Mika Hasanen

Sijaintivirheiden etsintäväline JAKO-järjestelmässä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Maanmittaustekniikan tutkinto-ohjelma

Insinöörityö

8.5.2014

Metropolia

Tekijä Otsikko Sivumäärä Aika	Mika Hasanen Sijaintivirheiden etsintäväline JAKO-järjestelmässä 28 sivua + 2 liitettä 8.5.2014
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	maanmittaustekniikka
Ohjaajat	sovellusasiantuntija Juha-Matti Vääränen lehtori Jussi Laari

Insinöörityössä kehitettiin Sijaintivirheiden etsintä -ikkuna Maanmittauslaitoksen JAKOkii-toimitustuotannon sovellukseen. Pohjana käytettiin JAKO-järjestelmään kuuluneita erillisiä sijaintivirheiden hakuikkunoita, joiden toiminnot yhdistettiin yhteen ikkunaan. Uusi ikkuna tarvittiin toimitustuotannon henkilöstön työnteon helpottamiseksi ja nopeuttamiseksi.

Kehitystyö tehtiin sovelluskehityksen menetelmillä. Ohjelmointikielenä General Electricin Smallworld GIS -teknologiaan pohjautuvassa JAKO-järjestelmässä oli käytössä Smallworld Magik. Työhön liittyi oleellisena osana ennen käyttöönottoa myös sovellusmuutosten dokumentointi ja testaus.

Ikkuna otettiin onnistuneesti käyttöön Maanmittauslaitoksessa sekä kiinteistörekisteriä pitävissä kunnissa ensimmäisen kerran helmikuun 2013 sovellusversiossa sekä käyttäjäpalautteen pohjalta paranneltuna helmikuussa 2014 uuden tuotantoversion myötä.

Ikkunan voidaan sanoa täyttäneen käyttäjien tarpeet, koska käyttäjäpalautetta helmikuun 2013 sovellusversion jälkeen ei ole tullut. Tulevaisuudessa Sijaintivirheiden etsintä -ikkuna on tarkoitus ottaa käyttöön myös JAKOmtj-sovelluksessa.

Avainsanat	JAKO-järjestelmä, sijaintivirheet, JAKOkii

Author	Mika Hasanen
Title	Quality control tool in JAKOcadastre system
Number of Pages	28 pages + 2 appendices
Date	8 May 2014
Degree	Bachelor of Engineering
Degree Programme	Land Surveying
Instructors	Juha-Matti Vääränen, Application specialist Jussi Laari, Senior Lecturer

The purpose of this Bachelor's thesis was to develop a quality control tool for National Land Survey of Finland's JAKOcadastre Geographic Information System application. JAKOcadastre is used to maintain the nation-wide cadastre system of Finland. The application had 11 separate geometry error search tools which were now combined into a single quality control tool. The new tool was needed to minimize the time needed for geometry error checks and to improve the usability of the application.

The new tool was made with the methods of application development. The programming language used was General Electric's Smallworld Magik. In addition to the actual programming, software testing and documentation were also integral parts of the work.

The tool was introduced in the February 2013 update of JAKOcadastre. After some user feedback regarding the experienced slow search speed of the tool a new version was introduced in February 2014. It can be said that the users are happy with the latest version as no negative feedback has been received since the last update. In the future the tool will be implemented in JAKOtds (Topographic Data System) as well.

Keywords	GIS, Magik, Smallworld

Sisällys

Lyhenteet

GE	*General Electric.* Yhdysvaltalainen monialayritys.
IDE	*Integrated development environment.* Sovelluskehitystyössä käytettävä ohjelmointiympäristö.
JAKO	Maanmittauslaitoksessa käytössä oleva paikkatietojärjestelmä.
JAKOkii	Maanmittauslaitoksen toimitustuotantosovellus.
JAKOmtj	Maanmittauslaitoksen sovellus maastotietokannan ylläpitämiseen
KITE	Kiinteistötehtävät -palveluryhmä.
MDT	*Magik Development Tools.* Magik-ohjelmointikielen ohjelmointiympäristö.
RWO	*Real world object.* Olio, joka kuvaa oikean maailman kohdetta, kuten rajamerkkiä.

1 Johdanto

1.1 Työn tausta

Maanmittauslaitoksen julkisten verkkosivujen mukaan Maanmittauslaitos (MML) on maa- ja metsätalousministeriön alaisuudessa oleva viranomainen, jonka tehtäviin kuuluvat maanmittaustoimitukset, kiinteistötietojen ja kartta-aineistojen ylläpito sekä lainhuudot ja kiinnitykset. MML:n palveluksessa on noin 1 800 työntekijää 35 paikkakunnalla. JAKOkii on Maanmittauslaitoksessa toimitustuotannon käytössä oleva sovellus. [1]

Työn tekeminen alkoi kesällä 2012 työskennellessäni sovelluskehittäjäharjoittelijana Maanmittauslaitoksella. Työtehtävän minulle antoi JAKOkii-toimitustuotannon sovelluksesta vastaava sovellusryhmän vetäjä (nykyisin vastuualuepäällikkö ja palveluryhmän vetäjä) Mikko Peltokorpi.

Työn taustalla oli tarve toimitustuotantoprosessin nopeuttamiseen. Muutoksen oli tarkoitus säästää sovelluksen käyttäjien työaikaa ja sitä kautta säästää työaikakuluissa toimitusta kohden.

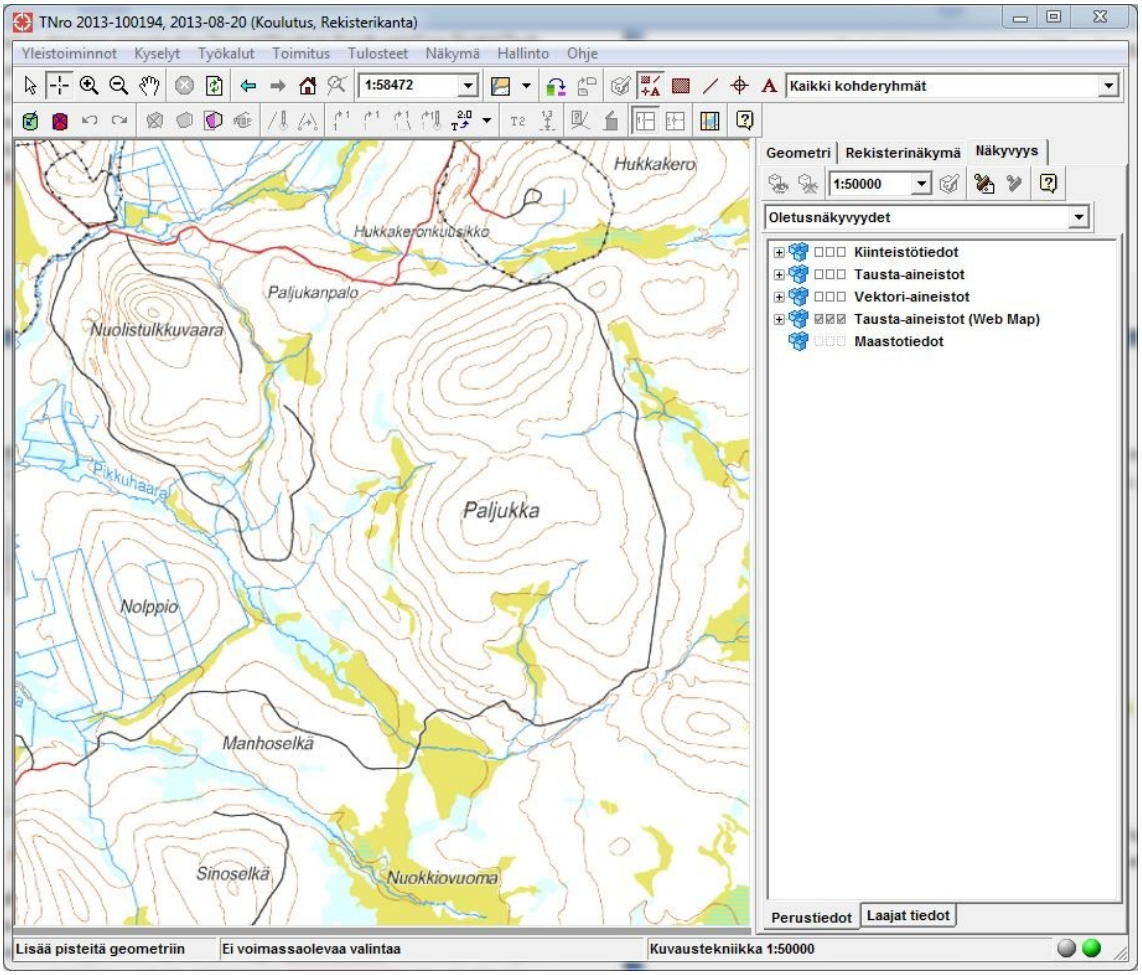
1.2 Ongelmanasettelu

JAKOkii-sovelluksessa on ollut sijaintivirheiden etsintään käytössä 11 kpl erillisiä ikkunoita, joilla sovelluksen käyttäjät joutuivat ajamaan tarkistuksia. Usein tarkistuksia jouduttiin tekemään usealla ikkunalla peräkkäin. Työaikaa säästyy, kun kaikki sijaintivirheiden tarkistukset on mahdollista tehdä yhdellä kertaa ja haut toimivat nopeammin.

Tavoitteena on yhdistää kaikki tarkistusikkunat samaan sijaintivirheiden hakuikkunaan. Ikkunan pitää olla tehokas ja helppokäyttöinen väline käyttäjille.

2 Sovelluskehitys JAKO-tietojärjestelmäympäristössä

2.1 JAKO-järjestelmä



Kuva 1. JAKOkii-sovellus

JAKO-järjestelmä (kuva 1) on Maanmittauslaitoksessa käytössä oleva paikkatietosovellus, joka pohjautuu yhdysvaltalaisen monialayritys General Electricin (GE) Smallworld Core Spatial Technology -paikkatieto-ohjelmistoon. Ohjelmiston on alun perin kehittänyt brittiläinen Smallworld-yhtiö. Maanmittauslaitoksessa JAKO-järjestelmän suunnittelu ja pohjatyöt on aloitettu vuonna 1994 ja kehittämistä jatkettiin uudelleenorganisoidun projektin mukaisena vuonna 1996. [2] Järjestelmä otettiin käyttöön maaliskuussa 1998 [3].

