TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TOMMI KANERVO

WEB-POHJAINEN KYSELYTYÖKALU JA SEN ARVIOINTI

Diplomityö

Tarkastaja: professori Kari Systä
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunta-neuvoston kokouksessa 6. huhtikuuta 2016

TIIVISTELMÄ

TOMMI KANERVO: Web-pohjainen kyselytyökalu ja sen arviointi
Tampereen teknillinen yliopisto
Diplomityö, 88 sivua, 18 liitesivua
Huhtikuu 2017
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Ohjelmistotuotanto
Tarkastaja: professori Kari Systä

Avainsanat: kyselytyökalu, web-sovellus, pitkän aikavälin käyttäjäkokemus

Retrospektiivisessä pitkän aikavälin käyttäjäkokemustutkimuksessa ollaan kiinnostuneita tuotteen tai palvelun käyttöön liittyvistä muistikuvista. Eräs tapa kerätä retrospektiivistä käyttäjäkokemustietoa on käyränpiirtotehtävä, jossa vastaaja piirtää käyttäjäkokemuksensa laadusta kuvaajan. Tässä työssä jatkokehitettiin ja arvioitiin käyränpiirtotehtäviin perustuvaa web-kyselytyökalua, jolla voi toteuttaa retrospektiivisiä pitkän aikavälin käyttäjäkokemuksen etätutkimuksia.

Jatkokehitykseen kuului lukuisia ominaisuuksia ja näkymiä, joista tärkeimmät ovat vastausten tarkasteluun kehitetty näkymä, mobiilivastausnäkymä ja pitkittäistutkimuksia tukeva kyselytyyppi. Järjestelmä kehitettiin PHP-kielellä CodeIgniter-kehyksen avulla. Kuvaajat toteutettiin Highcharts-kirjastolla.

Työssä myös toteutettiin järjestelmän nykyisen toteutuksen arviointi. Arvioinnissa toteutusta verrattiin järjestelmälle asetettuihin vaatimuksiin, joista rakennettiin laatukriteeripuu. Arvioinnissa huomioitiin järjestelmän sopivuus käyttöön, tehokkuus ja ylläpidettävyys. Arvioinnin pohjalta järjestelmän havaittiin olevan ominaisuusvalikoimaltaan kattava, mutta joidenkin näkymien käytettävyydessä on vielä puutteita. Teknologiavalinnat ja perusarkkitehtuuri todettiin onnistuneiksi. Eräästä näkymästä löytyi merkittävä tehokkuusongelma. Ylläpidettävyysarvioinnissa kooditiedostojen pituus ja rakenne havaittiin ongelmalliseksi. Tehokkuuteen ja ylläpidettävyyteen vaikuttivat negatiivisesti myös eräät monimutkaiset syötteenkäsittely- ja kommunikointisekvenssit. Lisäksi kehitysprosessissa itsessään oli puutteita erityisesti testauksen ja dokumentoinnin suhteen.

Työssä saatiin järjestelmään onnistuneesti kehitettyä lähes kaikki projektin aikana toteutettaviksi suunnitellut ominaisuudet. Lisäksi arvioinnissa saatiin kattava käsitys toteutuksen vahvuuksista ja ongelmakohdista. Lopuksi muodostettiin lista parannus- ja jatkokehityssuosituksista. Suosituksista olennaisimmat ovat järjestelmän kattava testaaminen, tulosten tarkastelunäkymän suorituskyvyn parantaminen käyräobjektien laiskalla alustamisella, syötteenkäsittelyn erottaminen omaksi kerroksekseen ja joidenkin epäideaalien sekvenssien uudelleensuunnittelu.

ABSTRACT

TOMMI KANERVO: Web-based survey tool and its evaluation
Tampere University of Technology
Master of Science Thesis, 88 pages, 18 Appendix pages
April 2017
Master's Degree Programme in Information Technology
Major: Software Engineering
Examiner: Professor Kari Systä

Keywords: survey tool, web application, long term UX

Retrospective long term user experience research is interested in memories connected to the use of a product or a service. One method for collecting retrospective user experience data is a curve drawing task, in which the respondent draws a graph of the quality of their user experience. This thesis project was about the follow-up development and quality evaluation of a web survey tool based on curve drawing tasks and intended for use in remote retrospective long term user experience studies.

