Datan visualisointi kokonaisarkkitehtuurin tukena — datan visualisointityökalun toteutus Arter Oy:n ARC-ohjelmistoon D3.js-kirjaston avulla

Noora Huttunen

Haaga-Helia ammattikorkeakoulu

AMK-opinnäytetyö

2021

Tradenomin tutkinto

Tiivistelmä

Tekijä
Noora Huttunen

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Datan visualisointi kokonaisarkkitehtuurin tukena —
datan visualisointityökalun toteutus Arter Oy:n ARC-ohjelmistoon D3.js-kirjaston avulla

Sivu- ja liitesivumäärä
51 + 12

Tämän toiminnallisen opinnäytetyön tarkoituksena oli suunnitella ja kehittää Arter Oy:n ARC-ohjelmistoon ensimmäinen versio uudesta visualisointiominaisuudesta, jota ARCia käyttävät organisaatiot voisivat hyödyntää kokonaisarkkitehtuurinsa kuvaamisessa. Työkalun tehtävä oli mahdollistaa tiettyä ARC-mallia edustavien yksittäisten ARC-elementtien tarkastelu kaaviossa, jonka esitysmuoto ja muut ominaisuudet muuttuisivat dynaamisesti asiakkaan määrittämien asetusten mukaan ja ARCin sisältämän tiedon päivittyessä.

Opinnäytetyössä kuvataan aluksi projektin teoreettinen viitekehys, joka jakautuu neljään datan visualisointia ja kokonaisarkkitehtuuria käsittelevään kappaleeseen. Ensimmäisessä kappaleessa kuvataan datan visualisoinnin merkitystä ja hyötyjä sekä tekijöitä, jotka huomioimalla voidaan varmistaa tuotetun visualisoinnin onnistuneisuus. Tärkeää on esimerkiksi tarkoituksenmukaisen esitystavan valinta, sisällöllisten ja visuaalisten ominaisuuksien tasapainottaminen sekä kognitiivisten rajoitteiden huomioiminen. Toisessa kappaleessa taas esitellään projektin toiminnalliseen osuuteen valittu JavaScript-visualisointikirjasto D3.js.

ARC-ohjelmistoa käytetään organisaatioiden kokonaisarkkitehtuurin kuvaamisen työkaluna, joten opinnäytetyön teoreettisen osuuden kolmannessa kappaleessa perehdytään sekä kokonaisarkkitehtuurin merkitykseen ja hyötyihin että sen edellytyksiin ja riskeihin. Lisäksi tarkastellaan myös tiedonhallintalain ja kokonaisarkkitehtuurin yhteyttä. Tietoperustan viimeinen kappale puolestaan käsittelee datan visualisoinnin roolia kokonaisarkkitehtuurin työvälineenä.

Opinnäytetyön jälkimmäinen osa liittyy projektin toiminnalliseen osuuteen ja siinä käydään läpi mm. visualisointityökaluun kohdistuneet toiminnalliset vaatimukset, kehityksen pääasialliset osa-alueet ja lopullinen tuotos. Toteutuskuvauksen ensimmäinen osa käsittelee projektin alussa toteutettua D3-versiopäivitystä, jonka haasteena oli D3:n modulaarisen rakenteen yhdistäminen Require.js:n mahdollistamaan asynkroniseen moduulinlataukseen. Tämän jälkeen käydään läpi uuden kaaviotyypin lisäämisen edellyttämät backend-muutokset ja tarvittavan tiedon välittäminen frontend:in Backbone-mallille. Visualisointityökalusta esitellään yleisellä tasolla sen ohjelmallinen rakenne ja tarkemmin sen kolme kiinnostavinta ominaisuutta.

Projekti toteutettiin 14.9.2020 - 31.1.2021 välisenä aikana ja kaikki sille asetetut tavoitteet saavutettiin. Projektin loppuessa uusi toiminnallisuus sisältää suunnitellut perusominaisuudet ja sen jatkokehitys voidaan aloittaa. Projektin ohjausryhmä arvioi projektin onnistuneen hyvin. Opinnäytetyön viimeisessä osassa käydään läpi opinnäytetyöprosessin etenemistä ja haasteita sekä oppimistavoitteiden täyttymistä.

Asiasanat
ohjelmistokehitys, datan visualisointi, D3.js, JavaScript, kokonaisarkkitehtuuri

Sisällys

1 Johdanto....1
1.1 Toimeksiantajan esittely....1
1.2 ARC-ohjelmiston esittely....2
1.3 Projektin lähtökohdat....3
1.4 Projektin tavoitteet ja tehtävänasettelu....4
1.5 Projektin rajaukset....5
1.6 Käsitteet....5
2 Datan visualisointi....7
2.1 Visualisointi data-analyysin välineenä....7
2.2 Datan visualisoinnin tarjoamat hyödyt yrityksille....8
2.3 Hyvät visualisointikäytännöt....8
2.3.1 Esitystapa ja kohderyhmä....9
2.3.2 Luettavuuden ja ulkonäön tasapaino....9
2.3.3 Kognitiiviset tekijät....9
2.4 Puutteellisen visualisoinnin riskit....10
3 D3.js datan visualisoinnin välineenä....12
3.1 D3:n käytön vaativuus ja suosio....12
3.2 DOM-elementtien valinta....13
3.3 DOM-elementtien visuaaliset esitystavat....13
3.4 D3:n modulaarisuus....13
4 Kokonaisarkkitehtuuri ja tiedonhallintalaki....15
4.1 Kokonaisarkkitehtuurin hyödyt....16
4.2 Kokonaisarkkitehtuurityön vaatimukset ja riskit....16
4.3 Tiedonhallintalaki....17
4.4 Kokonaisarkkitehtuurin tulevaisuus....17
5 Datan visualisointi kokonaisarkkitehtuurin välineenä....18
6 Suunnitelmakuvaus....21
6.1 Uuden kaavioiden lisäysnäkymän suunnittelu....23
6.2 Visualisointikirjaston valinta....23
6.3 Projektin riskit....24
6.4 Kehitysprosessin etenemisen suunnittelu....24
7 Toteutuskuvaus....25
7.1 D3:n päivittäminen....26
7.2 Backend-muutokset....27
7.2.1 Chart.java-luokan laajentaminen....28
7.2.2 Kaavioiden kokoamisen muutokset....29
7.3 Kaavion käyttämät moduulit....31

7.4 Backbone-mallin sisältämän datan käsittely....31
7.5 Kaavion ohjelmallinen rakenne....33
7.6 Pudotuslista-akseleiden kategorianimien käsittely ja asemointi....35
7.7 Datapisteiden päällekkäisyyden estäminen....38
7.8 Kaavion lähentäminen....40
7.8.1 Lähennystason valintatavat....40
7.8.2 D3 Brush....41
7.8.3 Zoom-funktio....42
7.9 Yhteenveto toteutuksen tuloksista....43
8 Pohdinta....45
8.1 Tulosten tarkastelu....45
8.2 Rajauksen onnistuneisuus....46
8.3 D3:n soveltuvuus projektiin....47
8.4 Kaavion jatkokehitys....47
8.5 Opinnäytetyöprosessin ja oman oppimisen arviointi....48
8.5.1 Toiminnallisen osuuden haasteet....48
8.5.2 Kirjallisen osuuden haasteet....49
8.5.3 Oppimistavoitteiden täyttyminen....49
Lähteet....52
Liitteet....57

1 Johdanto

Datan määrä maailmassa kasvaa kiihtyvää vauhtia, joten vastaavasti myös datan ja sen sisältämän tiedon kuvaamisen ja analysoinnin menetelmät muuttuvat merkitykseltään yhä tärkeämmiksi. Yksi näistä menetelmistä on datan visualisointi, joka tarjoaa helpon ja nopean välineen datan sisällön tutkimiseen ja arviointiin. Tämän lisäksi datan visualisoinnilla on erityisen merkittävä rooli myös kommunikaation välineenä, sillä sen avulla voidaan välittää monimutkaistakin tietoa helposti hahmotettavassa muodossa.

Myös kokonaisarkkitehtuuria käytetään mm. tiedon hallinnan ja kommunikoinnin välineenä. Kokonaisarkkitehtuurissa organisaation toimintaa hallitaan ja kehitetään kokonaisuutena, jonka osia ja niiden välisiä suhteita tarkastellaan ja mallinnetaan. Kokonaisarkkitehtuuriin liittyvän tiedon monimutkaisuuden takia tätä mallintamista toteutetaan usein datan visualisointia hyödyntäen. Esimerkiksi organisaation rakennetta voidaan kuvata yhteyskaavioiden avulla, kun taas prosessien kuvauksiin soveltuvat hyvin erilaiset uimaratakaaviot.

Kokonaisarkkitehtuuriin suuren tietomäärän hallitsemiseen käytetään tyypillisesti erilaisia kokonaisarkkitehtuurityökaluja. Yksi näistä työkaluista on työpaikkani Arterin tuottama ARC-ohjelmisto, jonka avulla organisaatiot voivat kuvata rakenteitaan ja toimintaansa. Tämän toiminnallisen opinnäytetyön tavoite oli suunnitella ja kehittää ARC-ohjelmistoon uusi käyttäjien visualisointitarpeita vastaava ominaisuus, joka tarjoaisi heille helpon tavan saada ajankohtaista tietoa oman organisaationsa toiminnasta ja rakenteista. Tätä tietoa voitaisiin organisaatiossa käyttää myös strategisen päätöksenteon tukena sekä muutosvaikutusten arvioinnin ja kehitystavoitteiden muodostamisen apuvälineenä.

1.1 Toimeksiantajan esittely

Arter Oy on suomalainen ratkaisutalo, jonka ydinosaamisalueeseen kuuluu laadunhallinta, kokonaisarkkitehtuuri, prosessien kehittäminen ja tiedolla johtaminen. Arter tarjoaa asiakkailleen ohjelmistoja ja asiantuntija- sekä koulutuspalveluita liiketoiminnan kehittämiseen. Arterin kaksi pääasiallista ohjelmistoa ovat IMS ja ARC. Näistä IMS on suunnattu prosessien hallintaan, ja sen ensimmäinen versio julkaistiin jo vuonna 2001. Vuonna 2014 julkaistu ARC puolestaan tarjoaa organisaatioille työvälineen kokonaisarkkitehtuurin mallintamiseen ja hallintaan, jonka avulla voidaan kuvata organisaation rakenteita ja niiden välisiä yhteyksiä visuaalisesti. (Arter 2020a; Arter 2020b; Arter 2020c.)

Vuoden 2021 alussa Arterilla on 43 työntekijää, joista kolmasosa työskentelee ohjelmistokehityksen parissa. Arter liittyi vuoden 2019 kesäkuussa Hollantilaiseen IT-ratkaisuja tarjoavaan Total Specific Solutions:iin, johon kuuluu noin 60 itsenäisesti johdettua ohjelmistoalan liiketoimintayksikköä

Euroopassa (Arter 2020f). Vuonna 2019 Arterin liikevaihto oli 4,4 miljoonaa euroa, ja pandemiatilanteesta huolimatta vuoden 2020 liikevaihto nousi 4,8:aan miljoonaan euroon. Myös vuosi 2021 alkoi hyvissä merkeissä, sillä Arterille myönnettiin tammikuussa Great Place to Work -sertifiointi (Huusari 27.1.2021). Tulevaisuudessa Arterin ohjelmistoja pyritään kehittämään modulaarisempaan suuntaan, jotta asiakkailla olisi mahdollisuus valita joustavasti käyttöönsä juuri ne ominaisuudet, joita he tarvitsevat. (Aunola 11.11.2020.)

1.2 ARC-ohjelmiston esittely

ARC-ohjelmistoa käytetään organisaatioiden kokonaisarkkitehtuurin kuvaamisen ja hallinnan työkaluna. ARC on käytössä yli 150:ssä organisaatiossa, joista useat sijoittuvat julkiselle sektorille siihen kohdistuvien laissa määriteltyjen kuvaamisvaatimusten vuoksi. ARCia käytetään esimerkiksi useissa kunnissa, ammattikorkeakouluissa ja sosiaali- ja terveysalan yrityksissä. Organisaatioiden työntekijöillä on yleensä joko katseluoikeudet tai muokkausoikeudet ARCin sisältöön, mutta kuvausten luomisesta sekä tiedon lisäämisestä ja ylläpitämisestä ovat vastuussa pääkäyttäjät. (Aunola 11.11.2020.)

ARCissa organisaation ja sen toiminnan rakenteita kuvataan malleilla, joiden avulla voidaan määritellä, mistä asioista ARCiin halutaan kirjata tietoja ja millaisista asioista organisaation toiminta koostuu. Yleisiä malleja ovat esimerkiksi Strategia, Hanke, Prosessi ja Riski. Malleille voidaan myös määritellä yhteyksiä, joita voidaan tarkastella erilaisten yhteyskaavioiden avulla.

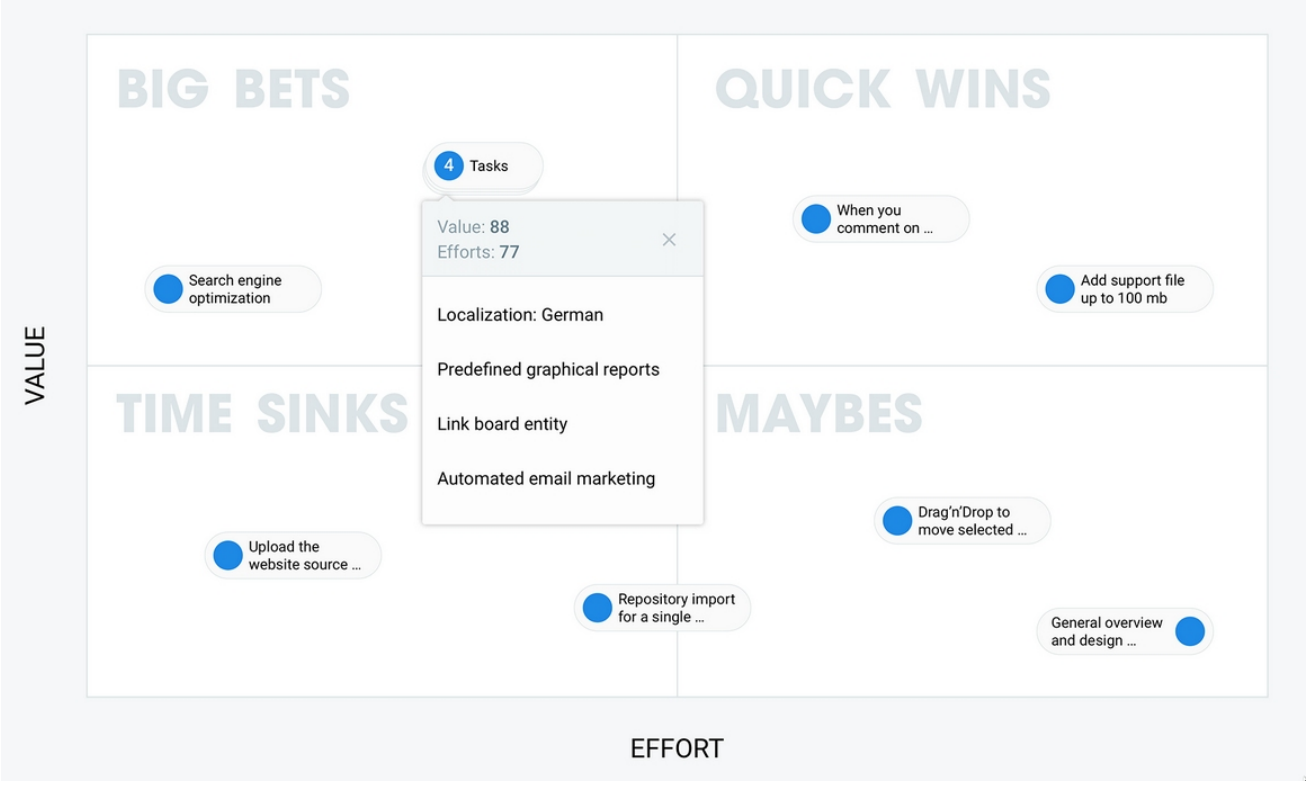
Kun mallit on määritelty, voidaan järjestelmään lisätä sisältöä elementtien muodossa. Kunkin mallin pohjalta voidaan luoda elementtejä, jotka perivät mallin ominaisuudet. Esimerkiksi Riski-mallia voisivat edustaa seuraavat Riski-elementit: Kilpailun lisääntyminen, Työtapaturma ja Kriittinen vika infrastruktuurissa. Elementit ovat siis edustamansa mallin yksittäisiä ilmentymiä. Elementtejä tarkastellaan järjestelmässä pääasiallisesti erilaisten taulukoiden avulla. Käytännössä kaikki ARCin lukuisat toiminnallisuudet liittyvät malleihin ja elementteihin tavalla tai toisella. Mallien ja elementtien ominaisuuksiin liittyvä tieto sijoitetaan kenttiin, jotka voivat olla tyypiltään esimerkiksi teksti-, lukuarvo-, pudotuslista-, monivalinta-, tai päivämäärämuotoisia.

Ohjelmallisesti ARC koostuu erillisestä Java-backend:istä ja JavaScript-frontend:istä, joiden välillä tieto liikkuu REST-rajapintojen avulla. Frontend:issä on käytössä myös Backbone.js-kirjasto, joka noudattaa ns. MVC-viitekehystä (lyhenne sanoista Model View Controller). Tämä tarkoittaa, että ARCin frontendin dataa käsitellään Backbone-malleina, joita esitetään DOM:issa erilaisten näkymien avulla. ARCin tietokantana puolestaan toimii MongoDB, johon tieto varastoidaan perinteisistä relaatiotietokannoista poiketen JSON:ia muistuttavassa muodossa.

1.3 Projektin lähtökohdat

ARC mahdollistaa jo useiden eri visualisointien tekemisen. Tällä hetkellä ARCissa ei kuitenkaan ole kaaviotyyppiä, joka kuvaisi tiettyä mallia edustavien elementtien jakaumaa ryhmittelemättömässä muodossa. Sekä ympyrä- että palkkikaavio jakavat elementit erilaisiin ryhmiin, mikä tarjoaa kyllä tietoa elementtien yleisestä tilanteesta, mutta saadakseen tarkempaa tietoa yksittäisistä elementeistä käyttäjien pitää edelleen tarkastella taulukoita, joihin elementtien tiedot on listattu. Tämä voi olla hankalaa ja hidasta etenkin, kun elementtejä on paljon.

Monet ARC-asiakkaat ovatkin esittäneet ARCin tuoteomistajalle toiveen elementtien tarkemman visualisoinnin mahdollistavan ominaisuuden kehittämisestä. Esimerkkinä he ovat yleensä maininneet projektinhallintaohjelmisto Hygger:in tarjoaman Value-Effort-matriisin, jota käytetään esimerkiksi tehtävien ja projektien priorisointiin niiden tuottaman arvon ja niihin vaadittavan työpanoksen mukaan. Kyseinen kaaviotyyppi jakautuu kuvan 1 osoittamalla tavalla neljään eri osa-alueeseen, joihin lisättävät tehtävät sijoitetaan. Kustakin tehtävästä on nähtävissä nimi, ja kun käyttäjä vie hiiren kursorin jonkin tehtävän päälle, hän pääsee näkemään myös tehtävän kuvauksen.



Kuva 1. Hygger:in Value/Effort-matriisikaavio. (Hygger 2020.)

ARC-asiakkaiden on siis tähän mennessä täytynyt käyttää jotain muuta ohjelmistoa, mikäli he ovat halunneet visualisoida ARCin elementtejä vastaavalla tavalla. Tämä on lisännyt päällekkäisen tiedon ja siten myös tarvittavan ylläpitotyön määrää. Samantyyppisen toiminnallisuuden sisällyttäminen

ARCiin taas varmistaisi sen, että luodun kaavion tiedot päivittyisivät automaattisesti mallien ja elementtien tietojen muuttuessa.

Siinä missä Hygger:in Value/Effort-matriisin akselit kuvaavat pysyvästi arvoa ja työpanosta, ARCin tapauksessa kaavioita luovan pääkäyttäjän pitäisi pystyä itse määrittelemään, mitä muuttujia kaavion akselit käyttävät. Näin kaaviota voitaisiin käyttää mahdollisimman monenlaisten elementtien kuvaamiseen. Jotta kaaviota voitaisiin käyttää esimerkiksi projektin budjetin ja toteutuneiden kustannusten vertailuun, sen tulisi suoriutua myös numeraalisia kenttiä käyttävien elementtien mallintamisesta.

Mikäli ARCiin siis toteutettaisiin riittävän monipuolinen elementtien jakautumista esittävä kaaviotyyppi, siitä hyötyisivät kaikki ARCia käyttävät organisaatiot ja siten myös kyseisten organisaatioiden tuhannet käyttäjät. Uusi kaaviotyyppi helpottaisi pääkäyttäjien arkea, toimisi päätöksenteon apuvälineenä ja välittäisi informaatiota organisaation toiminnasta ymmärrettävässä muodossa.

1.4 Projektin tavoitteet ja tehtävänasettelu

Asiakkaiden tarpeiden mukaisesti opinnäyteyöprojektin tavoitteeksi muodostui siis sellaisen visualisointityökalun suunnittelu ja kehittäminen, joka mahdollistaisi saman mallin elementtien jakauman tarkastelemisen tiettyjen vapaavalintaisten ominaisuuksien perusteella. ARCin aikaisemmista visualisointitavoista poiketen toiminnallisuuden tulisi myös mahdollistaa yksittäisten elementtien tarkastelu elementtikategorioiden sijasta. Projektin suunnitteluvaiheen aikana tätä yleistason tavoitetta täydennettiin tarkemmilla visualisointityökalun ominaisuuksiin liittyvillä tavoitteilla.

Koin projektin olevan otollinen tilaisuus kartuttaa omaa ammatillista osaamistani, joten muiden tavoitteiden lisäksi muodostin itselleni myös useita oppimistavoitteita. Varmistaakseni tuottamani visualisoinnin laadukkuuden koin tärkeäksi saada lisää tietoa etenkin dataan ja sen visualisointiin liittyen. Lisäksi halusin perehtyä aikaisempaa tarkemmin myös kokonaisarkkitehtuuriin, sillä tämä tarjoaisi minulle paremmat lähtökohdat ARCin käyttäjien tarpeiden ymmärtämiseen.

Ennen projektia olin Arterilla työskennellyt lähinnä IMS:in parissa, joten halusin projektin puitteissa myös syventää käsitystäni ARCin rakenteesta ja ominaisuuksista. Myös frontend-ohjelmointiin liittyvä osaamiseni oli melko rajallista, joten valitsemani JavaScript-visualisointikirjaston toimintaan perehtymisen lisäksi halusin parantaa osaamistani myös muiden frontend-teknologioiden (esimerkiksi CSS, HTML, Backbone.js, Require.js) osalta.

1.5 Projektin rajaukset

Projektin tarkoituksena oli toteuttaa uuden kaaviotyypin ensimmäinen versio, joka sisältäisi tarvittavat perusominaisuudet, mutta jota tultaisiin jatkokehittämään projektin jälkeen. Tämän lisäksi myös kaaviotyypin testaaminen ja tuotantoon vienti tapahtuisi vasta opinnäytetyöprojektin jälkeen normaalin kehitystyön puitteissa.

Projektin suunnitteluvaiheen aikana kävi ilmi, että uuden kaaviotyypin lisääminen edellyttäisi laajoja muutoksia kaavioiden lisäysnäkymään. Myös tämä työvaihe rajattiin pois projektin piiristä, sillä jo itse kaavion luomisprosessi osoittautui odotettua laajemmaksi kokonaisuudeksi. ARCin käyttö tapahtuu pääsääntöisesti tietokoneella, joten myös mobiililaitteiden tukeminen rajattiin pois kaavion ominaisuuksista.

1.6 Käsitteet

Data
Data on raakaa tietoa, jolta puuttuu konteksti, ja jota yleensä varastoidaan tiettyä käyttötarkoitusta varten siihen sopivassa formaatissa. Data voi olla muodoltaan esimerkiksi paperille kirjoitettuja arvoja, tai digitaalisessa muodossa tietokoneen muistissa olevia bittejä. Yleisimmin datalla viitataan nimenomaan sellaiseen tietoon, jota varastoidaan ja käsitellään digitaalisesti esimerkiksi tietokantojen kautta, tai erillisinä tiedostoina. (Norton 2020.)

Kokonaisarkkitehtuuri
Kokonaisarkkitehtuuri on organisaation kokonaisvaltainen hallintamalli. Kokonaisarkkitehtuurin avulla organisaation tai yrityksen toimintaa hallitaan ja kehitetään kokonaisuutena, joka koostuu osista ja niiden välisistä suhteista. Kokonaisarkkitehtuuri toimii suunnittelun, hallinnan ja strategisen johtamisen apuvälineenä, sillä sen avulla tiedot organisaation resursseista, varoista, kehityskohteista ja mahdollisuuksista ovat saatavilla selkeästi ja keskitetysti. (Satakieli 7.10.2014.)

HTML
HTML-lyhenne muodostuu sanoista Hypertext Markup Language. HTML:n avulla määritellään verkkosisällön merkitys ja rakenne. Hypertekstillä tarkoitetaan linkkejä, jotka yhdistävät verkkosivut toisiin verkkosivuihin joko yhden sivuston sisällä tai useamman sivuston välillä. HTML-merkintä sisältää tiettyjä ennaltamääriteltyjä tunnisteita, kuten <head>, <title> ja <body>, joita voidaan käyttää eri elementtityyppien määrittelyyn. (Mozilla 2020a.)

XML
XML-lyhenne muodostuu sanoista Extensible Markup Language. XML on samankaltainen merkintäkieli kuin HTML, mutta toisin kuin HTML, se ei sisällä ennaltamääriteltyjä tunnisteita. XML:ää

käytettäessä tunnisteet määritellään itse käyttötarpeen mukaan. XML:n perusrakenne on standardoitu, joten se soveltuu hyvin datan varastoimiseen. (Mozilla 2020b.)

CSS

CSS-lyhenne muodostuu sanoista Cascading Style Sheets. CSS:ää käytetään HTML- ja XML-dokumenttien tyylien määrittelemiseen. CSS kuvaa, millä tavalla eri elementtien tulisi piirtyä verkkosivulle. (Mozilla 2020c.)

SVG

SVG-lyhenne muodostuu sanoista Scalable Vector Graphics. SVG on XML:ään pohjautuva merkintäkieli, jota käytetään vektorigrafiikan määrittelyyn. Vektorigrafiikkana tehdyt kuvat voidaan skaalata minkä tahansa kokoisiksi ilman, että niiden laatu kärsii. (Mozilla 2020d.)

JavaScript

JavaScript on korkean tason komentopohjainen ohjelmointikieli, jota käytetään yleisesti verkkosivujen toiminnallisuuksien toteuttamiseen. JavaScript mahdollistaa verkkosivujen dynaamisen toiminnan. JavaScript-koodia ei käännetä etukäteen, vaan vasta suoritusvaiheessa ohjelman toiminnan aikana. (Mozilla 2020e.)

DOM

DOM-lyhenne muodostuu sanoista Document Object Model. DOM kuvaa esimerkiksi HTML- tai XML-dokumentin elementtejä olioina, joiden keskinäiset suhteet on esitetty puurakenteena. DOM:in kautta verkkosivujen sisältöä voidaan manipuloida ohjelmakoodin tai komentosarjojen, useimmiten JavaScript:in avulla. (Mozilla 2020f.)

D3.js

D3.js on JavaScript-kirjasto, jonka avulla voidaan manipuloida dokumentteja datan perusteella. D3 hyödyntää HTML:ää, SVG:tä ja CSS:ää datan visualisoinnissa ja DOM:in manipuloinnissa. (D3 2020a; D3 2020b.)