Maanmittauslaitoksen JAKO-tietojärjestelmään kuuluvat sovellukset JAKOtoma, JAKOkhr, JAKOinfo, JAKOmtj, ja JAKOkii, jota tämä insinöörityö käsittelee. JAKOkii-sovelluksella on noin 1 200 käyttäjää. JAKOkii-sovellusta käytetään Maanmittauslaitoksessa toimitustuotannon tehtävissä, kiinteistörekisterin ylläpidossa sekä tietopalvelussa. Lisäksi kuntakäytössä on KTJkii-rekisterinpitosovellus, jota käyttää noin 200 kuntien työntekijää. KTJkii-rekisterinpitosovellus vastaa JAKOkii-sovellusta, josta on riisuttu tuotantovälineet pois. [3]

JAKOmtj-sovelluksella ylläpidetään Maanmittauslaitoksen maastotietokantaa, joka on sijaintitiedoiltaan valtakunnan tarkin paikkatietoaineisto. [4]

2.2 Sovelluskehitys Maanmittauslaitoksessa

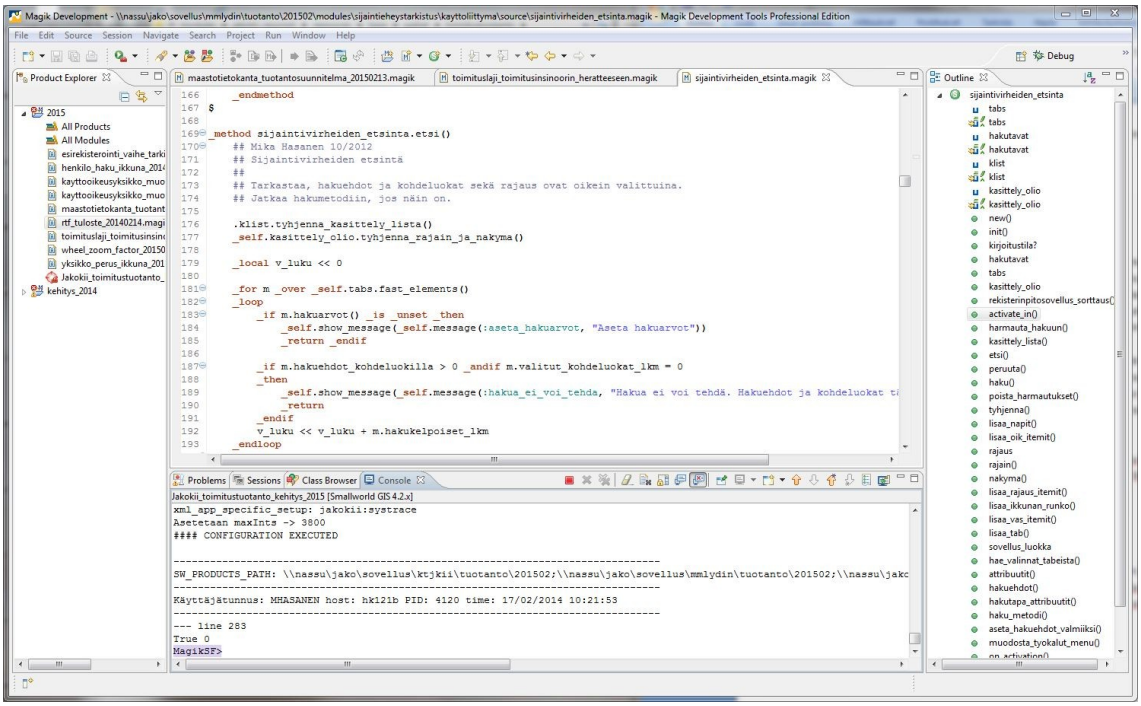
Maanmittauslaitos ylläpitää kiinteistötietojärjestelmää, joka on valtakunnallinen tietovarasto. Jotta tätä ylläpitotehtävää voidaan hoitaa asianmukaisesti ja virheettömästi, vaatii tämä sovellukselle tehtäviä muutoksia, jotka tulevat työnteon yhteydessä tarpeellisiksi. Tämän vuoksi JAKOkii ja KTJkii-sovelluksilla on ylläpitoprosessi, johon kuuluu korjaava, laajentava, sopeuttava ja ennaltaehkäisevä ylläpito. [5]

Laajentava ylläpito tarkoittaa toiminnon lisäystä sovellukseen. Sopeuttava ylläpito tarkoittaa sovelluksen muuttamista vastaamaan esimerkiksi säädösmuutoksesta johtuvaa muuttunutta tilannetta. Korjaava ylläpito tarkoittaa havaitun virheen korjaamista sovelluksessa. Ennaltaehkäisevällä ylläpidolla varaudutaan tiedossa olevaan tulevaisuuden tilanteeseen, joka vaatii sovellukseen muutoksia. Tällainen voi olla esimerkiksi tulevaisuudessa käyttöönotettava käyttöjärjestelmäpäivitys. [6]

Sovelluskehityksellä tarkoitetaan sovellusten, eli loppukäyttäjän ohjelmien, kehittämistä. Kehittäminen alkaa, kun on todettu, että sovellus tarvitsee muutoksen. Tämän jälkeen tehdään määrittely, millainen muutos tai laajennus tarvitaan. Määrittelyn perusteella sovelluskehittäjä suunnittelee, miten tulee muutoksen toteuttamaan. Varsinainen kehittäminen tehdään sovelluskoodia kirjoittamalla eli ohjelmoimalla.

Kehittäjä testaa muutosten toimivuutta rinnakkain sovelluskoodin kirjoittamisen kanssa. Kun muutoksen sovelluskoodi on kehittäjän mielestä valmis, kehittäjä testaa itse sen toimivuuden vielä uudestaan ja tämän jälkeen antaa muutoksen testattavaksi. Jos virheitä tai muutostarpeita ilmenee, palautuu asia kehittäjälle, joka tekee tarvittavat muutokset sovelluskoodiin.

Sovelluskehittäjällä on työvälineenään ohjelmointiympäristö eli IDE (Integrated Developing Environment). Itse käytän avoimen lähdekoodin Eclipse-sovellusta, johon on asennettu Magik Development Tools (MDT) -niminen lisäosa Magik-kehittämistä varten (kuva 2). Toinen Smallworldin sovelluskehittämisessä käytetty IDE on Emacs.



Kuva 2. Magik Development Tools -ohjelmointiympäristö

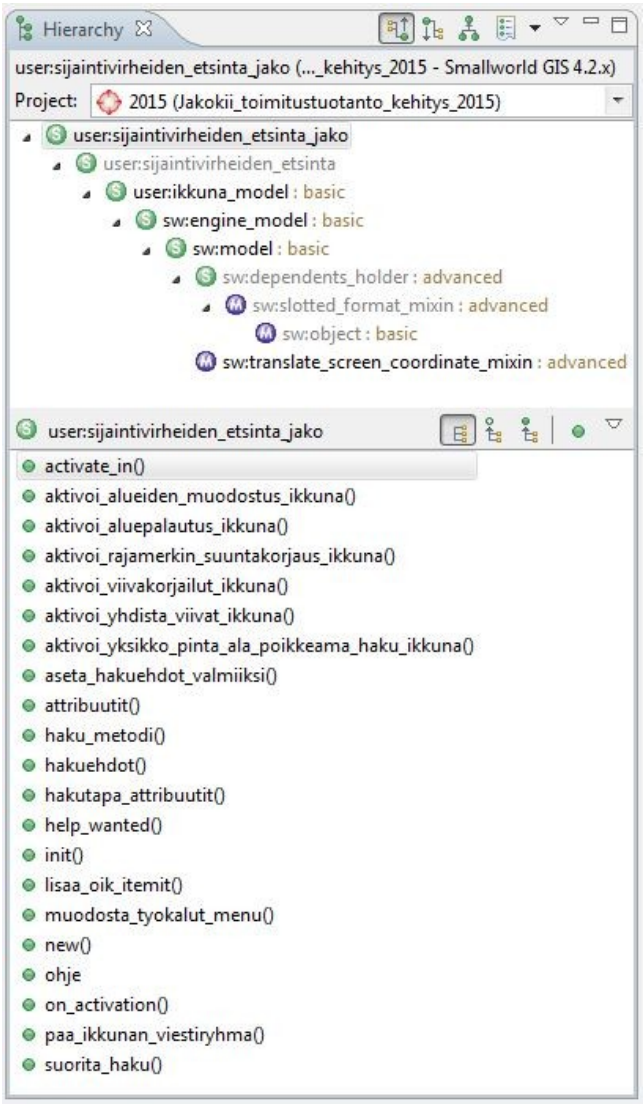
MDT sisältää työkalut, joita sovelluskehityksessä tarvitaan. Tiedostoikkunassa voidaan katsella ja muokata tekstitiedostoja. MDT näyttää tässä ikkunassa sovelluskoodin lukemista helpottavia merkintöjä. Luokkaselain (Class browser) on hakutyökalu, jolla voidaan nopeasti hakea sovelluskoodikirjastosta metodeita luokan ja olion nimen perusteella. Konsolissa näytetään Smallworldin virtuaalikoneen antamat viestit ja tätä kautta voidaan myös ajaa komentoja virtuaalikoneella.

Smallworld-sovelluksia kehitetään ohjelmointikielellä, joka on nimeltään Magik. Yleisistä ohjelmointikielistä poiketen Magik on käytössä vain General Electricin paikkatietoohjelmistojen sovelluskehityksessä. Siksi kieltä ei myöskään opeteta juuri missään oppilaitoksessa, joten sen käyttö tulee opetella itse. Magik muistuttaa kuitenkin muita tunnettuja olio-ohjelmointikieliä. Smallworldin ohjeen mukaan (Application development, Magik language) Magik on erittäin tehokas olio-ohjelmointikieli. Olio-ohjelmointi tarkoittaa paradigmaa, jossa ohjelma rakennetaan useista yhdessä toimivista olioista. [7]

Smallworldin ohjeessa (Object-orientation concepts in Magik) luokka määritellään yhteenkuuluvien olioiden kategoriaksi. Jokainen luokka pitää yleensä sisällään oman toiminnallisuuden ja datan. Luokan käyttämä data on tallentuneena luokkamuuttujiin (slotted exemplars) ja toiminnallisuus on luokan olioissa. [8]

Magik-kielessä olioksi luetaan mikä tahansa instanssi, joka kutsuttaessa pystyy vastaamaan viestiin. Olion toiminta määritetään metodissa (method). Metodi yksilöidään siten, että ensin kirjoitetaan sen luokan nimi, johon olio kuuluu ja tämän jälkeen itse olion nimi pisteellä erotettuna. Jokainen olio kuuluu johonkin luokkaan. Olio aktivoituu, kun sitä kutsutaan. Esimerkiksi olio *geometrian_kasittely.poista_duplikaatit()* poistaa toiseen *valitetty_rope*-nimiseen olioon mahdollisesti tallennetut kaksoiskappaleet. Esimerkki tästä metodista on liitteessä 1.