The follow-up development consisted of numerous features and views, the most important of which are a page for viewing all the response data, a mobile version of the response page, and a new type of survey for longitudinal studies. The system was developed in PHP with the CodeIgniter framework. The charts were implemented using the Highcharts library.

This thesis also includes a quality evaluation of the system's current state. The system's implementation was compared with its requirements, which were used to form a quality criterion tree. The evaluation focused on the system's suitability for use, performance, and maintainability. Based on the evaluation, the system was perceived to have a wide range of features, but the usability of some views still needs improvement. The choices of technologies and basic architecture of the system were found successful. A significant performance problem was found in one of the views. The maintainability evaluation found the length and structure of the code files problematic. Complicated input handling and communication sequences affected the performance and maintainability in a negative way. The development project itself had flaws as well, especially related to testing and documentation.

The project was successful in that nearly all the features planned for this development phase were implemented. Additionally, the evaluation provided a comprehensive understanding of the strengths and problems of the system's current state. Finally, a list of improvement recommendations was formed. The most important suggestions were comprehensive testing of the system, improving the performance of the response data view by lazy initialization of the curve objects, separating input handling as its own layer, and redesigning some nonideal sequences.

ALKUSANAT

Tämä työ on tehty Tampereen teknillisen yliopiston tietotekniikan laitokselle osana diplomi-insinöörin tutkintoani. Ajatus työhön syntyi jo kesällä 2014, kun aloin työskennellä CCD MobiLe -projektissa DrawUX-järjestelmän parissa. Haluan kiittää projektissa työnantajana toiminutta Ihmiskeskeisen teknologian yksikköä ja erityisesti projektipäällikkö Jari Varsaluomaa aktiivisesta tuesta jatkokehitysprojektissa ja sen jälkeen. Kiitän myös diplomityön ohjannutta ja tarkastanutta professori Kari Systää avusta ja palautteesta.

Tampereella, 21.3.2017

Tommi Kanervo

SISÄLLYSLUETTELO

LYHENTEET JA TERMIT

Ajax	Asynchronous JavaScript And XML, joukko selainpuolen teknologioita asynkronisten web-sovellusten kehittämiseen
CIT	Critical Incident Technique, haastattelumenetelmä, jossa vastaaja raportoi kriittisimpiä kokemuksiaan
CodeIgniter	PHP-pohjainen websovelluskehys
CORPUS	Change Oriented analysis of the Relationship between Product and USer, haastattelutekniikka, jossa vastaaja vertaa nykyisiä mielipiteitään tuotteesta käytön alkuun
CSS	Cascading Style Sheets, tyylimerkkauskieli web-sivun ulkoasun määrittämiseen
DOM	Document Object Model, ohjelmointirajapinta mm. HTML-dokumenttien käsittelyyn
DRM	Day Reconstruction Method, pitkittäistutkimukseen soveltuva itseraportointimenetelmä, jossa raportoidaan koko päivän kokemukset kerralla
ESM	Experience Sampling Method, kenttätutkimusmenetelmä, jossa tilanne keskeytetään haastatteluja varten
HCI	Human–computer interaction, ihmisten ja tietokoneiden välistä vuorovaikutusta tutkiva tieteenala
Highcharts	kirjasto interaktiivisten diagrammien esittämiseen selaimessa
HTML	HyperText Markup Language, merkkauskieli web-sivujen luomiseen
HTTP	Hypertext Transfer Protocol, web-järjestelmien käyttämä tiedonsiirtoprotokolla
JavaScript	yleensä web-ympäristössä käytetty komentosarjakieli, jonka avulla web-sivulle voi toteuttaa dynaamista toiminnallisuutta
jQuery	suosittu JavaScript-kirjasto/selainkehys, joka tarjoaa helpomman syntaksin mm. DOMin käsittelyyn ja Ajax-kommunikointiin
JSON	JavaScript Object Notation, yksinkertainen tiedostoformaatti tiedonvälitykseen
MVC	arkkitehtuurimalli, jossa sovellusaluelogiikka, tietokantaoperaatiot ja käyttöliittymä erotetaan toisistaan
PHP	PHP: Hypertext Preprocessor, usein web-palvelinohjelmoinnissa käytetty komentosarjakieli
PostgreSQL	Avoimen lähdekoodin tietokannan hallintajärjestelmä
SQL	Structured Query Language, standardoitu kyselykieli relaatiotietokannoille
SVG	Scalable Vector Graphics, graafisten objektien kuvauskieli, jonka avulla voidaan piirtää vektorikuvia
URL	Uniform Resource Locator, verkkosivun osoite
WebQEM	Web Quality Evaluation Method, kvantitatiivinen asiantuntijalähtöinen web-sivustojen laadunarviointistrategia
XML	Extensible Markup Language, jäsennettyjen dokumenttien merkintäkieli
XMLHttpRequest	Oliomuotoinen ohjelmointirajapinta selain-palvelin-tiedonsiirtoon

