousmalli (engl. waterfall model) perustuu oletukselle siitä, että jokaisen vaiheen edeltäjä on tehty valmiiksi ja sen sisältö on lyöty lukkoon eikä siihen ole tarvetta palata (Benington, 1983).

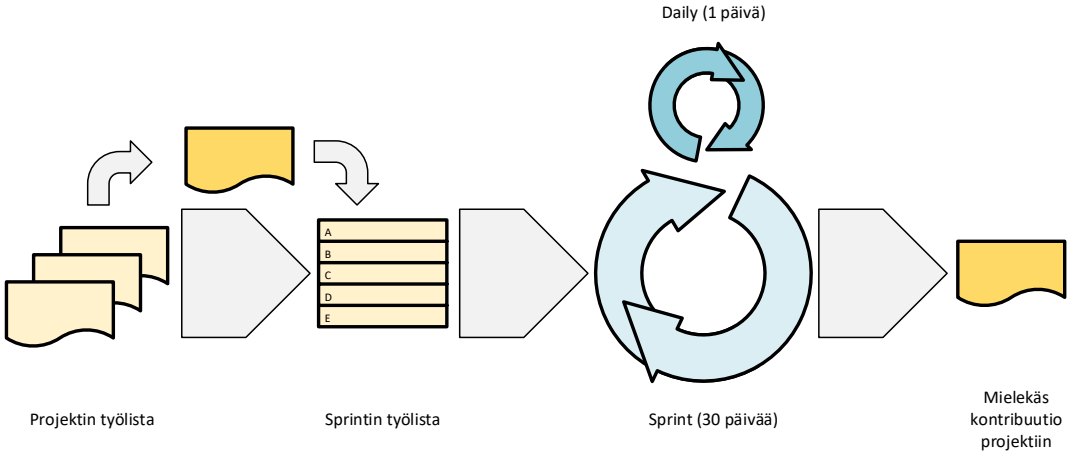
Miten suunnitteluprosessin rakenne tulisi siis määritellä, kun sisällönjalostus toimii versionhallinnan varassa? Ratkaistavia kysymyksiä ovat muun muassa:

- Ketkä suunnitteluun osallistuvat ovat velvoitettuja/oikeutettuja tallentamaan muutoksia (”*tekemään committeja”*) versionhallintaan?
- Millaiset rajat tallentamisen frekvenssille tulisi asettaa (esim. ”ei useammin kuin kerran päivässä” tai ”ei harvemmin kuin kerran viikossa”)?
- Millaiset rajat muutoksen koolle ja sisällölle tulisi määrittää, jotta talletettavat muutokset ovat prosessinäkökulmasta hyödyllisiä ja käyttäjänäkökulmasta helposti tulkittavia (esim. ”commit tehdään aina kun suunnitelmassa tapahtuu vähintään korttelitasoon kohdistuvia muutoksia” tai ”useampaan kortteliin kohdistuvat muutokset tehdään selkeyden vuoksi omina commiteinaan”)?

Kuvassa 1 on kuvattu yksi mahdollinen periaate työn jakamiseen, mutta tämän lisäksi tarvitaan myös toimiva yksilö- ja ryhmätyöprosessi. Perinteisen vesiputousmallin tilalle on kasvavissa määrin syntynyt muun muassa ketteriä (engl. *agile*) kehitysmetodeja. Tunnetuimpia ovat

SCRUM (kts. kuva 5), Kanban sekä Lean. Metodeilla on useita työn laadun ja tehokkuuden kasvattamiseen tähtääviä periaatteita:

- muuttuvien vaatimusten hyväksyminen ja kyky reagoida niihin
- työn pilkkominen hallittaviin pieniin osiin
  (yleensä noin 1-2 päivää kestäviin tehtäviin)
- kehityksen suuntaa lukitsevien valintojen tekeminen mahdollisimman myöhään
- pienien mielekkäiden kokonaisuuksien valmistelu mielekkäässä tahdissa
- odottamisen määrän ja muiden työntekoa haittaavien esteiden minimointi
- tehtävien poiminta delegoinnin sijaan



Kuva 5. Esimerkki SCRUM-metodista. Projektista poimitaan jokin kokonaisuus toteutettavaksi esimerkiksi kuukauden aikana. Kyseinen kokonaisuus pilkotaan tehtäviksi, joita tehdään yhden päivän sykleissä. Jokaisen päivän aikana kaikki kehittäjät kertaavat mitä ovat edeltävänä päivänä tehneet, mitä aikovat tehdä seuraavaksi ja mitä mahdollisia esteitä työn edistämiselle on syntynyt. Ryhmän ohjaajan tehtävänä on varmistaa syklin toimivuus hallitsemalla työlistaa, auttaa esteiden purkamisessa ja edesauttaa projektin säännöllistä edistymistä.