Magik-kielessä on mahdollista periyttää ylempiä luokkia alemmille luokille. Perittyjen luokkien, eli yläluokkien, oliot ja luokkamuuttujat ovat käytössä myös alaluokissa. Smallworldin ohjeen mukaan (Overview of object-oriented programming) periyttäminen on hyödyllistä, kun rakennetaan modulaarista sovellusta. Tällöin olio voi käyttää samaa sovelluskoodia ilman, että tätä tarvitsisi erikseen kopioida uudelle luokalle. Uuden luokan voidaan ajatella olevan perittävän luokan erityistapaus. Kuvasta 3 nähdään, että luokan *sijaintivirheiden_etsinta_jako* perintäketju on varsin pitkä.



Kuva 3. MDT:n hierarkia-näkymä, jossa näkyy *sijaintivirheiden_etsinta_jako* -luokan perimät luokat sekä luokan sisältämät metodit.

Kun tehdään uusi ikkuna, tämän rakentamisessa kannattaa käyttää periyttämistä hyödyntämällä olemassa olevia yleisiä luokkia. Esimerkiksi luokka *ikkuna_model* on tällainen (kuva 3). Se sisältää ikkunoissa yleisesti tarvittuja toimintoja ja perii itse Smallworldin yleisiä luokkia.

Smallworld-sovellusta ajetaan virtuaalikoneen päällä, joten sovelluskoodia voidaan kääntää dynaamisesti koko sovellusta uudelleen käynnistämättä. Komentoja voidaan myös ajaa suoraan Smallworldin konsolin kautta, mikä säästää aikaa ja helpottaa olennaisesti kehittäjien työtä.

2.3 JAKO-järjestelmän kehittäminen Maanmittauslaitoksessa

Vuodenvaihteessa 2014 voimaan astuneen organisaatiomuutokseen jälkeen JAKO-järjestelmän kehittämisestä vastaa Tukipalvelut-toimintayksikkön kuuluvan Sovelluspalvelut-tukiyksikön KII- ja MARA-tuotannon sovellukset -vastuualue. JAKOkii-sovelluksen kehittämisestä vastaa Kiinteistötehtävät-palveluryhmä (KITE). [9]

Sovelluksen kehittäminen tapahtuu nk. vuosikellon tahdissa. Kello rakentuu helmikuun toisella viikolla tapahtuvan versionvaihdon ympärille. Ennen versionvaihtoa tärkeitä tapahtumia ovat testauskierrokset, joiden aikana sovellukseen tehdyt muutokset testataan ja havaitut virheet kirjataan ylös. Testauskierroksia tehdään useimmiten neljä. Tämän jälkeen kehittäjät korjaavat virheet ja korjatut kohdat testataan uudestaan. [9]

Versionvaihtojen välillä voidaan pieniä tärkeitä muutoksia jakaa käyttäjille paikkauksina. Touko-kesäkuussa julkaistaan suurempi paikkajakelu, jossa jaellaan kerralla enemmän hieman suurempia muutoksia. [9]

3 Sijaintivirheet

3.1 Yleistä

Sijaintivirheet ovat usein käyttäjälle huomaamattomia, mutta korjaamattomina paikkatietojärjestelmän eheydelle vahingollisia virheitä. Eheysvirheet ovat erityisen haitallisia etenkin aineiston luovutuksissa. [10] Toimitus- ja rekisteröintiprosessissa sijaintivirheiden etsintä suoritetaan yleensä aina, kun sijaintiaineistoon on tehty muutoksia, eli esimerkiksi kun järjestelmään on ladattu uusia pisteitä ja viivoja ja on muodostettu uusia alueita. Sijaintivirheiden etsintä voidaan tehdä esimerkiksi ennen kiinteistöjaotuksen muutosta tai toimituksen rekisteröintiä. [11]

Sijaintivirheitä etsitään kartalle merkityistä geometriakohteista, joita ovat pisteet, viivat ja alueet. Näillä geometriakohteilla on linkki RWO-tietueeseen (real world object), jossa on määritelty geometriakohteen ominaisuustiedot. RWO-tietueessa määritetään geometriakohteen kohdeluokka, joka voi olla esimerkiksi kiinteistöraja tai rajamerkki.

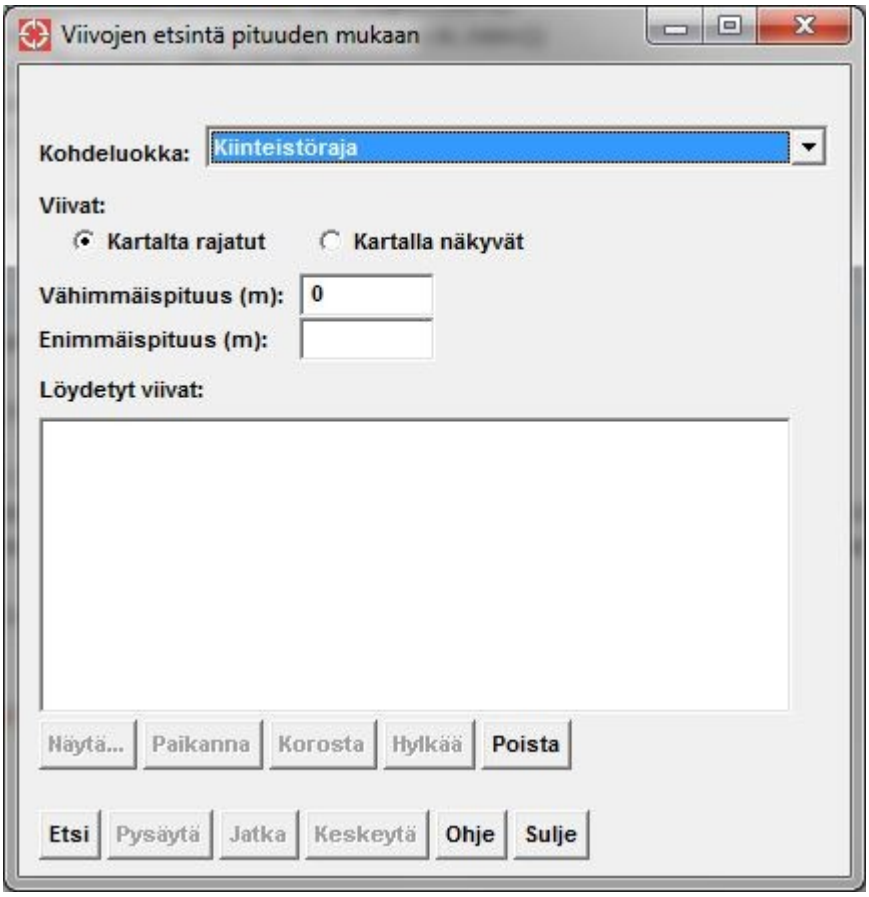
Pisteiden kohdeluokista rajamerkki on ainoa, joka tarkastetaan sijaintivirheiden varalta. Alueiden kohdeluokista Sijaintivirheiden etsintä -välineellä voidaan tehdä hakuja vain palstoille. Poikkeuksena tähän on Ongelma-alueet-hakuehto, joka on kuitenkin poistumassa käytöstä. Jokaista alueen kohdeluokkaa rajaa tätä vastaava viiva, joka on oma kohdeluokkansa. Esimerkiksi kiinteistöä rajaa kiinteistöraja.

Kiinteistörekisteriyksikkö tarkoittaa kiinteistörekisteriin merkittyä kiinteistöä tai muuta rekisteriyksikköä, jolla on kiinteistötunnus [12]. Kiinteistörekisteriyksikön palsta tarkoittaa kiinteistörajan rajaamaa aluetta [10]. Käyttöoikeusyksiköllä tarkoitetaan JAKO-järjestelmässä kiinteistörekisteriyksikköön kohdistuvia sijainnillisia tai sijainnittomia oikeuksia, rajoituksia ja kaavoja. Käyttöoikeusyksikköjen ja käyttöoikeusyksikkörajan kohdeluokkia on useita, alueista mainittakoon esimerkiksi suojelukohde, kiinteistönmuodostamislain mukainen rasite ja yleiskaava. Arvokohde tarkoittaa maanmittaustoimituksissa arvioitavaa kohdetta. Sellaisia ovat esimerkiksi arvioitava rakennus, puustokuvio ja ranta-arvokuvio. [10]

Suunnittelupalsta on alue, jolla merkitään jakoyksikön tai käyttöyksikön alueellinen ulottuvuus. Suunnittelupalsta kuvaa tulevaa kiinteistöjaotusta. Käyttöyksikkö tarkoittaa maankäytön kokonaisuutta, joita käsitellään jako- ja järjestelytoimituksissa. Esimerkiksi halkomistoimituksessa käyttöyksikkö tarkoittaa muodostettavaa uutta kiinteistörekisteriyksikköä. Jakoyksikkö on kiinteistörekisterin yksikkö, jota käsitellään jakotoimituksessa itsenäisenä. [10]

3.2 Erilliset hakuikkunat

JAKO-järjestelmässä on ollut käytössä erilliset hakuikkunat (kuva 4) jokaista tässä luvussa mainittua hakua varten. Ikkunat on tehty heti JAKO-järjestelmän ensimmäiseen versioon. Haku vanhoilla ikkunoilla vie käyttäjältä huomattavan paljon työaikaa, koska jokainen haku pitää tehdä erikseen.



Kuva 4. Esimerkki vanhanmallisesta erillisestä hakuikkunasta.

3.3 Sijaintivirheiden kuvaukset

JAKOkii-sovellusoppaassa (Sijaintivirheiden hakuehdot) on esitelty yleisimmät sijaintivirheet korjausvaihtoehtoineen, joita toimitustuotannossa voi tulla eteen. Sovellusopas on sijaintivirheiden osalta laadittu pääpiirteissään vanhojen hakuikkunoiden aikakautena, mutta opasta on päivitetty Sijaintivirheiden etsintä -ikkunan tultua käyttöön toimitustuotannossa. Ohjeen saa auki esimerkiksi Sijaintivirheiden etsintä -ikkunan Ohje-painikkeesta.

3.3.1 Solmuttumattomat viivat

Solmuvirhe tarkoittaa tapausta, jossa viivan perään on lisätty saman kohderyhmän viiva, mutta nämä kaksi viivaa eivät kuitenkaan ole solmuttuneet keskenään. Seurauksena on kaksi erillistä viivaa, jotka käyttäjä on tarkoittanut yhdeksi viivaksi.

Solmuttumattomia viivoja voidaan korjata monella tavalla. Yhtä viivaa voidaan jatkaa pakottamalla se jatkumaan solmupisteeseen saakka. Solmupiste voidaan myös siirtää solmuttumattoman viivan päätepisteeseen. Tällöin kaikki solmupisteeseen yhdistyvät viivat muuttuvat, koska solmupiste on niiden yhteinen päätepiste. Jos solmupistettä ei ole, tulee sellainen tehdä katkaisemalla toinen tai molemmat viivoista.