1. JOHDANTO

DrawUX on Tampereen teknillisen yliopiston ihmiskeskeisen teknologian yksikössä kehitetty verkkopohjainen kyselytyökalu, joka on tarkoitettu ensisijaisesti pitkän aikavälin käyttäjäkokemuksen tutkimiseen [1]. DrawUX:n kotisivu kirjoitushetkellä on https://drawux.cs.tut.fi/. Työkalun tärkeimmät käyttäjäryhmät ovat tutkijat, jotka voivat luoda työkalulla kyselyjä ja tarkastella kyselyihin saatuja vastauksia, sekä vastaajat, jotka voivat vastata kyselyyn henkilökohtaisen linkin kautta tai anonyymisti.

Pitkän aikavälin käyttäjäkokemuksen tutkimuksessa tarkastellaan käyttäjäkokemuksen kehittymistä pidemmällä aikavälillä kuin tavallisissa käyttäjäkokemus- ja käytettävyystutkimuksissa, joissa huomio on yleensä vain käytön alussa. Kuitenkin pidemmän aikavälin käyttäjäkokemus vaikuttaa merkittävästi käyttäjän jatkokäyttäytymiseen, joten sen tutkimiselle ja uusille tutkimustyökaluille on tarvetta. Pitkän aikavälin käyttäjäkokemusta voidaan tutkia esimerkiksi poikkileikkaustutkimuksella, jossa tarkastellaan kerralla eritasoisia käyttäjiä ja oletetaan kokemuksista löytyvien erojen johtuvan eroista kokemuksen pituudessa, tai pitkittäistutkimuksella, jossa samoja käyttäjiä tarkastellaan useina ajanhetkinä. DrawUX:ssa on otettu kolmas näkökulma, *retrospektiivinen* tutkimus, jossa käyttäjää pyydetään muistelemaan ja kuvaamaan kokemustensa kehittymistä ajan mittaan. Retrospektiiviset tutkimukset, kuten pitkän aikavälin käyttäjätutkimukset ylipäänsä, ovat vielä melko harvinaisia, mutta joitakin työkaluja retrospektiivisiä käyttäjäkokemustutkimuksia varten on kuitenkin kehitetty. Esimerkiksi CIT-pohjaisissa menetelmissä vastaajaa pyydetään kertomaan tarkemmin kriittisimmistä käyttökokemuksistaan [2], ja CORPUS-haastattelumenetelmässä vastaaja vertaa nykyistä kokemustaan ostohetken jälkeiseen kokemukseen ja selittää mahdolliset muutokset [3]. DrawUX:ssa tutkimuksen pohjaksi on valittu piirtotehtävä, jossa vastaaja piirtää kokemuksensa laatua kuvaavan käyrän kuvaajalle, jonka x-akseli kuvaa aikaa ja y-akseli jotakin laatuattribuuttia, kuten tuotteen miellyttävyyttä tai käytön helppoutta. DrawUX on kehitetty erityisesti kahden piirtotehtäviä hyödyntävän aiemman työkalun pohjalta. Nämä työkalut ovat paperipohjainen haastattelutilanteisiin tarkoitettu UX Curve [4] sekä kahden eri muistiteorian perusteella kehitetty, tietokoneella käytettävä iScale [5].

Tämän työn ensimmäinen osa on DrawUX-järjestelmän jatkokehittäminen osana Tampereen teknillisen yliopiston CCD MobiLe -projektia. Jatkokehitysprojektin tavoitteena oli parantaa ohjelmiston sopivuutta käyttöön lisäämällä ominaisuuksia, joista on hyötyä työkalua käyttäville tutkijoille ja vastaajille, pitäen samalla myös muut laatuominaisuudet, kuten käytettävyyden, tehokkuuden ja ylläpidettävyyden, mahdollisimman hyvällä tasolla. Järjestelmän kehitykseen kuului perinteiseen web-ohjelmointiin kuuluvia osa-alueita, kuten selainohjelmointia, palvelinohjelmointia ja tietokannanhallintaa. Projektin

alkaessa DrawUX sisälsi jo perustoiminnallisuuden, kuten kyselyeditorin ja vastausnäkymän. Tämän jatkokehitysprojektin aikana näkymiin lisättiin runsaasti uusia toimintoja, kuten uusia kysymystyyppejä ja kustomointimahdollisuuksia. Järjestelmään lisättiin myös joitakin täysin uusia näkymiä, joista tärkeimmät ovat vastausten tarkasteluun tarkoitettu Data Viewer sekä mobiiliversio vastausnäkymästä.