Menetelmien sisältöä ei tässä loppuraportissa käydä läpi tämän tarkemmin. Niiden tavoitteet vastaavat kuitenkin hyvin taustatutkimuksessa tunnistettuihin prosessin haasteisiin, ja näin ollen on perusteltua sisällyttää ne hypoteesiin. Kappaleiden 2.1. ja 2.2.1 sisällön yhteenvetona voidaan todeta kokeiluhankkeen hypoteesin olevan:

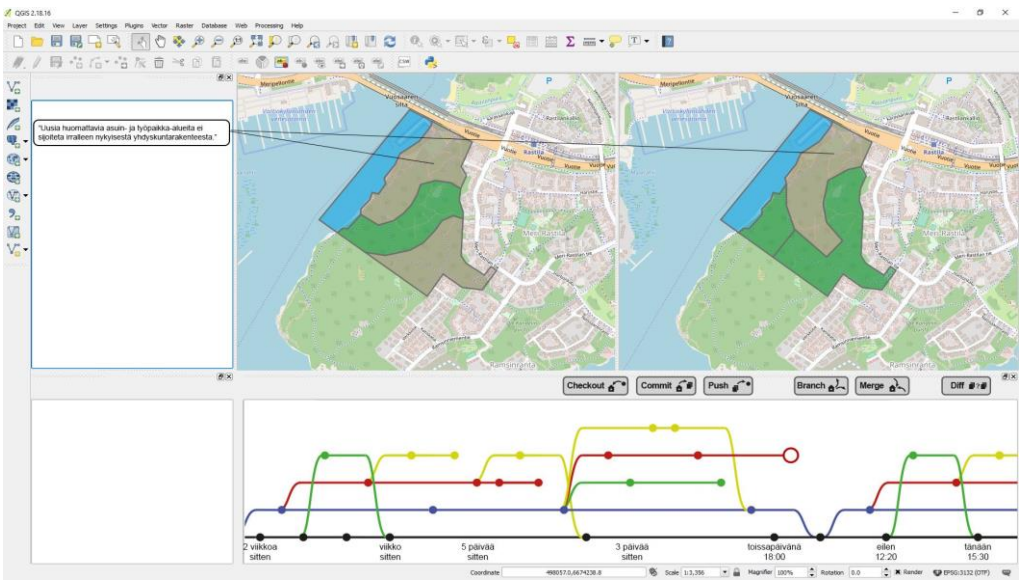
”Kaavoitusprosessin laatua voidaan kehittää johdonmukaistamalla sen tiedonhallintaa versionhallinnan teknologioilla sekä niihin läheisesti liittyvillä yksilö- ja ryhmätyökäytännöillä. Olennainen osa tiedonhallintaa on suunnitelman sisällön kehittymisen seuranta ja jäljitettävyys.”

2.2.2. Spesifikaatio

Hypoteesin perusteella hankkeen tavoitteeksi määriteltiin ohjelmistoprototyypin kehitys. Kerättyyn aineistoon ja analyysiin pohjautuen tehtiin seuraava karkea määrittely:

- Ohjelmisto pyrkii imitoimaan suunnittelupiirtämisen työtehtävää tarjoamalla karttanäkymän spatiaaliselle sisällölle (karttaprimitiivit), sekä muun tai muita näkymiä spatiaalista tietoa kuvailevalle (semanttiselle) tai siihen liittyvälle sisällölle.
- Ohjelmisto tallentaa kaiken sisällön yksinkertaiseen JSON-skeemaan, jonka pääasiallisena tehtävänä on ylläpitää sisältöjen välisiä viittauksia.
- Kyseisten sisältöjen versiointi toteutetaan avoimen lähdekoodin Git-ohjelmiston varaan sen vakauden, laajan tuen ja joustavuuden vuoksi.
- Ohjelmisto tarjoaa muita graafisia Git-työkaluja (mm. gitk, Gitkraken, GitLab) karkeasti vastaavan helppolukuisen näkymän tiedossa tapahtuneisiin muutoksiin.
- Ohjelmiston diff-toiminto avustaa kahden eri muutosjoukon välisten erojen ja yhtäläisyyksien intuitiivisessa tarkastelussa.
- Ohjelmisto tarjoaa yksinkertaiset luonnospiirtotyökalut sisällön luomiseen ja olemassa olevan sisällön kommentointiin (annotointi), sekä kuvailevan sisällön syöttäminen ja muokkaaminen.
- Ohjelmiston on pyrittävä ylittämään tai minimissään saavuttamaan nykyisten suunnittelupiirtämisen metodien intuitiivisuus käyttöliittymän osalta.
- Ohjelmiston käyttö ei saa johtaa suunnittelussa tarvittavien työvaiheiden määrän kasvuun.

Spesifikaation pohjalta luotiin kuvassa 6 esitetty QGIS-ohjelmistoon upotettu mock-up, jossa spatiaalinen ja semanttinen sisältö, niiden välinen viittaus, versioiden vertailu ja suunnitelman rakenne esitetään yhdessä näkymässä.



Kuva 6. Ohjelmiston ensimmäinen mock-up -luonnos.

Spesifikaatiossa ja erityisesti skeeman määrittelyssä erityisen olennaisessa asemassa ovat suunnitelman sisällön kehittymisen seurattavuus sekä henkilöriippumattomuus. Spesifikaatiolla haluttiin varmistaa, että kaavaprojektissa toisinaan tapahtuvat henkilövaihdokset eivät johda tilanteeseen, jossa suunnitelmassa tehtyjen ratkaisujen ymmärtäminen vaatii alkuperäisen projektista lähteneen tekijän konsultointia.