2 Datan visualisointi

Datan visualisoinnilla tarkoitetaan datan esittämisessä visuaalisessa kontekstissa, esimerkiksi erilaisten kuvaajien tai karttojen avulla. Visualisoinnin pääasiallinen tarkoitus on välittää informaatiota helposti hahmotettavassa muodossa. Datan visualisointi onkin tehokas kommunikaation väline, joka tarjoaa uuden tavan tarkastella olemassa olevaa tietoa (Toucan Toco 2020). Tämän lisäksi datan tarjoama kokonaiskuva on helpompi hahmottaa, kun se voidaan nähdä yhdellä silmäyksellä. (Heitzman 29.1.2019.)

Mikään ei sinänsä olisi esteenä sille, että kaikki data esitettäisiin erilaisissa taulukoissa tai tekstimuotoisissa raporteissa. Raporttien laatiminen on kuitenkin työlästä ja taulukoiden lukeminen on ihmisille huomattavasti hankalampaa kuin visuaalisen informaation omaksuminen, joten yksinkertaiseenkin tehtävään, kuten korkeimman arvon löytämiseen, saattaa kulua paljon aikaa. Tämä ongelma korostuu tarkasteltavan datan määrän kasvaessa. Visualisoinnin avulla voidaan siis myös yksinkertaistaa datan ja sen sisältämän tiedon tarkastelua. (Saranya 22.7.2019.)

2.1 Visualisointi data-analyysin välineenä

Datan arvo liittyy sen sisältämään informaatioon, joka saadaan esiin, kun data ensin käsitellään ja sen jälkeen analysoidaan. Datan analysointi on prosessi, jossa dataa käsitellään erilaisten tilastollisten ja loogisten tekniikoiden avulla. Analysoinnin tavoitteena on kuvata, tiivistää ja arvioida dataa ja näin erottaa tarkasteltava ilmiö ns. kohinasta, eli muilla tekijöillä selittyvästä tilastollisesta vaihtelusta. (The Office of Research Integrity 2020.)

Data-analyysi voi olla luonteeltaan joko eksploratiivista tai evaluoivaa. Eksploratiivisessa analyysissä tutustutaan dataan ja muodostetaan yleiskuva sen sisällöstä. Tässä vaiheessa pyritään myös löytämään relevantteja kysymyksiä datan sisältöön liittyen. Evaluoiva analyysi puolestaan pyrkii mallintamaan tutkittavaa ilmiötä yksityiskohtaisemmin. Tämän mallintamisen avulla ilmiötä pyritään sekä ymmärtämään että mahdollisesti myös ennustamaan. (Toivonen, Salmenkivi & Verkamo 2003.)

Datan visualisointi on tärkeä data-analyysin osa, sillä se tarjoaa helpon ja nopean välineen datan sisällön tutkimiseen ja arviointiin. Visualisoinnin avulla voidaan ymmärtää ja tulkita dataa sekä vetää johtopäätöksiä sen sisällöstä. Graafinen esitystapa voi myös mahdollistaa kokonaan uusien havaintojen tekemisen, mistä johtuen datan visualisoinnilla on merkittävä rooli etenkin eksploratiivisessa analyysissä. Visualisoinnin avulla voidaan helposti havaita myös mahdolliset datan sisältämät virheelliset arvot, jotka voidaan poistaa ennen analysointivaihetta (Tableau 2020). (eduCBA 2020; Toivonen ym. 2003, 9.)

2.2 Datan visualisoinnin tarjoamat hyödyt yrityksille

Yleisluontoisempien hyötyjen lisäksi datan visualisointi tarjoaa myös erityisiä hyötyjä yritysmaailmaan, jossa sillä on potentiaali ratkaista useita liiketoiminnallisia ongelmia. Jotta yritykset voisivat varmistaa oman kilpailukykynsä, niiden on erityisen tärkeää havainnoida, mihin suuntaan markkinat ovat kehittymässä. Datan visualisointi on tähän tarkoitukseen hyvä työväline, sillä sen avulla voidaan havaita kerätystä datasta trendejä ja muita säännönmukaisuuksia. Kun datan sisällöstä on saatu hyvä kokonaiskuva, voidaan arvailun sijasta tehdä tietoon perustuvia päätöksiä esimerkiksi siitä, mihin yrityksen kannattaa kohdistaa resurssinsa. (Boost Labs 25.10.2019; Saranya 22.7.2019.)

Datan visualisointi voi myös helposti näyttää yhteyden muutosten ja niistä seuraavien tulosten välillä. Mikäli esimerkiksi havaitaan, että yrityksen myynti on kääntynyt laskuun tietyllä alueella, asiaan voidaan reagoida nopeasti. Kun yritys on ensin perehtynyt siihen, mikä tilanteen on aiheuttanut, voidaan tehdä tarvittavat korjaustoimenpiteet ja tutkia niiden vaikutuksia. Erityisesti uudet datan visualisointiin käytettävät teknologiat mahdollistavat dynaamiset, reaaliajassa kehittyvät visualisoinnit, jotka tarjoavat yrityksille entistä suuremman etulyöntiaseman markkinoilla. (Boost Labs 25.10.2019.)

Datasta löydettyjen säännönmukaisuuksien perusteella voidaan myös ennustaa tulevaa. Markkinoiden kehityksen ennustamisen lisäksi voidaan arvioida myös mahdollisia riskejä ja niiden toteutumisen todennäköisyyttä. Esimerkiksi erilaiset teollisuudessa käytettävät koneistot tuottavat dataa omaan toimintaansa liittyen. Mikäli tässä datassa nähdään jotakin tavallisesta poikkeavaa, mahdollisiin virheisiin voidaan puuttua ennen kuin ne johtavat koneiston hajoamiseen tai jopa onnettomuuteen. Turvallisen toiminnan lisäksi tämä mahdollistaa myös merkittävät taloudelliset säästöt. (Digiteum 19.7.2019.)

Myös hyvä kommunikaatio on tärkeää niin yritysmaailmassa kuin muissakin ympäristöissä. Datan visualisointi tarjoaa hyvän välineen kommunikaation parantamiseen ja hyvä kommunikaatio puolestaan on yhteydessä työtyytyväisyyteen ja motivaatioon. Onkin tärkeää, että datasta nousseita havaintoja esitellään organisaation sisällä mahdollisimman laajasti sen sijaan, että tietoa välitettäisiin vain yrityksen päätöksentekijöille. (Boost Labs 25.10.2019.)

2.3 Hyvät visualisointikäytännöt

Datan visualisoinnin pääasiallinen tehtävä on muuttaa monimutkainen tieto yksinkertaiseen muotoon (Digiteum 19.7.2019). Visualisoitava tieto pitäisi siis esittää siten, että katsojien tarvitsee nähdä mahdollisimman vähän vaivaa sen omaksumisen eteen (Tableau 2020). Visualisoinnin helppolukuisuus voidaan varmistaa noudattamalla visualisointiin liittyviä hyviä käytäntöjä. Monet näistä käytännöistä ovat sidoksissa tiettyihin visualisointitapoihin, mutta on olemassa myös yleisluontoisia käytäntöjä, joita on hyvä noudattaa käytettävästä visualisointitavasta riippumatta.

2.3.1 Esitystapa ja kohderyhmä

On hyvin tärkeää valita käytettävä visualisointitapa vasta siinä vaiheessa, kun tiedetään, mitä tietoa halutaan esittää. Tällöin voidaan valita käytettäväksi esimerkiksi tarkoituksenmukaisin kaaviotyyppi ja siten varmistaa toteutetun visualisoinnin helppolukuisuus. Kun tiedetään, mitä kaaviolla halutaan kuvata, kannattaa siitä karsia pois ylimääräinen tieto, joka ei liity olennaisella tavalla kaavion ydinviestiin. (Tableau 2020.)

Tiedon oleellisuuden arvioinnissa on tärkeää ottaa huomioon, mille kohderyhmälle kaavio tullaan esittämään. Mikäli kaavio yritetään tehdä liian yleislukuiseksi, riskinä on, että yksittäisten ryhmien tiedonsaanti vaikeutuu. Onkin hyödyllisempää tuottaa useita eri kaavioita vastaamaan eri kohderyhmien tietotarpeisiin kuin yrittää sisällyttää kaikki tieto yhteen kaavioon. (Tableau 2020.)

2.3.2 Luettavuuden ja ulkonäön tasapaino

Vaikka kaavioiden selkeys onkin kriittistä, myös kaavioiden liiallinen yksinkertaisuus voi haitata tiedon omaksumista. Mikäli kaavio on hyvin yksinkertainen, se voi näyttää tylsältä, jolloin se ei kiinnitä katsojien huomiota. Lisäksi katsojat eivät myöskään jaksa katsella kaaviota niin kauaa, että he sisäistäisivät sen tarjoaman informaation. Toisaalta kauniskin kaavio voi olla täysin hyödytön, mikäli se ei sisällä järkevää viestiä. (Tableau 2020.)

Visualisointeja tehtäessä joudutaankin aina punnitsemaan sekä luettavuutta että ulkonäköä. Datan ja visuaalisten ominaisuuksien tulee aina toimia yhdessä, sillä juuri laadukkaan analyysin yhdistäminen hyvään tarinankerrontaan on datan visualisoinnin ydintarkoitus. (Tableau 2020.) Onnistuneen visualisoinnin tulisi olla selkeä ja helppolukuinen, mutta sen tulisi myös esittää tieto innostavalla tavalla (Crooks 28.7.2017).

2.3.3 Kognitiiviset tekijät

Taulukko- ja tekstimuotoisen datan käsittelyyn verrattuna visuaalisen informaation käsittely on ihmisille luotevampaa ja kykenemmekin prosessoimaan kuvia huomattavasti nopeammin kuin tekstiä (Boost Labs 25.10.2019). MIT:n tutkimuksen mukaan jo kolmentoista millisekunnin katseluaika voi mahdollistaa kuvan onnistuneen prosessoinnin (Trafton 16.1.2014). Lisäksi aivomme voivat käsitellä visuaalista tietoa myös suuremmissa osissa (Digiteum 19.7.2019). Visuaalisen informaation tehokas prosessointi varmistaa, että huomiomme kiinnittyy visuaalisiin asioihin helposti. Tästä johtuen myös datan visualisointi herättää tehokkaasti katsojien kiinnostuksen ja suuntaa heidän huomionsa visualisoinnin kertomaan viestiin. Visualisoinnit jäävät myös paremmin ihmisten muistiin esimerkiksi tekstimuotoiseen informaatioon verrattuna. (Tableau 2020.)

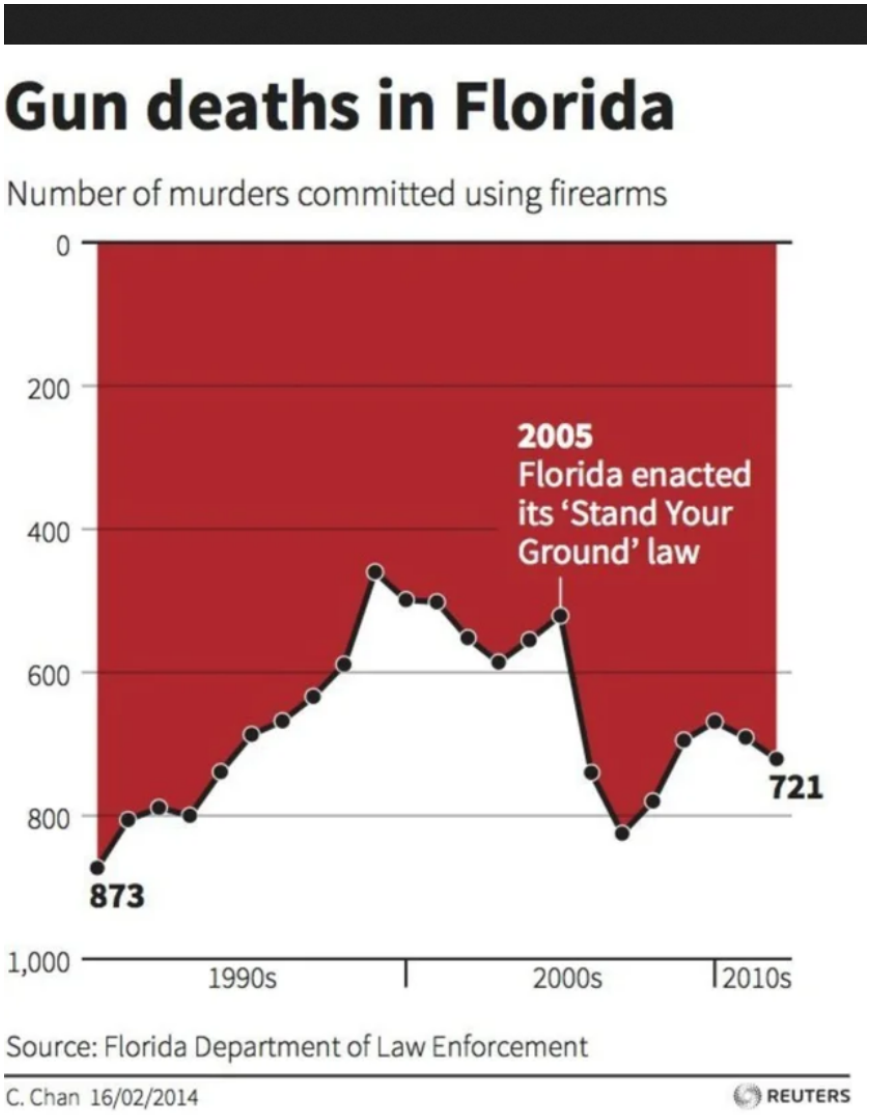
Visualisointien suunnitteluvaiheessa on tärkeää ottaa huomioon työmuistin rajoitteet. Ihmisen työmuisti on rajallinen, joten sen kognitiivista ylikuormittamista ja siitä seuraavaa informaatioähkyä tulisi välttää. Ihmisillä on kuitenkin taipumus yhdistellä pienemmistä tiedon palasista isompia lohkoja, eli merkityksellisiä kokonaisuuksia. Mikäli data voidaan siis esittää tavalla, joka mahdollistaa tiedon ryhmittelyn lohkoiksi, työmuistin rasitus vähenee (Thalmann, Souza & Oberauer 2019, 2). Suuri datamäärä voidaan esimerkiksi ryhmitellä eri kategorioihin tai suodattaa (Saranya 22.7.2019). Näin voidaan edesauttaa tiedon siirtymistä työmuistista pitkäkestoiseen muistiin, jonka kapasiteetti on huomattavasti työmuistia suurempi (Guo 22.12.2017; Cowan 18.3.2009).

Myös huomiokyvyn rajoitteet on hyvä ottaa huomioon. Niin tarpeeton informaatio kuin myös esimerkiksi ylimääräiset animaatiot kilpailevat katsojan huomiosta, joten niiden käyttö tulisi minimoida. Pahimmillaan ylimääräiset tekijät voivat johtaa ilmiöön, jonka englanninkielinen nimi on "cognitive tunneling". Tällöin katsojien huomio kiinnittyy esimerkiksi kiinnostavaan animaatioon, eivätkä he enää edes yritä ymmärtää kaavion tarjoamaa tietosisältöä. (Guo 22.12.2017.)

On myös tärkeää ymmärtää, että ihmiset tulkitsevat näkemiään asioita aina aikaisemman kokemuspohjansa avulla. Yleensä tämä edesauttaa uuden tiedon omaksumista, mutta mikäli visualisointitapa sotiikin katsojan aikaisempaa tietoa vastaan, katsojan kognitiivinen rasitus kasvaa. Ihmiset ovat esimerkiksi tottuneet siihen, että eri väreillä on tietynlainen semanttinen merkitys. Vihreä on yleensä hyvän vaihtoehdon väri, kun taas punaista käytetään varoitusvärinä. Siniseen väriin yhdistetään kylmyys, kun taas punainen kuvaa kuumuutta. Mikäli kyseisiä värejä käytetään näiden perusoletusten vastaisesti, kaavion luettavuus kärsii huomattavasti, kun katsojat joutuvat käyttämään kognitiivisia resurssejaan värien tulkitsemiseen. Kognitiivista rasitusta lisää myös hyvin erikoisten tai tavallisuudesta poikkeavien visualisointitapojen käyttö. Mikäli esimerkiksi piirakkakuvio jaettaisiinkin keskipisteen sijasta jostakin muusta kohdasta, katsojien ennakko-odotukset hankaloittaisivat kaavion lukemista. (Guo 22.12.2017.)

2.4 Puutteellisen visualisoinnin riskit

Laadukkaan visualisointi vaatii siis monien eri näkökulmien punnitsemista ja tarkkaa harkintaa, mikä ei aina ole helppoa tai nopeaa. Ehdottomasti tärkeintä on kuitenkin, että dataa visualisoidaan tavalla, joka ei anna sen sisällöstä virheellistä kuvaa. Yllättävän pienetkin tekijät saattavat johtaa kaavioon, joka voi antaa katsojille täysin väärän käsityksen visualisoitavasta asiasta (Crooks 28.7.2017).



Kuva 2. Floridan ampuma-asekuolemien määrän kehitystä kuvaava aluekaavio. (Engel 2014.)

Hyvä esimerkki harkitsemattomasta tyylikeinojen käytöstä on kuvan 2 kaavio, jossa kuvataan ampuma-asekuolemien määrän kehitystä Floridassa 1990-, 2000- ja 2010-lukujen aikana. Kuvaan on merkitty vuosi, jolloin Floridassa astui voimaan ns. "Stand Your Ground" -laki, joka oikeuttaa kansalaiset käyttämään väkivaltaisiakin voimakeinoja silloin, kun he kokevat olevansa vaarassa. Ensisilmäyksellä vaikuttaa siltä, että lain myötä ampuma-asekuolemien määrä kääntyi selkeään laskuun, mutta huolellisempi tarkastelu paljastaa kaavion käännetyn y-akselin. Todellisuudessa siis kuolemat kääntyivät lain myötä jyrkkään nousuun. Kannattaakin siis olla tietoinen siitä, että toisinaan visualisointien kertomat tarinat voivat olla ristiriidassa visualisoidun datan todellisen sisällön kanssa.

3 D3.js datan visualisoinnin välineenä

JavaScript-pohjaista datan visualisointikirjastoa etsivälle on tarjolla lukuisia eri vaihtoehtoja käyttötarpeesta ja muista käytettävistä teknologioista, esimerkiksi sovelluskehyksistä riippuen (Majorek 23.8.2020). Eräs kattavimmista ja suosituimmista kirjastoista on D3, jonka nimen lyhenne muodostuu sanoista Data-Driven Documents. Laajasti ilmaistuna D3:n pääasiallinen käyttötarkoitus onkin dokumenttien manipulointi datan perusteella, minkä D3 toteuttaa sitomalla annetun datan sisällön Document Object Model:iin, eli DOM:iin. D3:n ensimmäinen virallinen versio julkaistiin vuonna 2011, minkä jälkeen kirjastoon on lisätty valtava määrä uusia ominaisuuksia. (Wellesley 2020.)

D3 on kehitetty käyttämään toiminnassaan normaaleja web-standardeja, kuten HTML:ää, SVG:tä ja CSS:ää yksityisomistuksellisten esitysmuotojen sijasta, mikä mahdollistaa kyseisten teknologioiden kokonaisvaltaisen hyödyntämisen. Tämän toimintaperiaatteen ansiosta D3 soveltuukin lähes minkä tahansa teknologian kanssa käytettäväksi. Kannattaa kuitenkin ottaa huomioon, että samoin kuin D3, myös useat modernit sovelluskehykset, kuten React tai Vue, manipuloivat DOM:ia, mikä saattaa aiheuttaa ongelmia jos niiden välistä työnjakoa ei selkiytetä riittävästi. (Majorek 23.8.2020.)

D3:a voidaan käyttää useiden eri datamuotojen kanssa. D3 mahdollistaa esimerkiksi Javascript-taulukoiden ja -objektien sekä CSV-, JSON- ja XML-muotoisen datan käsittelyn. Tästä datasta voidaan luoda dynaamisesti elementtejä, joiden ominaisuudet voidaan myös määritellä dynaamisesti datan funktioiden avulla. (TutorialsTeacher 2020.)

Edellä mainittujen DOM-ominaisuuksien ja datan visualisointiin liittyvien ominaisuuksien lisäksi D3 tarjoaa myös animaatioon, analyysiin ja datan käsittelyyn liittyviä toiminnallisuuksia. D3 onkin siis itse asiassa enemmän kuin pelkkä visualisointikirjasto, sillä useat sen tarjoamista ominaisuuksista liittyvät datan visualisointiin vain etäisesti. (Meeks 11.6.2018.)

3.1 D3:n käytön vaativuus ja suosio

Vaikka tämä laaja toimintojen kirjo tekeekin D3:sesta äärimmäisen kattavan visualisointikirjaston, uusien D3-käyttäjien voi olla vaikeaa hahmottaa, mitä ominaisuuksia heidän tulisi käyttää. Tämän lisäksi saatavilla olevien D3-materiaalien taso vaihtelee ja kaikki niistä eivät sovellu aloittelevalle käyttäjälle (Wellesley College 2020). D3:n käytön opettelu onkin melko vaativa prosessi muihin suosittuihin visualisointikirjastoihin verrattuna. (Meeks 11.6.2018.)

Tästä huolimatta D3 on kuitenkin äärimmäisen suosittu, sillä se tukee lähes kaikkia datan visualisoinnin muotoja (Wellesley College 2020). Vuoden 2021 alussa D3:lla on GitHubissa hieman yli 95 tuhatta tähteä, kun taas esimerkiksi toisella suositulla visualisointikirjastolla, Chart.js:llä on tästä määrästä hieman yli puolet (D3 2020b; Chart.js 2020).

3.2 DOM-elementtien valinta

Yksi D3:n yleisimmistä ominaisuuksista on selection, jota käytetään haluttujen dokumentin elementtien valitsemiseen. Vastaavanlaisia valintatoimintoja tarjoavat myös CSS ja jQuery. Valituille elementeille voidaan määritellä ominaisuuksia ja myös niiden tyylejä ja HTML-sisältöä voidaan muuttaa. Siinä missä CSS:n ja jQuery:n avulla on mahdollista määritellä elementeille eri ominaisuuksia vakioiden avulla, D3 mahdollistaa elementtien ominaisuuksien ja tyylien muuttamisen dynaamisesti datan funktioiden avulla. (D3 2020a.)

Mikäli datana käytetään esimerkiksi kokonaislukuja sisältävää taulukkoa, data voidaan tällöin sitoa funktion avulla D3:n selection:iin niin, että sen ensimmäiseen elementtiin yhdistetään taulukon ensimmäinen arvo, toiseen toinen arvo, ja niin edelleen. Kun data on kerran yhdistetty selection:in elementteihin, D3 muistaa niiden sisällön, eikä dataa tarvitse sitoa elementteihin enää uudestaan. Uudesta datasta voidaan luoda elementtejä enter-valinnan avulla ja kun elementtejä ei enää tarvita, ne voidaan poistaa exit-valinnan avulla. (D3 2020a.)

3.3 DOM-elementtien visuaaliset esitystavat

D3:lla ei ole erillistä tapaa elementtien visuaaliseen esittämiseen, vaan se hyödyntää olemassa olevia web-standardeja. Datan pohjalta voidaan esimerkiksi luoda SVG-elementtejä, kuten ympyröitä, neliöitä tai viivoja, joille voidaan antaa erillisiä tyylimäärittelyjä CSS:n avulla. SVG on varteenotettava valinta silloin, kun halutaan varmistaa luotavien elementtien skaalautuvuus, eli se, että elementit piirtyvät tarkkoina niiden koosta riippumatta. SVG:n suorituskyky on hyvä, kun käsitellään joko rajallista määrää objekteja tai suurta aluetta. Mikäli kuitenkin tarkoituksena on käyttää pientä aluetta tai suurta määrää objekteja, HTML:n tarjoama canvas-elementti voi olla parempi vaihtoehto sen heikosta skaalautuvuudesta huolimatta. (D3 2020a; Tutorialspoint 2020.)

Esitettävien elementtien visuaalisia ominaisuuksia voidaan muuttaa D3:n tarjoamien transitioiden eli siirtymien avulla. Tällaisten siirtymien manuaalinen toteuttaminen olisi työlästä, joten D3 helpottaa prosessia suorittamalla tarvittavan laskennan sisäisesti ilman erillisiä määrittelyitä. Siirtymiin liittyy monesti myös animaatioita. Mikäli esimerkiksi ympyräelementin väri halutaan vaihtaa punaisesta siniseksi, D3 voi esittää tämän muutoksen visuaalisesti miellyttävällä, lineaarisella tavalla. (TutorialsTeacher 2020.)

3.4 D3:n modulaarisuus

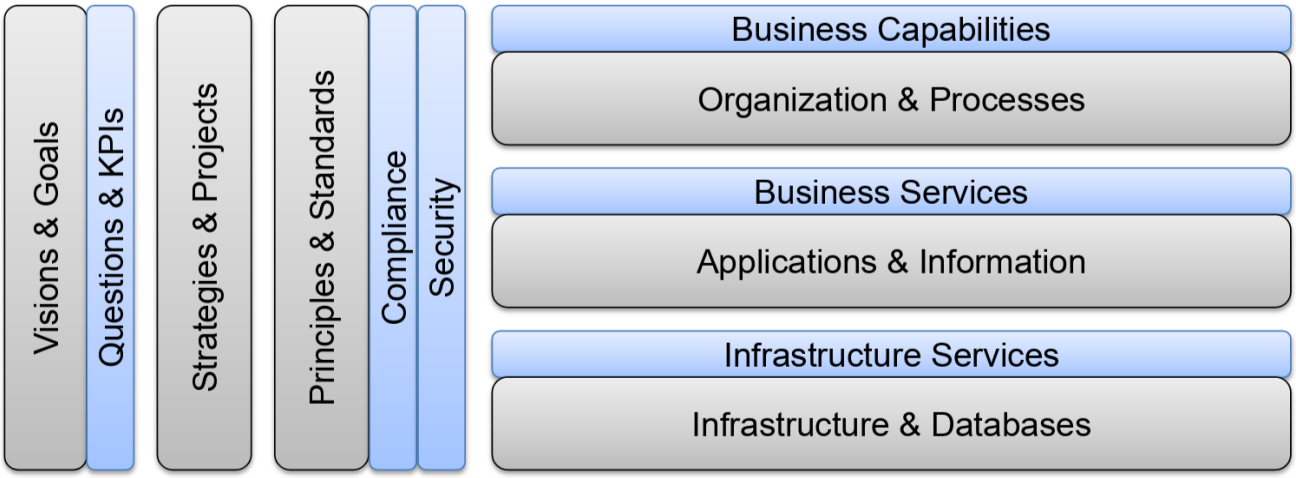
Ei olisi järkevää, että hyödyntääkseen vain yhtä D3:n toiminnallisuutta käyttäjät joutuisivat lisäämään projektiinsa koko D3-kirjaston. Tästä syystä D3:n rakenne muutettiin modulaariseksi vuonna 2016 (Bostock 28.6.2016). Nykyään D3 siis muodostuu useista erillisistä kirjastoista, joita voidaan käyttää

myös itsenäisesti. Jokaisella D3-moduulilla on oma GitHub-repositorio ja julkaisusykli nopeamman kehityksen varmistamiseksi. Modulaarisen lähestymistavan on tarkoitus myös kannustaa D3:n käyttäjiä osallistumaan D3:n kehitystyöhön mahdollistamalla uusien ominaisuuksien ja moduulien helpompi kehittäminen (Viau 15.2.2017). (D3 2020b.)

4 Kokonaisarkkitehtuuri ja tiedonhallintalaki

Vaikka arkkitehtuurista puhuttaessa mieleen saattaa ensimmäisenä nousta sanan perinteisempi, rakennustaiteeseen liittyvä merkitys, organisaatio- ja tietojärjestelmäkontekstissa arkkitehtuurilla tarkoitetaan kuitenkin jotain aivan muuta. Yleisesti tällä tasolla arkkitehtuuri on työväline, tapa, ja toimintamalli, jolla voidaan jäsentää, kuvata ja suunnitella tarkasteltavan kohteen rakenteen osia, siihen liittyviä elementtejä ja niiden suhteita sekä toisiinsa että ympäristöön. Arkkitehtuurista käytetään tarkastelutavasta riippuen useita eri käsitteitä, joista yleisimpiä ovat esimerkiksi kokonaisarkkitehtuuri, integraatioarkkitehtuuri, tietojärjestelmä-/sovellusarkkitehtuuri ja ohjelmistoarkkitehtuuri. Nämä eri arkkitehtuurin osa-alueet ovat yhteydessä toisiinsa ja muodostavat yhdessä kokonaisuuden. (Kehmet 2020.)