Toinen osa työtä on järjestelmän toteutuksen arviointi, jossa keskitytään kolmeen laatuominaisuuteen. Näistä ensimmäinen on sopivuus käyttöön, eli tarkastellaan miten hyvin järjestelmän ominaisuudet vastaavat käyttäjien tarpeisiin. Tähän arvioinnin osaan kuuluvat sekä toiminnalliset että ei-toiminnalliset ominaisuudet ja arviointiin otetaan järjestelmän käyttäjän näkökulma. Toinen osa arviointia koskee tehokkuutta, joka mittaa ohjelman vaatimaa ajan ja resurssien kulutusta [6]. Myös tässä arvioinnin osassa näkökulmana on järjestelmän käyttäjä. Viimeinen arvioitava ominaisuus on ylläpidettävyys eli se, kuinka helposti järjestelmää voidaan muokata virheiden korjaamiseksi, sen laatuominaisuuksien parantamiseksi tai muuttuneeseen ympäristöön sovittamiseksi [7]. Arvioinnin tässä osassa tarkastelun näkökulma vaihdetaan käyttäjästä kehittäjään.

Arviointi on laadullinen versio WebQEM-arvioinnista. WebQEM [8] on kvantitatiivinen asiantuntijalähtöinen laadunarviointistrategia, jossa käytetään ISO/IEC 9126-1-standardiin pohjautuvaa laatuvaatimuspuuta. Järjestelmään kohdistuvat vaatimukset rakennetaan hierarkkiseksi puumalliksi, jossa mennään niin matalalle tasolle, että alimman tason kriteerien toteutumisen astetta voidaan mitata jollakin metriikalla, ja ylemmän tason kriteerien toteutumisen aste voidaan näin ollen laskea alikriteerien arvojen kautta sopivilla funktioilla. Aidosta WebQEM-arvioinnista poiketen tässä työssä ei kuitenkaan tarkastella vaatimusten toteutumisen astetta numeerisesti, vaan kriteeripuu toimii ainoastaan järjestelmällisenä mallina siitä mihin asioihin huomio kiinnitetään laadullisessa asiantuntija-arvioinnissa. Laatukriteeripuun laatimiseen käytettiin monipuolisesti projektin vaatimuksiin liittyvää materiaalia, kuten asiakaspalautetta, projektin aikaista kirjanpitoa, projektin sisäistä viestintää sekä aloituspalaverin pöytäkirjaa.

Arvioinnin tavoitteena on saada käsitys ohjelmiston laadusta kolmen valitun laatuattribuutin suhteen, löytää ohjelmasta kuhunkin valittuun laatuominaisuuteen erityisen positiivisesti ja negatiivisesti vaikuttavia yksittäisiä ratkaisuja sekä muodostaa lista parannusehdotuksista. Tämän yksittäisen järjestelmän kontekstin lisäksi arvioinnin on tarkoitus antaa suuntaviittaa myös muiden vastaavien järjestelmien toteutuksessa mahdollisesti vastaantulevista laatuun vaikuttavista yksityiskohdista.

Luvussa 2 esitellään tarkemmin pitkän aikavälin käyttäjäkokemus käsitteenä ja siihen liittyviä tutkimusmenetelmiä. Erityisesti luvussa keskitytään kuvaamaan käyränpiirtoa hyödyntäviä retrospektiivisiä menetelmiä, joiden pohjalta myös DrawUX:n käyränpiirtotehtävä on kehitetty. Luvussa käsitellään myös web-kyselyjä yleisesti. Luku 3 esittelee DrawUX-järjestelmän tärkeimmät käyttötapaukset ja näkymät. Luvussa 4 kuvataan järjestelmän toteutus eli käytetyt teknologiat, arkkitehtuuri, tähän työhön kuuluvan

osuuden laajuus sekä toteutusprosessin vaiheet. Luvussa 5 esitellään ohjelmistojen arvioinnin ja erilaisten laatuominaisuuksien teoriaa sekä tärkeimmät arvioinnissa esiin nousseet positiiviset ja negatiiviset huomiot järjestelmän laadusta. Luvussa 6 arvioidaan mahdollisten korjausten tarvetta ja realistisuutta ja esitetään priorisoitu lista parannusehdotuksista jatkokehitystä varten. Lopuksi luvussa myös arvioidaan arvioinnin onnistumista. Luku 7 on yhteenveto koko työstä.