JSON-skeeman määrittely tehtiin seuraavien periaatteiden pohjalta:

- Jokaisella objektilla on attribuutteina
    - uniikki commit-kohtainen tunniste
    - lista [0..n] tunnisteesta, jotka viittaavat objektin edeltäjiin (engl. *ancestors*)
- Spatiaalisilla objekteilla on lisäksi attribuutteina
    - GeoJSON-muotoinen esitys piste-, polku- ja alueprimitiiveillä EPSG 3879-koordinaatistossa. Objektien täytyy olla yhtenäisiä suunnittelualueen osajoukkoja, joten Multi-tyyppiset kokoelmat esitetään erillisinä objekteina. Alueprimitiivi saa sisältää aukkoja (genus > 0).
    - Mikäli kyseessä on lähtötiedoista tai ulkoisesta aineistosta tuotu objekti, vaaditaan linkki objektin lähteenä olevaan ulkoiseen aineistoon (sähköposti, selvitys, Paikkatietovipusen aineisto ym.).
- Semanttisilla objekteilla on lisäksi attribuutteina
    - Tupleista koostuva lista, jonka avulla objektin sisältämän kuvaustekstin osiin voidaan liittää joko julkishallinnossa kehitteillä olevaan maankäytön sanastoon tai yksikön sisäiseen sanastoon sisältyviä termejä sekä tyypitettyä numeerista tai muuta tietoa.
    - Mikäli lista edeltäjistä on tyhjä, vaaditaan linkki objektin lähteenä olevaan ulkoiseen aineistoon, josta objekti on jalostettu (sähköposti, selvitys, Paikkatietovipusen aineisto ym.).
- Jokainen commit sisältää
    - listan spatiaalisista objekteista
    - listan semanttisista objekteista
    - listan viittauksista näiden välillä siten, että jokainen semanttinen objekti viittaa vähintään yhteen spatiaaliseen objektiin.

Objekteille suunnittelun aikana tehtäviä valideja operaatioita ovat:

- Yhden objektin jakaminen kahdeksi tai useammaksi objektiksi
    - Esimerkki: yleisesti asumiskäyttöön määritelty alue jaetaan useampaan alueeseen, joita koskevat tarkemmin määritellyt reunaehdot.
    - Toteutus: alkuperäinen spatiaalinen objekti poistetaan, sen tilalle luodaan n kappaletta uusia, joiden edeltäjäksi merkitään poistetun objektin tunniste. Jos kyseessä on karttaobjekti, jokaiselle luodulle objektille kopioidaan viittaukset poistetun objektin semanttisiin viitteisiin.
- Objektin sisällön muokkaaminen

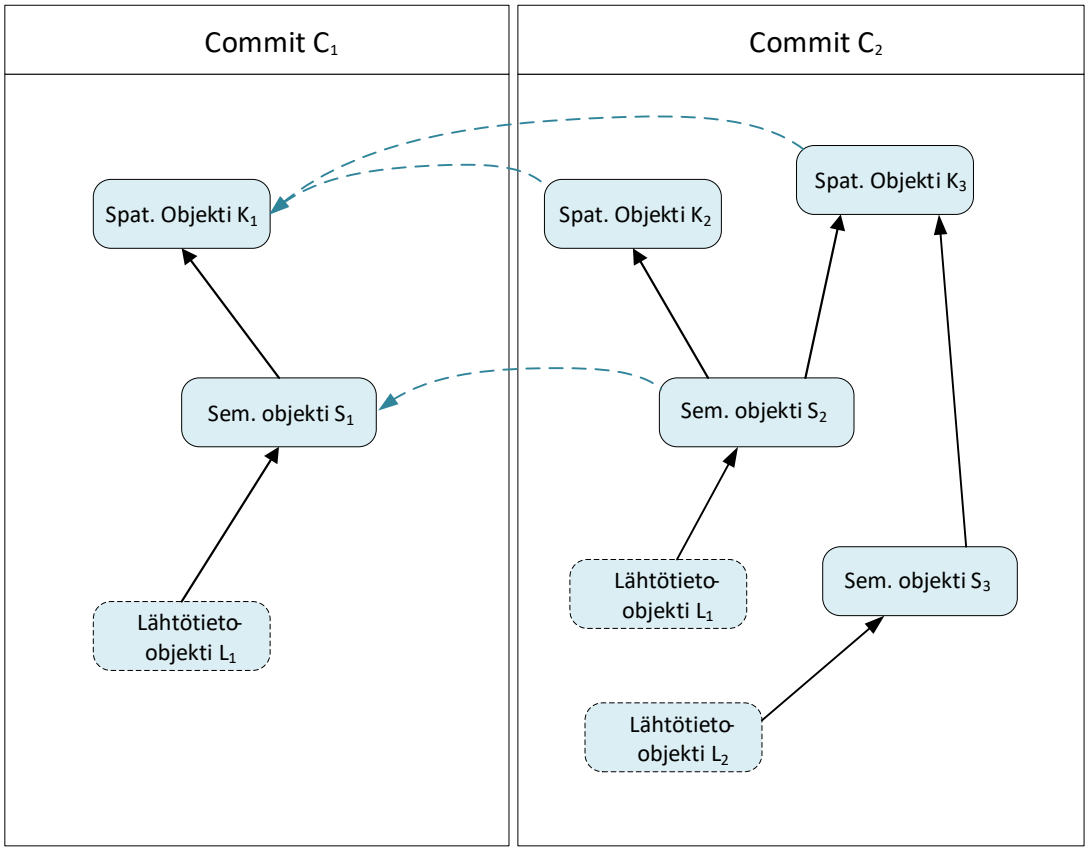
    - Esimerkki: kuvaavaa tekstiä korjataan tai tarkennetaan, karttaobjektin muotoa, sijaintia tai tyyppiä muokataan.
    - Toteutus: ei aiheuta muutoksia objektien välisiin suhteisiin, ainoastaan sisältö päivitetään.
- Kahden tai useamman objektin yhdistäminen yhdeksi
    - Esimerkki: kahdessa eri commitissa suunnittelijat ovat luonnostelleet päämäärältään samantapaista ratkaisua, nämä päätetään yhdistää.
    - Toteutus: alkuperäiset objektit poistetaan, uuden luotavan objektin edeltäjiksi merkitään poistettujen objektien tunnisteet. Jos kyseessä on karttaobjekti, luodulle objektille kopioidaan viittaukset poistettujen objektien semanttisiin viitteisiin.
- Objektin luominen
    - Esimerkki: uusi spatiaalinen objekti luodaan kuvaamaan alueelle suunniteltavaa uutta toimintoa tai lähtötietoaineistosta jalostetaan uusi suunnittelua rajaava reunaehto.
    - Toteutus: karttaobjektin ollessa kyseessä, se piirretään ohjelman tarjoamilla piirtotyökaluilla, jonka jälkeen se linkitetään yhteen tai useampaan semanttiseen objektiin. Semanttisen objektin ollessa kyseessä, sille valitaan luomisen yhteydessä lähde, kuten esimerkiksi yleiskaavamääräys, selvityksessä saatu tulos tai muu ulkopuolinen tieto.
- Objektin poistaminen
    - Esimerkki: semanttinen tai spatiaalinen objekti on muuttunut suunnitelman kannalta tarpeettomaksi.
    - Toteutus: jos spatiaalisen objektin poistamisen jäljiltä suunnitelmaan jää viittauksettomia semanttisia objekteja, ne korostetaan käyttöliittymässä. Semanttisen objektin poistamisen yhteydessä varoitetaan, mikäli operaation jäljiltä suunnitelmaan jäisi viittauksettomia spatiaalisia objekteja. Kyseiset objektit korostetaan käyttöliittymässä.