Kokonaisarkkitehtuurin juuret ovat englanninkielisessä termissä Enterprise Architechture, josta käytetään myös käännöstä yritysarkkitehtuuri. Siinä missä yritysarkkitehtuuri kuitenkin liittyy nimenomaan liiketoimintaan, kokonaisarkkitehtuuria voidaan käyttää kuvaamaan minkä tahansa organisaation toimintaa ja rakenteita. Esimerkiksi Suomessa julkisen hallinnon tietohallintoa on jo vuodesta 2011 asti lailla velvoitettu kuvaamaan omaa kokonaisarkkitehtuuriaan (laki julkisen hallinnon tietohallinnon ohjauksesta 10.6. 634/2011). (Isokallio 2005, 22-24.)



Kuva 3. Kokonaisarkkitehtuurin tärkeät kerrokset ja kokonaisvaikutteiset osat. (Roth, Zec & Matthes 2014, 14)

Kokonaisarkkitehtuuri tarkoittaa siis systemaattista lähestymistapaa organisaation toiminnan ja sen rakenteiden jäsentämiseen, kehittämiseen ja hallinnointiin. Organisaation toimintaa hallitaan ja kehitetään kokonaisuutena, joka koostuu osista. Näiden osien välisiä suhteita tarkastellaan ja mallinnetaan esimerkiksi kuvan 3 osoittamalla tavalla. Kokonaisarkkitehtuurityö kuuluu organisaation strategiatyöhön, johtamisprosessiin sekä talouden ja toiminnan suunnitteluun. (Kehmet 2020.)

Kokonaisarkkitehtuuriin sisältyy yleensä hierarkisia kuvauksia esimerkiksi työyhteisön periaatteista, prosesseista, tiedosta, järjestelmistä, käsitteistä, sovelluksista, infrastruktuurista, teknologioista ja muista resursseista sekä siitä, miten nämä osa-alueet ovat yhteydessä toisiinsa. Nämä kuvaukset toteutetaan ns. kokonaisarkkitehtuurikehyksen, eli taulukon, joka pitää sisällään eri tasoihin (esim. käsitteellinen, looginen, fyysinen) ja näkökulmiin liittyvää tietoa, avulla. KA-kehys toimiikin eräänlaisena kuvausten selitteenä ja ohjenuorana siitä, miten eri asioiden dokumentointi tulisi toteuttaa. (Kehmet 2020; Satakieli 7.10.2014.)

4.1 Kokonaisarkkitehtuurin hyödyt

Arkkitehtuurityö tarjoaa organisaatiolle useita eri hyötyjä. Aikaisemmista menetelmistä poiketen kokonaisarkkitehtuuri yhdistää organisaation strategian, liiketoimintojen tavoitteet ja teknologian, mikä edesauttaa organisaation strategisten tavoitteiden jalkauttamista huomattavasti. Kokonaisarkkitehtuuri kiinnittää myös huomion organisaation kokonaisuuden kannalta kriittisiin aktiviteetteihin, mikä mahdollistaa tehokkaamman resurssien käytön. Kokonaisarkkitehtuurin yhteydessä määritellään yleensä myös organisaation nykytila ja tavoitetila, mikä auttaa organisaatiota näkemään erot tämän hetkisen tilanteen ja tavoitellun tilan välillä. Kun nämä erot on tunnistettu, voidaan alkaa suunnitella, millä toimenpiteitä tulisi tehdä, jotta tavoitetila muuttuisi nykytilaksi. Kokonaisarkkitehtuurin avulla voidaan esimerkiksi priorisoida erilaisia projekteja ja investointeja. (Wallenius 6.1.2019.)

Päätöksenteon tukemisen lisäksi kokonaisarkkitehtuuri tarjoaa myös yhdenmukaisen tavan kuvata organisaation tilaa, mikä parantaa organisaation sisäistä kommunikaatiota. Kokonaisarkkitehtuuri voi myös nopeuttaa uusien ratkaisuiden suunnittelua ja toteuttamista mikäli nyky- ja tavoitetilojen kartoituksen yhteydessä havaitaan sellaisia olemassa olevia rakenteita, joita on mahdollista uusiokäyttää myös kehitteillä olevissa ratkaisuissa. (Wallenius 6.1.2019.)

4.2 Kokonaisarkkitehtuurityön vaatimukset ja riskit

Kokonaisarkkitehtuuri ei ole kuitenkaan vailla ongelmia. Arkkitehtuurityö vaatii taloudellista panostamista ja sitoutumista. Mikäli kokonaisarkkitehtuurin toteuttamiseen ei varata riittävästi resursseja, tai arkkitehtuuria tukevia päätöksiä ei tehdystä työstä huolimatta tehdä, kokonaisarkkitehtuurin tulokset jäävät todennäköisesti vaatimattomiksi. Kokonaisarkkitehtuurityö vaatii yleensä myös erikoisosaamista, minkä takia organisaatiossa tämä työ saattaa jäädä vain yhden avainhenkilön varaan. Näin ollen myös kokonaisarkkitehtuuriin liittyvä työ saattaa pahimmassa tapauksessa pysähtyä kokonaan kyseisen henkilön lähtiessä pois organisaatiosta.

Lisäksi myös kokonaisarkkitehtuurin mallintamiseen käytettävät työkalut voivat olla monimutkaisia, mikä voi aiheuttaa yllättäviä ongelmia niiden käyttön ja integraatioon liittyen. Kokonaisarkkitehtuurityöstä hyötyminen edellyttää myös, että organisaatiossa tiedetään, mikä on juuri

heidän toimintansa kannalta tärkeää. Erilaisia järjestelmäkuvauksia voidaan tehdä paljonkin, mutta jos työtä ei rajata oikein, niiden tarjoama hyöty on rajallinen. (Wallenius 6.1.2019.)

4.3 Tiedonhallintalaki

Viime aikoina kokonaisarkkitehtuuri-termin käyttö on jossain määrin vähentynyt vuoden 2020 alussa voimaan astuneen tiedonhallintalain myötä. Tämä ei kuitenkaan tarkoita sitä, että kokonaisarkkitehtuurin merkitys olisi vähentynyt, vaan sen periaatteet ovat edelleen voimassa. Uuden tiedonhallintalain tavoitteena on yhtenäistää ja päivittää julkisen hallinnon tiedonhallintaan kohdistuvaa lainsäädäntöä määrittelemällä valtakunnalliset periaatteet viranomaistietojen käsittelyyn ja tiedonhallinnan kuvaamiseen, eli käytännössä kokonaisarkkitehtuurityöhön. Tiedonhallintalain avulla pyritään varmistamaan viranomaisten tietoaineistojen yhdenmukainen ja laadukas hallinta, tietoturvallinen käsittely, sekä turvallinen ja tehokas hyödyntäminen (Tuominen 12.02.2020). (Lehtinen 7.11.2019.)

Siinä missä aiempi lainsäädäntö määritteli kokonaisarkkitehtuurin kuvaamiseen liittyvät vaatimukset hyvin karkealla tasolla, uusi tiedonhallintalaki tarkentaa julkiseen hallintoon kohdistuvia kuvausvelvoitteita, ja edelleen täydentää niitä erilaisten suositusten kautta. Sen sijaan, että velvoitteet kohdistuisivat kokonaisarkkitehtuurin noudattamiseen, julkisen hallinnon organisaatioita vaaditaan nyt toteuttamaan tietyt tiedonhallinnan kuvaukset tiedonhallintamallin muodossa. Tiedonhallintamallissa tulee kuvata vähintään organisaation toimintaprosessit, tietovarannot, tietoaineistot, tietojärjestelmät sekä tietoturvajärjestelyt (Kuntaliitto 4.12.2019). Yleisesti ottaen voidaankin siis sanoa, että tiedonhallintamalli on näkökulma arkkitehtuuriin, jossa tarkastellaan nimenomaan tiedon hallintaa organisaation rakenteissa, ja jonka kuvaamisessa käytetään kokonaisarkkitehtuurin menetelmiä (Lehtinen 7.11.2019). (Halsas 04.04.2019.)

4.4 Kokonaisarkkitehtuurin tulevaisuus

Kokonaisarkkitehtuuriin liittyy siis useita eri näkökulmia, joiden tärkeys ja ajankohtaisuus saattavat vaihdella organisaatioiden toimialasta ja tietotarpeista riippuen. On kuitenkin vaikeaa kuvitella, että kokonaisarkkitehtuuri ja siihen liityvien työtapojen merkitys vähenisi tulevaisuudessa. Päin vastoin, tuntuu todennäköiseltä, että niiden tärkeydestä voidaan tulla entistä tietoisemmiksi. Esimerkiksi psykoterapiakeskus Vastaamoon vuoden 2020 lokakuussa kohdistunut tietomurto on nostanut erityisesti tiedonhallinnan ja tietoturvan kriittisyyden jälleen yleiseen tietoisuuteen (Yle 2020). Onkin kiinnostavaa nähdä, tullaanko tämän tapauksen seurauksena kiristämään tiedonhallintaan liittyvää lainsäädäntöä entisestään.

5 Datan visualisointi kokonaisarkkitehtuurin välineenä

Datan visualisointi ja kokonaisarkkitehtuuri palvelevat siis samankaltaisia päämääriä, sillä molempien tavoitteena on muuttaa olemassa oleva tieto helpommin ymmärrettävään ja hallittavaan muotoon. Molemmat toimivat myös datan sisältämän tiedon havainnollistamisen ja kommunikoinnin työkaluina, joita voidaan käyttää esimerkiksi päätöksen tekemisen ja muutoksen seurannan tukena.

Datan visualisointi on tehokas tiedonvälitystapa etenkin silloin, kun käsiteltävä tieto on hyvin monimutkaista. Ei olekaan ihme, että datan visualisoinnilla on tärkeä rooli myös kokonaisarkkitehtuurin hallinnassa. Kokonaisarkkitehtuurissa tehtäviä visualisointeja voidaan käyttää esimerkiksi monimutkaisen tiedon kommunikointiin ja analysointiin, sidosryhmien osallistumisen edistämiseen sekä läpinäkyvyyden lisäämiseen. (Roth, Zec & Matthes 2014, 4.)

Kokonaisarkkitehtuurin tietomäärästä johtuen sen hallintaan käytetään tyypillisesti erilaisia kokonaisarkkitehtuurityökaluja, joiden pääasiallinen tehtävä on (tieto)mallien ylläpitäminen ja raporttien tuottaminen. Nämä raportit sisältävät usein myös visualisointeja, tai vaihtoehtoisesti järjestelmien luomat visualisoinnit voivat jo itsessään toimia raportteina. (Roth ym. 2014, 4.)

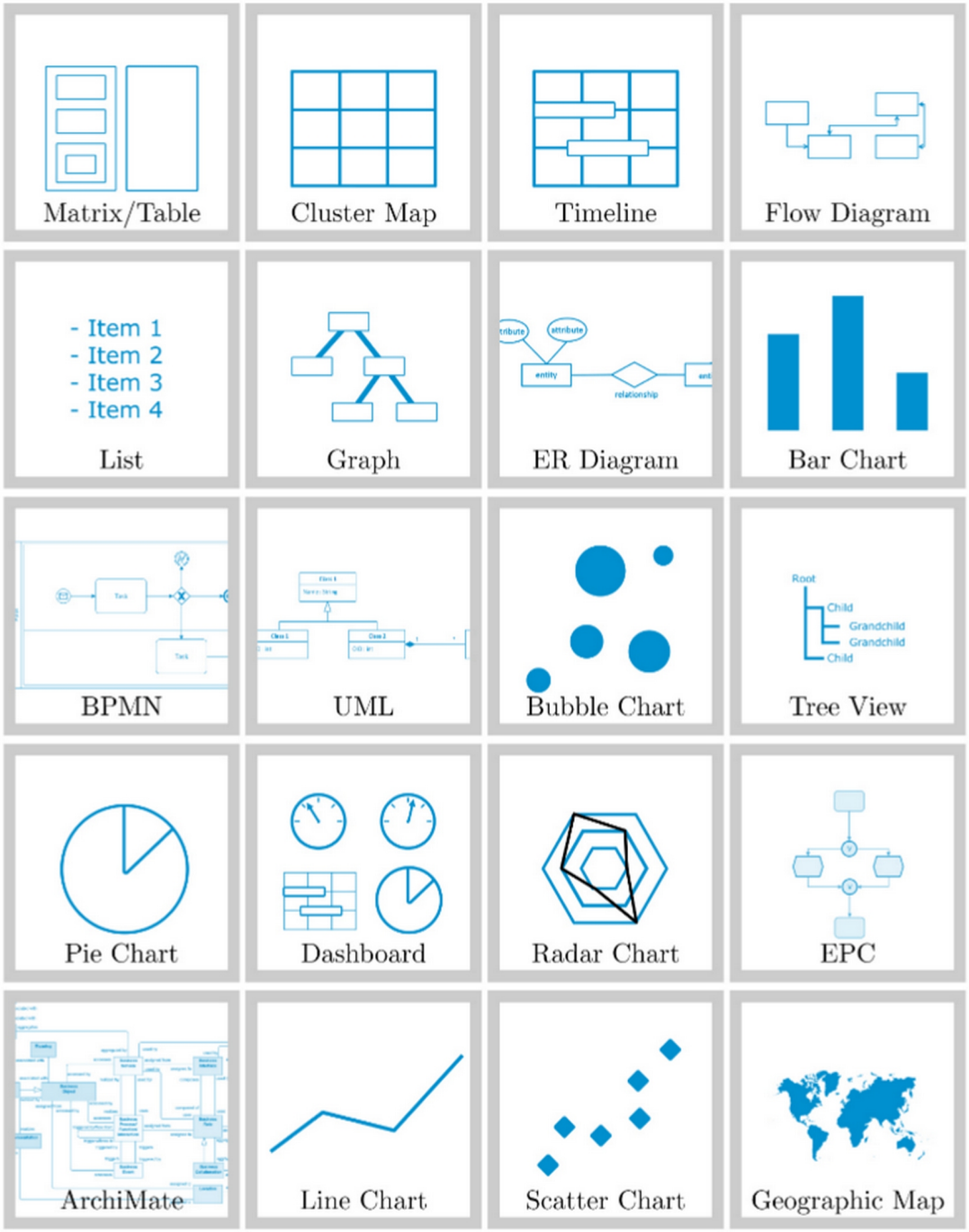
Münchenin yliopiston informaatioteknologisen tiedekuntaan kuuluva Sebis-tutkimusryhmä on toteuttanut muutamia laajamittaisia kokonaisarkkitehtuurissa käytettävien visualisointityökalujen ominaisuuksiin ja käyttöön liittyviä tutkimuksia, joista viimeisin julkaistiin vuonna 2014. Tutkimuksessa muodostettiin profiilit 18:sta eri kokonaisarkkitehtuurityökalusta, joihin sisältyi esimerkiksi valittujen työkalujen visualisointikyvykkyyksien kartoitus. Tämän lisäksi tutkimuksessa haastateltiin kokonaisarkkitehtuurin parissa työskenteleviä henkilöitä siitä, miten he käyttivät eri visualisointityökaluja ja millaisia vaatimuksia heillä oli niihin liittyen. Tutkimukseen osallistui yhteensä 109 henkilöä. Saksalaiset ja yhdysvaltalaiset muodostivat vastaajien suurimman osan, mutta tämän lisäksi tutkimukseen osallistui henkilöitä myös Etelä-Afrikasta, Australiasta, Venäjältä, Kanadasta, Intiasta ja monista muista maista. (Roth ym. 2014, 4-8, 349.)

Vastausten perusteella kokonaisarkkitehtuurin ammattilaisten eniten käytetyt visualisointityökalut olivat Microsoft:in Visio, PowerPoint ja Excel, joiden lisäksi vastaajat käyttivät myös useita muita tutkimuksen ensimmäisessä osassa läpikäytyjä kokonaisarkkitehtuurin hallintatyökaluja. Suurin osa vastaajista (49%) suhtautui käyttämiinsä työkaluihin neutraalisti. Vain 11% vastaajista suunnitteli siirtyvänsä käyttämänsä työkalun uuteen versioon, kun taas kokonaiset 86% vastaajista aikoi tulevaisuudessa korvata käyttämänsä työkalun jollain muulla, joko saman tai eri valmistajan, työkalulla. (Roth ym. 2014, 353.)

Kyselyyn vastanneilla oli selkeästi siis sellaisia tarpeita, joihin heidän käyttämänsä työkalut eivät vastanneet. Tutkimuksen mukaan käyttäjät toivoivat esimerkiksi parempia kustomointimahdollisuuksia ja visualisointien interaktiivisuutta. Toisaalta vastaajilla oli tarve myös yksinkertaisemmille ja intuitiivisemmille visualisoinneille. Tämän lisäksi he toivoivat raportteja, joiden käsittely ei edellyttäisi ulkoisten työvälineiden käyttöä. Visualisointien toivottiin mahdollistavan myös eri näkökulmat ja versionhallinta. (Roth ym. 2014, 366-367.)

Tutkimuksessa havaittiin, että eri sidosryhmien visualisointitarpeet poikkesivat toisistaan. Suurimmat erot liittyivät visualisoitavan tiedon päivitysintervalleihin. Siinä missä alemman johdon visualisointeja päivitettiin tyypillisesti joko neljännesvuosittain tai kuukausittain, esimerkiksi liiketoiminta-analyytikot, ratkaisuarkkitehdit ja keskijohto vaativat visualisointien huomattavasti säännöllisempää, esimerkiksi viikoittaista, päivitystä. Monet ratkaisuarkkitehdit halusivat tiedon päivittyvän jopa päivittäin. (Roth ym. 2014, 383.)

Myös käytetyimmät visualisointitavat vaihtelivat sidosryhmittäin. Kuplakaavio oli yleisin kaaviotyyppi johdolle kommunikoitaessa, kun taas UML-kaaviot olivat tärkein kuvaamisen muoto yritys- ja ratkaisuarkkitehdeille. Pylväs- ja piirakkakaaviot puolestaan olivat suosituimpia keskijohdon, yritysarkkitehtien ja IT-johtajien parissa. Kuvassa 4 on esitetty saatujen vastausten perusteella 20 yleisintä kokonaisarkkitehtuurityössä käytettyä visualisointitapaa. Tutkimuksen mukaan kolme suosituinta kaaviotyyppiä olivat siis matriisi, klusterikartta ja aikajana. Kokonaisuudessaan osallistujien vastauksessa esiintyi 26 erilaista visualisointitapaa. (Roth ym. 2014, 46, 371.)



Kuva 4. 20 yleisintä kokonaisarkkitehtuurissa käytettyä datan visualisointitapaa. (Roth ym. 2014, 46)

Sebis:in tutkimus oli laajuudestaan johtuen epäilemättä hyvin työläs toteuttaa, mikä voi selittää, miksi uutta vastaavaa tutkimusta ei ole tehty kokonaisarkkitehtuurityökalujen kehityksestä huolimatta. Viimeisen kuuden vuoden aikana tutkimuksessa käsitellyt työkalut ovat epäilemättä kehittyneet merkittävästi, joten myös käyttäjien tarpeet ovat todennäköisesti muuttuneet tutkimuksen jälkeen. Onkin siis vaikeaa arvioida, missä määrin Sebis:in tutkimustuloksia voidaan yleistää kokonaisarkkitehtuurin nykyisiin visualisointitarpeisiin ja -käytäntöihin.

6 Suunnitelmakuvaus

Ennen opinnäytetyöprojektin alkua projektille muodostettiin ohjausryhmä, johon kuului ARCin tuoteomistaja, Arterin sovellusarkkitehti ja tuotekehityspäällikkö. Kahden projektin ensimmäisen viikon aikana järjestin kolme suunnittelupalaveria, joista ensimmäiseen osallistui lisäkseni koko ohjausryhmä, ja jälkimmäisiin ainoastaan ARCin tuoteomistaja ja sovellusarkkitehti. Palaverien aikana kävimme läpi, minkä tyyppisiä käyttötarpeita suunnittelemaamme visualisointityökaluun kohdistui, ja muodostimme niiden pohjalta listan toteutettavista ominaisuuksista. Taulukko 1 esittää suunnitteluvaiheen aikana muodostetut vaatimukset eri prioriteettikategorioihin jaettuna. Osa listatuista vaatimuksista tarkentui edelleen projektin aikana.

Taulukko 1. Suunnitteluvaiheen aikana muodostetut vaatimukset

Prioriteetti	Vaatimus	Tarkennus
A	Kaavion tulee muuttaa muotoaan dynaamisesti hajonta- ja matriisikaavion välillä visualisoitavan datan luonteesta riippuen.	Kaavion tulee voida käyttää akselimuuttujinaan suhde- ja välimatka-asteikollista dataa, sekä järjestysasteikollista dataa. ARCin näkökulmasta tämä edellyttää, että akseleiksi voidaan valita sekä numeraalisia kenttiä että pudotuslistakenttiä.
A	Käyttäjän tulee voida määritellä 0-2 raja-arvoa yhdelle numeraalista muuttujaa käyttävälle akselille.	Raja-arvolla tarkoitetaan kaavion halki menevää viivaa, joka liittyy tiettyyn akselin arvoon. Raja-arvojen avulla käyttäjä voi korostaa tiettyjä kaavion alueita.
A	Pudotuslistoja käyttäville akseleille kaavion tulee piirtää raja-arvot automaattisesti siten, että ne asettuvat pudotuslistan kategorioiden väliin.	Käytettävän pudotuslistan pituudelle ei aseteta tiukkoja rajoitteita, mutta mikäli pudotuslistan arvoja on yli yhdeksän, raja-arvojen ei tarvitse piirtyä kaavioon.
A	Käyttäjän tulee voida määritellä kaaviolle taustaväri, tai raja-arvoja käytettäessä eri taustavärejä raja-arvojen määrittelemille alueille. Tämä mahdollistaa kaavion eri alueiden semanttisen korostamisen (liite 1).	Käyttäjän tulisi voida valita kaavion käyttämät värit vapaasti esimerkiksi värivalitsimen avulla.

B	Käyttäjän tulee voida lähentää kaaviota.	Lähentämismahdollisuus vähentää elementtien mahdollisesta suuresta määrästä aiheutuvaa haittaa.
B	Käyttäjän tulee voida määritellä kaavioon taustamuuttuja, jota voidaan käyttää datan suodattamiseen.	Käyttäjä voi halutessaan valita taustamuuttujaksi jonkin mallin pudotuslistakentistä. Myös datan suodatus helpottaa suurten elementtimäärien tarkastelua.
B	Käyttäjän tulee voida tarkastella yksittäisiin elementteihin liittyvää tietoa.	Kun käyttäjä vie kursorin elementin datapisteen päälle, aukeaa infolaatikko, joka sisältää elementin nimen ja sen saamat x- ja y-akselien muuttujien arvot.
B	Kaavion datapisteiden päällekkäisyys tulee estää.	Mikäli datapisteet menisivät päällekkäin, käyttäjä voisi tarkastella vain päällimmäiseen datapisteeseen liittyvän elementin tietoja. Tämä olisi ongelma varsinkin pudotuslistoja käytettäessä, sillä useat elementit saavat samoja pudotuslistan arvoja.
C	Kaavion elementtien datapisteet voisivat olla ympyröiden sijasta joko elementtien omia ikoneita, tai suodatusta käytettäessä elementtien edustamien kategorioiden ikoneita.	ARCin yhteyskaavioissa käytetään myös elementtien ja mallien ikoneita, joten ikonien käyttö tekisi kaavion ilmeestä yhdenmukaisemman muiden kaaviotyyppien kanssa.

Kaavioon kohdistuneet vaatimukset liittyivät siis sen yleiseen toiminnallisuuteen ja muihin ominaisuuksiin. Tarkkoja tekniseen toteutukseen liittyviä vaatimuksia emme kuitenkaan muodostaneet, vaan minua rohkaistiin käyttämään omaa harkintaani sopivien teknisten ratkaisujen etsimisessä. Tämän lisäksi pääsin myös suunnittelemaan ja priorisoimaan työtehtäväni itsenäisesti. Muodostinkin annettujen vaatimusten pohjalta listan kaavion konkreettisista ominaisuuksista, jotka asetin toteutusjärjestykseen taulukon 2 osoittamalla tavalla.

Taulukko 2. Kaavion ominaisuuksien suunniteltu toteuttamisjärjestys

1.	Numeraalisia kenttiä käyttävät akselit
2.	Raja-arvot
3.	Datapisteympyrät
4.	Taustavärit

5.	Datapisteiden infolaatikot
6.	Kaavion lähentäminen
7.	Datapisteiden suodatus
8.	Datapisteiden päällekkäisyyden estäminen
9.	Pudotuslistakenttiä käyttävät akselit
11.	Datapisteikonit

Pyrin järjestämään tehtävät siten, että aloittaisin mahdollisimman yksinkertaisista ominaisuuksista ja etenisin vähitellen toteutukseltaan haastavampiin ominaisuuksiin, vaikkakin datapisteympyröiden vaihtamisen elementtien ikoneiksi jätin viimeiseksi sen matalan prioriteetin takia. Kaavion ominaisuuksiin liittyvien tehtävien lisäksi tulisin projektin loppuvaiheessa tekemään myös uuden kaaviotyypin edellyttämät backend-muutokset.

6.1 Uuden kaavioiden lisäysnäkymän suunnittelu

Suunnitteluvaiheen aikana kävi ilmi myös, että ARCin kaavioiden luontinäkymää tulisi muuttaa merkittävästi, jotta sitä voitaisiin käyttää myös uuden kaaviotyypin lisäämiseen. Tähän liittyen tein useita mockup-vedoksia (liitteet 2-6) suunnitteluprosessimme tueksi. Minulla ei ollut aikaisempaa kokemusta UX-suunnittelusta, joten yllätyin, kuinka monia erilaisia asioita minun tuli ottaa huomioon hahmotelmia tehdessäni. Vaikeaa oli etenkin näkymän visuaalisen selkeyden säilyttäminen kaaviotyyppien erilaisista tietotarpeista huolimatta. Viimeisin mockup-hahmotelmani (liite 5) kuitenkin ratkaisi suurimman osan muiden vedosten ongelmista ja sai hyväksynnän projektin ohjausryhmän lisäksi myös Arterin UX-suunnittelijalta. Vaikka suunniteltujen muutosten toteuttaminen päädyttiinkin lopulta rajaamaan pois projektin piiristä, näkymän suunnitteluvaihe oli joka tapauksessa hyödyllinen, sillä sen aikana käsitykseni kaavion tarvitseman tiedon muodosta selkeytyi huomattavasti.

6.2 Visualisointikirjaston valinta

Muiden suunnittelun vaiheiden jälkeen minun tuli myös arvioida eri visualisointikirjastojen soveltuvuutta suunniteltujen ominaisuuksien toteuttamiseen. Visualisointikirjastoja vertaillessani kiinnitin erityistä huomiota niiden suosittuuteen, monipuolisuuteen ja maturiteettiin. Näillä kriteereillä tarkasteltuna edukseen erottuivat kaksi kirjastoa: Chart.js ja D3.js, joista päädyin valitsemaan D3.js:n. Vaikka näistä vaihtoehdoista Chart.js olisikin ollut aloittelijaystävällisempi valinta ja siten kenties parempi vaihtoehto oman taitotasoni huomioon ottaen, D3:n toimintaperiaatteet tarjosivat mielestäni paremmat edellytykset etenkin kaavion erikoisempien ominaisuuksien, esimerkiksi taustavärien, toteuttamiseen. Tämän lisäksi D3 oli parempi valinta myös kirjastoriippuvuuksien minimoimisen kannalta, sillä suurin osa ARCin aikaisemmista visualisoinneista oli myös toteutettu D3:n avulla.