3.3.2 Päällekkäiset viivat

Päällekkäiset viivat tarkoittavat tapausta, jossa samaan kohdeluokkaan kuuluvia viivoja on kokonaan tai osittain päällekkäin. Jos viivat ovat kokonaan päällekkäin, toinen viivoista on useimmiten turha, ja se voidaan poistaa. Jos viivat ovat vain osittain päällekkäin, toinen viiva kannattaa katkaista ja pakottaa solmuttumaan ensimmäisen viivan kanssa ja viivasta yli jäävä osuus poistaa.

3.3.3 Leikkaavat viivat

Leikkaavat viivat tarkoittaa tapausta, jossa kaksi samaan kohdeluokkaan kuuluvaa viivaa leikkaavat toisensa, mutta viivojen leikkauspisteessä ei ole solmupistettä. Virhe voidaan korjata katkaisemalla toinen viivoista ja pakottamalla se solmuttumaan toisen viivan kanssa. Tarpeen vaatiessa ylimääräinen osa viivasta poistetaan. Jos viivojen on tarkoitus leikata, riittää korjaukseksi toisen viivoista katkaisu risteävällä viivalla.

3.3.4 Itsensä leikkaavat viivat

Itsensä leikkaava viiva tarkoittaa nimensä mukaisesti viivaa, joka leikkaa itseään. Tällainen tilanne voi syntyä esimerkiksi käyrän viivan digitoinnissa, kun uusi digitointipiste on annettu digitointisuuntaan nähden taaksepäin. Silmukka voi olla niin pieni, ettei käyttäjä tätä omalla silmällä havaitse. Virhe korjataan poistamalla ylimääräinen silmukka.

3.3.5 Viivojen pituudet

Hyvin lyhyet viivat ovat usein virheitä. Näitä mikroviivoiksi kutsuttuja kohteita, joiden pituus on yleensä alle 10 cm, on vaikea havaita karttaikkunassa. Virhe voidaan korjata poistamalla valikoidusti liian lyhyet viivat.

3.3.6 Viivat, jotka eivät muodosta aluetta

Tapaus tarkoittaa viivaa, jonka toisella tai molemmilla puolilla ei ole muodostettua aluetta. Virhe voidaan korjata muodostamalla alue tai poistamalla ylimääräinen viiva.

3.3.7 Poistuvat rajat

Jos käyttäjän poistuviksi merkitsemien kiinteistörajojen molemmin puolin on eri kiinteistötunnukset, on kyseessä sijaintivirhe, koska tällöin poistuvaksi merkittyä rajaa ei saa poistaa. Tässä tapauksessa virhe voidaan korjata kiinteistötunnuksen ollessa virheellinen, muuttamalla kiinteistötunnus ja tarkastamalla, että rajan kuvaus on tehty oikein ja tarpeen vaatiessa muuttamalla kuvaus.

Sijaintivirhe on myös tilanne, jossa kiinteistörajojen kummallakin puolella on samat kiinteistötunnukset ja toisella palstalla on tehty palstamuutoksia. Virhe voidaan korjata merkitsemällä raja poistuvaksi, jos tämä on unohdettu tehdä.

3.3.8 Päällekkäiset rajamerkit

Nimensä mukaisesti päällekkäiset rajamerkit ovat määrätyllä toleranssilla, yleensä alle 1 cm:n etäisyydellä toisistaan olevia, päällekkäisiä rajamerkkejä. Rajamerkit voivat olla identtisiä ominaisuuksiltaan tai niissä voi olla eroavaisuuksia. Virhe voidaan korjata joko poistamalla ylimääräiset identtiset rajamerkit tai korvaamalla rajamerkki tietokannassa olevalla toisella rajamerkillä. Viivojen solmuttuminen rajamerkkiin tulee huomioida.

3.3.9 Rajasta irralliset rajamerkit

Jos rajamerkki ei ole solmuttunut rajaan eli topologisesti kiinni rajassa, on tässä virhe. Irralliset rajamerkit ovat haitallisia aineistonluovutuksissa, koska ne eivät nouse mukaan siirtotiedostoihin. Tilanne voidaan korjata siirtämällä rajamerkki rajalle tai siirtämällä olemassa olevat rajat kiinni irralliseen rajamerkkiin.

3.3.10 Ongelma-alueet

Ongelma-alue tarkoittaa aluetta, joka ei muodostu kohdeluokan rajaviivoista. Virhe voi syntyä aineiston latauksessa. Tällainen virhe on hyvin harvinainen toimitustuotannossa. Hakuehto poistettiin käytöstä helmikuun 2014 versiossa, koska arveltiin, ettei sille ole käyttöä.

3.3.11 Palstavirheet

Palstavirhe-hakuehto hakee useita palstoihin liittyviä virheitä, joita ovat pinta-alavirhe, aluevirhe, yksikkövirhe, linkkivirhe, RWO-virhe ja symbolivirhe. Pinta-alavirhe tarkoittaa palstan RWO-tietueen virheellistä pinta-alatietoa, eli tilannetta jossa pinta-alaksi on tallennettu nolla tai tätä pienempi arvo. Aluevirhe tarkoittaa sitä, että palstan alue-tietue on tyhjä, eli palstan RWO:lla ei ole yhteyttä mihinkään alueeseen.

Yksikkövirhe tarkoittaa sitä, että palstan kiinteistörekisteriyksikön yksikkö-id on yhtä suuri tai pienempi luku kuin nolla. Yksikkövirheeksi luetaan myös tilanne, jossa kiinteistörekisteriyksikkö johon palsta kuuluu on lakannut.

Linkkivirhe tarkoittaa, että palstan aluetta rajaavan viivan tietue on tyhjä. RWO-virhe on tilanne, jossa palstan aluetta rajaavan viivan RWO-tietue on tyhjä, eli rajaava viiva on pelkkä viiva ilman tietoa sen todellisesta kohdeluokasta.

Symbolivirhe eli pistevirhe tarkoittaa, että palstan rajaviivan viimeisen noden eli päätöspisteen RWO-tietue tai RWO:n sijaintitieto on tyhjä.

3.3.12 Ristiriitaiset palstatunnukset

Jos palstalla on useita tunnuspisteitä, ovat palstatunnukset ristiriitaisia. Virhe korjataan poistamalla ylimääräinen tunnuspiste tai siirtämällä se oikeaan palstaan. Sama tunnus vierekkäisillä palstoilla on myös virhe. Tilanne korjataan poistamalla virheellinen tunnus ja tekemällä uusi.

4 Työn vaiheet

4.1 Määrittely

Määrittely tarkoittaa yleensä ennen varsinaisen sovelluskehittämisen aloittamista tehtävää kuvausta siitä, millainen tulevan sovellusosan tulisi olla. Määrittelyssä annetaan sovelluskehittäjälle yleensä tieto siitä, mikä on muutoksen tarkoitus, miten uuden sovellusosan pitäisi toimia ja millainen käyttöliittymä siinä pitäisi olla.

Sovellusryhmän (nykyisin palveluryhmän) vetäjä Mikko Peltokorpi antoi minulle määrittelyn kehittämistyöstä. Tämä määrittely oli tarkoituksellakin väljä, mutta se tarkentui työn edetessä. Minulle annettiin paljon vapauksia vaikuttaa lopputulokseen.

Sain tätä työtä varten määrittelynä tietoa muutoksen tarkoituksesta, ja alustavan idean käyttöliittymästä. Työn tarkoitus säilyi koko ajan samana, mutta käyttöliittymästä tuli lopulta täysin erilainen kuin määrittelijä oli alustavasti suunnitellut. Työn edetessä sain lisää toiveita ikkunassa tarvittavista ominaisuuksista määrittelijältä, joka puolestaan sai kuulla testaajilta palautetta uudesta ikkunasta.

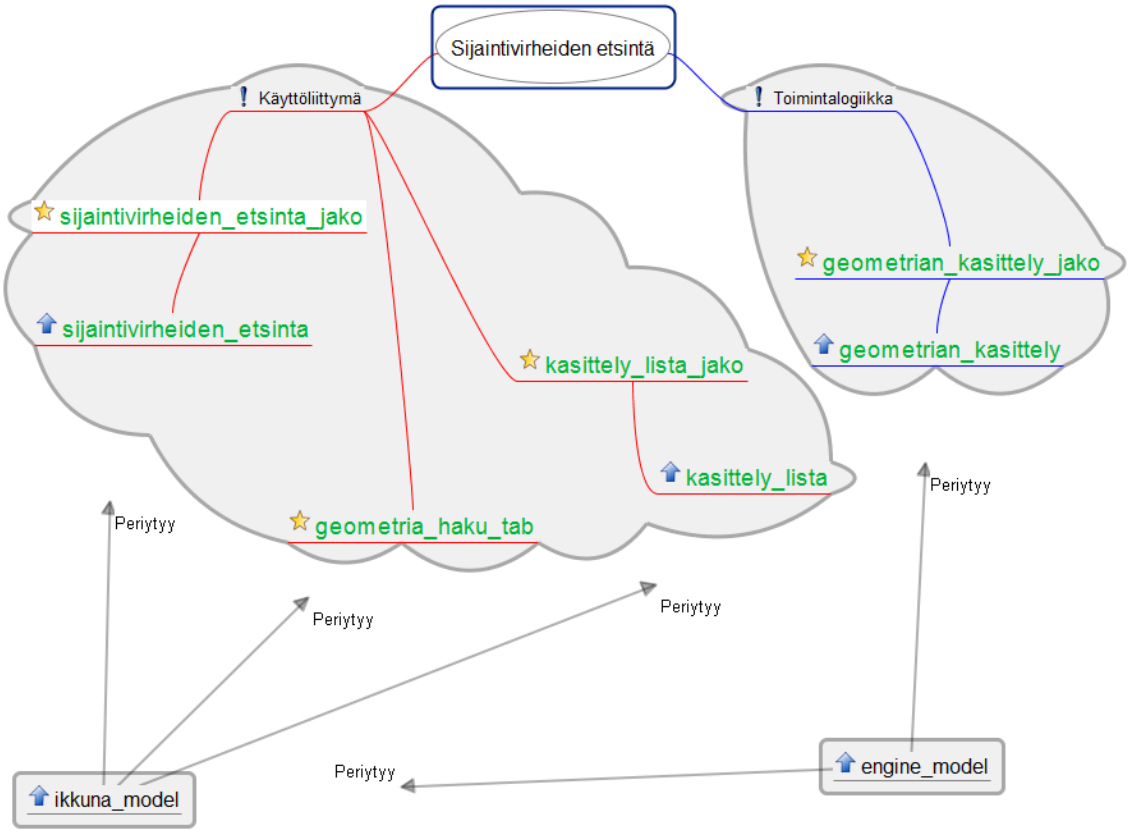
Määrittelyä tehtiin koko kehittämistyön ajan aina, kun uusia asioita tuli eteen. Katsoimme usein määrittelijän kanssa kehitteillä olevaa ikkunaa ja pohdimme, mikä olisi paras ratkaisu käyttöliittymän ja ikkunan toiminnan kannalta.