Objektien tunnisteet ovat commit-kohtaisia ja ne luodaan uudelleen jokaisen commitin tallennuksen yhteydessä (objektin edeltävä tunniste lisätään sen edeltäjät sisältävään listaan). Tämä ehto varmistaa sen, että yhdistettäessä objekteja säilytetään yksikäsitteinen viittaus edeltäjiin.

Skeeman varassa tapahtuva suunnittelu voidaan käynnistää pääpiirteittäin tuomalla ensimmäiseen commitiin lähtötietodokumentteja, joista jalostetaan niihin liittyviä semanttisia ja spatiaalisia objekteja. Suunnitelmaan voidaan esimerkiksi tuoda aluksi suunnittelualueen rajaus (spatiaalinen), alueen alle jäävät yleiskaavan pikselit (spatiaalinen), sekä kyseisiin pikseleihin liittyvät kaavamääräykset (semanttinen) sekä suunnittelualuetta koskeva yleistavoite (semanttinen).

Edellä mainittujen operaatioiden seurauksena objektien ja niiden välisten viitteiden muutoksista muodostuu graafi, jossa jokainen objekti voidaan jäljittää ajassa taaksepäin edeltävien

muutosten kautta yhteen tai useampaan lähtötietona toimineeseen aineistoon. Tämä on yksi skeeman tärkeimmistä tehtävistä (kuva 7).



Kuva 7. Esimerkki objektien varaan muodostuvasta graafista työskentelyn edetessä.

2.2.3. Toteutus

Alkuperäisen projektisuunnitelman mukaisesti tehtiin ensimmäisen kuukauden aikana sarja käyttäjähaastatteluja, joissa pyrittiin käyttöliittymäasiantuntijan kanssa selvittämään mahdollisimman tarkkaan, miten suunnittelupiirtämistä ja suunnittelun päivittäistä tiedonhallintaa tehdään. Haastattelujen aikana kysyttiin myös työkaluihin liittyvistä preferensseistä ja käytössä kohdatuista ongelmista.

Rinnakkain haastattelujen kanssa tehtiin selvitystä teknologiavalinnoista, kuten tuettavista käyttöjärjestelmistä, tarvittavista kirjastoista ja mahdollisesta olemassa olevien työkalujen muokkaamisesta spesifikaatioon sopivaksi. Tarkempi tarkastelu tehtiin QGIS- ja Blender-ohjelmistoille. Kyseiset ohjelmistot valikoituivat tarkasteluun seuraavin perustein:

- Molemmat ovat avointa lähdekoodia.
- QGIS on jo käytössä suunnitteluorganisaatiossa, se tukee natiivisti kaupungin WFS- sekä WMS-rajapintoja, ja sitä on hyödynnetty Kaavatietomalli-projektissa.
- Blender tarjoaa äärimmäisen laajan rajapinnan sekä käyttöliittymän että toiminnallisuuden lähes rajattomaan muokkaamiseen.

Tarkastelun jälkeen molemmat vaihtoehdot hylättiin, QGIS sen vakaan version rajapinnan rajoitteiden ja tuoreemman version epävakauden vuoksi, Blender taas rajapinnan dokumentaation puutteiden vuoksi.