6.3 Projektin riskit

Suurin projektin riskeistä liittyi ehdottomasti siihen, että kaikki teknologiat, joita tulisin projektin aikana käyttämään, olivat minulle joko kokonaan tai osittain vieraita. Projektin ohjausryhmä ei myöskään osannut arvioida, kuinka haastavaa suunniteltujen ominaisuuksien toteuttaminen tulisi olemaan. Oli siis odotettavissa, että projekti tulisi sisältämään paljon selvitystyötä.

Epävarmuustekijöitä liittyi myös D3:een. Visualisointikirjastoihin perehtyessäni olin lukenut, että D3.js saattaa monipuolisuutensa takia olla hyvin hämmentävä etenkin uusille käyttäjille ja siten sen oppimisprosessi on yleensä varsin hidas (Meeks 11.6.2018). Lisäksi suurin osa edistyneemmistäkin D3-käyttäjistä päätyy usein lähinnä kopioimaan valmiita esimerkkejä ja muokkaamaan niitä omiin tarpeisiinsa sopiviksi (Sweeney 16.12.2018). Näiden tekijöiden takia työskentelyni tulisi siis todennäköisesti olemaan hyvin haastavaa etenkin projektin alussa. Myöskään kollegani eivät välttämättä voisi tarjota minulle mittavaa tukea, sillä D3:n parissa eniten työskennelleet henkilöt olivat siirtyneet pois Arterilta jo aikaisemmin. Tämän lisäksi oli myös todennäköistä, että ARCin käyttämä D3:n versio tulisi aiheuttamaan ongelmia. Projektin alkaessa vuoden 2020 syyskuussa uusin D3:n versio oli 6.2.0, kun taas ARC käytti vuoden 2016 toukokuussa julkaistua versiota 3.5.17. Monet D3:n toiminnallisuuksista olivat muuttuneet versioiden välillä, joten uusien toimintatapojen soveltaminen vanhaan versioon voisi osoittautua hankalaksi.

Muut projektin riskeistä liittyivät ARCin tiedon määrään ja monimuotoisuuteen. ARCin mallien ja siten myös niihin pohjautuvien elementtien sisältö riippuu täysin asiakkaasta, mikä tekee kaavion mahdollisten käyttötarkoitusten ennakoinnista vaikeaa. Saman mallin elementtien määrää ei myöskään ole rajoitettu, joten kaavion rajallinen tila ei olisi välttämättä riittävä kaikissa tapauksissa.

6.4 Kehitysprosessin etenemisen suunnittelu

Sovimme projektin ohjausryhmän kanssa, että tulisin esittelemään heille projektini etenemistä ja uusia kaavion ominaisuuksia kerran kuukaudessa. Näissä palavereissa ohjausryhmän jäsenet voisivat antaa minulle palautetta ja kehitysehdotuksia sekä ottaa esille mahdollisia ongelmakohtia. Lisäksi voisin halutessani järjestää myös ylimääräisiä palavereja, mikäli tarvetta ilmenisi.

Puuttuvasta kokemuksestani ja kaavion ominaisuuksien keskinäisistä riippuvuussuhteista johtuen projektin eri osa-alueiden aikatauluttaminen oli äärimmäisen hankalaa, joten emme sopineet projektin kuukausille erityisiä tavoitteita, vaan tulisin edistämään projektia parhaani mukaan oman osaamiseni puitteissa. Kaikki kaavion ominaisuudet tultaisiin joka tapauksessa toteuttamaan, joten tarkkarajaisten tehtävien muodostaminen ja seuranta ei olisi tuonut projektille juurikaan lisäarvoa. Projektin jälkeisessä kaavion jatkokehityksessä puolestaan tultaisiin soveltamaan Arterilla tyypillisesti käytettävää Kanban-menetelmää.

7 Toteutuskuvaus

Suunnitteluvaiheen aikana havaitsemani D3:een liittyvät riskit realisoituivat heti toteutusvaiheen ensimmäisellä viikolla. Päädyin siis ehdottamaan projektin ohjausryhmälle, että aikaisemmasta suunnitelmastamme poiketen sisällyttäisin projektiini myös ARCin D3-version päivittämisen taulukossa 3 listatuista syistä johtuen. Ehdotukseni sai kannatusta, joten tämän kappaleen alussa käyn läpi D3:n päivitysprosessin, jonka aikana perehdyin Require.js:n toimintaan ja siihen, miten ARC hyödyntää käyttämiään moduuleita.

Taulukko 3. D3:n päivitykseen johtaneet tekijät.

1.	Löytämäni tieto liittyi lähes poikkeuksetta D3:n uudempiin versioihin, jotka poikkesivat toiminnaltaan käyttämästäni versiosta.
2.	Mikäli jokin tekemistäni ominaisuuksista ei toiminut, en tiennyt, olinko tehnyt jotakin väärin, vai oliko jokin käyttämistäni toiminnallisuuksista muuttunut eri versioiden välillä.
3.	Monet kohtaamistani ongelmista tai puutteista oli korjattu D3:n uusissa versioissa.
4.	D3:n päivittämistä oli suunniteltu Arterilla jo aikaisemminkin, ja koin projektini olevan looginen konteksti sen toteuttamiseen.
5.	Halusin välttää ARCin teknisen velan lisäämistä.
6.	Uusi D3-versio tarjoaisi paremman pohjan tulevien visualisointien kehittämiselle ja vanhojen visualisointien ylläpidolle.

Päivityksen vaiheiden jälkeen siirryn kuvaamaan toteuttamaani kaaviota. Aloitan backend:ista käymällä lyhyesti läpi Chart.java-luokkaan ja kaavioiden kokoamisprosessiin tekemäni muutokset. Kuvaan myös, millä tavalla käsittelin frontend:issä kaavion Backbone-mallia ja sen sisältämää tietoa. Tämän jälkeen toteutuksen kokonaisuuden havainnollistamiseksi esittelen varsinaisen kaavion piirtämisen ja sen toiminnan ohjelmallisen rakenteen. Kaavio on ominaisuuksiltaan monimuotoinen, joten käyn läpi syvällisemmin ainoastaan ne kaavion toiminnallisuudet, jotka olivat mielestäni toteutukseltaan kaikkein haastavimpia ja kiinnostavimpia.

Toteutuskuvauksen osuudet sisältävät useita havainnollistavia koodiesimerkkejä, joista monet olivat alkuperäisessä muodossaan hyvin pitkiä. Olenkin paikoitellen korvannut esimerkkien vähemmän olennaisia rivejä kolmella peräkkäisellä pisteellä. Tuotetun kaavion visuaalisesta luonteesta johtuen siihen liittyviin kuviin (kuva 23, liitteet 16-21) perehtyminen voi tarjota paremmat lähtökohdat kaavion toteuttamista käsittelevien kappaleiden asiasisällön ymmärtämiseen.

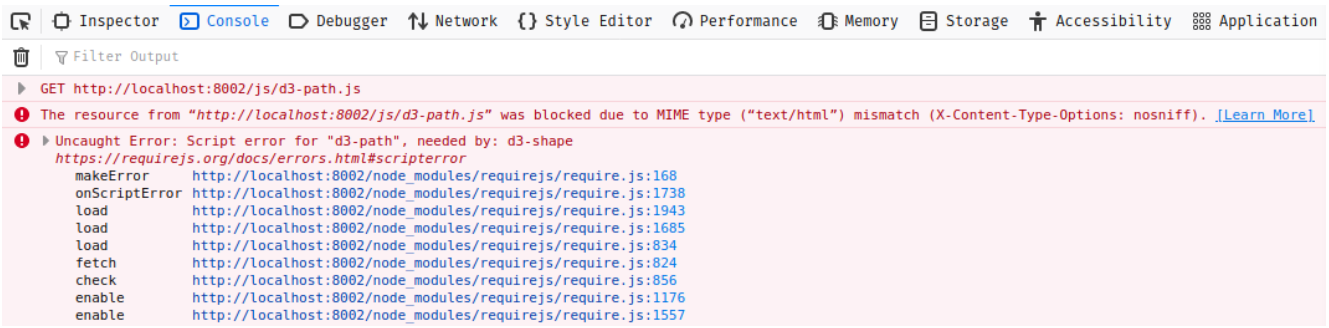
7.1 D3:n päivittäminen

ARCin frontend:in moduulien päivitykseen käytetään Yarn-pakentinhallintatyökalua. Aloitin siis D3:n päivittämisen vaihtamalla ARCin package.json-tiedostoon haluamani D3-version numeron, minkä jälkeen ajoin Yarn:in install-komennon. Alun paketinhallintakonfliktien jälkeen uudet D3-moduulit ilmestyivät node_modules-kansioon. Tämä oli kuitenkin päivityksen helpoin vaihe D3:n muuttuneesta rakenteesta johtuen.

ARCin frontend:in moduulien hallintaan käytetään AMD-rajapintaa (lyhenne sanoista Asyncronous Module Definition) toteuttavaa Require.js-kirjastoa (Require.js 2020). Tämä tarkoittaa, että ARCin käyttämät moduulit ladataan asynkronisesti vasta tarpeen vaatiessa, mikä tehostaa ohjelmiston toimintaa etenkin, kun käytettäviä moduuleita on paljon (AMD 2020). Sen lisäksi Require.js helpottaa myös suuren moduulimäärän hallintaa mahdollistamalla moduulien tiedostopolkujen ja muiden määritysten keskitetyn hallinnan, joka tapahtuu tyypillisesti main.js-tiedostossa. Require.js tarjoaa myös shim-toiminnallisuuden, jonka avulla voidaan määritellä riippuvuudet myös vanhoille skripteille, jotka eivät tue AMD-moduulimäärittelyitä (Buiza 2.5.2016).

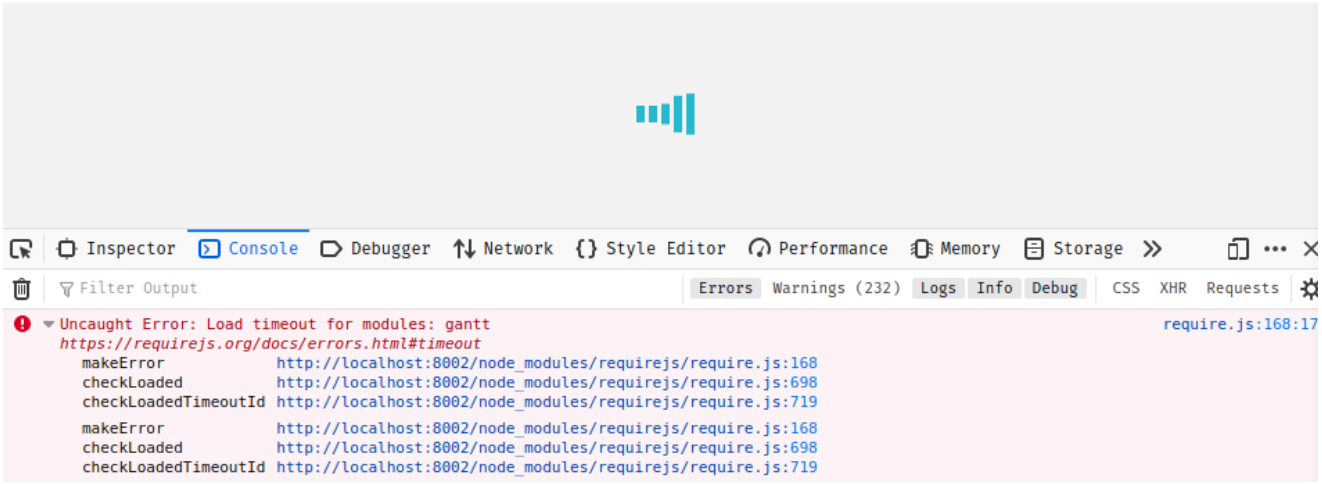
Vanha ARCin D3-versio koostui vain yhdestä moduulista, joka piti sisällään kaikki D3:n funktiot. Tästä moduulista oli Require.js:n avulla luotu globaali objekti, joka vietiin käytettäväksi sitä tarvitseviin tiedostoihin. D3:n rakennemuutoksen jälkeen siitä ei ole kuitenkaan enää voinut tehdä globaalia objektia AMD:tä käytettäessä, sillä tämä poistaisi AMD-rajapinnan tarjoamat hyödyt (Bostock 10.1.2014). Vaikka D3:sta onkin edelleen saatavilla kokonainen versio, jonka olisin saanut käyttööni muilla tavoilla, tämä valtava moduuli olisi joka tapauksessa ollut ARCin kaavioiden tarpeisiin liian raskas. AMD:n järkevä hyödyntäminen edellytti siis jokaisen tarvittavan moduulin tiedostopolun ja nimen eksplisiittistä määrittelyä uuteen D3-versioon siirryttäessä.

Toisin kuin oletin, kaikki D3:n moduulit eivät käytännössä olleetkaan täysin itsenäisiä, vaan osalla niistä oli keskinäisiä riippuvuuksia. Moduulit pyrkivät automaattisesti löytämään toisensa tietyillä ennalta määritellyillä nimillä, mikä ei käynyt mielestäni selkeästi ilmi moduulien sisällöstä tai saamistani virheilmoituksista. Vaikka alussa käyttämäni camelCase-muotoiset nimet eivät olleetkaan Require.js:n näkökulmasta virheellisiä, D3:n moduulit eivät löytäneet toisiaan ennen kuin lopulta muutin ne väliviivalliseen muotoon.



Kuva 5. Muodoltaan väärien Require.js-määrittelyiden aiheuttamat virheet.

Tarvittavien moduulien onnistunut määrittely ei kuitenkaan itsessään riittänyt. Kaikki ARCin aikaisemmista visualisoinneista etsivät edelleen globaalia D3-objektia. Tämän lisäksi ARCin vanhan Projektit-osion käyttämään Gantt-kaaviotoiminnallisuuteen liittyi shim-määrittelyitä. Require.js ei kuitenkaan saanut enää ladattua tarvittavia moduuleja, mikä esti ARCin frontendin toiminnan kokonaan.



Kuva 6. Gantt-moduulin rikkoutuneista riippuvuuksista aiheutunut virhe.

Pyrin kiertämään näitä ongelmia välittämällä D3:n yhdistelmämoduulin ja muita tarvittavia moduuleita niitä tarvitsevien tiedostojen käytettäväksi, mutta tämäkään ei juuri auttanut, sillä vanhat kaaviotyypit käyttivät useita deprekoituja ja toiminnaltaan muuttuneita funktioita. Tilanteen korjaaminen olisi edellyttänyt huomattavaa refaktorointia, joka olisi entisestään viivästyttänyt uuden kaaviotyypin toteutusta. Rajasin siis vanhojen kaavioiden korjaamisen pois opinnäytetyöprojektini piiristä ja kommentoin kaikki ARCin käynnistymisen estävät koodin osat pois käytöstä.

7.2 Backend-muutokset

ARCin Chart.java-luokka määrittelee ympyrä- ja palkkikaavioiden tarvitseman tiedon rakenteen. Muihin kaavioihin verrattuna uusi hajonta-/matriisikaavio tulisi kuitenkin tarvitsemaan tietoa hyvin erilaisessa muodossa, joten aluksi suunnittelin tekeväni kokonaan uuden ScatterChart.java-luokan.

Lopulta kuitenkin luovuin ideasta, sillä halusin, että myös uusi kaaviotyyppi pääsisi hyödyntämään kaavioiden toimintaan liittyviä service-funktioita mahdollisimman rajallisilla muutoksilla.

7.2.1 Chart.java-luokan laajentaminen

Chart-oliolla oli jo ennestään chartType-attribuutti, jolle lisäsin arvon uutta kaaviotyyppiä edustamaan. Tämän jälkeen lisäsin tiedostoon uusia attribuutteja kaaviotani varten. Muiden kaavioiden tapaan kaavion käyttämistä kentistä tallennettasiin tietokantaan ainoastaan id:t. Kaavioiden kentät, jotka kuvassa 7 on merkitty JPA:n transient-annotaatiolla, kasattaisiin ja tallennettaisiin välimuistiin vasta ajon aikana.

```
private String xAxisFieldId;
@Transient
private Field xAxisField;
private String yAxisFieldId;
@Transient
private Field yAxisField;
private String filteringDropdownFieldId;
@Transient
private Field filteringDropdownField;
private List<Double> xAxisLimitValues;
private List<Double> yAxisLimitValues;
private List<ScatterBackgroundRect> backgroundRects;
```

Kuva 7. Uuden kaaviotyypin tarvitsemat tiedot.

Suurin osa lisäämistäni attribuuteista olivat sisällöltään melko yksiselitteisiä taustavärien toteutusta lukuunottamatta. Suunnitteluvaiheen aikana olin ajatellut taustaa yhtenä elementtinä, jonka väri vaihtelisi sen alueesta riippuen. Tämän lähestymistavan tekninen toteutus osoittautui kuitenkin haastavaksi. Päädyin siis ratkaisuun, jossa jokaista taustavärialuetta edustaisi erillinen elementti, joka sijoitettaisiin oikeaan kaavion kohtaan raja-arvojen ja akseleiden minimi- ja maksimiarvojen avulla.

Kaavioon liittyvät laskutoimitukset tehtäisiin kuitenkin vasta frontend:issä, joten backend:issä akseleiden ääriarvot eivät vielä olleet tiedossa. Tämä puolestaan tarkoitti, että myöskään kaavion ulkoreunoille asettuvien suorakuutioiden mittoja ei voitu tietää. Minun piti siis kehittää vaihtoehtoinen tapa, jolla voisin varmistaa, että kuhunkin alueeseen yhdistyisi varmasti juuri sille määritelty väri.

Päädyin siis lisäämään ScatterBackgroundRect-luokalle uuden attribuutin, joka saisi arvokseen kahden alkion mittaisen Array-listan. Alkiot edustaisivat tietynlaisia järjestyskoordinaatteja, joista ensimmäinen kuvaisi, kuinka monennessa kolumnissa kyseinen suorakuutio on, ja toinen, mille riville suorakuutio sijoittuu. Esimerkiksi paikka [0,0] tarkoittaisi, että suorakuutio sijoittuisi aivan kaavion

vasempaan alareunaan. Paikassa [1,0] sijaitseva suorakuutio sijaitsisi myös y-akselin alhaisimman raja-arvon ja x-akselin välissä, mutta ensimmäisen suorakuution oikealla puolella, toisessa kolumnissa.

Taulukko 4 havainnollistaa koordinaattiarvojen muodostumista käyttäen esimerkkinään kuvitteellista taulukkoa, jonka x- ja y-akseleille on määritelty yhteensä neljä raja-arvoa. Taulukon harmaapohjaiset solut edustavat kaavion akseleita, kun taas valkoiset solut muodostavat todellisen kaavioalueen.

Taulukko 4. Taustavärisuorakuutioiden saamat koordinaatit.

2	0, 2	1, 2	2, 2
1	0, 1	1, 1	2, 1
0	0, 0	1, 0	2, 0
	0	1	2

7.2.2 Kaavioiden kokoamisen muutokset

ARCissa kaavioita voidaan näyttää kahdessa eri näkymässä eli yksittäisen mallin sivulla tai etusivun paneeleissa. Nämä näkymät tarvitsevat tiedon kaavion käyttämien kenttien sisällöstä. Mikäli taas kaavio avataan esimerkiksi muokkausnäkymään, sen kenttien sisältöä ei tarvita. Tästä johtuen ARCin toimintaa on pyritty tehostamaan sisällyttämällä kaavioihin vain se tieto, joka valittua esitystapaa varten tarvitaan. Kaavion kenttien tieto kasataan siis vasta siinä vaiheessa, kun kaavio avataan katseltavaksi. Tämän prosessin aikana kaaviot tallennetaan myös välimuistiin sekä kasatussa että kasaamattomassa muodossa. Lisäksi kaavioiden kasaamisvaiheessa tutkitaan mm. myös, onko kaavion tarvitsemia kenttiä poistettu.

Kaavioiden käsittelyprosessi tapahtuu kokonaisuudessaan usean eri funktion toimesta, joista monet olivat hyvin pitkiä ja monimutkaisia. Helpotinkin siis omaa työtäni refaktoroimalla muutaman erityisen pitkän funktion pienemmiksi loogisiksi kokonaisuuksiksi. Tekemäni muutokset tekivät toteutuksesta mielestäni huomattavasti aikaisempaa selkeämmän.

Refaktoroinnin aikana tulin kiinnittäneeksi huomiota siihen, että poistettuja kenttiä käyttävät kaaviot karsittiin tuloksista pois silloinkin, kun niitä oltiin välittämässä muokkausnäkymään. Tällaisia kaavioita

ei siten olisi ollenkaan mahdollista poistaa ARCin kautta, joten ne jäisivät turhaan pyörimään asiakkaiden tietokantoihin. Vaikka tämä ei ollutkaan erityisen vakava haitta, tuoteomistajan suostumuksella muutin toimintalogiikkaa kuitenkin hieman kuvan X osoittamalla tavalla.

```
charts = charts.stream().filter(chart→ {
if (isEditView) {
  return true;
}
if (chartHasMissingContent(chart) ||
                  chartHasDeprecatedContent(chart)) {
  return false;
}
return true;

}).collect(Collectors.toList());
```

Kuva 8. Refaktoroitu kaavioiden suodatusvaihe.

ARCin pylväs- ja ympyräkaavioiden ryhmittelevästä luonteesta johtuen ne eivät tarvitse tietoa niistä pudotuslistan arvoista, joita yksikään niiden kuvaamista elementeistä ei edusta. Uuden kaavion matriisimuotoinen esitystapa puolestaan tarvitsee ehdottomasti kaikki pudotuslistojen arvot riippumatta siitä, liittyikö niihin elementtejä, mistä johtuen päädyin käyttämään vaihtoehtoista pudotuslista-arvojen kokoamistapaa (kuva 9).

```
} else {
  List<String> contentIds = new ArrayList<>();
  for (Field field : fields) {
    contentIds.add(field.getContentId());
  }
  List<DropdownItem> allDropdownItems = dropdownItemService
          .findAllNonDeprecatedByDropdownIds(contentIds, tenantId);

for (DropdownItem dropdownItem : allDropdownItems) {
    for (Field field : fields) {
      if (field.getContentType() == Field.CONTENT_TYPE_DROPDOWN &&
         field.getContentId()
         .equals(dropdownItem.getParentId())) {

        field.getDropdownItemContents()
         .add(listAssembler.assembleListItem(dropdownItem));
      }
```

Kuva 9. Matriisikaavion akseleiden pudotuslistakenttien sisällön kasaaminen.

7.3 Kaavion käyttämät moduulit

Edellisessä kappaleessa kuvaamieni backend-muutosten jälkeen frontend:in chart.js-tiedosto sai käyttöönsä uuden kaavion piirtämiseen tarvitut tiedot. Chart.js-tiedosto oli kuitenkin itsessään jo hyvin pitkä, joten päätin luoda omalle kaaviotyypilleni erillisen tiedoston. Minun tuli siis välittää chart.js-tiedoston Backbone-malli uudelle tiedostolle (liite 8) ja määritellä sen käyttämät moduulit Require.js:n define-funktiolla (kuva 10).

```
define([
   'jquery',
   'backbone',
   'common',
   'd3-array',
   'd3-axis',
   'd3-brush',
   'd3-force',
   'd3-format',
   'd3-scale',
   'd3-selection',
   '../../../styles/icomoonToMDIconCodes',
   'i18next'
], function ($, Backbone, Common, d3Array, d3Axis, d3Brush, d3Force,
d3Format, d3Scale, d3Selection, icomoonToMDIconCodes, i18n) {
```

Kuva 10. scatterChart.js-tiedoston käyttämät moduulit.

7.4 Backbone-mallin sisältämän datan käsittely

Uuden scatterChart.js:n vastaanottama Backbone-malli sisälsi kaavion piirtämisen kannalta myös paljon ylimääräistä tietoa. Kopioin siis tarvitsemieni attribuuttien arvot scatterChart-objektin kontekstiin initialize-funktiossa. Tietoa oli paljon, joten siirsin suurimman osan sen keräämiseen ja käsittelyyn liittyvistä toiminnoista erillisiksi funktioiksi.

```
      initialize: function (options) {
         this.parent = options.parent;
         this.target = options.target;
         this.renderLocation = options.renderLocation;
         this.initChartBaseProperties();
         this.initChartDimensions()
         this.render()
      },
```

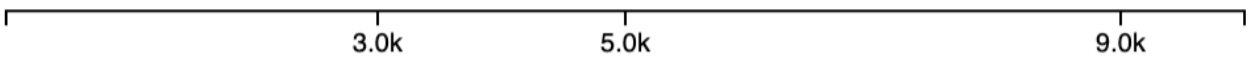
Kuva 11. Kaavion tarvitseman tiedon kasaamisesta ja määrittelystä vastaava funktio.

Tekemäni valmisteluvaiheen funktiot siis keräsivät kaavion perusominaisuuksiin, kuten käytettyihin kenttiin sekä raja-arvoihin ja taustaväreihin, liittyvän tiedon ja määrittelivät kaavion mittasuhteet. Render-funktiota (kuva 12) tehdessäni pääsin käyttämään Underscore.js-kirjastoa ensimmäistä kertaa käydessäni läpi suodatuskategorioita (liite 9) ja elementtejä ja muuttaessani niihin liittyvää tietoa hyödyllisempään muotoon (liite 10). Elementtien määrittelyn jälkeen myös akseleiden ääriarvojen laskeminen oli vihdoinkin mahdollista, sillä ne pohjautuivat elementtien arvoihin kuvan 14 osoittamalla tavalla.

```
render: function () {
   this.handleFilterData()
   this.handleElementData()
   if (this.data.length === 0) {
      Common.makeNotification({
         type: 'info',
         text: 'general.noResults',
         target: this.target,
         automaticClose: false
      });
      return false;
   }
   this.calculateXAxisMinAndMax()
   this.calculateYAxisMinAndMax()
   if (this.backgroundRects) {
      this.addBackgroundRectCoordinates()
   }
   this.drawScatterChart()
  },
```

Kuva 12. Render-funktion sisältö.

Vaikka D3 tarjoaakin ordinaalisen skaalan, joka olisi ollut looginen valinta pudotuslistoja käyttäville akseleille, päädyin toteutuksen yksinkertaistamiseksi hyödyntämään pelkkiä numeerisia skaaloja akselien käyttämistä kenttätyypeistä riippumatta. Pudotuslistakenttiä käytettäessä akselien tick-viivat (kuva 13) piilotettaisiin ja tick-nimikkeet formatoitaisiin käyttämään numeroiden sijasta pudotuslistan kategorioiden nimiä. Kaikki elementit saisivat siis aina numeeriset xValue- ja yValue-attribuutit, mutta pudotuslistoja käytettäessä elementeille asetettaisiin lisäksi myös xDropdownValue- ja yDropdownValue-attribuutit elementtien edustamien pudotuslistakategorian nimen varastoimista varten.



Kuva 13. Numeerista skaalaa käyttävä akseli, jolla on viisi tick-viivaa ja kolme tick-nimikettä. (Henshaw 30.9.2019.)

Mikäli siis esimerkiksi x-akselille valittaisiin pudotuslista, jolla olisi kolme kenttää, elementit saisivat x-akselin arvoja 0, 1, ja 2. Näiden arvojen perusteella saatoin määritellä x-akselin minimi- ja maksimiarvot. Jotta jokainen pudotuslistaluokkaa edustava alue olisi yhtä leveä, määrittely tuli toteuttaa kuvan 14 osoittamalla tavalla.

```
calculateXAxisMinAndMax: function () {
    if (this.xAxisContentType ===
        parseInt(Common.TABLE_COLUMN_TYPES.NUMERIC.value)) {

        this.xMax = Math.ceil(d3Array.max(this.data,
                    function (d) { return d.xValue }) * 1.05)
        this.xMin = Math.floor(d3Array.min(this.data,
                    function (d) { return d.xValue }) * 1.05)

    } else if (this.xAxisContentType ===
        parseInt(Common.TABLE_COLUMN_TYPES.DROPDOWN.value)) {

        this.xMin = -0.5
        this.xMax = this.xDropdownItemContents.length - 1 + 0.5
    }
},
```

Kuva 14. X-akselin ääriarvojen muodostamisen logiikka.