4.2 Sovelluskehitys

4.2.1 Luokkarakenne

Luokkarakenne tarkoittaa sitä, miten metodit on luokkiin järjestelty ja millaisia eri luokkia on tehty sekä miten niiden periyttäminen tehdään. Periaatteena luokkajaossa on, että käyttöliittymä ja toimintalogiikka eriytettäisiin. Versiossa 2013/2 luokkarakennetta ei ehditty tehdä sellaiseksi, että käyttöönotto JAKOmtj-versiossa olisi mahdollista, mutta versioon 2014/2 tämä tehtiin.

Metodit, joita voidaan käyttää yleisesti muissakin kuin JAKOkii-sovelluksessa, pyritään siirtämään omaan luokkaansa, joka kirjastoidaan sovelluksen ytimeen, jossa sovelluskoodi on kaikkien sovellusten käytössä. Tällöin nämä luokat voidaan periyttää jokaisen sovelluksen omille luokille. Tarvittaessa on myös mahdollista ylimääritellä metodeita sovelluskohtaisesti. Tällöin oliota ei peritä, vaan siitä tehdään oma versio tälle sovellukselle.



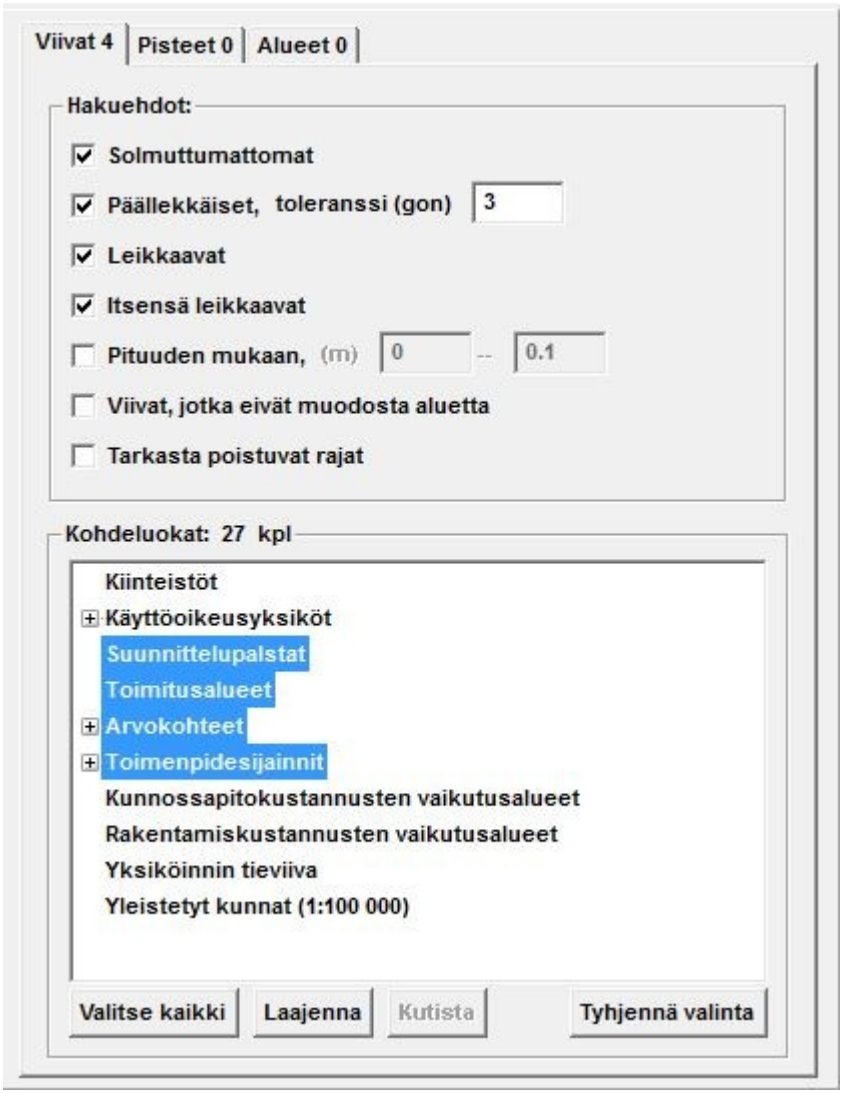
Kuva 5. Sijaintivirheiden etsintä -välineen luokkarakenne.

Kuvassa 5 on esitetty kaavio luokkarakenteesta. Tähdellä merkittyjä luokkia käytetään ikkunassa sellaisenaan. Sinisellä nuolella merkittyjä luokkia käytetään vain periyttämiseen. Nämä luokat on tehty sitä varten, jos JAKOmtj:ssä otetaan Sijaintivirheiden etsintäväline käyttöön. Silloin JAKOmtj:lle tehdään esimerkiksi *kasittely_lista*:n tapauksessa luokka *kasittely_lista_mtj*, joka perii luokan *kasittely_lista*.

Luokan *engine_model* perivät kaikki Sijaintivirheiden etsinnän luokat. Luokassa *ikkna_model* on käyttöliittymissä tarvittuja komponentteja, joten kaikki käyttöliittymäluokat perivät tämän luokan.

4.2.2 Käyttöliittymä

Kun olin ensin paperilla ja kuvankäsittelyohjelmalla suunnitellut käyttöliittymän, rakensin sen sovelluskoodilla. Käyttöliittymän teossa oli haasteellista, miten saan eroteltua viivojen, pisteiden ja alueiden osiot toisistaan, mutta kuitenkin mahdutettua kaiken tämän samaan ikkunaan.

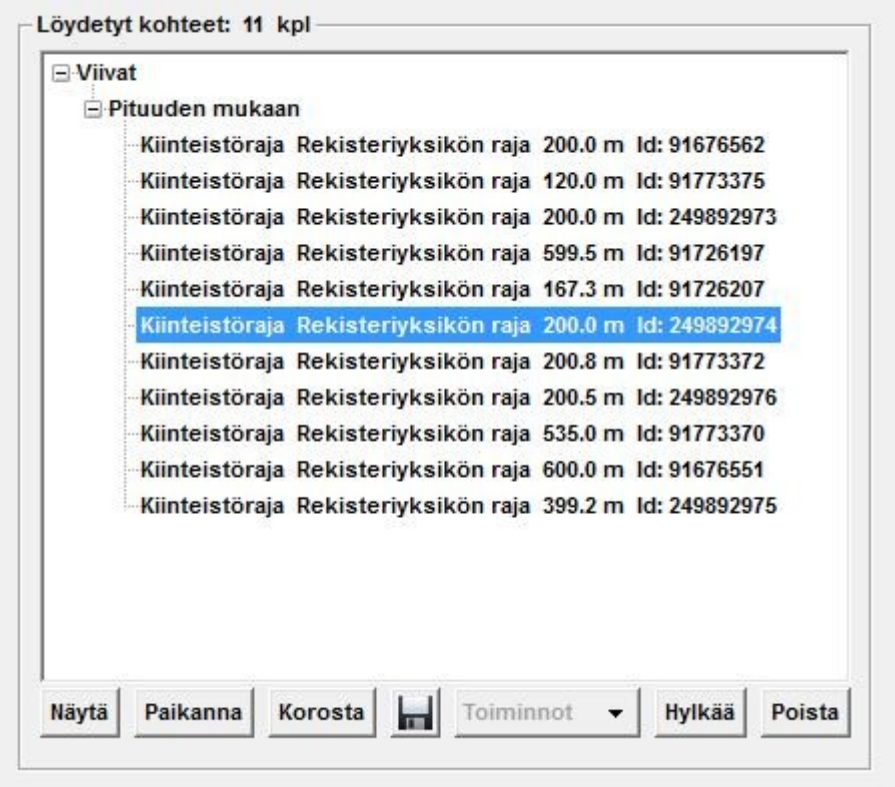


Kuva 6. Hakuehtojen ja kohdeluokkien valinta

Päädyin ratkaisuun, jossa jaoin ikkunan kahteen osaan. Vasemmalle puolelle sijoitin hakuehdot ja kohdeluokat (kuva 6). Ne toteutin välilehdillä, jossa jokaiselle tyypille (viivat, pisteet, alueet) on oma välilehtensä. Välilehdestä voidaan siirtyä toiseen klikkaamalla yläosassa näkyvää välilehden nimeä. Välilehden nimen perään asetin luvun näyttämään, kuinka monta hakuehtoa välilehdeltä on valittu. Välilehden yläosassa valitaan hakuehdot klikkaamalla hakuehdon vieressä olevaa ruutua. Samassa yhteydessä voidaan myös syöttää kenttään hakua ohjaava arvo, esimerkiksi Pituuden mukaan -hakuehdossa lyhyin ja pisin hakuun mukaan tuleva viiva.

Kohdeluokat-osiossa välilehden alaosassa valitaan haettavat kohdeluokat, joita voidaan valita yksi tai useampia. Tämä lista on toteutettu hierarkkisena puunäkymänä,

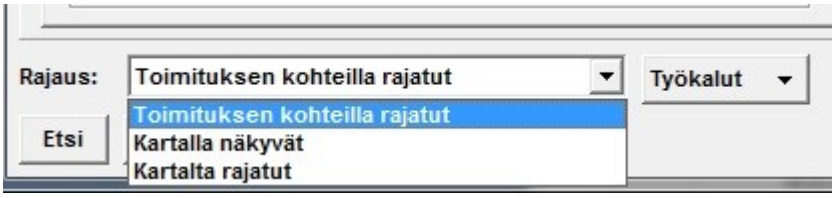
jossa esimerkiksi valittaessa arvokohteet tulee mukaan kaikki arvokohdelajit. Puun vieressä olevasta plus-merkkiä muistuttavasta painikkeesta voidaan laajentaa lista näyttämään myös alemman tason kohteet ja valita kohdeluokkia pelkästään niistä. Kohdeluokat-osion alaosaan sijoitin toimintopainikkeet Valitse kaikki, Laajenna, Kutista ja Tyhjennä valinta. Näillä voidaan ohjata listaa. Samat toiminnot onnistuvat myös listan kohdalla hiiren oikealla painikkeella klikatessa avautuvasta valikosta.