Toteutuksessa päädyttiin hyödyntämään seuraavia avoimen lähdekoodin teknologioita:

- Electron – sovelluskehys, jolla ohjelma voidaan kääntää Windows, Mac kuin Linux-ympäristöön
- Leaflet – nopea projektioita ja WFS/WMS-aineistoja tukeva karttakirjasto
- React – kirjasto käyttöliittymäkomponenttien rakentamiseen
- NodeGit – kirjasto Git-versiohallinnan kantojen käyttöön

Versionhallinnan käyttöä ja skeeman testausta varten päädyttiin lisäksi toteuttamaan Python-ohjelmointikielellä työkalun käyttötapauksia simuloiva työkalu, jolla ohjelman testausta varten voitiin tuottaa nopeasti vaihtelevaa aineistoa. Kyseinen työkalu tekee stokastisia muokkauksia mock-up-suunnitelmaan ja tuottaa niistä commiteja kantaan. Syntynyttä prosessia voidaan sen jälkeen tarkastella versionhallinnassa.

Itse kehitystyö tehtiin iteratiivisesti viikon sykleissä karkeasti SCRUM-metodia mukaillen. Jokaisen viikon alussa pidettiin kokous, jossa toteutumaa verrattiin jäljellä olevaan aikaan ja kehittäjien kanssa tehtiin yhteinen päätös seuraavan viikon alkuun mennessä toteutettavista osista. Samalla jäljellä olevat tehtävät käytiin läpi ja priorisoitiin uudelleen, mikäli sille havaittiin tarvetta.

Projektin edistymiseen vaikutti huomattavasti se, että kehitysajasta vain osa voitiin hyödyntää varsinaiseen implementointiin (ohjelmointiin). Kehitettäviä ominaisuuksia reflektoitiin suhteessa relevanttien saatavilla olevien ja kaavoituksessa jo käytettävien ohjelmien toteutuksiin. Lisäksi suunnittelussa hyödynnettiin aihepiiriä sivuavaa tutkimuskirjallisuutta.

2.3. Toteuma

Projektille määriteltiin projektisuunnitelmassa seuraavat osatavoitteet:

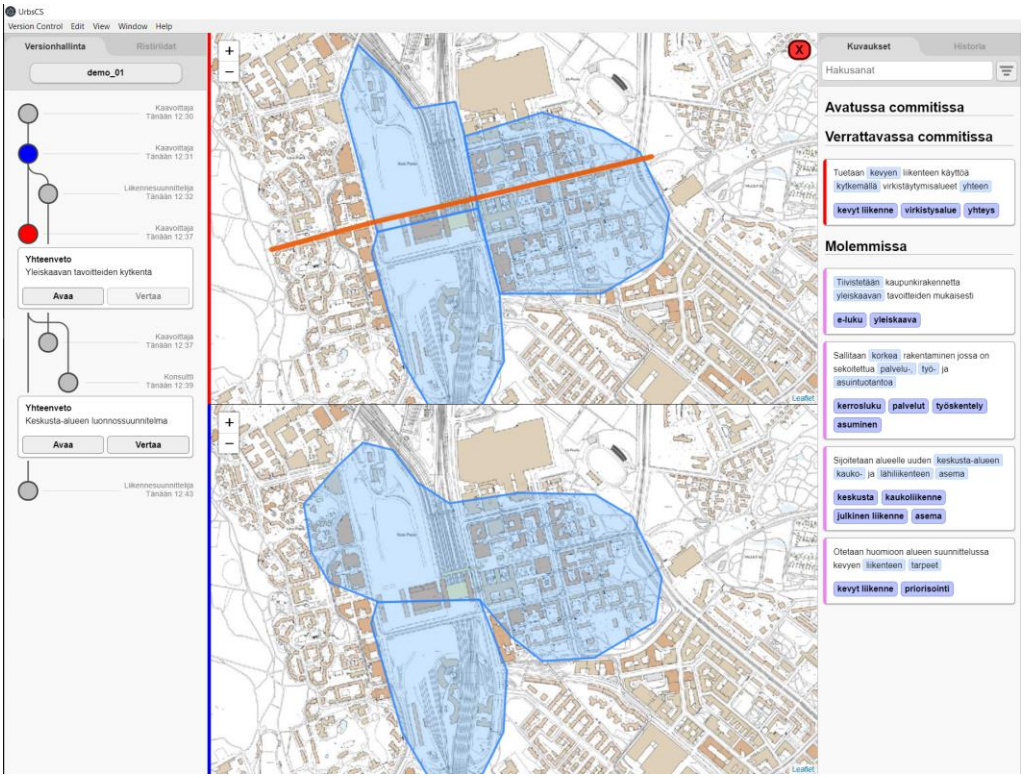
1) suunnitteluprosessin vaiheiden tallentamiseen käytettävän versionhallintatyökalun kehitystyö
2) kyseisen työkalun iteratiivinen testaaminen kehitystyön aikana
3) tutkimushypoteesin mahdollinen validointi ja uudelleenmäärittely

Ensimmäinen osatavoite saatiin projektin puitteissa toteutettua, lukuun ottamatta suunnitelman muokkaukseen tarvittavia työkaluja. Ohjelmiston lopullinen versio kykenee esittämään kantaan tallennetut commitit ja vertaamaan niitä keskenään. Toteumaa voi visuaalisesti verrata alkuperäiseen luonnokseen kuvissa 6 ja 8. Projektisuunnitelman mukaisesti viimeinen ke-

hityskuukausi käytettiin ohjelmistossa ilmenneiden virheiden ja puutteiden korjaukseen ja viimeistelyyn, sillä erotuksella että toteuttajien syksyn aikana tapahtuneen sairastelun vuoksi toteutukseen haettiin yksi kuukausi lisäaikaa (tammikuu 2019).