Akseleiden minimi- ja maksimiarvojen laskemisen jälkeen saatoin myös muuttaa taustavärisuorakuutioiden väliaikaiset backend-koordinaatit todellisiksi akselikoordinaateiksi. D3:n suorakuutioelementtien mitat määritellään tyypillisesti SVG-elementin koordinaateilla, joita en kuitenkaan itse voinut käyttää, sillä suorakuutioiden tuli reagoida kaavion lähentämiseen. Annoin siis suorakuutioille uudet koordinaattiattribuutit x1, x2, y1 ja y2, jotka muodostin raja-arvojen ja akseleiden käyttämien skaalojen ääriarvojen avulla.

7.5 Kaavion ohjelmallinen rakenne

Suoritettuaan muut tehtävänsä render-funktio kutsuu drawScatterChart-funktiota, joka piirtää kaavion ja hallitsee siihen liittyviä DOM-elementtejä. Useat drawScatterCart:in funktioista kutsuvat edelleen

myös muita funktioita, ja kaaviota tehdessäni minulla oli toisinaan vaikeuksia muistaa, mihin tiedoston kohtaan nämä funktiot sijoittuivat. Päädyinkin kuvan 15 mukaisella tavalla lopulta lisäämään drawScatterChart:iin kommentteina myös niiden funktioiden nimet, joita ei suoraan kutsuttu drawScatterChart:in toimesta. Uudessa muodossaan funktio tarjoaa kattavan kokonaiskuvan kaavion piirtämisen vaiheista ja sen toiminnallisuuksista.

```
    drawScatterChart: function () {
        this.createContainersAndSvg()
        this.appendClipAreas()
        this.appendXAxis()
        this.appendYAxis()
        //this.formatXDropdownLabels
        //this.formatYDropdownLabels
        //this.positionYDropdownLabels
        //this.wrapLabelText
        //this.mouseover
        //this.mouseleave
        //this.mousemove
        //this.formatXAxisTickLabelTooltip
        //this.formatYAxisTickLabelTooltip
        this.appendAxisLabels()
        if (this.backgroundRects){
            this.appendBackgroundRects()
        }
        this.appendLimitValueLines()
        this.appendTooltip()
        this.appendBrush()
        //this.idled
        //this.brushEnded
        this.appendDatapoints()
        //this.formatDatapointTooltip
        this.createForceSimulation()
        //this.ticked
        //this.zoom
        this.appendControlsForDatapointOverlapping()
        if (this.filterCategories.length > 0) {
            this.appendFilters()
        }
        //this.toggleFilterData
        //this.appendControlsForDatapointOverlapping()
        //this.checked
      },
```

Kuva 15. Kaavion piirtämisen ja toiminnan funktiot.

Elementtien lisäämisjärjestys muodostui yllättävän tärkeäksi tekijäksi kaavion toiminnan kannalta. Esimerkiksi appendClipAreas-funktio nimensä mukaisesti lisää kaavioon ns. leikkausalueita, joihin voidaan yhdistää eri elementtejä clip-path -attribuutin avulla (esim. liite 11). Jotta elementeille voitaisiin määritellä clip-path, tulee leikkausalueet siis lisätä kaavioon ennen siihen yhdistettäviä elementtejä. Leikkausalueet varmistavat, että kaaviota lähennettäessä sen ulkopuolelle siirtyvät elementit tai niiden osat eivät jää näkyviin vaan leikkautuvat pois.

Myös värisuorakuutioiden, raja-arvoviivojen, lähennykseen liittyvän brush-elementin ja datapisteiden lisäämisessä vain yksi järjestys tuotti haluamani lopputuloksen. Eri elementtien lisäysjärjestyksen vaatimuksista johtuen uusien elementtien lisääminen kaavioon olikin usein yllättävän haastavaa, ja tämä ongelma paheni progressiivisesti projektin loppua kohden kaavion ominaisuuksien määrän kasvaessa.

En voi käydä kaavion toimintaa läpi erityisen yksityiskohtaisella tasolla sen laajuuden takia, joten seuraavaan kappaleeseen olen valinnut esiteltäväksi kolme mielestäni haastavinta ja kiinnostavinta asiakokonaisuutta: pudotuslista-akseleiden kategorianimien käsittelyn ja asemoinnin, datapisteiden päällekkäisyyden estämisen sekä lähentämisen. Näennäisestä tarkkarajaisuudestaan huolimatta näiden osa-alueiden vaikutukset kaavioon ovat laaja-alaiset ja lisäksi ne kattavat myös suurimman osan käyttämistäni tekniikoista.

7.6 Pudotuslista-akseleiden kategorianimien käsittely ja asemointi

Myös pudotuslistoja käyttävien akselien arvoja käsiteltiin siis numeraalisessa muodossa, joten kategorioille piti ensin antaa oikeat nimet. Tähän tarkoitukseen käytin akseleiden tickFormat-funktiota kuvan 16 osoittamalla tavalla. Pudotuslistojen arvot tulivat backend:istä aina oikeassa järjestyksessä, joten saatoin hyödyntää niitä ilman erityistä käsittelyä.

```
} else if (this.yAxisContentType ===
          parseInt(Common.TABLE_COLUMN_TYPES.DROPDOWN.value )) {
     this.yAxis = d3Axis.axisLeft(this.y)
          .tickValues(this.numericValuesForYDropdownCategories)
          .tickFormat(function(i) {
                         return this.yDropdownItemContents[i].name
                    }.bind(this))
          .tickSize(0).tickPadding(15);
}
```

Kuva 16. Pudotuslistakategorioiden nimien käyttäminen akselin tick-arvoina.

Pudotuslistakategorioiden nimet eivät saaneet ulottua viereisten kategorioiden alueille, mikä edellytti pitkien nimien muokkaamista sopivaan esitysmuotoon. Olin jo valinnut nimille pienen fonttikoon tilankäyttöä tehostaakseni, mutta tämän lisäksi minun piti toteuttaa myös pitkien nimien rivittäminen ja katkaisu. Päädyin siihen, että kahden ensimmäisen rivin jälkeen nimi katkaistaisiin ja sen perään lisättäisiin katkaisun merkiksi kolme pistettä. Halusin kuitenkin, että käyttäjä voisi halutessaan nähdä myös nimen lyhentämättömän version. Muutin siis aikaisempaa toteutustani siten, että tooltip-elementtien muodostamiseen oli mahdollista käyttää myös nimien tekstielementtien sisältämää tietoa.

```
this.yAxisTickLabels = this.yAxisGroup.selectAll(".tick text")

...
    .on('mousemove', this.formatYAxisTickLabelTooltip.bind(this))

  setTimeout(this.formatYDropdownLabels.bind(this), 300)
```

Kuva 17. Tick-nimikkeiden ja niiden infolaatikoiden muodostamisen alkuvaihe.

Tekstien rivittäminen edellytti setTimeout-funktion käyttöä. Rivitysprosessin aikana wrapLabelText-funktio (kuva 19) muuttaa sille välitetyt tekstielementit tspan-elementeiksi, mitä varten tarvitaan tieto elementtien alkuperäisestä pituudesta. Jos tekstielementtiä ei ole vielä piirretty ennen sen pituuden tutkimista, getComputedTextLength tulkitsee tekstin pituuden olevan 0. Tällöin tekstin rivitystä ei tapahdu, sillä wrapLabelText-funktiolle annettu tekstin maksimipituus ei ikinä ylity. Siispä pitkätkin tekstit täytyy lisätä kaavioon aluksi yhtenä rivinä, jotta niiden todellinen mitta saataisiin laskettua ennen rivitystä. Valitsin setTimeout-funktiolle välitetyksi viiveeksi 300ms, sillä sitä alhaisemmat arvot muuttivat tekstien liikkeet epämiellyttävän äkilliseksi nytkähtelyksi.

Tekstien maksimipituudet oli helppo laskea kaavioon määriteltyjen raja-arvojen avulla. Pudotuslistakenttiä käytettäessä kaikki kaavion alueet olisivat yhtä pitkiä ja leveitä, joten käytännössä mitkä tahansa peräkkäiset arvot olisivat soveltuneet laskentaan yhtä hyvin. Päädyin kuitenkin laskemaan alueiden mitat akseleiden minimiarvojen ja ensimmäisten raja-arvojen avulla (kuva 18).

```
    formatYDropdownLabels: function () {
       this.dropdownAreaMaxHeight = this.y(this.yMin) -
                            this.y(this.yAxisLimitValues[0])
       this.yAxisTickLabels
          .call(this.wrapLabelText, this.dropdownAreaMaxHeight - 10)
       this.positionYDropdownLabels()
    },
```

Kuva 18. Y-akselin kategorianimien formatointi.

Y-akselin koordinaattiarvojen käsittely oli aina hieman hankalampaa x-akseliin verrattuna, sillä y-skaalan korkeammat arvot vastasivat SVG-elementin y-koordinaatin pienempiä arvoja. Tästä johtuen laskutoimitukset piti tehdä aina vähentämällä näennäisesti pienemmästä arvosta suurempi arvo. Myös y-akselin tekstit vaativat erikoiskäsittelyä, sillä moniriviset tekstit laajenivat kaavioalueen päälle. Vaikka olisinkin voinut siirtää tekstejä pysyvästi kauemmas kaavioalueesta, halusin tekstien muuttavan paikkaansa vasta silloin, kun vähintään yksi teksteistä olisi useamman rivin mittainen.

```
wrapLabelText: function(text, width) {

        ...
        while (word = words.pop()) {
           line.push(word);
           tspan.text(line.join(" "));
           if (tspan.node().getComputedTextLength() > width) {
              line.pop();
              tspan.text(line.join(" "));
              line = [word];
              tspan = text.append("tspan")
                          .attr("x", 0).attr("y", y)
                          .attr("dy", ++lineNumber *
                                      lineHeight + dy + "em")
              if ( lineNumber ⩽ 1) {
                 tspan.text(word);
              } else {
                 tspan.text(" ... ");
                 break;
```

Kuva 19. Osa wrapLabelText-funktion toimintalogiikasta.

Y-akselin tekstejä rivitettäessä wrapLabelText-funktiolle (kuva 19) välitetty width-parametri kuvaa siis kategoria-alueiden korkeutta. Vastaavasti x-akselin tapauksessa tekstien maksimipituus määräytyy kategoria-alueiden leveyden mukaan. Funktiossa teksti jaetaan sanoiksi, joita lisätään rivielementille, kunnes annettu maksimipituus ylittyy. Tämän jälkeen rivistä tehdään tspan-elementti, joista ensimmäinen sijoitetaan tekstin alkuperäiseen kohtaan ja seuraavat tspan-elementit allekkain ensimmäisen rivin alapuolelle.

Jokainen tekstirivi muodosti siis oman tspan-elementin, joten ratkaisin y-akselin tekstien sijoitteluun liityvän ongelman akseliryhmään kuuluvien tspan-elementtien määrää tutkimalla. Mikäli määrä oli sama kuin alkuperäisten tekstien määrä, tekstien sijaintia ei tarvinnut muuttaa. Jos tspan-elementtejä taas oli teksteihin verrattuna enemmän, vähintään yksi teksteistä oli jaettu usealle riville. Tällöin valitsin kaikki yAxisGroup-elementtiin kuuluvat tspan-elementit ja muutin niiden y-attribuutin arvoa, eli siirsin tekstejä vasemmalle antaakseni niille lisää tilaa. Vaikka y-attribuutin muuttaminen siirsikin

elementtejä niiden omaa y-akselia pitkin, tässä tapauksessa tekstit liikkuivat niiden orientaatiosta johtuen SVG:n x-akselin suuntaisesti.

7.7 Datapisteiden päällekkäisyyden estäminen

Vaikka datapisteiden päällekkäisyys olisi ollut ongelma myös numeraalisia akselikenttiä käytettäessä, päällekkäisyyden estäminen oli ehdottoman välttämätöntä erityisesti pudotuslistakenttien tapauksessa elementtien saamien arvojen rajallisen määrän takia. Sen lisäksi, että päällekkäisyys olisi estänyt muiden datapisteiden alle jäävien elementtien tarkastelun, käyttäjä olisi voinut myös saada täysin virheellisen käsityksen datapisteiden määrästä ja jakaumasta.

Tavoitteenani oli siis, että numeraaliset datapisteet jäisivät mahdollisimman lähelle omaa todellista paikkaansa menemättä kuitenkaan päällekkäin toistensa kanssa, kun taas pudotuslista-arvoja saavat datapisteet voisivat sijoittua mihin tahansa oman alueensa rajojen sisällä. Mikäli numeraalisia datapisteitä olisi paljon, riskinä olisi, että ne asettuisivat liian kauas todellisesta paikastaan, mikä voisi myös olla harhaanjohtavaa. Päätinkin lopulta liittää kaavioon valintaruudun, jonka avulla käyttäjä voisi itse hallita datapisteiden päällekkäisyyttä, sillä tämä mahdollistaisi tarvittaessa myös elementtien palauttamisen niiden todellisille paikoille.

Eräs suhteellisen yksinkertainen tapa datapisteiden päällekkäisyyden rajoittamiseen on ns. "jittering", jossa datapisteille arvotaan uudet, satunnaiset paikat niiden oikean paikan läheisyydestä. Laskennan satunnaisuudesta johtuen tämä tapa ei kuitenkaan tarjonnut haluamaani tarkkaa kontrollia datapisteiden hallintaan, joten päädyin sen sijaan tutkimaan, soveltuisiko D3:n force-ominaisuus tähän tarkoitukseen paremmin.

D3:n force-moduulin avulla voidaan luoda simulaatio (kuva 20) ja asettaa se vaikuttamaan haluttuun dataan ja siten myös siihen liittyviin elementteihin. Simulaation avulla sen piirissä olevia elementtejä voidaan hallita simulaatioon lisättävien voimien avulla. Elementeille voidaan esimerkiksi määritellä niihin kohdistuvat x- ja y-voimat, jotka alkavat vetää elementtejä puoleensa tiettyjä SVG-elementin koordinaatteja kohti. Elementeille voidaan myös määritellä joko negatiivinen tai positiivinen varaus, jonka perusteella elementit joko hylkivät toisiaan tai vetävät toisiaan puoleensa. Lisäksi elementteihin voidaan kohdistaa myös useita muita voimia.

Kun simulaatio luodaan, se käynnistyy automaattisesti, minkä jälkeen se alkaa muuttaa elementtien paikkoja määriteltyjen voimien perusteella. Simulaatio joko luo itse hallitsemilleen elementeille x- ja y-attribuutit paikkojen laskentaa varten tai muuttaa niiden arvoja mikäli ne on jo määritelty. Elementtien paikan muutos aiheuttaa tick-tapahtuman, johon voidaan yhdistää haluttuja toimenpiteitä. Simulaatio ei itsessään kuvaa elementtien paikkojen muutosta visuaalisesti, vaan tätä varten tick-tapahtumaan pitää yhdistää funktio, joka siirtää elementtejä simulaation laskemiin koordinaatteihin.

```
createForceSimulation: function () {
   this.overlappingDatapoints = false
    ...
   this.simulation = d3Force.forceSimulation(this.data)
      .force("x", this.forceX)
      .force("y", this.forceY)
      .force("charge", this.charge)
      .force("collide", this.forceCollision)
      .on("tick", this.ticked.bind(this))
},
ticked: function () {
   this.datapoints
      .attr("x", function(d) { return d.x })
      .attr("y", function(d) { return d.y })
},
```

Kuva 20. Simulaation luominen, voimien määrittely ja datapisteiden siirtäminen.

Simulaation käynnistyessä sen liike-energiaa kuvaava alpha-taso on 1, joka laskee kohti nollaa alphaDecay-funktion parametrin määrittelemällä tavalla. Kun alpha-taso laskee nollaan, simulaation elementit pysähtyvät. Elementit saadaan uudelleen liikkeelle, kun alpha-taso nostetaan yhteen ja simulaatio käynnistetään uudelleen.

```
checked: function (event) {
   if(event.target.checked){
      this.charge.strength(0)
      this.forceCollision.strength(0)
   } else {
      if (this.xAxisContentType ===
    parseInt(Common.TABLE_COLUMN_TYPES.DROPDOWN.value) &&
      ...
         this.charge.strength(-20)
      } else {
         this.charge.strength(-10)
      }
      this.forceCollision.strength(1)
   }
   this.simulation.alpha(1.0)
   this.simulation.restart()
  },
```

Kuva 21. Datapisteiden päällekkäisyyden muuttamisen funktio.

Käyttämäni x-, y- ja charge-voimat saivat siis kaavion datapisteet hakeutumaan mahdollisimman lähelle todellista sijaintiaan, mutta negatiivisesta varauksestaan johtuen ne pysyivät silti erillään toisistaan. "Päällekkäiset datapisteet" -asetuksen valitseminen puolestaan poistaa datapisteiden negatiivisen varauksen, nostaa simulaation alpha-tason yhteen ja käynnistää simulaation uudelleen. Tällöin datapisteet eivät enää hylji toisiaan, joten x- ja y-voimat vetävät datapisteet todellisiin paikkoihinsa. Asetuksen kääntäminen pois päältä puolestaan palauttaa datapisteiden negatiivisen varauksen alkuperäiseen arvoonsa, jolloin datapisteet ponnahtavat taas erilleen toisistaan.

Aluksi päällekkäisyysasetuksen muuttamisen vaikutukset ulottuivat myös niihin datapisteisiin, jotka eivät menneet päällekkäin muiden datapisteiden kanssa, mikä aiheutti niiden turhaa tärähtelyä. Sain kuitenkin tämän visuaalisen haitan poistettua määrittämällä elementtien varauksille suhteellisen pienen maksimietäisyyden. Päädyin lopulta myös lisäämään elementeille törmäysvoiman, sillä päällekkäisyyden estäminen ainoastaan elementtien negatiivisen varauksen avulla osoittautui haastavaksi. Suuret päällekkäiset elementtimäärät vaativat yhä vahvemman negatiivisen varauksen käyttöä, mikä puolestaan aiheutti elementtien vauhdikasta sinkoilua paikasta toiseen päällekkäisyysasetusta muutettaessa.

Matriisimaista kuvaustapaa käytettäessä datapisteiden sijoittelu oli vapaampaa, joten tässä tapauksessa asetin elementit hylkimään toisiaan hieman aggressiivisemmin. Ongelmana oli kuitenkin edelleen, että datapisteet valuisivat ennemmin tai myöhemmin viereisille alueille elementtien määrän kasvaessa. Datapisteiden päällekkäisyysasetusta muuttamalla käyttäjä kuitenkin saisi selville datapisteiden todelliset kategoriat, minkä lisäksi myös lähentäminen lieventäisi ongelmasta aiheutuvaa haittaa. Ongelman varsinainen korjaaminen olisi kuitenkin edellyttänyt elementtien liikkumisalueen rajaamista ns. "bounding box" -määrittelyllä, mikä olisi vaatinut elementtien ja raja-arvojen tuntemattoman määrän takia kenties pitkiäkin matemaattisia pohdintoja, joihin päätin paneutua vasta jatkokehityksen aikana.

7.8 Kaavion lähentäminen

Lähentäminen oli toteuttamistani kaavion ominaisuuksista ehdottomasti hankalin, sillä käytännössä kaikkien kaavion osien piti reagoida lähentämiseen jollain tavalla. Lähentämistoiminnallisuutta lisätessäni minun piti myös tehdä laajamittaisia muutoksia moniin kaavion aikaisempiin ominaisuuksiin saadakseni ne yhteensopiviksi lähentämisen kanssa, mikä johti projektin vaikeustason yhtäkkiseen nousuun.

7.8.1 Lähennystason valintatavat

Lähennettävän alueen valintaan ja lähentämiseen on useita eri tapoja, joista käyttäjän kannalta intuitiivisin vaihtoehto on todennäköisesti lähentäminen hiiren rullan avulla. Tässä toteutustavassa on

kuitenkin omat riskinsä etenkin ARCin kontekstissa. Kaaviot näkyvät mallien sivuilla kaikkein ylimpänä, mikä tarkoittaa, että jos käyttäjä haluaa nähdä mallin muita tietoja, hänen tulee vierittää sivua ensin alas. Tässä tilanteessa hänen kursorinsa on todennäköisimmin juuri kaavion päällä. Käyttäjälle olisi epäilemättä hyvin ärsyttävää, jos sivun liikkumisen sijaan hiiren rulla yllättäen lähentäisikin kaaviota.

Vertailin eri toteutustapoja myös testikäyttämällä D3:n sivuilta löytyviä esimerkkikaavioita. Edellä mainitun ongelman lisäksi en myöskään pitänyt siitä, miltä kaavion lähentäminen tuntui hiiren rullaa käyttäen. Tarkan lähennysalueen valinta tuntui hankalalta, ja etenkin lähennystasojen ja -alueiden muuttaminen oli hidasta. Löysin kuitenkin mielestäni huomattavasti paremman lähennystavan, joka oli toteutettu D3:n brush-työkalun avulla. Vaikka brush:in käyttö tuntui aluksi hieman erikoiselta, mielestäni se oli kuitenkin huomattavasti miellyttävämpää hiiren rullan käyttöön verrattuna. Esimerkiksi lähennetyn alueen vaihtaminen kaavion reunasta vastakkaiselle reunalle oli huomattavasti nopeampaa. Tämän lisäksi brush mahdollisti myös tarkemman alueiden valitsemisen.

7.8.2 D3 Brush

Brush-työkalua käytetään elementtien tai niiden osien manuaaliseen valitsemiseen. Esimerkiksi kaavioni tapauksessa halutun alueen valitseminen aloitettaisiin viemällä hiiren kursori kaavion päälle ja painamalla hiiren painike alas. Kun tämän jälkeen kursoria liikutetaan, brush-työkalu indikoi valinnan kohteena olevaa aluetta suorakuutiolla, jonka koko ja muoto vaihtuu kursorin liikkeiden mukaan. Alue valitaan vasta, kun hiiren painike päästetään ylös. Brush ei siis itsessään lähennä kaaviota, vaan brush:ia käytetään vain lähennettävän alueen valitsemiseen. Lähennys voidaan palauttaa takaisin alkuperäiselle tasolle kaksoisklikkaamalla kaaviota.

Jotta kaavion lähennystasoa voitasiin muuttaa valitun alueen perusteella, alueeseen liittyvä tieto pitää välittää eteenpäin. Tämä tieto sisältyy brush:in lähettämään end-tapahtumaan, johon voidaan liittää funktio, joka ottaa tapahtuman vastaan ja käsittelee sen sisältämän tiedon. Tapahtuma sisältää käyttäjän valitsemat koordinaatit, joiden avulla voidaan laskea uudet arvot akseleiden käyttämien x- ja y-skaalojen domain-alueille.

Tämän lisäksi pitää ottaa huomioon myös tilanne, jossa käyttäjä onkin tuplaklikannut kaaviota palauttaakseen lähennyksen oletustason. Tällöin tapahtuman mukana ei tule valintaa ja brush:in toiminta jäädytetään hetkellisesti hyödyntäen JavaScript:in sisäänrakennettua setTimeout-metodia, jolle välitetään callback:ina idled-funktio. Tämä funktio nollaa idleTimeout-muuttujan arvon määritellyn ajan jälkeen, jolloin brush aktivoituu uudelleen. Muuttujaa käyttämällä varmistetaan, että tuplaklikkauksen tapauksessa huomioidaan vain ensimmäinen klikkaus, eikä skaalojen domain-arvoja turhaan muuteta kahta kertaa peräkkäin.

```
brushEnded: function (event) {
      var idleDelay = 350;
      var s = event.selection;
      if (!s) {
         if (!this.idleTimeout) {
            return this.idleTimeout =
                        setTimeout(this.idled.bind(this),
                                   idleDelay);
         }
         this.x.domain([this.xMin, this.xMax]);
         this.y.domain([this.yMin, this.yMax]);
      } else {
         this.x.domain([s[0][0],
                        s[1][0]].map(this.x.invert, this.x));
         this.y.domain([s[1][1],
                        s[0][1]].map(this.y.invert, this.y));
         this.scatter.select(".brush").call(this.brush.move, null)
      }
      this.zoom();
},
```

Kuva 22. Käytettäviä skaaloja muutetaan vastaanotetun valinnan mukaisesti.

Kuten aikaisemminkin, setTimeout-metodin käyttö edellytti myös tässä tapauksessa JavaScript:in kontekstin ymmärtämistä ja huomioon ottamista. Mikäli en olisi välittänyt scatterChart-objektin kontekstia setTimeout-metodille, idled-funktion käyttö ei olisi ollut mahdollista, sillä JavaScript olisi etsinyt sitä Window-objektin kontekstista, johon setTimeout-metodi kuuluu. Käytettävä konteksti tuli siis välittää setTimeout-metodille sitomalla se idled-funktioon bind-funktion avulla.

7.8.3 Zoom-funktio

Vaikka brush:in lisääminen kaavioon olikin haastavaa, sen käytöstä löytyi useita havainnollistavia esimerkkejä, joita saatoin noudatella omassa toteutuksessani. Sen sijaan zoom-funktion lisääminen oli huomattavasti vaikeampi prosessi lähennettävien elementtien eroista ja suuresta määrästä johtuen. Lisäksi esimerkiksi taustavärialueet eivät olleet yleinen hajontakaavion ominaisuus, joten oli vaikeaa arvioida, kuinka saisin ne lähentymään oikein.

Alun perin olin määritellyt suorakuutioelementeille x- ja y-attribuuttien lisäksi leveyden ja pituuden. Leveys ja pituus säilyivät kuitenkin staattisina lähentämisestä huolimatta, joten minun piti vaihtaa tapaa, jolla suorakuutioiden mitat laskettiin. Poistin siis suorakuutioelementtien aikaisemmat attribuutit ja korvasin ne elementtien ulkomittoja kuvaavilla x1-, x2-, y1-, ja y2-attribuuteilla. Aikaisemmasta

poiketen nämä attribuutit sisälsivät akseliarvoja, jotka voitiin lähennyksen aikana muuttaa SVG-koordinaateiksi. Tämän muutoksen jälkeen myös suorakuutiot lähentyivät odotetulla tavalla.
Eri elementtien onnistunut lähentäminen edellytti myös transitioiden käytön opettelua, sillä ilman niitä elementit olisivat vaihtaneet paikkaansa ensin katoamalla ja sitten ilmestymällä yllättäen uuteen paikkaansa. Olin jakanut kaavion ulko- ja sisäosan eri elementeiksi selkeyden ja clip-alueiden monipuolisemman hyödyntämisen takia, joten minun piti tehdä kyseisille alueille myös erilliset transitiot.