Kuva 7. Löydetyt kohteet -lista

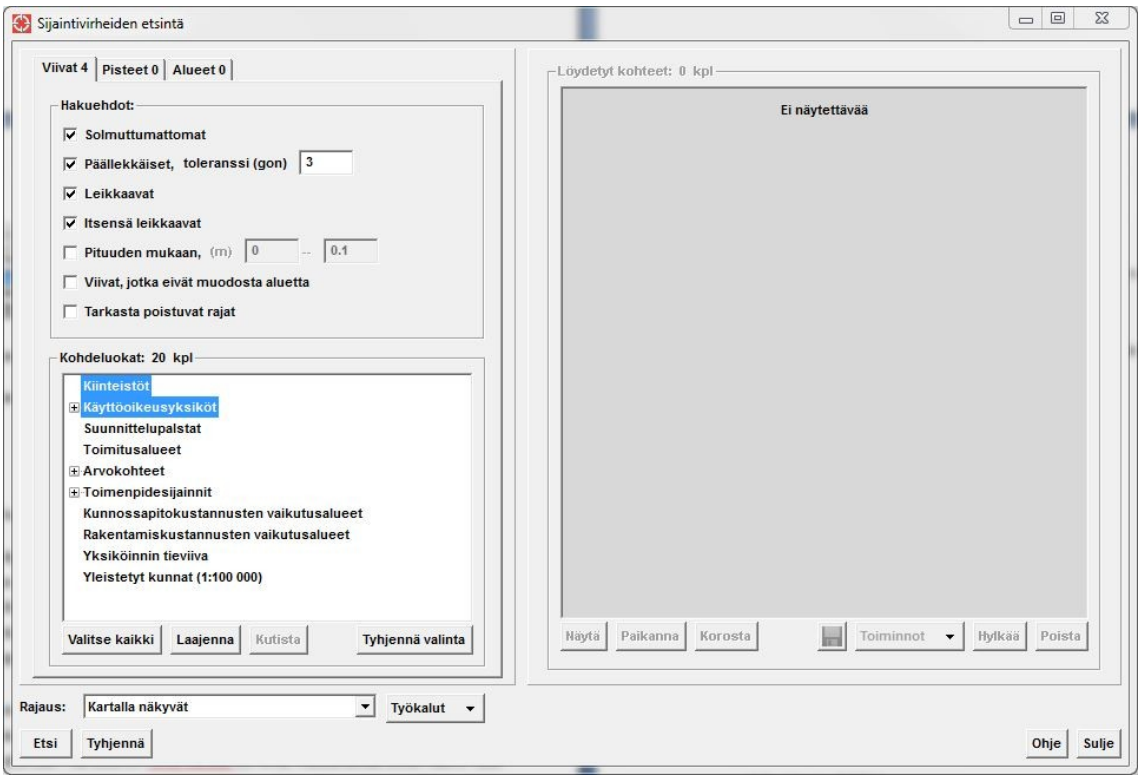
Oikealle puolelle ikkunassa sijoitin listan, johon löydetyt kohteet lisätään haun jälkeen (kuva 7). Listan toteutin puunäkymä-mallisena (*tree view*). Koska puunäkymässä on mahdollista rakentaa hierarkkisia listoja, oli tällainen lista siksi mielestäni paras valinta, koska löydetyt kohteet saadaan järjesteltyä suoraan hakuehtonsa alle. Listan alle sijoitin myös toimintopainikkeen Näytä, Paikanna ja Korosta, joilla voidaan avata listasta valitun kohteen ikkuna sekä paikantaa ja korostaa kohde karttaikkunassa. Hylkää-painike poistaa kohteen näkyvistä listasta, mutta ei poista itse kohdetta tietokannasta. Poista-painike poistaa kohteen myös tietokannasta. Kun tietynlainen kohde on valittu listasta, Toiminnot-painikkeesta voidaan avata valikko, jossa on toimintoja tietyille vir-

heille. Tallennuspainikkeen, jonka tunnistaa disketti-kuvakkeesta, kautta voidaan tallentaa lista tekstitiedostoksi.



Kuva 8. Rajauksen valinta

Ikkunan alaosassa on valinta, jolla asetetaan hakualue eli rajaus (kuva 8). Vaihtoehtoja on kolme: kartalla näkyvät, kartalta rajatut ja toimituksen kohteilla rajatut.



Kuva 9. Sijaintivirheiden etsintä -ikkuna kokonaisuudessaan

Koko ikkunaan kohdistuvat painikkeet ovat ikkunan alaosassa (kuva 9). Etsi-painike käynnistää haun. Tyhjennä-painike tyhjentää koko ikkunan valinnat. Ohje-painike avaa

ikkunan ohjeen ja Sulje-painike sulkee ikkunan. Työkalut-painikkeesta avautuvan valikon kautta voidaan avata ikkunan käytön yhteydessä yleisimmin tarvittuja välineitä.

Sijaintivirheiden etsintä -ikkuna käynnistetään pääkarttaikkunan Työkalut-valikon ”Etsi sijaintivirheet…” -painikkeesta.

4.2.3 Hakuprosessi

Kun sain käyttöliittymän rakennettua mietin, miten itse haku voisi toimia. Tähän jouduin käyttämään paljon aikaa, koska haun tulee toimia mahdollisimman tehokkaasti. Toimintaa muutettiin vielä versioon 2014/02. Pääpiirteissään tein haun toimimaan seuraavasti: Hausta tehtiin erillinen prosessi, joka pyörittää vain ikkunasta valitut hakuehdot ja kohdeluokat läpi ja kerää samalla löytyneet kohteet, eli mahdolliset virheet, muuttujaan.

Hakuprosessi hakee tietokannasta yksitellen käsittelyyn jokaisen hakuehtoon sopivan geometriakohteen, jolle jokainen valittu hakumetodi tekee tarkastelun. Tarkastelussa pyritään löytämään mahdollinen virhe. Version 2014/02 muutoksessa siirsin kaikki haut samaan silmukkaan, niiltä osin kuin tämä oli mahdollista. Tämä muutos nopeutti hakuja, kun tietokannasta haetaan sama geometriakohde vain kerran. Silmukka tarkoittaa ohjelmoinnissa rutiinia, joka jatkuu niin kauan kuin asetetut reunaehdot täyttyvät. Tässä tapauksessa siis niin pitkään, kuin rajauksen sisältämiä kohteita on tarkistamatta.

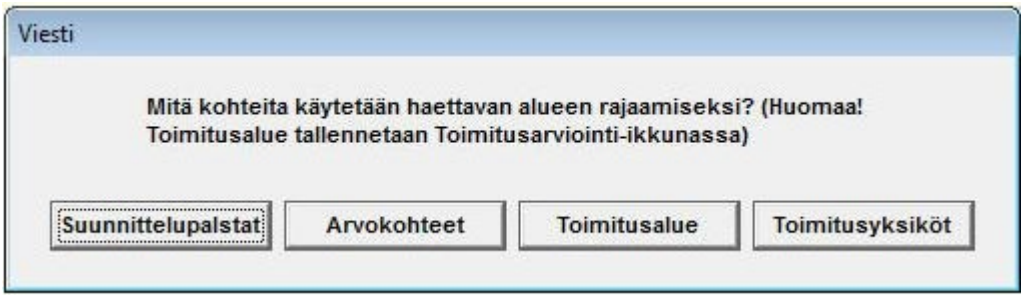
Kun kaikki geometriakohteet on käyty valitulla rajauksella tietokannasta läpi, syötetään löydetyt kohteet listalle. Listan kohteet viittaavat suoraan tietokannan kohteisiin, joten kun listan toiminnolla poistetaan geometriakohde tai muokataan sitä, se häviää myös tietokannasta.

4.2.4 Hakumetodien sovittaminen

Minulla oli käytössä vanhojen hakuikkunoiden sovelluskoodi, jota käytin hyväkseni uusia hakumetodeita tehdessäni. Usein nämä vaativat vain pientä sovittamista uutta hakuikkunaa varten.

4.2.5 Haun rajaukset

Minulla oli käytössäni sovelluskoodia vanhojen ikkunoiden rajauksiin, ”Kartalla näkyvät” ja ”Kartalta rajatut”, jotka toteutin samalla tavalla. Kartalla näkyvät -rajaus muodostaa hakualueen pääkarttaikkunassa (kuva 1) näkyvän alueen mukaiseksi. Tämä on nopein tapa rajata hakualue. Kartalla rajatut -rajauksessa käyttäjän tulee rajata pääkarttaikkunasta geometrilla sulkeutuva alue, jonka sisällä haku tehdään.



Kuva 10. Kysymysikkuna, joka esitetään rajauksen ”Toimituksen kohteilla rajatut” yhteydessä.

Uutena rajauksena haluttiin mukaan ”Toimituksen kohteilla rajatut”, joka on käytössä kun JAKOkii-sovelluksessa on maanmittaustoimitus avattuna. Se ottaa rajaukseen mukaan, riippuen toimituksen tyypistä ja sen sisältämistä alueista, toimitusyksiköt tai arviointitoimituksissa joko suunnittelupalstat, arvokohteet tai toimitusalueen eli arviointialueen. Näistä vain yksi vaihtoehto on kerrallaan mahdollinen. Jos vaihtoehtoja on avoinna olevalla toimituksella useita, käyttäjälle esitetään kysymysikkuna (kuva 10), jossa käyttäjä valitsee alueet, joilla hän haluaa rajauksen tehdä.

4.3 Kirjastointi

Kirjastointi tarkoittaa muutoksien viemistä sovelluskoodikirjastoon. Käytännössä sovelluskoodikirjasto on hakemisto Maanmittauslaitoksen sisäisellä verkkolevyllä. Kirjastointi tehdään tallentamalla sovelluskoodin sisältävä tekstitiedosto hakemusrakenteeseen sille kuuluvaan paikkaan, tilanteesta riippuen joko uuteen tai olemassaolevaan tiedostoon.

Sovelluskoodia voidaan ajaa tietokoneella vasta sen kääntämisen jälkeen. Sovelluskoodikirjastosta muodostetaan *image*, eli tiedosto josta käännetty sovellus ladataan

käyttäjälle. Uuden sovellusversion muutokset tulevat käytettäviksi kirjastoinnin jälkeisessä *imagessa*. Tuotantosovelluksessa uusi *image* otetaan käyttöön vain versionvaihdon yhteydessä. Paikkaukset mahdollistavat pienempien muutosten käyttöönoton sovellusversioiden välissä ilman tarvetta uuteen *imageen*.

Kirjastoinnissa on tärkeää, että kirjastointi tehdään oikeaan hakemistoon ja tiedostoon. Samalla viedään kirjastoon myös tarvittavat resurssitiedostot. Resurssitiedostoja ovat esimerkiksi viestitiedostot. Viestitiedostosta sovellus hakee näytettävät ikkunan viestit, eli selkokieliset tekstit, esimerkiksi painikkeissa lukevat toiminnot. Viestitiedostot tehdään sekä suomen- että ruotsinkielisinä, jotta sovelluksen viestit toimisivat kummallakin kielellä. Resurssitiedosto on myös tallennuspainikkeen diskettikuvake, jonka vein kirjastoon.