Toisen osatavoitteen mukaisesti projektiin oli suunnitelmassa mitoitettu myös työkalun testaaminen potentiaalisessa pilottiympäristössä, mutta se jouduttiin rajaamaan pois hankekokonaisuudesta kahdesta syystä: kohdeympäristössä olisi ollut tarjolla soveltuvia pilotointimahdollisuuksia vasta vuoden 2019 puolella, ja lisäksi toteutuksen toiminnallinen vajaus olisi pakottanut jatkamaan kehitystyötä muun rahoituksen varassa pilotointia varten tarvittavien ominaisuuksien implementoimiseksi. Testaukseen liittyvät riskit oli tiedostettu jo alkuperäisessä projektisuunnitelmassa.

Kolmas osatavoite toteutui valitun kehitysmetodin kautta. Kehityssyklien aikana rinnakkain tehdyn tutkimuksen tuloksia verrattiin viikoittaiseen toteumaan ja spesifikaatiota täydennettiin ja korjattiin tehdyn reflektion mukaan.



Kuva 8. Ohjelmiston toteutunut versio ja siinä esitetty mock-up-suunnitelma.

Projektin tuloksiin kuuluvat:

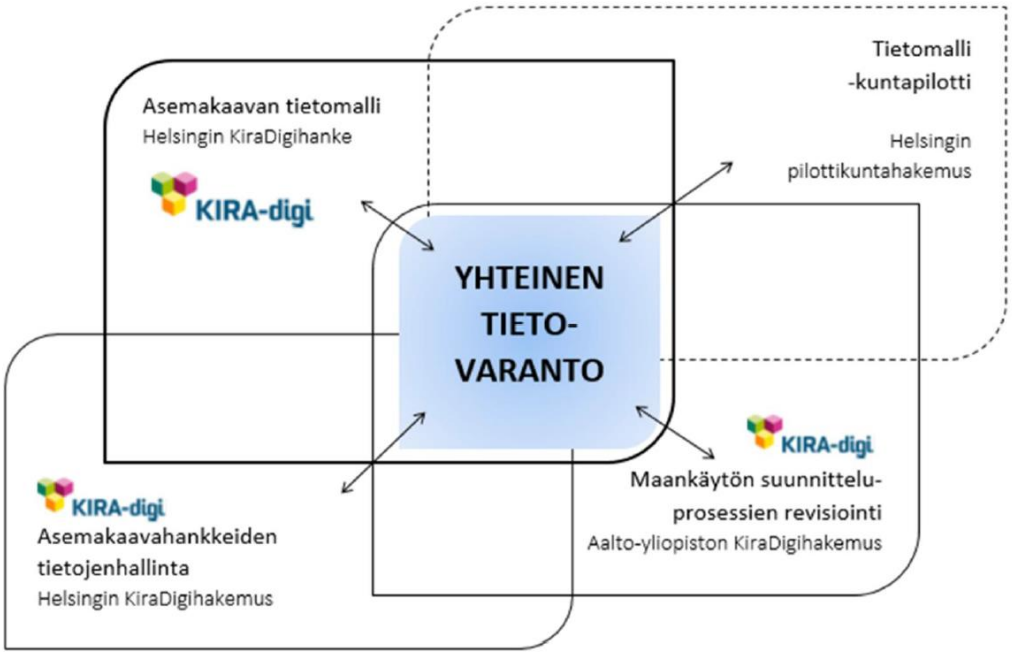
- Syntynyt ohjelmistoprototyyppi
- Tehty tieteellinen (myöhemmin julkaistava) tutkimus

Suhteessa käytettävissä oleviin resursseihin (yksi backend-ohjelmoija, yksi käyttöliittymäkehittäjä ja yksi tutkija projektinvetäjänä), kehitykseen tarvittavan taustatyön määrään ja uuden

suunnittelumetodia hyödyntävän työkalun kehittämiseen liittyvään epävarmuuteen, voidaan projektin katsoa pääosin onnistuneen tavoitteissaan.

2.3.1. Hyödynnettävyys

Projektin tuloksia on käyty läpi kohdeorganisaation (KYMP) kanssa arvioitaessa niiden hyödynnettävyyttä. Luonnostelutyökalujen ja laajempien käyttäjätestien puutteen vuoksi toteutuneen ohjelmiston välitön hyödynnettävyys on vajavainen. Muiden tuloksien välitön hyödynnettävyys on arvioitu hyväksi, sillä niitä voidaan hyödyntää suunnitteluprosessien kehittämistyössä.



Kuva 9. Kuvaus hankkeiden yhteistyöstä suhteessa toisiinsa ja yhteiseen hyödynnettävään ja tuotettavaan tietoon (Arja Kasanen).

Käytyjen keskustelujen pohjalta on todettu tulosten muodostavan tärkeän pohjan kaavoituksen tiedonhallinnan ja prosessien kehittämiselle (kuva 9). Tiivistä yhteistyötä hankkeiden välillä jatketaan kokeiluhankkeen päätyttyä, ja tulosten hyödynnettävyyden odotetaan kasvavan merkittävästi tämän myötä.

2.3.2. Viestintä ja tulosten jakaminen

Projektin tuloksista tullaan viestittämään Ympäristöministeriön MRL-uudistuksessa toimiville tahoille, erityisesti Maankäyttöpäätökset-työryhmälle, haastatteluja antaneille suunnitteluorganisaatioille sekä julkiselle yleisölle. Laajempi tuloksista tiedottaminen tullaan kohdistamaan vaiheeseen, jossa projektin integraatio on saatu vietyä riittävän pitkälle kuvassa 9 esitettyjen yhteistyöprojektien kanssa. Projektin aikana syntyneet tutkimustulokset julkaistaan avoimesti niiden läpäistyä akateemisen vertaisarvioinnin.