2. TEORIAA JA TAUSTOJA

DrawUX on erityisesti pitkän aikavälin käyttäjäkokemuksen tutkimiseen tarkoitettu työkalu. Järjestelmän ydin on käyränpiirtotehtävä, jonka avulla voidaan kerätä retrospektiivistä käyttäjäkokemustietoa, eli tietoa siitä miten vastaaja muistaa käyttäjäkokemuksensa laadun kehittyneen tuotteen käytön mittaan.

Luvussa 2.1 esitellään pitkän aikavälin käyttäjäkokemukseen liittyviä tutkimusmenetelmiä, joista erikoistapauksena nostetaan esiin DrawUX:n käyttämä käyränpiirtomenetelmä. Luvussa 2.2 käsitellään web-kyselyjä yleisesti.

2.1 Pitkän aikavälin käyttäjäkokemus

Käyttäjäkokemukselle on useita määritelmiä. ISO 9241-210 -standardin [9] mukaan käyttäjäkokemus koostuu käsityksistä ja reaktioista, jotka syntyvät tuotteen, järjestelmän tai palvelun käytön tai sen käytön odotuksen seurauksena. Käyttäjäkokemukseen kuuluvat käyttäjän tunteet, uskomukset, mieltymykset, havainnot, fyysiset ja psyykkiset vasteet sekä käytös ja aikaansaannokset käytön aikana, ennen käyttöä ja käytön jälkeen. Nielsen-Norman Groupin mukaan käyttäjäkokemus koostuu kaikista loppukäyttäjän ja firman sekä palvelujen ja tuotteiden väliseen vuorovaikutukseen liittyvistä asioista [10]. McNamaran ja Kirakowskin [11] mukaan käyttäjäkokemuksen tutkimisessa ollaan kiinnostuneita käyttäjän ja tuotteen välisestä vuorovaikutuksesta pelkkää toiminnallisuutta ja käytettävyyttä laajemmin, ja tarkoituksena on saada tietoa käyttäjän henkilökohtaisista kokemuksista liittyen tuotteen käyttöön.

Tavanomaisen tehtäväsuuntautuneen ergonomisen käytettävyyden lisäksi käyttökokemukseen liitetään siis hedonistisia näkökohtia kuten kauneus, hauskanpito, mielihyvä ja henkilökohtainen kasvu, jotka vastaavat inhimillisiin tarpeisiin olematta kuitenkaan välttämättömiä varsinaisen tehtävän suorittamisen kannalta. Van der Heijdenin [12] mukaan kokemus tuotteen tai palvelun hyödyllisyydestä menettää ennustusvoimansa, kun kyseessä on ennen kaikkea hedoninen järjestelmä, jonka ensisijainen tarkoitus ei ole auttaa jonkin tehtävän suorittamisessa tai päämäärän saavuttamisessa. Tällöin koettu ilo ja käytön helppous ovat tärkeämpiä tekijöitä, kun tutkitaan tuotteen hyväksymistä käyttöön.

Yritysmaailmassa hyvä pitkän ajan käyttökokemus nähdään olennaisena, koska sillä voidaan parantaa käyttäjäuskollisuutta. Nimenomaan pitkän aikavälin vuorovaikutus tuotteen kanssa vaikuttaa käyttäjien päätöksiin, kuten siihen, miten käyttäjä puhuu tuotteesta muille ja ostaako käyttäjä saman yrityksen tuotteita. Tähän mennessä akateemiset

tutkimukset ja yritysmaailma ovat keskittyneet tutkimaan käytön alkua ja erityisesti ensimmäistä käyttökertaa, jolloin saadaan tietoa tuotteen opittavuudesta ja käyttöönottokynnyksestä. Tällaisista tutkimuksista on hyötyä erityisesti tuotteen kehitysvaiheessa, mutta interaktiivisten tuotteiden käytön tutkiminen kiinnostaa tutkijoita myös ensimmäisen käyttökerran jälkeen, sillä käyttäjäkokemukseen vaikuttavien näkökohtien merkittävyys muuttuu ajan myötä. [4]