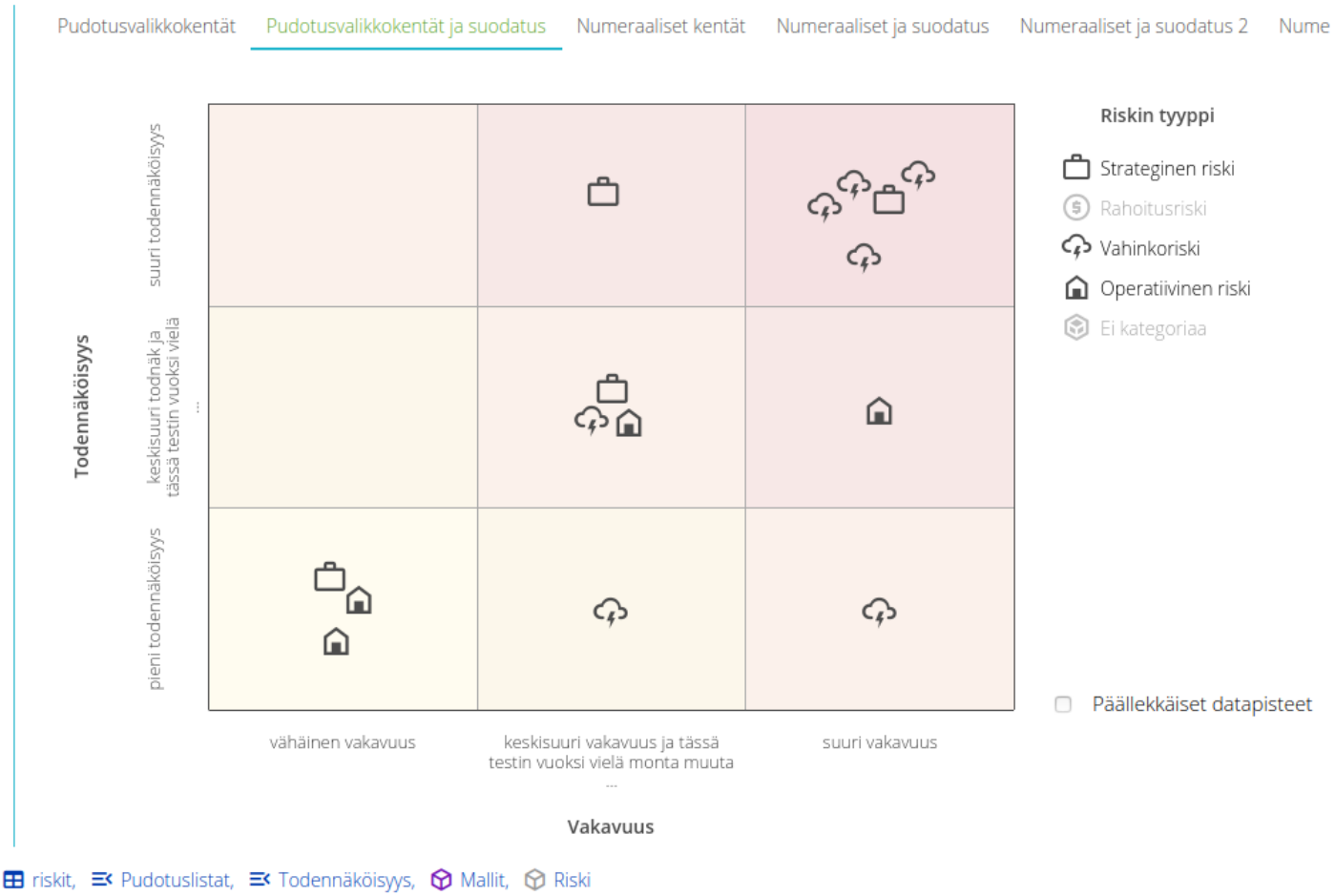
Datapisteet puolestaan olivat simulaation hallinnassa eikä niiden suora manuaalinen siirtäminen siten ollut mahdollista. Sain kuitenkin siirrettyä datapisteitä välillisesti niihin kohdistuneiden x- ja y-voimien paikkoja vaihtelemalla. Vaikka tämä tapa siirsikin datapisteitä onnistuneesti, poikkeavasta siirtotavasta johtuen datapisteet siirtyivät uusiin paikkoihinsa huomattavasti nopeammin esimerkiksi raja-arvoviivoihin verrattuna. Illuusio yhdestä kaaviokokonaisuudesta siis rikkoutui, mikä ei onneksi kuitenkaan vaikuttanut kaavion varsinaiseen käytettävyyteen millään tavalla.

7.9 Yhteenveto toteutuksen tuloksista

Projektin tuloksena oli siis uuden raja-arvoja ja taustavärejä hyödyntävän ARCin kaaviotyypin ensimmäinen versio, joka muuttaa esitystapaansa dynaamisesti matriisin (kuva 23) ja hajontavaavion (liitteet 17-18) välillä. Kaaviolla elementtien datapisteinä toimivat joko elementtien omat ikonit (liite 16), tai suodatusta käytettäessä niiden edustamien kategorioiden ikonit (liite 19). Datan suodatuksen lisäksi myös datapisteiden päällekkäisyyttä voidaan hallita valinta-asetuksen avulla (liite 21). Kaaviota voidaan myös lähentää (Liite 20), mikä mahdollistaa kaavion alueiden lähemmän tarkastelun.

Riski (Malli)

Kuvaajat



riskit, Pudotuslistat, Todennäköisyys, Mallit, Riski

Kuva 23. Matriisimuotoinen kaavio suodatuksella.

8 Pohdinta

Tässä osiossa tutkin projektin yleistä onnistuneisuutta sekä sitä, vastasiko lopullinen kaavio sille asetettuja vaatimuksia. Lisäksi arvioin projektin rajauksen ja teknologiavalintojen onnistuneisuutta sekä käyn läpi kaavion jatkokehityksen seuraavat askeleet. Kappaleen lopussa arvioin myös opinnäytetyöprosessin etenemistä ja oppimistavoitteideni täyttymistä.

8.1 Tulosten tarkastelu

Kaikki suunnitteluvaiheen aikana muodostetut kaavion ominaisuuksiin ja toimintaan liittyvät kriteerit täyttyivät, vaikka kehitysprosessi etenikin huomattavasti hitaammin kuin olin odottanut. Osittain tämä johtui projektin laajuuden muutoksista (esim. D3-version päivittäminen), mutta pääasiallisesti aikataulu ylittyi, sillä minulla ei projektin alussa ollut tarvittavaa tietoa sen todellisen laajuuden arviointia varten. Kaikeksi onneksi Arterin näkökulmasta projektin etenemiselle ei ollut tiukkaa aikataulua ja projektin ohjausryhmä suhtautui etenemisvauhtiini ymmärtäväisesti. Taulukko 5 listaa projektin ohjausryhmän jäsenten suoritusarviot projektini eri osa-alueista.

Taulukko 5. Projektin ohjausryhmän arvio projektin onnistumisesta asteikolla 1-5.

Arvioija	**Aihe**	**Arvosana (1-5)**
ARCin tuoteomistaja	Kehitysvaatimusten täyttyminen	5
Sovellusarkkitehti	Kehitysprosessin eteneminen ja tekninen toteutus	4
Tuotekehityspäällikkö	Asiakasarvon tuottaminen	5

Lisäksi ohjausryhmän jäsenet kommentoivat antamiaan arvosanoja ja projektia yleisesti omista näkökulmistaan:

"Projektissa toteutettu kaavio on onnistuneesti kehitetty sille asetettujen vaatimusten mukaan. Odotukset ovat jopa ylittyneet, koska kaavioon on pystytty implementoimaan sellaisia käyttöä helpottavia toiminnallisuuksia, jotka nousivat mahdollisuuksina esiin työn edetessä. Lopputuloksesta näkee, että siinä on ajateltu asiakaslähtöisesti kaavion loppukäyttäjää. Toteutetut toiminnallisuudet ovat intuitiivisia. Vaatimusten toteuttamiseen on löydetty selvästi järkevimmät ja parhaiten käyttötarkoitusta palvelevat ratkaisut."

- ARCin tuoteomistaja

"Projektin scatter plot -tyyppinen kaaviokomponentti toteutettiin ja sovitettiin Backbone-näkymän rakenteeseen onnistuneesti päivitettyä D3-kirjastoa käyttäen. D3-kirjaston päivitys oli työläs prosessi kirjaston uudistuneen modulaarisen rakenteen ja RequireJS-moduulihallinnan yhteensovittamisen takia. Lopputuloksena syntynyt kaaviokomponentti on hyvin jäsennelty, pilkottu alifunktioihin ja sovitun koodityylin mukainen. Päivitystyötä riittää projektin jälkeen, jotta muut D3-kirjastoon tukeutuvat komponentit ohjelmistossa saadaan myös sovitettua uudistuneen version kanssa yhteensopiviksi."

- Sovellusarkkitehti

"Toteutus täydentää ohjelmistomme visualisointikyvykkyyttä, mikä on sen parhaiten asiakasarvoa tuottavaa aluetta. Toteutus on joustava eikä sitoudu ennalta määriteltyihin rakenteisiin, mikä vastaa tuotelinjaustamme hyvin. Toteutus on tehty itsenäisesti käyttöliittymän logiikkaa ja design pattern:eja noudattaen. Toteutus ylitti alun perin asetetut odotusarvot."

- Tuotekehityspäällikkö

Saamieni arvosanojen ja kommenttien perusteella vaikuttaa siis siltä, että projektin ohjausryhmä piti projektia kaiken kaikkiaan hyvin onnistuneena.

8.2 Rajauksen onnistuneisuus

Projektin toiminnallisen osuuden vahvempi rajaaminen olisi voinut olla perusteltu valinta. Halutessani olisin voinut sisällyttää opinnäytetyöhöni esimerkiksi vain kaavion numeraalisia akseleita käyttävän muodon. Projektin alusta asti oli kuitenkin selvää, että asiakkaiden tarpeet liittyivät suurimmaksi osaksi nimenomaan matriisimuotoiseen esitystapaan, joten sen rajaaminen pois projektin piiristä olisi ollut mielestäni kyseenalainen valinta.

Tarvittaessa olisin voinut toteuttaa myös muita kaavion toiminnallisuuksia vasta jatkokehityksen aikana, mutta tällöin parhaan leikkauskohdan valitseminen olisi ollut vaikeaa ominaisuuksien keskinäisistä riippuvuuksista johtuen. Myös D3:n päivittämisen olisin ainakin teoriassa voinut siirtää myöhemmäksi, mikä olisi todennäköisesti kuitenkin vaikeuttanut kehitystyötäni huomattavasti. Koen siis, että valitsemani rajaus oli viime kädessä onnistunut, sillä se kattoi huolellisen suunnitteluvaiheen, kaavion suunniteltujen ominaisuuksien toteuttamisen ja myös muutaman muun yleishyödyllisen osa-alueen.

8.3 D3:n soveltuvuus projektiin

Voin suositella D3:a vastaaviin projekteihin, sillä se sopi erinomaisesti myös kaavion epätyypillisten ominaisuuksien toteuttamiseen. D3:n toimintaperiaatteista johtuen se mahdollisti hyvin pikkutarkkojenkin muutosten tekemisen, mikä toisaalta teki kirjaston käytöstä myös haastavaa. Mielestäni siis uusien D3-käyttäjien kannattaa joko aloittaa huomattavasti yksinkertaisemmasta projektista tai vähintään varata riittävästi aikaa D3:n käytön opetteluun, sillä D3:n ominaisuuksien onnistunut soveltaminen edellyttää usein hyvin syvällistäkin perehtymistä.

D3:a käyttäessäni yllätyin myös siitä, ettei kirjasto tarjonnut muutamia yksinkertaisiakaan ominaisuuksia valmiina pakettina. Esimerkiksi infolaatikot ovat ymmärtääkseni tavallinen ominaisuus muissa visualisointikirjastoissa, kun taas D3:lla nekin piti rakentaa alusta alkaen itse. Olen aikaisemmin käyttänyt esimerkiksi ApexCharts.js-kirjastoa, ja vaikka se olikin ominaisuuksiltaan huomattavasti vaatimattomampi, se tarjosi useat ominaisuudet täysin valmiina. Myös ApexCharts-kaavion rajallisia kustomointimahdollisuuksia oli helppo soveltaa. Luulenkin, että tämä ensimmäinen datan visualisointikokemukseni sai minut lukemastani tiedosta huolimatta karkeasti aliarvioimaan D3:n käytön haastavuuden.

8.4 Kaavion jatkokehitys

Vaikka kaavio onkin ominaisuuksiensa puolesta valmis, sen toimintaan liittyy vielä muutamia ongelmia. Tällä hetkellä esimerkiksi datan suodatus ainoastaan piilottaa valitut elementit, jotka jäävät kuitenkin edelleen force-simulaation piiriin. Datan suodatuksen tulisi siis poistaa elementit kokonaan, jotta jäljelle jäävät elementit ryhmittyisivät paremmin. Elementtien datapisteiden tulisi myös kohdella kaavion raja-arvoviivoja todellisina aluerajoina, jolloin elementtien määrän kasvaessakaan datapisteet eivät tulvisi viereisten alueiden puolelle.

Ikoneita käyttäessäni (liite 12) huomasin myös, että tiettyjen elementtien ikonikoodit eivät vastanneetkaan elementeille määriteltyjä ikoneita. Esimerkiksi elementtien oletusikoni näkyi kaaviossani taulukkoa kuvaavana ikonina. Tämä kielii siitä, että käyttämäni ikonikoodilistauksen koodit poikkeavat muualla ARCissa käytettävistä koodeista. Ratkaisin asian väliaikaisesti muuttamalla elementti-ikonin oikeaan muotoonsa, mutta ongelman juurisyyn korjaaminen vaatii tarkempaa perehtymistä ARCin ikonien tilanteeseen.

Edeltävistä korjaustarpeista huolimatta jatkokehityksen prioriteettina on tällä hetkellä D3-päivityksen rikkomien kaaviotyyppien korjaaminen ja sen jälkeen myös suunniteltujen muutosten toteuttaminen kaavioiden lisäysnäkymään. Tarvittavien korjausten ja lisäysten jälkeen kaavio-ominaisuus tulee etenemään ensin sisäiseen testaukseen. Tämän jälkeen kaaviota tullaan mahdollisesti testaamaan myös Arterin yhteistyökumppanin, Sofigate:n, toimesta. Lisäksi pyrimme muodostamaan myös ARC-

asiakkaista koostuvan testiryhmän, jolta keräämme palautetta uudesta kaaviotyypistä. Vasta näiden vaiheiden jälkeen kaavio on valmis tuotantoon lisättäväksi, joten menee vielä hetki, ennen kuin ARC-asiakkaat pääsevät uuteen ominaisuuteen käsiksi.

8.5 Opinnäytetyöprosessin ja oman oppimisen arviointi

Ennen projektin alkua sain Arterilta kaksi aihe-ehdotusta, joista toinen olisi ollut huomattavasti lähempänä omaa mukavuusaluettani. Valitsin kuitenkin aiheekseni uuden visualisointityökalun toteuttamisen sen mukanaan tuomista epävarmuustekijöistä huolimatta, sillä koin aiheen tarjoavan minulle huomattavasti paremmat mahdollisuudet ammatilliseen kehitykseen. Vaikka tämä arvio pitkin epäilemättä paikkansa, käytännössä myös kaikki projektin aikana kohtaamistani hankaluuksista liittyivät aihevalintaani tavalla tai toisella.

Minulle täysin vieraasta aihealueesta johtuen oletukseni projektin laajuudesta olivat täysin epärealistiset, ja projektin alussa pelkäsin jopa valineeni liian suppean aiheen. Projektisuunnitelmaan kirjaamani aika-arvio oli siis enemmänkin arvaus projektin potentiaalisesta kestosta. Aloitin projektin 14.9.2020 ja alun perin suunnitelmani oli saattaa projekti päätökseen marraskuun loppuun mennessä. Lopulta kuitenkin päädyin työskentelemään projektin parissa aina vuoden 2021 tammikuun loppuun asti.

8.5.1 Toiminnallisen osuuden haasteet

Vaikka aikataulun pitkittymisellä ei itsessään ollutkaan negatiivisia seurauksia, olin koko ajan hyvin tietoinen projektin hitaasta etenemisestä, mistä aiheutuneella stressillä oli huomattavia vaikutuksia mm. ongelmanratkaisukykyyni ja suunnitelmallisuuteeni. Tavallisista toimintatavoistani poiketen aloin projektin aikana esimerkiksi huomaamattani priorisoisoida projektin etenemisnopeutta tekemieni ratkaisujen laadun kustannuksella. Ajan säästämisen sijaan tämä kuitenkin lähinnä hidasti projektia entisestään, sillä jouduin usein vaihtamaan tekemiäni ratkaisuja parempiin projektin edetessä.

Myös projektin vaikeustaso oli huomattavasti arvioimaani korkeampi ja kohtasin huomattavia vaikeuksia näennäisesti yksinkertaistenkin asioiden toteuttamisessa. Projektin alussa lohduttauduin sillä, että työskentelyni tulisi todennäköisesti helpottumaan osaamiseni ja tietojeni karttuessa. Todellisuudessa projektin vaikeustaso säilyi kuitenkin lähes samanlaisena koko projektin ajan, sillä aloitin työskentelyni suhteellisesti helpoimmista ominaisuuksista, joiden jälkeen projektin vaativuus nousi samaan tahtiin oman taitotasoni kanssa. Vaikka osa oppimistani asioista olikin yleistettävissä seuraavien toiminnallisuuksien kehittämiseen, opin myös uusia asioita aivan projektin loppuun asti. Projektiin sisältyi siis ehkä liikaakin uusia asioita melko lyhyellä aikavälillä, mikä lisäsi sen kuormittavuutta.

Hankaluuksia aiheutui myös eri ominaisuuksien keskinäisistä riippuvuuksista. Sen sijaan, että olisin saanut jonkin osa-alueen valmiiksi esimerkiksi projektin puolivälissä, jouduin jatkuvasti muuttamaan kaavion aikaisempia ominaisuuksia uusia toiminnallisuuksia lisätessäni. Vaikka esimerkiksi kaavion akselit olivat ensimmäiset lisäämäni elementit, tein niihin muutoksia vielä projektin viimeisen viikon aikana. Tästä syystä minusta monesti tuntui siltä, etten varsinaisesti edennyt minnekään, vaikka itse kaavio kehittyikin jatkuvasti parempaan suuntaan. Uudet ominaisuudet toivat mukanaan myös uusia ongelmia, joten myöskään käsittelemieni ongelmien kokonaismäärä ei tuntunut vähenevän. Välillä minun olikin vaikeaa pysyä motivoituneena, koska tiesin, että vaikka saisin senhetkisen ongelmani ratkaistua, edessäni olisi välittömästi tuntematon määrä uusia, entistä hankalampia ongelmia.

8.5.2 Kirjallisen osuuden haasteet

Myös opinnäytetyöni kirjallinen osio eteni hitaasti ja epätasaisesti. Minulla oli useita virheellisiä käsityksiä siitä, millaisia vaatimuksia toiminnallista opinnäytetyötä kuvaavan raportin rakenteeseen ja sisältöön kohdistui. Projektin aikana oli myös hyvin vaikeaa arvioida, mitkä sen osista tulisivat lopulta olemaan tärkeitä projektin kokonaisuuden kannalta. Monet kohtaamistani haasteista tuntuivat hyvin merkityksellisiltä projektin aikana, mutta lopulta niiden rooli jäikin melko pieneksi. Kuvasin siis monia käsittelemiäni aiheita alun perin aivan liian yksityiskohtaisesti.

Projektin alussa pelkäsin myös opinnäytetyöni teoreettisen osuuden jäävän liian suppeaksi. Lisäksi olin todella kiinnostunut etenkin datan visualisoinnista, joten päädyin käsittelemään useita eri aiheita, jotka eivät lopulta liittyneetkään riittävällä tasolla toiminnallisen osuuden sisältöön, ja jotka siten jouduin karsimaan pois opinnäytetyön lopullisesta versiosta. Luovuin esimerkiksi kappaleista, joissa käsittelin datan eri muotoja, datan visualisoinnin historiaa, datan ja tiedon suhdetta, sekä useita kaaviotyyppejä, kuten hajonta- ja kuplakaaviota, pylväs- ja palkkikaaviota, histogrammia, viiva- ja aluekaaviota sekä piirakkakaaviota.

Minulla oli siis huomattavia ongelmia visualisointiin liittyvän teoriaosuuden aiheiden valinnassa ja rajaamisessa, ja osion rakenne ja sisältö muuttuikin projektin aikana merkittävästi. Toisaalta, vaikka opinkin datasta ja sen visualisoinnista huomattavasti enemmän kuin opinnäytetyöni laajuus olisi edellyttänyt, koen, että läpi käymäni "ylimääräinen" tieto tarjosi lopulta minulle paremmat lähtökohdat tuottamani visualisoinnin suunnitteluun ja sen lopputuloksen laadun arviointiin.

8.5.3 Oppimistavoitteiden täyttyminen

Datan visualisointiin liittyvän tiedon lisäksi sain projektin aikana myös viimeinkin selkeämmän käsityksen kokonaisarkkitehtuurin luonteesta ja sisällöstä. Vaikka tähänastiset työtehtäväni eivät sinänsä olekaan edellyttäneet kokonaisarkkitehtuurin ymmärtämistä, Arterilla on etenkin viime aikoina rohkaistu myös ohjelmistokehittäjiä perehtymään asiaan tarkemmin, jotta saisimme paremmat

edellytykset ARCin ominaisuuksien käytettävyyden ja ARC-asiakkaiden tarpeiden arviointiin. Projektin loppupuolella Arterilla järjestettiin esimerkiksi sisäinen koulutus juuri ARCin käyttötarkoituksiin liittyen, ja huomasin ilokseni, että aikaisemmasta poiketen koulutuksen sisältö ei enää tuntunutkaan itselleni vieraalta opinnäytetyöni aikana oppimieni asioiden takia.

Myös tiedonhallintalaki on Arterille hyvin tärkeä aihe, sillä suurin osa uusista ARC-asiakkaista tekee kuvauksia juuri tiedonhallintalain edellyttämillä tavoilla. Ennen opinnäytetyötäni en kuitenkaan tiennyt käytännössä mitään tiedonhallintalaista tai sen yhteydestä kokonaisarkkitehtuuriin. Yleisesti ottaen opinnäytetyöni aikana sain siis huomattavasti paremman käsityksen Arterin tuotekehityksen laajemmasta kontekstista.

Opinnäytetyön toiminnallisen osan tavoitteeni toteutuivat myös hyvin, sillä sain runsaasti uutta tietoa ARCin ohjelmallisesta toiminnasta ja minulle aiemmin vieraista frontend-teknologioista. Projektin aikana käytin ensimmäistä kertaa mm. ARCin tietokantaa, MongoDB:tä (liite 7), ja opin, miten kaavioita käsitellään ARCin backendissa. Välittäessäni uuden kaaviotyypin tarvitsemaa tietoa ARCin backend:ista frontend:in käytettäväksi sain myös lisää kokemusta Java-koodin refaktoroinnista aikaisempaa koodia siistiessäni.

Tein ARCiin ensimmäistä kertaa myös kirjastopäivityksen, mikä puolestaan edellytti ARCin moduulienhallintaan ja Require.js:n toimintaan perehtymistä. Lisäksi sisäistin vihdoinkin, miten Backbone.js vaikuttaa ARCin frontendin rakenteeseen ja sen käsittelemään tietoon. Käytin ensimmäistä kertaa myös Underscore.js:n tarjoamia apufunktioita (liite 6).

Projektin aikana D3.js:n yleiset periaatteet ja käyttämieni D3-moduulien sisältö tulivat minulle tutuiksi. D3 on kuitenkin hyvin laaja kirjasto, joka sisältää kymmeniä moduuleja, joten en voi vielä pitää D3-osaamistani syvällisenä tai laajana. Projektin aikana sain kuitenkin paljon tarvittavaa perustietoa, mikä epäilemättä tarjoaa minulle huomattavasti aikaisempaa vahvemmat edellytykset myös muiden D3-toiminnallisuuksien hyödyntämiseen tulevaisuudessa.

Nimensä mukaisesti D3.js on JavaScript-pohjainen kirjasto, joten opin projektin aikana myös tärkeitä asioita JavaScript:in toiminnasta. Olen oppimisyrityksistäni huolimatta aina pitänyt monia JavaScript:in toiminnallisuuksia erittäin hämmentävinä. Projektin aikana kuitenkin sisäistin viimeinkin esimerkiksi JavaScript:in kontekstia kuvaavan this-avainsanan sekä callback- ja anonyymifunktioiden käyttötarkoituksen.

Sain hyvää harjoitusta myös CSS:n käyttöön liittyen. Perehdyin aikaisempaa tarkemmin esimerkiksi flexbox-sijoittelumoduulin käyttöön (liite 13). Tämän lisäksi käytin kaaviossani useita tyylimäärittelyjä, joiden olemassaolosta en aiemmin ollut tietoinen. Suurin osa näistä asetuksista, kuten shape-

rendering- (liite 14) ja transform-attribuutit (liite 15) sekä erilaiset elementtien sijoittelun asetukset liittyivät nimenomaan SVG-elementtien tyylimäärittelyihin. En ennen projektia tiennyt juuri mitään myöskään ikoneista tai niiden käytöstä, ja yllätyin esimerkiksi siitä, että saatoin lisätä kaaviooni ikonit normaalina tekstinä ikonifonttia hyödyntäen.

Vaikka kaavioiden lisäysnäkymän uusiminen rajautui lopulta pois projektin piiristä, sain projektin suunnitteluvaiheen aikana yllätyksekseni hieman kokemusta myös UI-/UX-suunnittelusta. Tämä sai minut aikaisempaa tietoisemmaksi siitä, mitä tekijöitä on hyvä ottaa huomioon, kun käyttöliittymään tehdään muutoksia. En ollut myöskään aikaisemmin käyttänyt mockup-työkaluja, ja uskon, että rajallisista Figma-taidoistani voi olla hyötyä myös tulevaisuudessa.

Kaiken kaikkiaan opinnäytetyöprojektini siis sekä syvensi olemassa olevaa osaamistani, että tarjosi minulle täysin uutta tietoa useaan työni osa-alueeseen liittyen. Kohtaamistani hankaluuksista huolimatta koenkin ehdottomasti tehneeni oikean aihevalinnan oman ammatillisen kehitykseni kannalta. Etenkin heikot frontend-taitoni ovat vaivanneet minua jo pitkään, mutta projektin aikana ne kohenivat merkittävästi. Olen aina haaveillut full stack-osaamisesta, ja olinkin äärimmäisen iloinen, kun projektin lopussa tajusin sen kattaneen useita ohjelmistokehityksen eri osa-alueita tietokantapuolesta aina käyttäjäkokemuksen suunnitteluun asti.

Lähteet

AMD 2020. AMD. Luettavissa: https://github.com/amdjs/amdjs-api/wiki/AMD. Luettu: 13.01.2021.

Arter 2020a. Keitä me olemme? Luettavissa: https://www.arter.fi/tietoa-meista/. Luettu: 23.11.2020.

Arter 2020b. IMS-ohjelmisto. Luettavissa: https://www.arter.fi/ohjelmistot/ims-ohjelmisto/. Luettu: 23.11.2020.

Arter 2020c. ARC-ohjelmisto. Luettavissa: https://www.arter.fi/ohjelmistot/arc-ohjelmisto/. Luettu: 23.11.2020.

Arter 2020d. Arter osaksi menestyvää kansainvälistä ohjelmistotaloa. Luettavissa: https://www.arter.fi/uutiset-ja-tiedotteet/arter-osaksi-menestyvaa-kansainvalista-ohjelmistotaloa/. Luettu: 23.11.2020.

Aunola, S. 11.11.2020. Head of Research and Development. Arter. Haastattelu. Helsinki.

Boost Labs. 25.10.2019. 9 Amazing Benefits of Data Visualization Every Business Needs to Know. Luettavissa: https://boostlabs.com/blog/9-benefits-of-data-visualization/. Luettu: 29.11.2020.

Bostock, M. 10.1.2014. Something wrong with the http://d3js.org/d3.v3.min.js package #1693. Luettavissa: https://github.com/d3/d3/issues/1693. Luettu: 13.01.2021.

Bostock, M. 28.6.2016. v4.0.0. Luettavissa: https://github.com/d3/d3/releases/tag/v4.0.0. Luettu: 13.01.2021.

Buiza, D. 2.5.2016. RequireJS shim example. Luettavissa: https://www.webcodegeeks.com/web-development/requirejs-shim-example/. Luettu: 25.1.2021.

Chart.js 2020. Luettavissa: https://github.com/chartjs/Chart.js. Luettu: 26.1.2021.

Cowan, N. 18.3.2009. What are the differences between long-term, short-term, and working memory? Luettavissa: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2657600/. Luettu: 14.3.2021.

Crooks, R. 28.7.2017. Why Most People's Charts & Graphs Look Like Crap. Luettavissa: https://blog.hubspot.com/marketing/data-visualization-mistakes. Luettu: 23.11.2020.

Digiteum. 19.7.2019. Importance of Data Visualization for Your Business. Luettavissa: https://www.digiteum.com/data-visualization-your-business. Luettu: 23.11.2020.

D3 2020a. Data-Driven Documents. Luettavissa: https://d3js.org/. Luettu: 15.11.2020.

D3 2020b. D3: Data-Driven Documents. Luettavissa: https://github.com/d3/d3. Luettu: 26.1.2021.

eduCBA 2020. Data Analysis Process. Luettavissa: https://www.educba.com/data-analysis-process/. Luettu: 21.12.2020.