3. Kehittämistarpeet

3.1. Potentiaaliset hyödyntämiskohteet

Projektin aikana skeemaan pohjautuen tunnistettiin lukuisia käyttökohteita siihen tallentuvalle suunnittelutiedolle. Projektin kehitystyön sekä sen rinnalla käytyjen keskustelujen myötä kirjattiin alla mainitut jatkokehitystarpeet.

3.1.1. Suunnittelun sisällöllinen analyysi

Semanttisten objektien tyypityksen ja sanastointegraation ansiosta suunnitelman sisällön kehitystä voidaan seurata. Tämä mahdollistaa esimerkiksi rajatun attribuuttien välisen automaattisen ristiriitojen paljastamisen, tai toteutuneen suunnitelman ja sen lähtötilanteen sisältöjen vertaamisen. Suoran tilastollisen analyysin lisäksi voidaan hyödyntää NLP (Natural Language Processing) -ohjelmistokirjastoja sisällön semanttiseen tarkasteluun. Analyysiin voidaan myös yhdistää spatiaalisen sisällön ominaisuuksia (pinta-ala, muoto, etäisyydet, topologiset suhteet jne.).

Sisällöllistä analyysia voidaan käyttää sekä suunnittelun sisällön johdonmukaisuuden varmistamisen automatisointiin, suunnitelmien jälkikäteiseen laadunvarmistukseen, laajempaan useiden suunnitelmien sisällön tilastolliseen analysointiin esimerkiksi yleiskaavan toteuman seuraamisessa ja ohjauksessa.

3.1.2. Prosessin toteutuman analyysi

Edellä mainitun sisällön yhdistäminen versionhallinnan muutoshistoriaan tarjoaa keinoja seurata prosessin eri osien ja vaiheiden kestoa sekä kulkua. Prosessista voidaan identifioida usein toistuvia sisältöön liittyviä aiheita tai esimerkiksi osallisten välisiä tiedonhallinnan menetelmiin liittyviä rakenteita, jotka indikoivat viivästyksiä. Hyödyntämällä versionhallintaa kaavaprosessien tiedonhallinnan sydämenä voidaan projektien aikataulutuksen estimointia parantaa ja vahvistaa valmiuksia ennakoida tietyn tyyppisiin suunnittelutilanteisiin tai osallisiin liittyviä erityistarpeita. Versionhallinnan metatiedoista (muutosaika, tietosisältö, tekijä) voidaan johtaa osittainen prosessikuvaus, joka voidaan kääntää BPMN-kuvauskielelle ja täydentää seurantatutkimuksilla. Tason 3 BPMN-kuvaus on suoritettavissa (esim. Camunda Workflow Enginellä), joten mallinnettua prosessia voidaan simuloida mm. polkujen ja aikataulutuksen analysoimiseksi.

3.1.3. Reflektio ja prosessioppiminen

Sisältö- ja prosessianalyysi ovat teknologian mahdollistamia johdannaisia hyötyjä, mutta jo pelkkä työkalun onnistunut integraatio suunnitteluprosessiin tekee siitä reflektiivisen: suunnittelun tiedonkulun ja sisällön tekeminen näkyväksi antaa välitöntä palautetta prosessin rakenteesta ja auttaa tunnistamaan sekä jalkauttamaan hyväksi havaittuja käytäntöjä organisaation käyttöön laajemmin.

Olennaisinta prosessin laadun kehittämisen kannalta onkin sellaisen mekanismin (esimerkiksi versionhallinta) integrointi, joka strukturoi työtä muttei kahlitse sen rakennetta.

3.1.4. Ulkoisen tiedon sanitointi

Skeeman määrittelyä laajentamalla kaavoitusta tekevä organisaatio voi ohjata kaavoitukseen osallistuvia tahoja tuottamaan käytettävää tietoa muodossa, joka on organisaatiolle arvokasta. Esimerkiksi selvityksiä tekeviä tahoja voitaisiin ohjata syöttämään tulokset versionhallintaan suoraan rakenteellisessa muodossa rajapinnan kautta.

Tulevaisuudessa voidaan myös visioida osan konsulttityöstä tapahtuvan automatisoituna esimerkiksi siten, että tietyille suunnitelman commiteille tehdään laskennallisina ajoina toteutettavia vaikutusten arviointeja automaattisesti, ja laskennan tulokset tallennetaan suoraan versionhallintaan.

3.1.5. Suunnittelun joukkoistaminen

Mikäli kaavoitustyö siirretään laajemmin versionhallinnan varaan, voi kaupunki tarjota ajantasa-asemakaavan sijaan versiohallinnan kautta vakaan ajantasaisen asemakaavojen haaran käytettäväksi prototyyppiohjelmiston kaltaisen helppokäyttöisen työkalun avulla.

Kaupunki voisi täten hajautetun versionhallinnan periaatteiden mukaisesti antaa suuren yleisön, varjokaavojen tekijöiden ja työryhmien haarauttaa ja jalostaa omia suunnitteluideoitaan sekä julkisesta ajantasa-haarasta että toistensa luomista haaroista. Näin kaavaorganisaatio saa käyttöönsä joukkoistuksen kautta laajan ja orgaanisesti kehittyvän poolin ideoita, joista parhaita voidaan poimia osaksi sekä asema- että yleiskaavoitusta. Versionhallinnan rakenteen vuoksi ideoita voidaan poimia käyttöön hallitusti yksityisissä haaroissa tapahtuvaa varsinaista suunnittelua sotkematta.