Tuotteiden pidentyneet takuut lisäävät yritysten motivaatiota panostaa pidemmän aikavälin käyttökokemukseen, erityisesti siksi, että merkittävä osa takuuajalla palautettavista tuotteista on täysin toimivia. Palautuksen syynä on tällöin se, että tuote ei vastannut käyttäjän todellisiin tarpeisiin, mikä ei liity ongelmiin tuotteen käytön alussa, vaan ajan mittaan pysyviin ongelmiin. Toinen pitkän ajan käyttökokemuksen merkitystä lisäävä trendi on tulonlähteiden siirtyminen tuotteista palveluihin. Varsinainen tuote on yhä useammin alihintainen, ja voitot saadaan tuotteeseen liittyvistä palveluista. Tällöin tuotteen ja palvelun jatkuva käyttö on avainasemassa yrityksen menestyksen kannalta. [5]

Siitä, miten käyttökokemus kehittyy ajan mittaan, on kuitenkin vähemmän tutkimuksia ja jopa pidempää aikaväliä käsittelevät tutkimukset ovat tyypillisesti suhteellisen lyhyitä, korkeintaan muutaman viikon mittaisia. Pidempiä tutkimuksia on vaikeampi järjestää kustannusten, vaadittavan ajan ja osallistujien motivoinnin vuoksi. Tarvitaan siis parempia tapoja tehdä tutkimusta.

2.1.1 Pitkän ajan käyttökokemuksen tutkimusmenetelmiä

Pitkän aikavälin käyttökokemusta voidaan tutkia eri tavoilla ja eri näkökulmista. Karapanos et al. [5] erottelee kolme eri lähestymistapaa: poikkileikkaustutkimus, pitkittäistutkimus ja takautuva tutkimus. Poikkileikkaustutkimuksessa tutkitaan kerralla eritasoisia käyttäjäryhmiä, kuten vasta-alkajat ja tuotetta tietyn aikaa käyttäneet. Tämä lähestymistapa on suosituin HCI-alalla, mutta sen merkittävänä heikkoutena on oletus siitä, että mahdolliset ryhmien välillä havaitut erot selittyvät lähinnä sillä, kuinka pitkään ryhmät ovat käyttäneet tuotetta. Tämä oletus ei välttämättä pidä paikkansa, sillä yksilöjen väliset erot ja satunnaiset tekijät voivat vaikuttaa havaintoihin merkittävästi.

Pitkittäistutkimuksessa tutkitaan samaa käyttäjäryhmää vähintään kahtena eri ajanhetkenä. Mitä useampia näytteitä käytön aikana kerätään, sitä enemmän tietoa käyttökokemuksen kehittymisestä saadaan, mutta samalla tutkimus myös käy työläämmäksi ja hankalammaksi järjestää, erityisesti jos käytetään kenttätutkimusta tai kasvokkain haastattelua. Tämän vuoksi useamman kuin kahden näytepisteen pitkittäistutkimuksia on harvemmin käytetty. ESM (Experience Sampling Method) on kenttätutkimusmenetelmä, jossa käyttötilanne keskeytetään tietyin väliajoin käyttäjähaastatteluja varten. Näin saadaan dataa reaaliajassa, mutta käyttäjiin kohdistuva jatkuvien keskeytysten kuormitus rajoittaa mahdollisten tutkimusten pituutta. ESM:n puutteita paikkaamaan kehitettiin itseraportointimenetelmä DRM (Day Reconstruction Method), jossa käyttökokemuksen

kehittyminen tietyn päivän ajalta raportoidaan kerralla seuraavana päivänä. Vastaajaa pyydetään jakamaan edellinen päivä nimettyihin jaksoihin ja tämän jälkeen kuvailemaan tarkemmin jokaista jaksoa. Kuvaamalla kokonaisen päivän kokemukset kerralla säästetään aikaa ja vaivaa, eikä käyttötilannetta tarvitse keskeyttää. [13]

Menetelmiä, joissa käyttäjä joutuu muistelemaan aiempia kokemuksiaan, kutsutaan takautuviksi eli *retrospektiivisiksi*. Esimerkiksi erilaiset Flanaganin [14] kehittämään CIT (Critical Incident Technique) -tekniikkaan perustuvat menetelmät pyytävät osallistujia raportoimaan positiivisimpia ja negatiivisimpia kokemuksiaan käytön ajalta [2]. CORPUS (Change Oriented analysis of the Relationship between Product and USer) on haastattelutekniikka, jossa vastaajaa pyydetään arvioimaan nykyistä kokemustaan tuotteen tietyistä laatuominaisuuksista, ja tämän jälkeen heitä pyydetään vertaamaan nykyisiä kokemuksiaan ostohetken jälkeisiin kokemuksiin ja selittämään mahdolliset muutokset [3].