Finlex. 10.6. 634/2011. Laki julkisen hallinnon tietohallinnon ohjauksesta. Luettavissa: https://www.finlex.fi/fi/laki/alkup/2011/20110634. Luettu: 15.11.2020.

Guo, J. 22.12.2017. How Is Data Visualization Influenced By Our Cognitive Processes? Luettavissa: https://medium.com/@jiajingguo/how-is-data-visualization-influenced-by-our-cognitive-processes-281d8486abfe. Luettu: 23.11.2020.

Halsas, K. 04.04.2019. Tiedonhallintalaki lyhyesti. Luettavissa: https://www.arter.fi/tiedonhallintalaki-lyhyesti/. Luettu: 22.11.2020.

Heitzman, A. 29.1.2019. Data Visualization: What It Is, Why It's Important & How to Use It for SEO. Luettavissa: https://www.searchenginejournal.com/what-is-data-visualization-why-important-seo/288127/. Luettu: 23.11.2020.

Henshaw, G. 30.9.2019. Customizing Axes in D3.js. Luettavissa: https://ghenshaw-work.medium.com/customizing-axes-in-d3-js-99d58863738b. Luettu: 28.1.2021.

Huusari, L. 27.1.2021. Head of HR. Arter. Haastattelu. Helsinki.

Hygger 2020. Value-Efforts Backlog Prioritization. Luettavissa: https://hygger.io/. Luettu: 25.1.2021.

Isokallio, J. 2005. Yritysarkkitehtuuri. Systeemityö 2005, 3, s. 22-24. Luettavissa: http://www.pcuf.fi/sytyke/lehti/kirj/st20053/ST053-22A.pdf. Luettu: 26.1.2021.

Kehmet 2020. Kehittämismenetelmät. Arkkitehtuuri. Luettavissa: https://kehmet.hel.fi/poikkileikkaavat-toiminnot/arkkitehtuuri2/. Luettu: 15.11.2020.

Kuntaliitto. 4.12.2019. Tiedonhallintalaki astuu voimaan vuodenvaihteessa - mitä se tarkoittaa kuntasektorille? Ajankohtaista. Luettavissa: https://www.kuntaliitto.fi/ajankohtaista/2019/tiedonhallintalaki-astuu-voimaan-vuodenvaihteessa-mita-se-tarkoittaa. Luettu: 22.11.2020.

Lehtinen, N. 7.11.2019. Tiedonhallintalaki on uusi kokonaisarkkitehtuuri. Luettavissa: https://www.arter.fi/tiedonhallintalaki-on-uusi-kokonaisarkkitehtuuri/. Luettu: 22.11.2020.

Majorek, J. 23.8.2020. 14 JavaScript Data Visualization Libraries in 2020. Luettavissa: https://www.monterail.com/blog/javascript-libraries-data-visualization. Luettu: 15.11.2020.

Meeks, E. 11.6.2018. D3 is not a Data Visualization Library. Luettavissa: https://medium.com/@Elijah_Meeks/d3-is-not-a-data-visualization-library-67ba549e8520. Luettu: 9.11.2020.

Mozilla 2020a. MDN web docs. HTML: HyperText Markup Language. Luettavissa: https://developer.mozilla.org/fi/docs/Web/HTML. Luettu: 23.11.2020.

Mozilla 2020b. MDN web docs. XML: Extensible Markup Language. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/XML. Luettu: 23.11.2020.

Mozilla 2020c. MDN web docs. CSS: Cascading Style Sheets. Luettavissa: https://developer.mozilla.org/fi/docs/Web/CSS. Luettu: 23.11.2020.

Mozilla 2020d. MDN web docs. SVG: Scalable Vector Graphics. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/SVG. Luettu: 23.11.2020.

Mozilla 2020e. MDN web docs. Document Object Model (DOM). Luettavissa: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model. Luettu: 23.11.2020.

Mozilla 2020f. MDN web docs. JavaScript. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Luettu: 23.11.2020.

Norton, K. 2020. Data Definition & Meaning | What is Data?. Luettavissa: https://www.webopedia.com/TERM/D/data.html. Luettu: 15.11.2020.

The Office of Research Integrity 2020. Data Analysis. Luettavissa: https://ori.hhs.gov/education/products/n_illinois_u/datamanagement/datopic.html. Luettu: 21.12.2020.

Require.js 2020. Why AMD? Luettavissa: https://requirejs.org/docs/whyamd.html. Luettu: 13.01.2021.

Roth, S, Zec, M & Matthes, F. 2014. Enterprise Architecture Visualization Tool Survey 2014. Luettavissa: https://wwwmatthes.in.tum.de/pages/o790x7rho1te/EAVTS-2014-Final-Report. Luettu: 25.1.2021.

Saranya, K. 22.7.2019. Data Visualization: Importance and Benefits. Luettavissa: https://www.boldbi.com/blog/data-visualization-importance-and-benefits. Luettu: 29.11.2020.

Satakieli, T. 7.10.2014. Kokonaisarkkitehtuuri, mitä se on selkokielellä? Luettavissa: https://www.alfame.com/blog/kokonaisarkkitehtuuri-mita-se-on-selkokielella. Luettu: 23.11.2020.

SciForce 2019. Best Data Visualization Techniques for small and large data. Luettavissa: https://www.kdnuggets.com/2019/04/best-data-visualization-techniques.html. Luettu: 1.11.2020.

Sweeney, P. 16.12.2018. Why I no longer use D3.js. Luettavissa: https://medium.com/@PepsRyuu/why-i-no-longer-use-d3-js-b8288f306c9a. Luettu: 9.11.2020.

Tableau 2020. Data visualization beginner's guide: a definition, examples, and learning resources. Luettavissa: https://www.tableau.com/learn/articles/data-visualization. Luettu: 23.11.2020.

Thalmann, M, Souza, A & Oberauer, K. 2019. How Does Chunking Help Working Memory? Luettavissa: https://www.zora.uzh.ch/id/eprint/151291/1/Thalmann.et.al.Chunking.final.pdf. Luettu: 14.3.2021.

Toivonen, H, Salmenkivi, M & Verkamo, I. 2003. Tutkimustiedonhallinnan peruskurssi - Data-analyysi: johdanto. Luettavissa: https://www.cs.helsinki.fi/u/htoivone/teaching/tutihaK03/slides2b.pdf. Luettu: 21.12.2020.

Toucan Toco 2020. 7 Examples of Data Visualization. Luettavissa: https://toucantoco.com/blog/en/7-examples-of-data-visualization/. Luettu: 23.11.2020.

Trafton, A. 16.1.2014. MIT News Office. In the blink of an eye. Luettavissa: https://news.mit.edu/2014/in-the-blink-of-an-eye-0116. Luettu: 14.3.2021.

Tuominen, M. 12.02.2020. Tiedonhallintamallin toteutus ARC-ohjelmistolla. Luettavissa: https://www.arter.fi/tiedonhallintamallin-toteutus-arc-ohjelmistolla/. Luettu: 22.11.2020.

TutorialsTeacher 2020. What is D3? Luettavissa: https://www.tutorialsteacher.com/d3js/what-is-d3js. Luettu: 15.11.2020.

TutorialsPoint 2020. What is the Difference Between SVG and HTML5 Canvas? Luettavissa: https://www.tutorialspoint.com/What-is-the-difference-between-SVG-and-HTML5-Canvas. Luettu: 15.11.2020.

Vaisala 2019. Observations for a Better World. Vuosiraportti 2019. Luettavissa: https://www.vaisala.com/sites/default/files/documents/Vaisala_vuosiraportti_2019_web.pdf. Luettu: 25.1.2021.

Viau, C. 15.2.2017. D3 is Now Modular. Luettavissa: https://medium.com/@christopheviau/d3-js-modularity-d5eed78ba06e. Luettu: 15.11.2020.

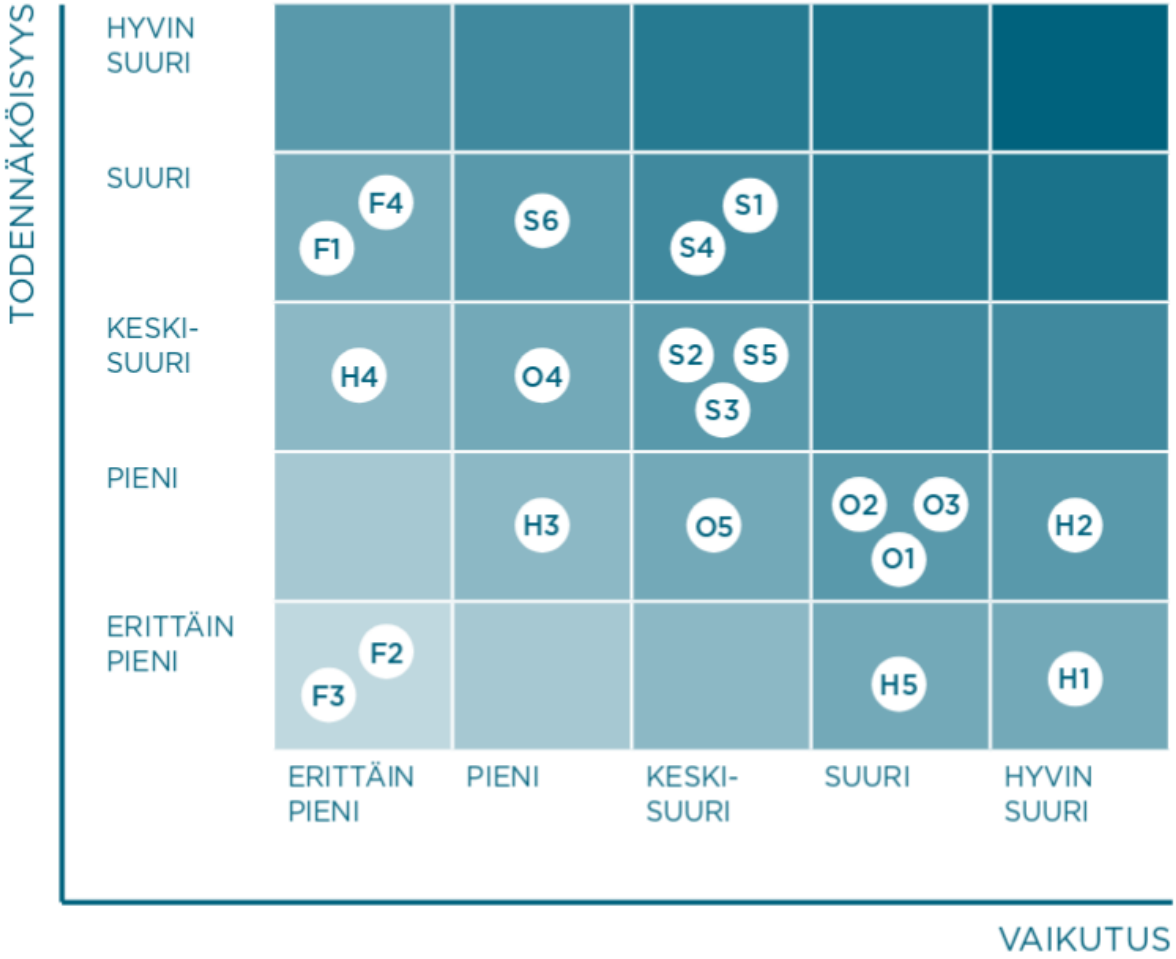
Wallenius, N. 6.1.2019. Kokonaisarkkitehtuuri - kaikki mitä aiheesta tarvitsee tietää. IT-johtaminen. Luettavissa: https://niklaswallenius.fi/kokonaisarkkitehtuuri-taydellinen-opas/#luku_4. Luettu: 15.11.2020.

Wellesley College 2020. CS 249. D3.js - An introduction. Luettavissa: http://cs.wellesley.edu/~mashups/pages/am5d3p1.html. Luettu: 15.11.2020.

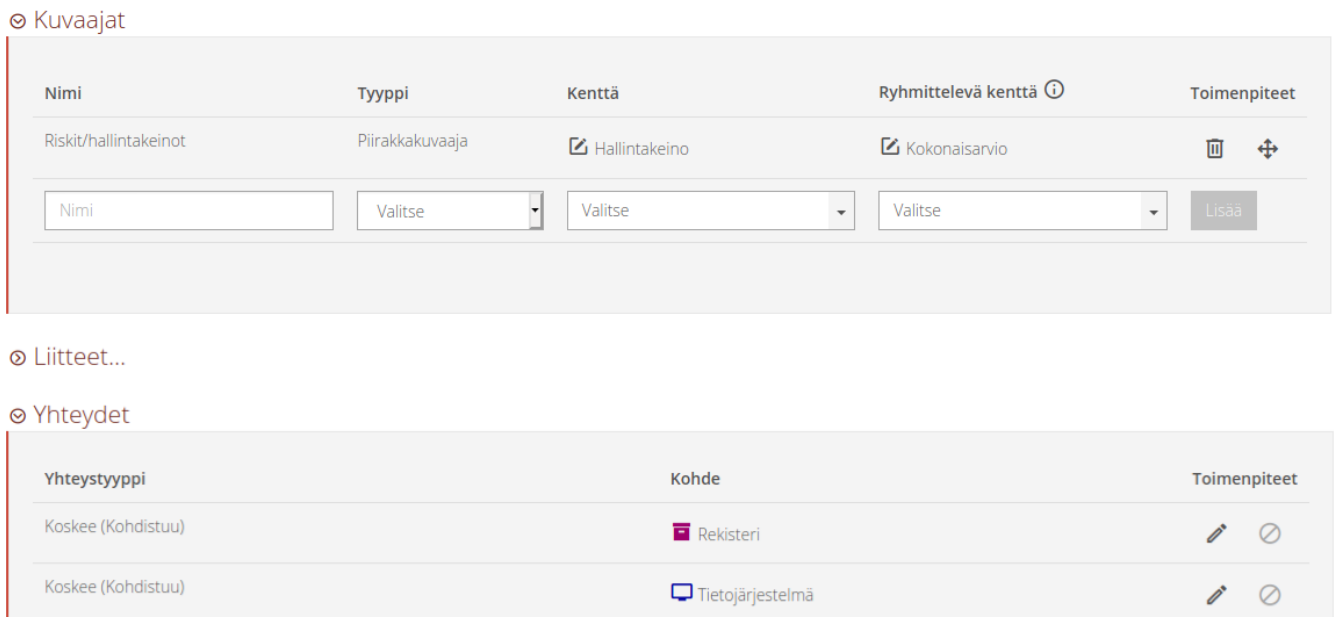
Yle. 3.11.2020. Vastaamon tietomurto aiheutti vyöryn: viikossa tehty yli 10 000 rekisterikieltopyyntöä, tavallisesti koko vuonna alle 300. Luettavissa: https://yle.fi/uutiset/3-11628308. Luettu: 22.11.2020.

Liitteet

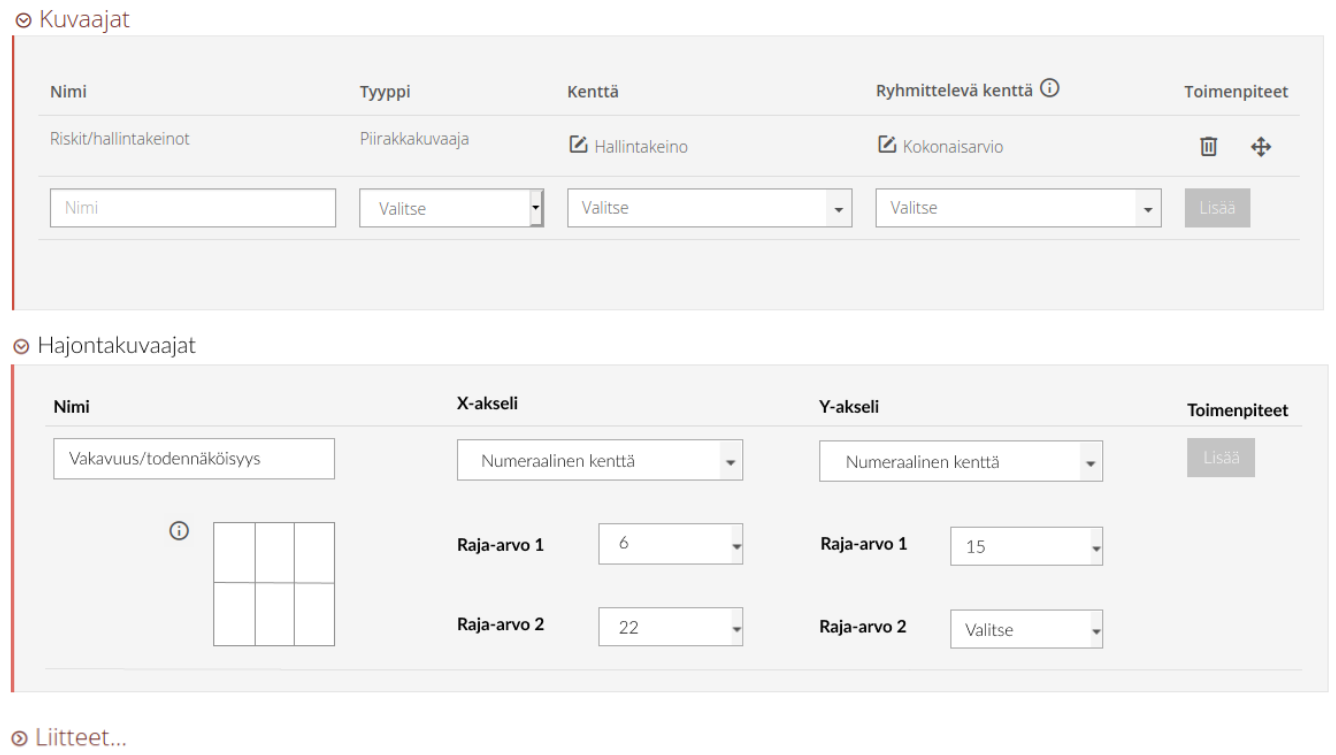
Liite 1. Vaisala-yhtiön vuoden 2019 vuosikertomuksesta poimittu matriisityyppinen kaavio, jonka tummemmat taustavärit vastaavat todennäköisyydeltään ja vaikutuksiltaan vakavampia riskialueita. (Vaisala 2019.)



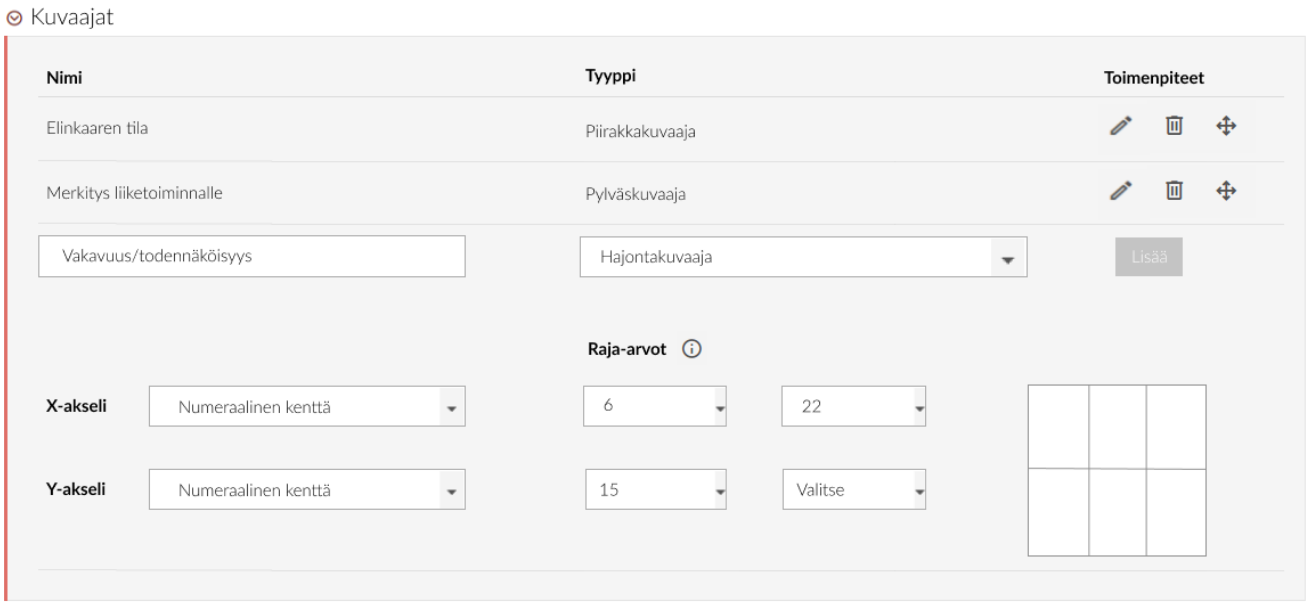
Liite 2. Alkuperäinen ARCin kuvaajien lisäysnäkymä.



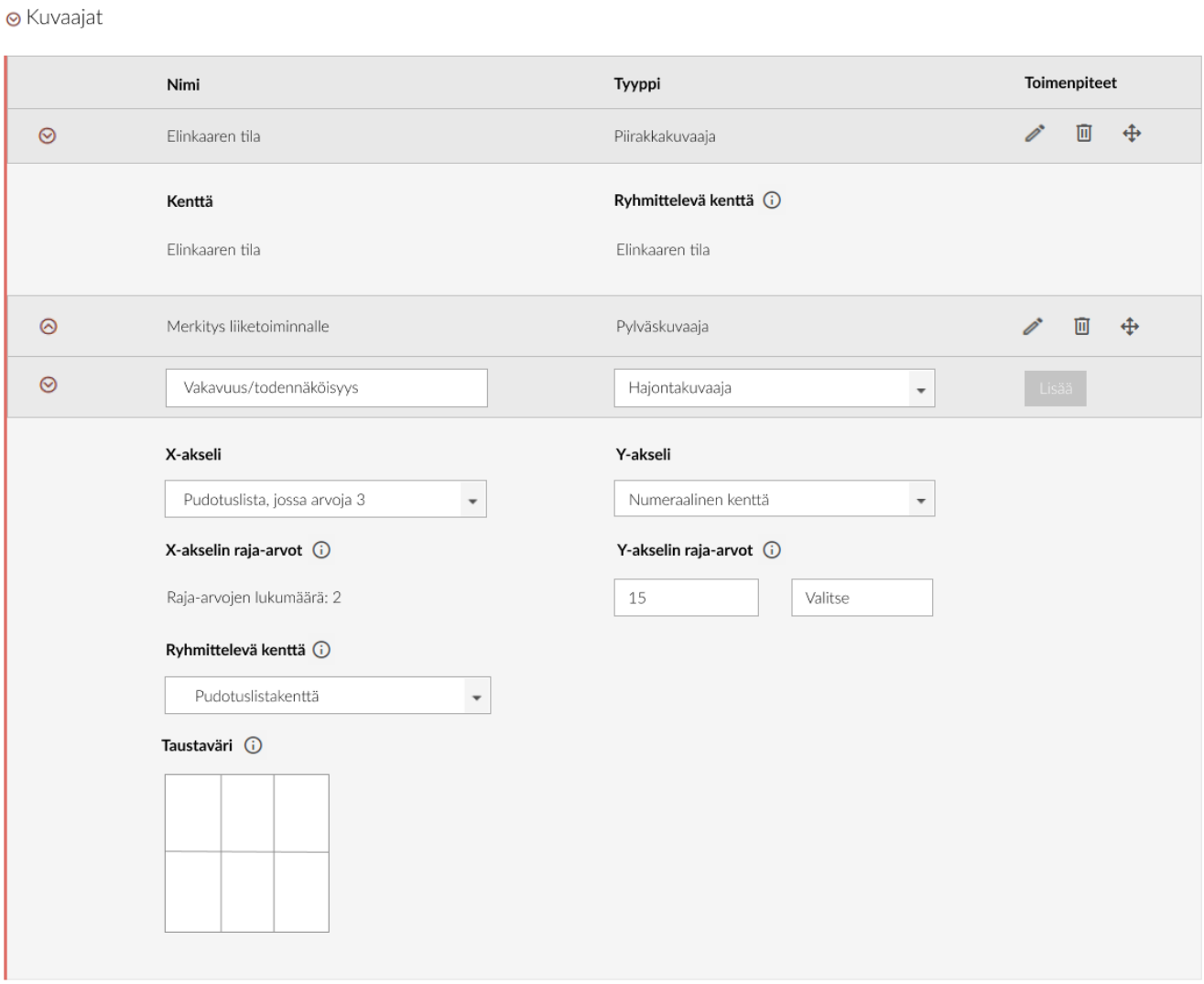
Liite 3. Ensimmäinen uuden kaavioiden lisäysnäkymän hahmotelmista, joka hyödyntää kokonaan erillistä hajontakuvaajien lisäämisnäkymää.



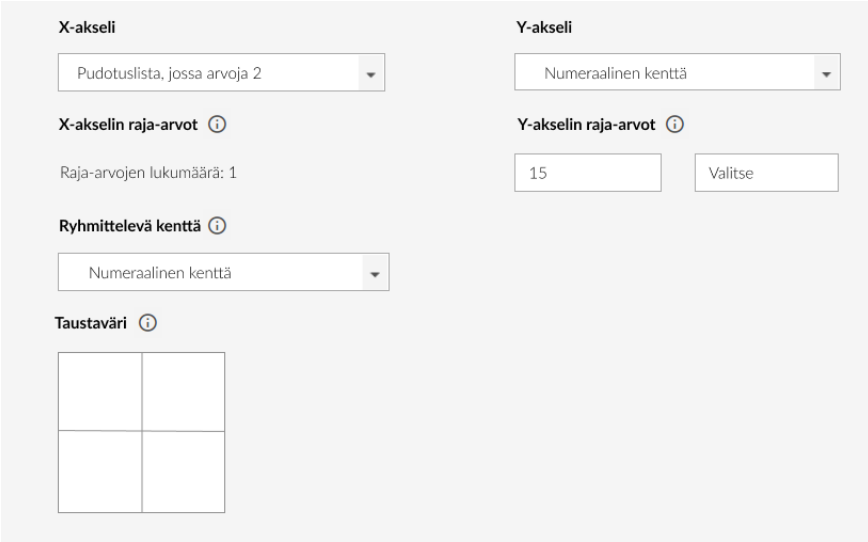
Liite 4. Toinen hahmotelma, jossa ongelmana on otsikoiden ja taustavärivalitsimen asemointi.

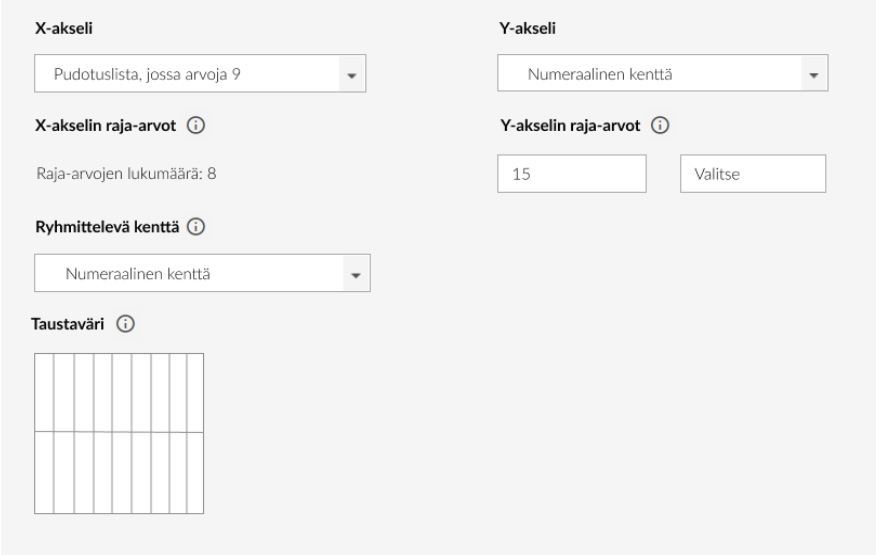


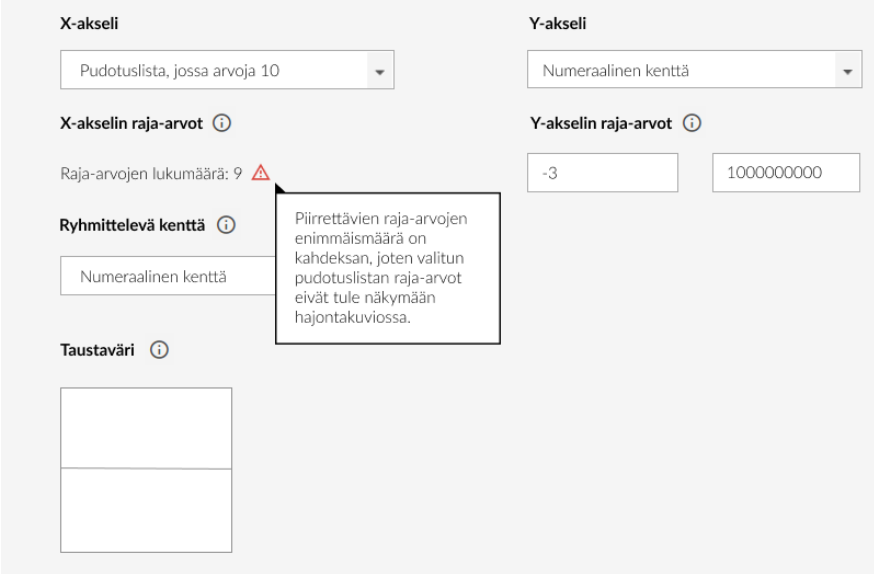
Liite 5. Lopullinen haitarimallinen ratkaisu.