Julkisen versionhallinnan rajapinnan voidaan hypoteettisesti myös ajatella toimivan esimerkiksi käyttöliittymänä maanomistajan suuntaan, jonne aloitteen tekijä syöttää alustavat tavoitteensa ja muut tiedot kaavahankkeen aloittamista varten.

3.1.6. Kaavan valmistelu ja vuorovaikutus

Työkalun välittömimmät loppukäyttäjiä koskevat hyödyt kohdistuvat kaavojen valmisteluvaiheeseen, jossa versionhallinnan tarjoamaa mahdollisuutta luonnostasoisen semanttisen karttasisällön tuottamiseen voidaan hyödyntää esimerkiksi konsensushakuisena neuvottelutyökaluna osallisten välillä. Työkalun avulla voidaan nopeuttaa osallisten välistä vuorovaikutusta, vähentää väärinymmärrysten riskiä johdonmukaistamalla tiedon esitystapaa ja tarjota mahdollisimman läpinäkyvä ja ajantasainen tilannekuva.

3.1.7. Hyödyntäminen yleiskaavatasolla

Kaikkien asemakaavaprojektien versionhallinta ja tiheä päivitystahti antaa laajat mahdollisuudet tarjota yleiskaavoitukseen laajaa ajantasaista tietoa kaavoituksen etenemisestä ja seurantatietoa esimerkiksi kerrosalojen toteutumasta ja muista olennaisista määreistä. Yleiskaavatasolla versionhallintaa ja strukturoitua sisältöä voidaan hyödyntää myös yleiskaavan toteuttamisohjelman ja nelivuotisen strategian sisältöjen koostamisessa, päivittämisessä, seurannassa ja analyysissa. Tarkemman hyödyntämispotentiaalin tunnistaminen ja validointi vaatii jatkotutkimusta yleiskaavatasolla sekä kaavatasojen välisellä rajapinnalla.

3.1.8. Kaavatasojen välinen linkitys

Tällä hetkellä suunnittelussa ohjausvaikutus muodostuu useimmiten ylempien kaavatasojen karttojen ja niiden symboleihin liitettyjen määräysten visuaalisesta tarkastelusta ja tulkinnasta. Versionhallintatyökalun avulla ylempien tasojen ohjausvaikutus voitaisiin sitoa objektien linkityksen kautta suunnittelun sisältöön muun lähtötiedon tavoin. Välittömin etu olisi eri tasojen välisten irrallisten tietojoukkojen linkittäminen ja sen seurannaisvaikutukset asemakaava-, yleiskaava- ja maakuntakaavatasojen sisältöjen ollessa läpinäkyvästi analysoitavissa ohjausvaikutuksen muodostamien linkkien kautta.

3.1.9. Työryhmätyö ja visiointi

Yhdistettynä intuitiivisiin luonnostyökaluihin voidaan versionhallintaa käyttää erikseen visio-, aluekehitys-, suunnitteluperiaate-, ja kaavarunkotyössä. Useiden osallisten erillisiä näkökulmia ja intressejä voidaan yhteensovittaa rinnakkaisten suunnitteluhaarojen avulla konsensus- tai kilpailumallisissa ryhmätyöskentelyprosesseissa.

3.2. Jatkokehitys

Kehitystyötä jatketaan yhteistyössä KYMP:n kanssa vuorovaikuttaen tiiviisti myös ministeriön ja muiden kuntakontaktien kanssa. Seuraavina konkreettisina kehitysaskelina on KYMP-projektien yhteensovituksen jatkaminen, työkalujen ja prosessien integraation luonnostelu, sekä yleiskaavatason intressien ja mahdollisten hyödyntämiskohteiden tarkempi kartoitus. Tässä yhteydessä olennaiseksi muodostuu myös käyttäjä- ja työryhmätason prosessien kuvaus esimerkiksi BPMN-kuvauskielellä. Yhteistyökeskusteluja jatketaan myös muiden kontaktien kanssa pitkän aikavälin tavoitteen ollessa versionhallinnan käytön laajentaminen ja versiohallitun yhdenmukaistetun tiedon tuottamisen ja hyödyntämisen kasvattaminen. Työkalulle haetaan aktiivisesti kumppaneita ja lisäresursseja sen kehitystyön jatkamiseen.

4. Lähteet

Nuseibeh, Bashar, and Steve Easterbrook. 2000. “Requirements Engineering: A Roadmap.” In *Proceedings of the Conference on The Future of Software Engineering*, 35–46. ICSE ’00. New York, NY, USA: ACM. https://doi.org/10.1145/336512.336523.

Boehm, Barry. 2003. “Value-Based Software Engineering.” *ACM SIGSOFT Software Engineering Notes* 28 (2): 3. https://doi.org/10.1145/638750.638775.

The Standish Group. 2014. CHAOS Report.

The Standish Group. 2015. CHAOS Report.

Johnson, James. 2018. CHAOS Report: Decision Latency Theory: It Is All About the Interval. The Standish Group.

Benington, H. D. 1983. “Production of Large Computer Programs.” *Annals of the History of Computing* 5 (4): 350–61. https://doi.org/10.1109/MAHC.1983.10102.