Retrospektiiviset menetelmät herättävät usein kysymyksiä siitä, kuinka luotettavina jälkikäteen muisteltuja kokemusraportteja voidaan pitää. Ihmisten muistikuvissa voi olla aukkoja ja vääristymiä. Kuitenkin Norman [15] väittää nimenomaan subjektiivisten muistikuvien, ei niinkään todellisten kokemusten eksaktien yksityiskohtien, olevan erityisen olennaisia, sillä subjektiivisesti koetut tapahtumat ja tunteet, jotka käyttäjä itse muistaa kokeneensa, vaikuttavat tulevaan käyttäytymiseen. Oishin ja Sullivanin [16] mukaan retrospektiiviset arvioinnit ovat jopa ennustusvoimaisempia kuin päivittäiset raportit. Retrospektiivisten tutkimusten tarkoitus ei kuitenkaan ole korvata pitkittäistutkimuksia, vaan retrospektiiviset tutkimukset ovat hyvä vaihtoehto silloin kun on kiinnostuttu enemmän muistikuvista kuin todellisista tapahtumista [5].

2.1.2 Käyränpiirto osana retrospektiivistä käyttökokemustutkimusta

DrawUX-työkalun ydin on käyränpiirtotehtävä, jossa vastaaja piirtää käyrän joka kuvaa käyttökokemuksen kehittymistä käytön aikana. Käyränpiirron oletetaan auttavan käyttäjää muistamaan enemmän yksityiskohtia kokemuksistaan. Goldschmidt [17] kuvailee ’interaktiivisia mielikuvia’ eli operaatiota, jossa luonnosta ja siihen liittyvää mielikuvaa tuotetaan lähes samanaikaisesti, jolloin luonnostelu ei ole ainoastaan olemassa olevan mielikuvan jäljentämistä paperille, vaan samanaikaista mielikuvan tarkennusta. Piirtäminen omien aiempien kokemusten raportointimenetelmänä oli käytössä jo Sonnemansin ja Frijdan [18] tutkimuksessa, jossa tutkittiin erilaisten tunteiden voimakkuutta. Osana tutkimusta vastaajia pyydettiin piirtämään tietokoneohjelmalla diagrammi tunneintensiteetin kehittymisestä, ja ohjelma esitti jatkokysymyksiä piirretyn käyrän muodon perusteella. Tässä ohjelmassa piirtäminen tapahtui liikuttamalla hiirtä vapaalla kädellä, kun taas DrawUX:ssa käyttäjä piirtää käyrän lisäämällä sille pisteitä. DrawUX:ssa

käytetty käyränpiirtotehtävä osana retrospektiivistä käyttäjäkokemustutkimusta pohjautuu erityisesti kahteen aikaisemmin kehitettyyn metodiin, iScaleen ja UX Curveen.

iScale on kyselytyökalu, josta on toteutettu kaksi hieman erilaista versiota. Toinen perustuu rakentavaan muistiteoriaan ja toinen tunnetilimuistiteoriaan. Kummassakin versiossa käyttökokemuskuvaaja muodostuu lineaarisista segmenteistä, jotka kuvaavat käyttökokemuksen paranemista (nouseva suora) tai heikkenemistä (laskeva suora). Kuhunkin segmenttiin liittyy tunniste, jota klikkaamalla vastaaja voi raportoida yhden tai useamman kokemuksen syyksi muutokselle, jota kyseinen segmentti kuvaa. [5]