JAKO-järjestelmä on suunniteltu siten, että se kootaan hierarkkisista olioryhmistä eli moduuliryhmistä. Olioryhmän koostuu tehtäviltään samantyyppisistä olioista, jotka muodostavat sovelluksessa palvelukokonaisuuden. Palvelukokonaisuus on yleensä kirjastoitu samaan hakemistoon. Sijaintivirheiden etsintä kirjastoitiin *geometrian_kasittely* -moduliin. [13]

4.4 Dokumentointi

Dokumentointi tarkoittaa muutosten kirjaamista ylös. Dokumentointi tehdään sekä paikkausten että versionvaihdossa käyttöön otettavien muutosten osalta aina sovelluskoodiin ja ylläpidon indeksiin. Uudesta sovellusosasta tai muusta suuremmasta muutoksesta voidaan tehdä myös erillinen dokumentti, jossa kuvataan sovellusosan toimintaa tarkemmin. Sovelluskoodissa dokumentointia tehdään kommentteina, joihin merkitään muutoksen tekijä, ajankohta ja tieto mitä ollaan muutettu, sekä muutoksen syy. Tämä helpottaa myöhemmin sovelluskoodiin palaamista.

KITE-palveluryhmässä kaikista muutoksista ja paikkauksista tehdään merkintä ylläpidon indeksiin (liite 2). Ylläpidon indeksiin merkitään havaittu ongelma ja se, miten tämä on korjattu, uudet ja muutetut metodit sekä muutosten tekijä ja kirjastointipäivämäärä. Ylläpidon indeksi on apuna testauksen suunnittelussa sekä mahdollisten tulevien ongelmien selvittelyssä. Siitä on apua myös JAKOkii-päivystäjälle, koska sovelluskoodiin

tehdyt muutokset löytyvät nopeasti ja havaittu ongelma voidaan ohjata muutoksen tehneelle kehittäjälle.

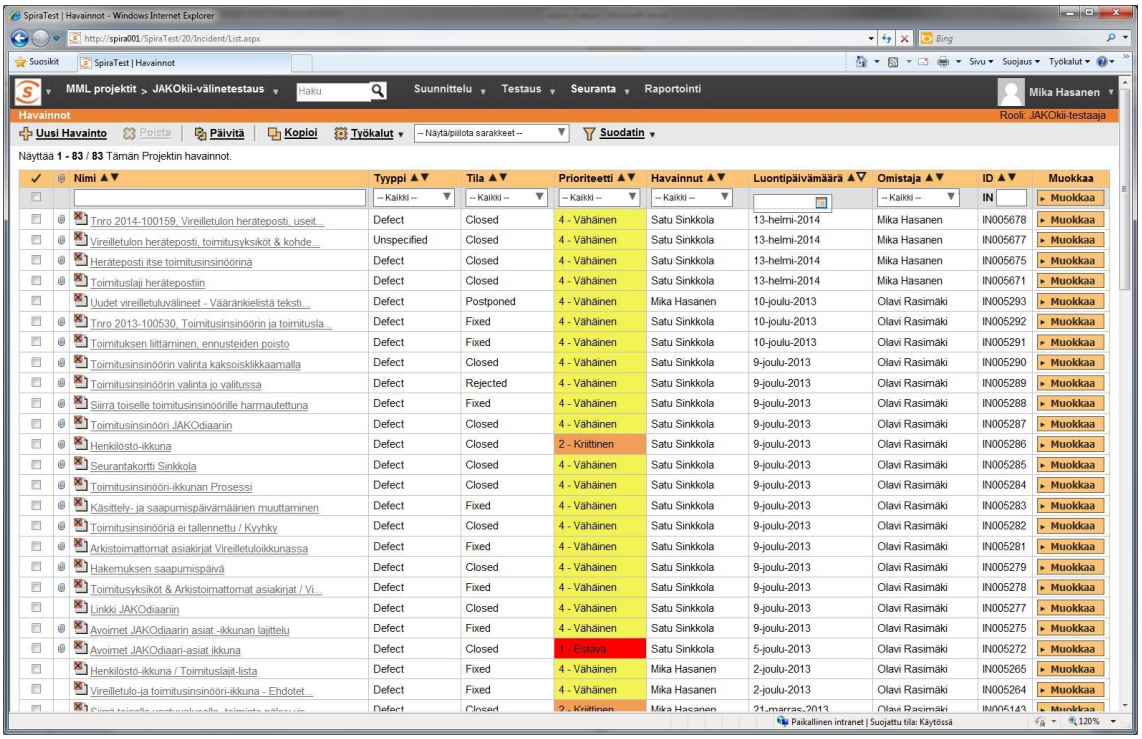
4.5 Ohjeistus

Muutokset JAKOkii-sovellukseen vaativat yleensä myös muutoksen sovelluksen ohjeisiin. Ohjeistus päivitetään aina vastaamaan uutta tuotantoversiota. Ohjeistukseen sisältyy myös muutostiedote, jossa kerrotaan, mitä muutoksia versionvaihdos tuo sovellukseen. KITE-palveluryhmässä on ohjeistaja, jonka vastuulla ohjeistus on. [6]

Sijaintivirheiden etsinnästä kirjoitettiin uusi luku ohjeisiin. Ohjeet laati ohjeistaja vanhojen ikkunoiden ohjeiden sekä ohjeistajalle ilmoittamieni muutosten pohjalta. Kehittäjänä tarkastin vielä ohjeet, kun ne saatiin valmiiksi. Ohjeet ovat käyttäjän apuna sovellusta käytettäessä.

4.6 Testaus

Maanmittauslaitoksessa JAKOkii-sovelluksen testaus jakaantuu kahteen osaan: välinetestaukseen ja versionvaihdon testaukseen [5]. JAKOkii-järjestelmän ylläpitoprosessiin kuuluu olennaisena osana uusien välineiden ja korjattujen välineiden testaus. Tätä kutsutaan välinetestaukseksi. Välinetestausta tehdään pitkin vuotta sitä mukaa, kun kehittäjät saavat rakennettua uusia välineitä ja korjattua vanhoja. [5] Välinetestauksessa testataan vain yhtä välinettä kerrallaan. Välineen käyttöliittymä ja sen toiminnot testataan erillään prosessista. Välinetestaus tehdään ennen versionvaihdon testausta ja testaukseen otetut välineet kirjataan testitapauksina Maanmittauslaitoksessa käytössä olevaan Spira-ohjelmistoon (kuva 11). [5]



Kuva 11. Spira-järjestelmä

Inflectra-nimisen yrityksen kehittämä Spira-ohjelmisto on tarkoitettu testausprosessin hallintaan. Tietokantaohjelmistoa käytetään web-selaimella. Kehittäjä ja testaaja laativat yhdessä jokaisen testattavan työn testitapaukset Spiraan. [5] JAKOkii-sovelluksen versionvaihdon testaukseen kootaan Spirasta vuoden aikana valmistuneiden välineiden testitapaukset, jotka testataan neljän työprosessin mukaan (perustoimitukset, arviointitoimitukset, tilusjärjestelyt ja kiinteistörekisterinpito). Versionvaihdon testaus tehdään Maanmittauslaitoksessa yleensä neljän testausviikon aikana siten, että testausviikkojen väliin jää kehittäjille kaksi tai kolme viikkoa aikaa korjata havainnot. Jokaista testausviikkoa varten rakennetaan sovelluskoodista uusi *image*. [5]

Versionvaihdon testausviikkojen ajaksi Pasilan palvelupisteeseen kutsutaan noin kymmenen testaajaa. Nämä henkilöt tekevät jokapäiväistä työtään omissa prosesseissaan. Testausviikkojen aikana he testaavat uutta sovellusversiota pyrkien käyttämään sovellusta kuten he itse toimitustuotannon töissään tekevät. Sovelluskehittäjät eivät välttämättä hallitse toimitustuotannon prosessienmukaista työskentelyä. Periaatteena on, ettei kehittäjä itse testaa omia muutoksiaan, vaan tämän tekee joku toinen. [5]

Kun testaaja aloittaa testaamisen, hän tutustuu ensin testitapaukseen. Tämän jälkeen hän avaa testitapauksen Spirassa ja aloittaa uuden testiajon. Hän testaa välineen testitapaukseen kirjoitettujen vaiheiden mukaan. Kun vaihe on testattu, testaaja merkitsee vaiheen hyväksytyksi (passed), jos testitapauksesta ei ole löytynyt huomauttamista. Jos jossakin testiajon vaiheessa on kuitenkin huomauttamista, testaaja merkitsee tämän hylätyksi (failed) ja kirjoittaa havainnon siihen liittyvine virheilmoituksineen. [5]

Kehittäjä näkee järjestelmässä vastuullaan olevien testattavien töidensä testiajojen tulokset. Jos testiajo on *open*-tilassa, on siinä usein jotain korjattavaa. Kehittäjä lukee järjestelmästä testaajan kommentit ja yrittää toistaa tilanteen omalla työasemallaan. Jos kehittäjä saa tapauksen toistettua ja virheen korjattua, hän voi muuttaa testiajon tilan korjatuksi (fixed). Tämän jälkeen testaaja testaa tapauksen uudestaan, ja tapaus palautuu kehittäjälle niin kauan, kunnes kaikki virheet on korjattu ja tapaus voidaan sulkea (closed). [5]

4.7 Käyttöönotto sovellusversiossa 2013/2

JAKOkii-sovelluksesta julkaistaan vuosittain uusi versio helmikuun toisella viikolla. Uuden version myötä otetaan käyttöön tämän sisältämät korjaukset ja laajennukset. Tänä aikana sovellus on muutaman päivän käyttökiellossa, kun muutoksia jaellaan eri palvelupisteisiin. Jos käyttäjällä on ongelmia sovelluksen kanssa, voi hän ottaa yhteyttä sovellustukeen, jossa mahdollisesta virheestä tehdään tarvittaessa tukitapaus ja se välitetään sovellusryhmän päivystäjälle. Päivystäjä korjaa virheen itse tai välittää sen toiselle kehittäjälle korjattavaksi. Usein ensimmäisinä päivinä käyttöönoton jälkeen saattaa ilmetä pieniä virheitä, joita ei testauksessa ole havaittu. Nämä korjataan pikaisesti paikkauksilla.

Ensimmäinen versio Sijaintivirheiden etsintä -ikkunasta otettiin käyttöön helmikuussa 2013. Itse en ollut tuolloin työsuhteessa Maanmittauslaitokseen. Kevään kuluessa ikkunan toiminnassa oli havaittu muutama pieni virhe, jotka korjattiin. Toisen version käyttöönotossa helmikuussa 2014 olin itsekin kehittäjänä käyttöönottoa todistamassa. Tietooni ei helmikuun loppuun mennessä tullut ikkunan toimintaa haittaavia virheitä.

Käyttöönottoon liittyy muutosten tiedottaminen käyttäjille, joka tehdään muutostiedotteella. Maanmittauslaitoksen palvelupisteissä koulutetaan myös tarpeen mukaan käyttäjiä, kun sovellukseen on tehty suurempia muutoksia.