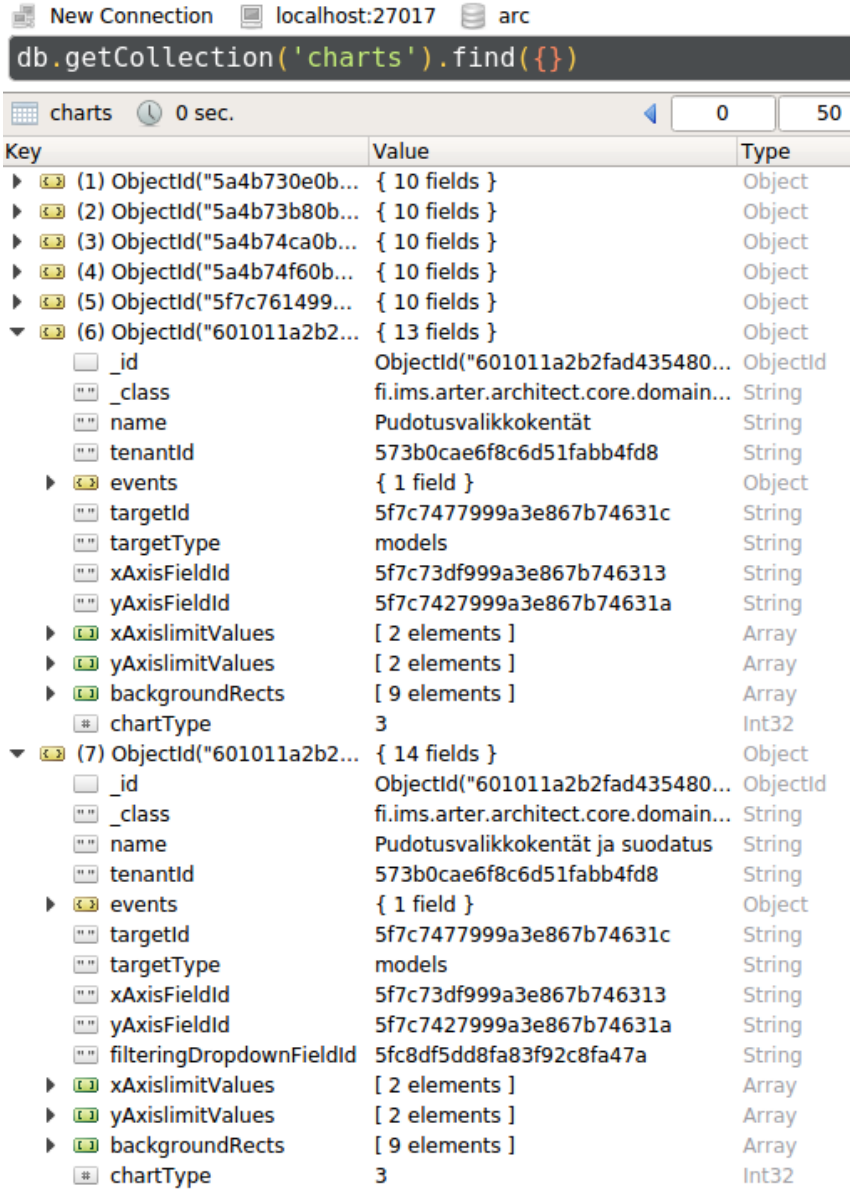
Liite 6. Raja-arvojen esikatselulaatikon/taustavärivalitsimen toimintaa havainnollistavia kuvia.







Liite 7. Kahden uuden matriisityyppisen kaavion tiedot tietokannassa.



Liite 8. Backbone-mallin välittäminen chart.js-tiedostosta scatterChart.js-tiedostoon.

```
if (this.chartType === Common.CHART_TYPES.BAR_CHART.value) {
    this.handleDrawBar();
} else if (this.chartType === Common.CHART_TYPES.PIE_CHART.value) {
    this.handleDrawPie();
} else { // this.chartType === Common.CHART_TYPES.SCATTER_CHART.value
    this.scatterChart = new ScatterChart ({
        model: this.model,
        target: this.$el,
        parent: this,
        renderLocation: this.$el[0]
    });
}
```

Liite 9. ScatterChart.js-tiedoston handleFilterData-funktio.

```
handleFilterData: function () {
   this.filterCategories = []
   if(this.model.get('filteringDropdownField')){
      _.each(this.model.get('filteringDropdownField')
            .dropdownItemContents, function(fieldContent) {
         var filterData = {
            fieldContentId: fieldContent.id,
            fieldContentName: fieldContent.name,
            icon: {
               className: fieldContent.icon.className,
               code: fieldContent.icon.code,
               color: fieldContent.icon.color
            },
         };
         this.filterCategories.push(filterData);
      }.bind(this))
      this.filterCategories.push({ fieldContentName:
            this.noCategoryText })
   }
},
```

Liite 10. handleElementData-funktion käsittelemän elementin rakenne.

```
var data = {
   elementId: element.id,
   elementName: element.get('listingName'),
   icon: {
      className: element.get('icon').get('className'),
      code: element.get('icon').get('code'),
      color: element.get('icon').get('color')
   },
   xValue: elementXValue,
   xDropdownValue: elementXDropdownValue,
   yValue: elementYValue,
   yDropdownValue: elementYDropdownValue,
   filterCategory: elementCategory
};
this.data.push(data);
```

Liite 11. Taustavärisuorakuutioiden lisäämisfunktio.

```
appendBackgroundRects: function () {
      this.scatter.selectAll("bgRect")
          .data(this.backgroundRects)
          .enter()
          .append("rect")
          .attr("id", "bgRect")
          .attr("clip-path", "url(#clip)")
          .attr("x", function (d) { return this.x(d.x1); }.bind(this))
          .attr("y", function (d) { return this.y(d.y2); }.bind(this))
          .attr("width", function (d) { return this.x(d.x2) -
                              this.x(d.x1) }.bind(this) )
          .attr("height", function (d) { return this.y(d.y1) -
                              this.y(d.y2); }.bind(this) )
          .style("fill-opacity", 0.15)
          .attr("fill", function (d) { return d.color }.bind(this) )
},
```

Liite 12. Ikonimuotoisten datapisteiden lisääminen.

```
      appendDatapoints: function () {
         this.datapoints = this.scatter.selectAll("datapoint")
            .data(this.data)
            .enter()
            .append('text')
            .style('font-size', '2.0em')
            .attr("dy", "0.5em")
            .attr('text-anchor', 'middle')
            .style('font-family', 'Material Icons')
            .attr("clip-path", "url(#clip)")
            .attr("class", function (d) {
               if (this.filterCategories.length > 0) {
                  return d.filterCategory.fieldContentName
               } else {
                  return "datapoint"
               }
            }.bind(this))
            .attr("x", function (d) { return
                              this.x(d.xValue) }.bind(this))
            .attr("y", function (d) { return
                              this.y(d.yValue) }.bind(this))
            .style("fill", "#4d4d4d")
            .style("display", "inline")
            .on('mouseover', this.mouseover.bind(this))
            .on('mouseout', this.mouseout.bind(this))
```

```
            .on('mousemove', this.formatDatapointTooltip.bind(this))
            .text(function (d) {
               if (d.icon && d.icon.code !== '') {

                  ...

                  if(d.icon.code === 'f757') {
                     d.icon.code = 'f72a' // new default element icon
                  }
                  if (d.filterCategory.fieldContentName !==
                    this.noCategoryText) {

                     return
icomoonToMDIconCodes.findCorrectIconCodeAsCharCodeFor(d.filterCategory.
icon.code)
                  } else {
                     return
icomoonToMDIconCodes.findCorrectIconCodeAsCharCodeFor(d.icon.code)
                  }
               } else {
                  return '\uf72a' // default element icon
               }
            }.bind(this))
      },
```

Liite 13. Kolumnimuotoinen flexbox-elementti.

```
this.optionsColumn = this.contentsRow.append("div")
            .attr("class", "options")
            .style("display", "flex")
            .style("flex-direction", "column-reverse")
```

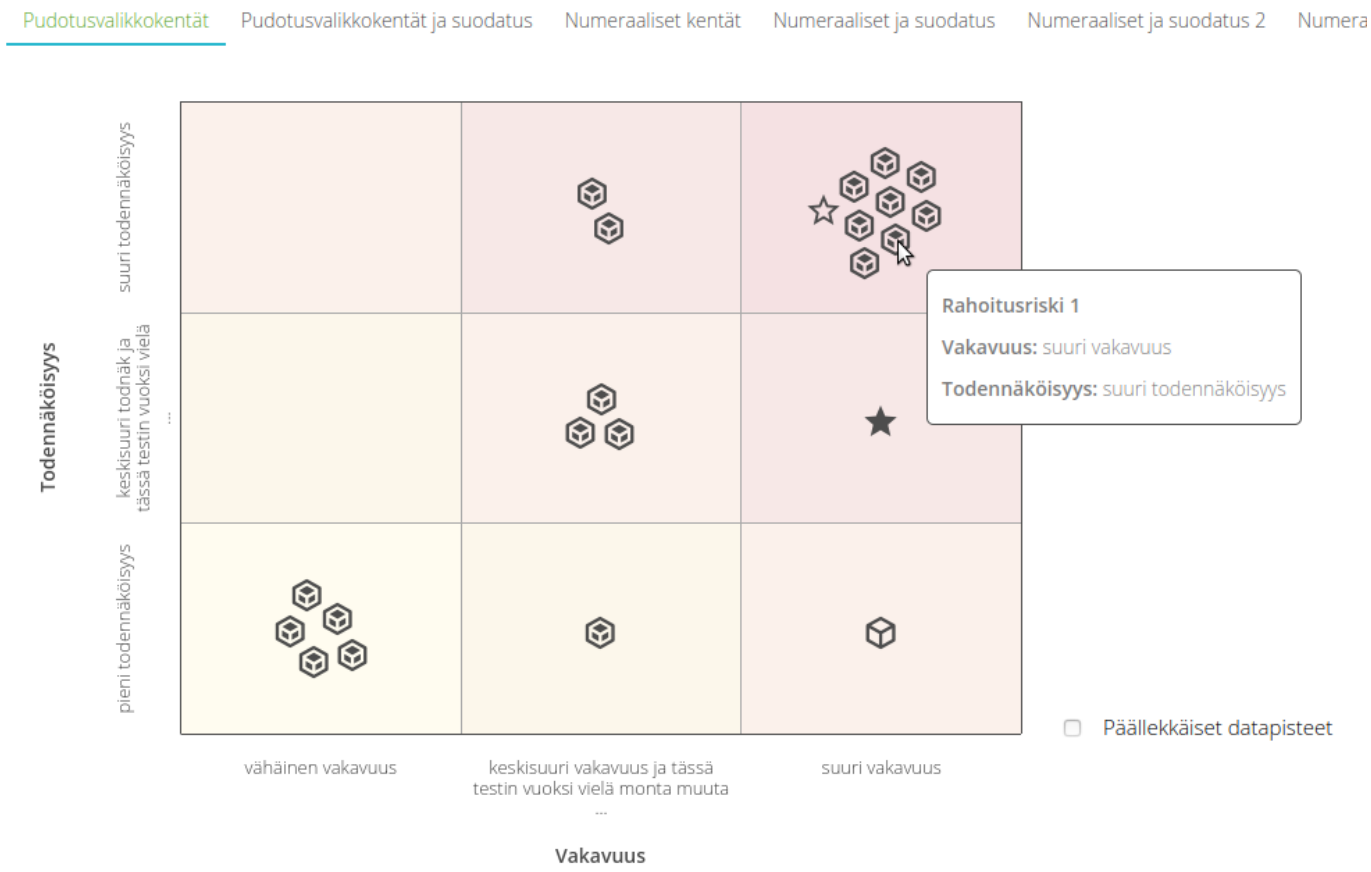
Liite 14. Raja-arvojen piirtämistyylin muutos.

```
         this.scatter.selectAll('.grid')
            .selectAll('line')
            .attr('stroke', 'darkgrey')
            .style('stroke-width', 1)
            .style("shape-rendering", "crispEdges")
```

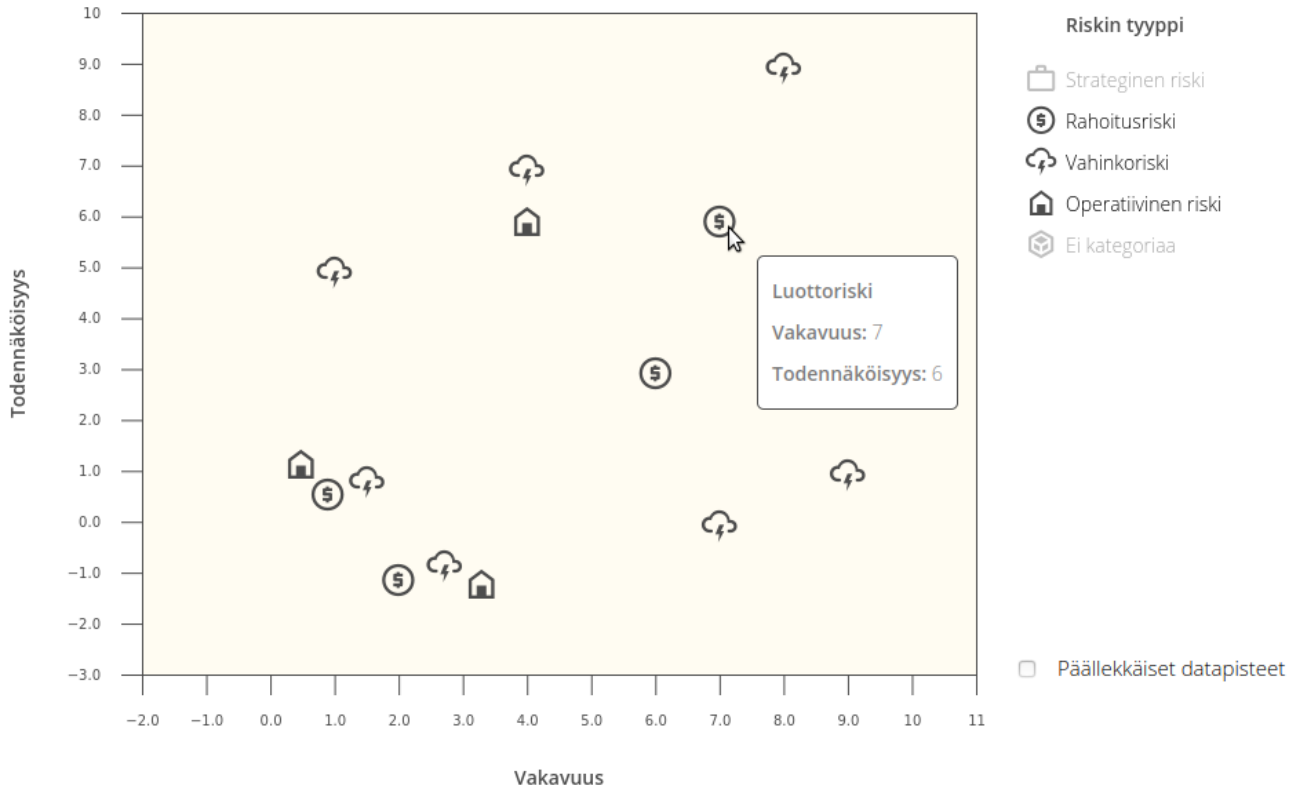
Liite 15. Suodatuskategorioiden nimien asemointi transform-attribuutilla.

```
         this.category.append('text')
            .attr("class", function(d) { return d.fieldContentName })
            .attr('transform', 'translate(' + 30 + ',' + 20 + ')')
            .text(function(d) { return d.fieldContentName });
```

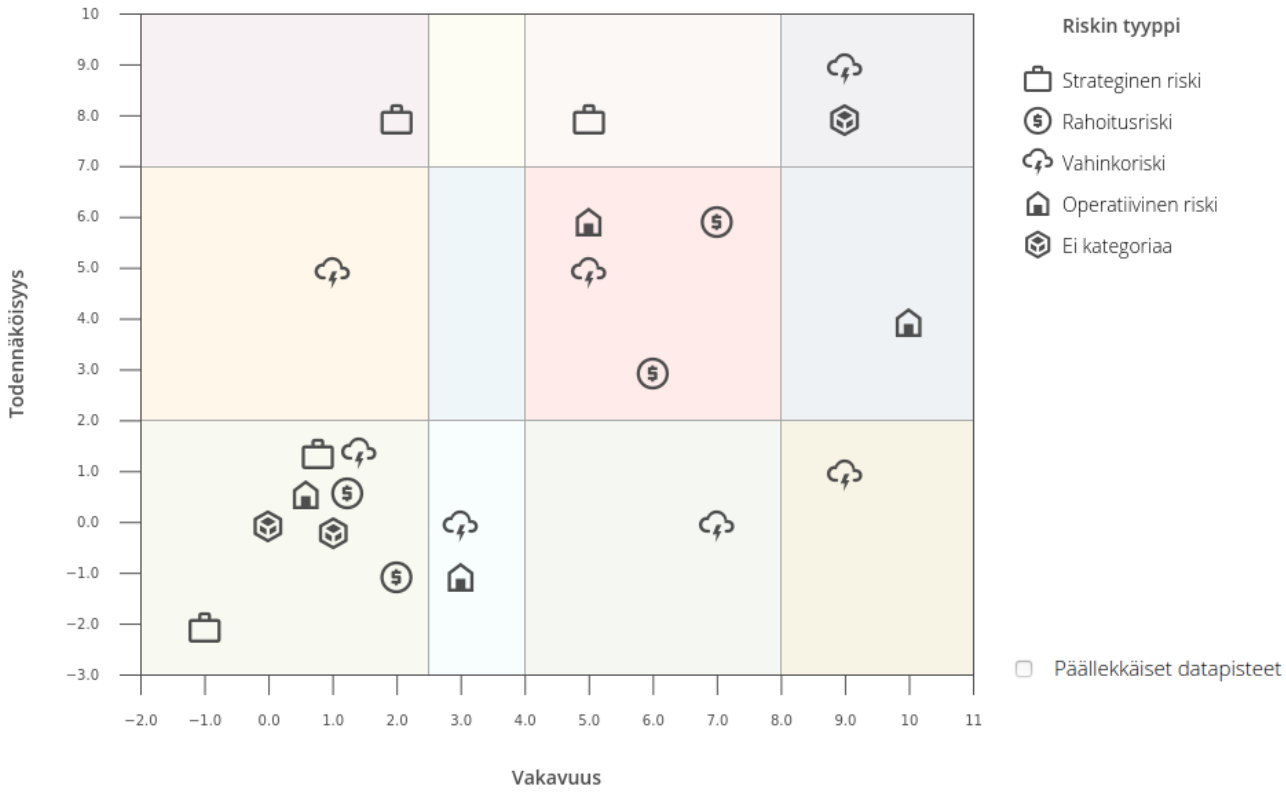
Liite 16. Riskikaavio ilman suodatusta. Käytössä elementtien omat ikonit.



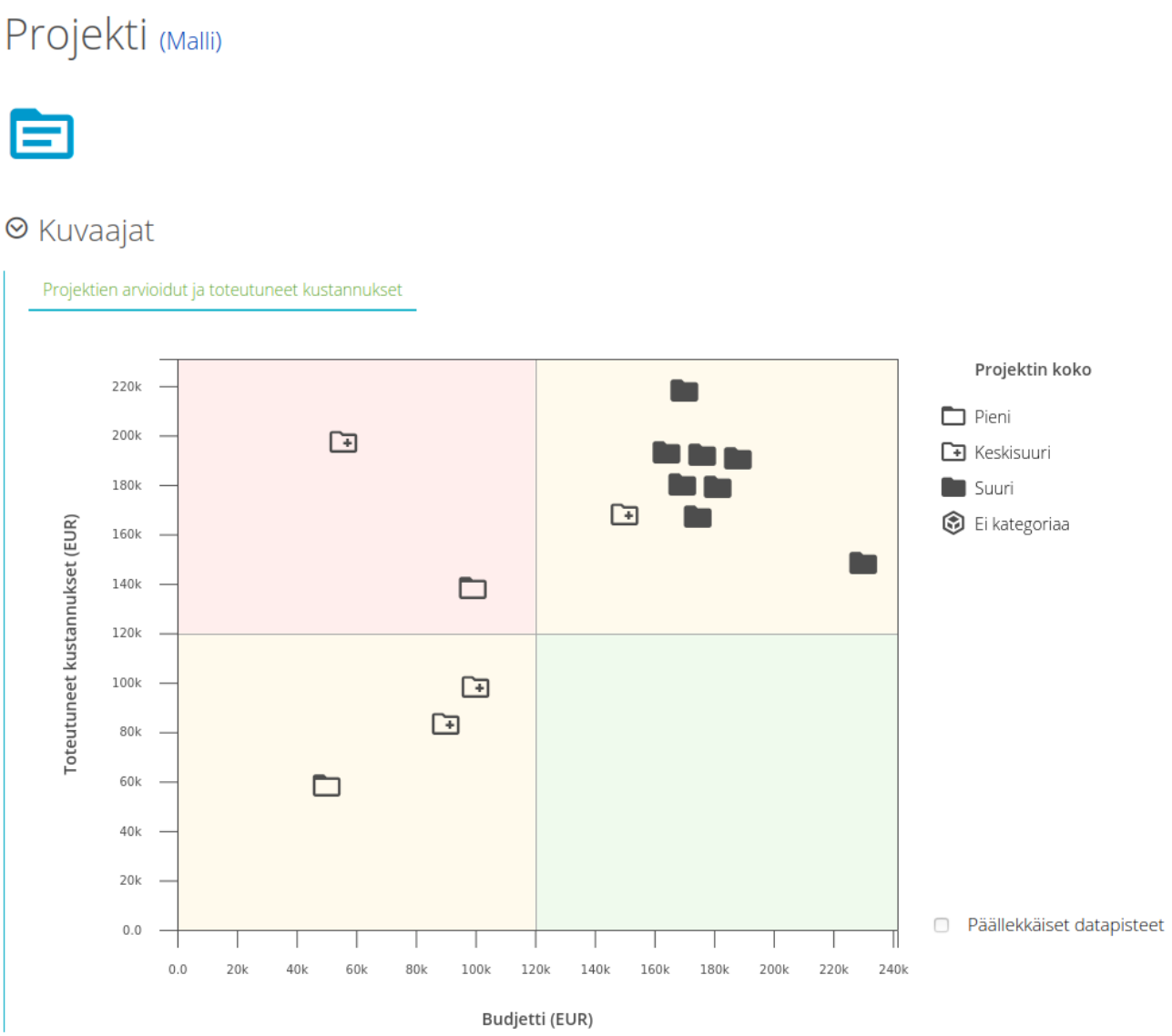
Liite 17. Suodatettu negatiivisia arvoja sisältävä numeraalinen riskikaavio ilman raja-arvoja.



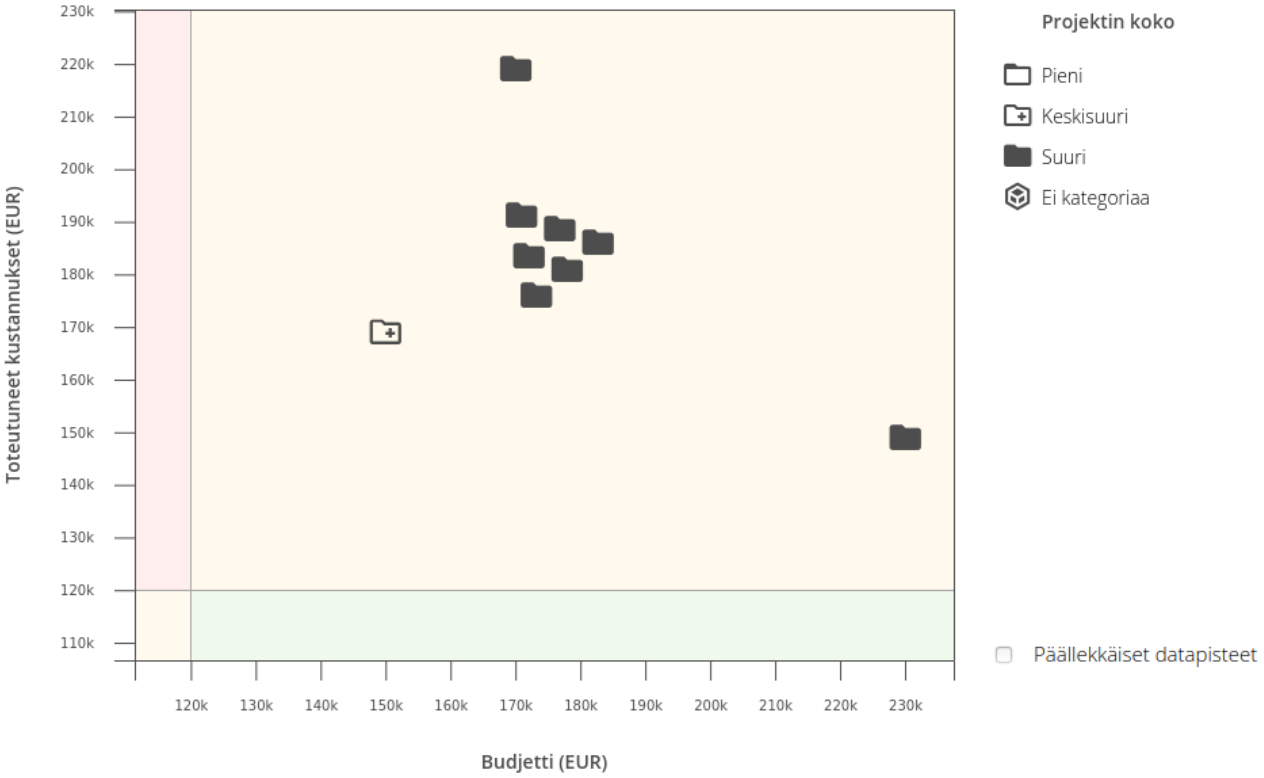
Liite 18. Raja-arvot vaihtelevilla etäisyyksillä.



Liite 19. Projekti-mallin elementtejä kuvaava kaavio numeraalisilla akselikentillä. Akseleiden pitkät tick-nimikkeet on formatoitu lyhyempään muotoon. Akselikenttien nimiin on liitetty myös akselikentille asetetut yksiköt (EUR).



Liite 20. Liitteen 19 kaavio lähennettynä.



Liite 21. Liitteessä 20 esitetyt datapisteet todellisilla paikoillaan.