4.8 Seuraava sovellusversio 2014/2

Sovelluksen käyttäjien yhteydenotot sovellustukeen johtivat muutamiin Sijaintivirheiden etsintä -ikkunaa koskeviin tukitapauksiin. Ristiriitaisten palstatunnusten ja palstavirheiden etsinnässä listalle nousi turhaan muutospalstoja, jotka tosiasiassa olivat aivan kunnossa.

Sovelluksen loki-ikkunan kautta välitettiin käyttäjälle tarpeettomia viestejä, jotka joissain tapauksissa saattoivat hämmentää käyttäjää. Lisäksi käyttäjät toivoivat, että Sijaintivirheiden etsintä -ikkunan haku varsinkin rajausta ”Toimituksen kohteilla rajatut” käytettäessä toimisi nopeammin. [14] Sovelluskoodia kirjastoitaessa ensimmäisen kerran päädyttiin väliaikaiseen ratkaisuun. Koska ikkuna on tarkoitus ottaa käyttöön myös JAKOmtj-sovelluksessa, pitäisi yhteiskäyttöön kelpaavat luokat kirjastoida sovelluksen ytimeen.

Edellä mainitut asiat otettiin huomioon, kun ikkunaan tehtiin muutoksia helmikuun 2014 versioon. Muutospalstojen nouseminen turhaan listoille korjattiin, ja tarpeettomia lokiilmoituksia vähennettiin. ”Toimituksen kohteilla rajatut” -rajausta muutettiin toimimaan nopeammin siten, että nyt käyttäjän tulee valita jokin esitettävistä vaihtoehdoista, kun edellisessä versiossa rajaus otti kaikki nämä mukaan hakuun. Sovelluskoodi myös kirjastoitiin sovelluksen ytimeen ja luokkajakoa muutettiin siten, että käyttöliittymä ja toimintalogiikka eriytettiin.

5 Johtopäätökset ja yhteenveto

Insinöörityössäni kehitettiin, testattiin ja otettiin käyttöön uusi väline JAKOkiisovelluksessa. Aikaa työn aloittamisesta käyttöönottoon kului useita kuukausia, sillä suurempia muutoksia voidaan ottaa käyttöön vain kerran vuodessa versionvaihdon yhteydessä. Sovelluksessa käyttöönotettu Sijaintivirheiden etsintä -ikkuna vähentää Maanmittauslaitoksen tuotantoprosesseissa sijaintitarkistuksiin käytettävää työaikaa.

Sijaintivirheiden etsintä -ikkunasta saatiin käyttäjiltä palautetta helmikuun 2013 version haun hitaudesta, kun Toimituksen kohteilla rajatut -vaihtoehto oli käytössä. Tulleen palautteen perusteella hakua nopeutettiin vuoden 2014 versioon. Vastuualuepäällikkö Mikko Peltokorven mukaan sovelluksen käyttäjiltä tuleva palaute on lähes poikkeuksetta luonteeltaan negatiivista. Kääntäen voidaan sanoa, että kun sovellukseen ollaan tyytyväisiä, ei käyttäjiltä tule minkäänlaista palautetta. [15] JAKOkii-sovellus on menestynyt Maanmittauslaitoksen sisäisissä asiakastyytyväisyyskyselyissä erittäin hyvin. Vuonna 2013 käyttäjät antoivat arvosanaksi JAKOkii-sovellukselle 3,89 asteikolla 1–5. [16]

Tulevaisuudessa Sijaintivirheiden etsintä -ikkuna pyritään ottamaan käyttöön myös JAKOmtj-sovelluksessa. JAKOmtj-sovelluksessa on vielä käytössä JAKOkii-sovelluksen vanhojen ikkunoiden tapaiset hakuikkunat. Myös JAKOmtj-sovelluksessa Sijaintivirheiden etsintä -ikkunan vaikutukset sovelluksen käytettävyyteen ja työaikaan olisivat merkittäviä.

Opin työn aikana paljon, sillä tämä oli ensimmäinen kerta, kun kehitin täysin uuden välineen JAKOkii-sovellukseen. Erityisesti taitoni luokkajaon suunnittelussa ja olio-ohjelmoinnin modulaarisen ohjelmointityylin hyödyntämisessä kehittyivät työn aikana. Myös kokonaiskuva sovelluskehitysprosessista selkiintyi, koska osallistuin prosessin jokaiseen vaiheeseen.

Lähteet

1 Toiminta ja tehtävät. 2014. Verkkodokumentti. Maanmittauslaitos. <http://www.maanmittauslaitos.fi/toiminta/organisaatio/toiminta-tehtavat>. Luettu 18.2.2014

2 Pykälä, Tarja. Johtava sovellusasiantuntija, Maanmittauslaitos. Sähköpostiviesti 27.2.2014.

3 Peltokorpi, Mikko. Vastuualuepäällikkö, Maanmittauslaitos. Keskustelu 25.2.2014.

4 Maanmittauslaitoksen maastotietokohteet. 2013. Verkkodokumentti. Maanmittauslaitos. <http://www.maanmittauslaitos.fi/sites/default/files/Maastotietokohteet_2013.pdf>. Päivitetty 12.12.2013. Luettu 5.3.2014.

5 Silvennoinen, Juhani. 2014. Sovellusasiantuntija, Maanmittauslaitos. Keskustelu 4.3.2014.

6 Ylläpitoprosessin kuvaus KITE-sovellusryhmässä. 2010. Maanmittauslaitos. Word-dokumentti. Päivitetty 30.6.2010.

7 Ohjelmointi I. 2011. Verkkodokumentti. Itä-Suomen yliopisto. <http://cs.joensuu.fi/~appro/materiaalit/o1/01-01.php>. Luettu 20.2.2014.

8 Smallworld Core Spatial Technology - Version 4.2 Main Documentation. 2010. Käyttöohje. GE Energy.

9 KITE-palveluryhmän verkkosivut. 2014. Verkkodokumentti. Maanmittauslaitos. Organisaation sisäinen verkko. Luettu 28.2.2014.

10 JAKOkii-sovelluksen sovellusopas. 2014. Verkkodokumentti. Maanmittauslaitos. Organisaation sisäinen verkko. Luettu 18.2.2014.

11 Pirinen, Anne. 2014. Sovellusasiantuntija, Maanmittauslaitos. Keskustelu 21.3.2014.

12 Kiinteistörekisterin ominaisuustiedot. 2012. Verkkodokumentti. Maanmittauslaitos. <http://xml.nls.fi/XML/Schema/sovellus/ktjkii/asiakasdokumentaatio/tuotekuvaukset/KiinteistorekisterinOminaisuustiedotRekisteriyksikkoRakenne.pdf>. Päivitetty 17.4.2012. Luettu 6.3.2014.

13 Mansikkamäki, Jouni. 1999. Smallworld ohjelmistostandardi. Verkkodokumentti. Maanmittauslaitos. Organisaation sisäinen verkko. Päivitetty 15.3.1999. Luettu 19.3.2014.

14 Sijaintivirheiden etsintävälineen virittelyä. 2013. Jaakkola, Esa, sovelluskehittäjä, Maanmittauslaitos. Word-dokumentti.

15 Peltokorpi, Mikko. 2014. Vastuualuepäällikkö, Maanmittauslaitos. Keskustelu 20.3.2014.

16 Järvelin, Kimmo. 2013. Keskushallinnon ja sisäisiä palveluja tuottavien valtakunnallisten yksiköiden asiakastyytyväisyystutkimus - syksy 2013. Simbain Consulting. PowerPoint-dokumentti. Organisaation sisäinen verkko. Päivitetty 31.10.2013. Luettu 20.3.2014.

Esimerkki Magik-sovelluskoodista

```
_method geometrian_kasittely.poista_duplikaatit(valitetty_rope)
    ## 2013-12-12 Mika Hasanen
    ## Poistaa ropesta kohteen, jos samaa on useampi kuin 1.

            rope2 << valitetty_rope.copy()
            _for i _over valitetty_rope.elements()
            _loop
                laskuri << 0
                _for j _over rope2.fast_elements()
                _loop
                    _if i[:sisalto] = j[:sisalto]
                    _then
                        laskuri +<<1
                    _endif
                _endloop
                _if laskuri > 1 _then
                    valitetty_rope.remove(i)
                _endif
            _endloop

_endmethod
$
```

Ylläpidon indeksin merkintä

KTJkii / Ylläpito
Laajennus/täydennys seuraavaan tuotantoversioon 2013/1

Laajennettu/täydennetty sovellusosa: Sijaintivirheiden etsintä

Havaittu ongelma / Laajennuksen tausta:	Haluttiin yhdistää kaikki sijaintihakutoiminnot yhteen ikkunaan.
Havaitsija / Laajentamisen vaatija:	Mikko Peltokorpi
Havainnon päivämäärä / Laajentamisen vaatimuspäivämäärä:	1.6.2012
Ratkaisu:	Miten näkyy sovelluksessa: Lisättiin uusi ikkunan, joka löytyy pääkarttaikkunan Työkalut-alasvetovalikosta. Muutettiin alasvetovalikkoa siten, että siitä poistettiin vanhat sijaintihakuikkunat ja lisäksi valikot ”Viivat”, ”Pisteet” ja ”Alueet”. Tekninen toteutus: Uudet source-tiedostot lisättiin polkuun \\nassu\jako\sovellusversio\ktjkii\tuotanto\201302\modules\toimitustuotanto\geometria_kasittely\kayttoliittyma\source. Tiedostojen polkua ja luokkajakoa joudutaan tulevaisuudessa kuitenkin muuttamaan, kun ikkuna otetaan käyttöön myös MTJ:ssä. Muokatut tiedostot: Pääkarttaikkuna alasvetovalikko: \\nassu\jako\sovellusversio\ktjkii\tuotanto\201302\modules\perus\sovellusmoduulit\swaf\jakokii_toimitustuotanto\resources\base\data\gui.xml Uudet metodit: ktj_sijaintieheystarkastus_plugin.aktivoi_sijaintivirheiden_etsinta_ikkuna() sijaintieheystarkastus_plugin.aktivoi_sijaintivirheiden_etsinta_ikkuna() Muokatut metodit: sijaintieheystarkastus_plugin.init()
Ratkaisija:	Mika Hasanen
Ratkaisun päivämäärä:	2012-10-10
Testaus:	[Testaus:[latauskomento, levyviittaus esim. \\nassu\jako...] [testaaja tekee linkin testausraporttiin tai kehittäjä kirjoittaa tekstin "Ei testata"]]
Ohje testaukseen:	[Ohje testaukseen: [testaussuunnitelma]]
Vaikutus ohjeistukseen:	Ohjeistettu 2012-10-23 Marjukka Heikkinen
Esimerkki testaukseen ja ohjeistukseen:	[Yksikön tunnus tai koordinaatit]