Rakentavan muistiteorian mukaan tunteita ei voida säilöä suoraan mustiin, vaan ne täytyy rakentaa uudelleen muistista löytyvän kontekstin pohjalta. Tässä lähestymistavassa yksityiskohtaisten kokemusten muistamisen oletetaan tapahtuvan aikajärjestyksessä siten, että tietyn tapahtuman muistaminen auttaa muistamaan ajallisesti seuraavia olennaisia tapahtumia. Rakentavassa iScale-versiossa vastaaja aloittaa käyränpiirron ensimmäisestä muistikuvastaan (tuotteen hankintahetki) ja rakentaa uudelleen muistikuvan senhetkisestä tunnereaktiostaan ja siihen liittyvästä laatuarviostaan. Tämä muistikuva vuorostaan auttaa vastaajaa muistamaan seuraavan olennaisen kokemuksen. Käyrän segmenttien lisäys tapahtuu siis aikajärjestyksessä alusta loppuun. Aina uuden segmentin lisäyksen jälkeen käyttäjää pyydetään kirjoittamaan vastaava kuvaus, sillä samanaikaisen raportoinnin ja piirtämisen oletetaan auttavan vastaajaa muistamaan enemmän yksityiskohtia kokemuksistaan. [5]

Tunnetili (*value-account*) -muistiteorian mukaan taas on olemassa muistisäiliö eri ärsykkeisiin liittyville tunnereaktioille, ja tätä tunnemuistia voidaan käyttää apuna kontekstiyksityiskohtien muistamiseen [19]. Lähestymistapa olettaa, että ihminen muistaa tunnepitoisen yleisvaikutelman ja vasta tämän jälkeen tarkempia yksityiskohtia tapahtumista. Tunnetiliteoriaan perustuva versio iScalesta alkaa pyytämällä käyttäjää kuvaamaan käyttökokemuksen kehittymisen yleisen muodon, minkä perusteella iScale muodostaa yhtenäisen suoran aikajanan alusta loppuun, ja käyttäjä lisää yksityiskohtia kokemuksistaan jakamalla kuvaajan pienempiin segmentteihin. Tässä iScalen versiossa kokemusten raportointi tapahtuu vasta kuvaajan piirron jälkeen erillisenä askeleena, jonka aikana käyttäjä voi lisätä yhden tai useamman kuvauksen jokaiselle segmentille. [5]

Kun iScale-menetelmiä verrattiin käyrän piirtämiseen vapaasti paperille, iScalen todettiin pärjäävän hyvin vertailussa. Vapaalla kädellä piirtäminen koettiin ilmaisuvoimaisemmaksi ja vastaajat pitivät mahdollisuudesta lisätä annotaatioita kuvaajiin, mutta iScalen hyvinä puolina pidettiin helpompaa muokattavuutta ja interaktiivisuutta. Tunnetiliversion koettiin antavan parempi näkymä käyttökokemuksen kokonaisuuteen aikajanalla, mutta aikajärjestyksessä etenevä rakentava iScale auttoi vastaajia muistamaan useampia kokemuksia ja lisätyt kokemusraportit sisälsivät tarkempaa informaatiota. Tulokset tukivat teoriaa siitä, että aikajärjestyksen noudattaminen segmenttien lisäyk-

sessä tukee kontekstin uudelleenrakentamista vastaajan mielessä ja auttaa näin muistamisprosessia. Rakentava iScale johti myös yhdenmukaisempiin kuvaajiin, kun samoja vastaajia pyydettiin piirtämään samaa käyttökokemusta vastaava kuvaaja kahdella eri kerralla. Tämä viittaa siihen, että rakentava iScale tuki vastaajien muistia paremmin. Vastaajien palautteen perusteella iScaleen lisättiin aikajana-annotaatio, jolla voidaan kuvata segmenttien alku-ja loppuajankohdat, sekä segmentteihin liittyvien kokemusten visualisointimahdollisuus. Lisäysten tarkoituksena oli antaa vastaajalle parempi yleiskatsaus aikajanaan. [5]

Lähes samaan aikaan iScalen kanssa kehitettiin itsenäisesti UX Curve -menetelmä [4]. Tarkoituksena oli kehittää helppo tapa auttaa vastaajia muistamaan käyttökokemukseen liittyviä tärkeitä yksityiskohtia. Siinä missä iScale ja DrawUX ovat enimmäkseen verkkokäyttöön suunniteltuja työkaluja, UX Curve on tarkoitettu kasvokkain tapahtuvan haastattelun tueksi. Työkalun tavoitteena on erityisesti auttaa muistamaan tapahtumien järjestys, sillä tapahtumien kronologisen kulun on todettu vaikuttavan käyttäjän nykyisiin arvioihin sekä ennustavan tulevaa käytöstä [16]. UX Curvessa vastaaja saa käyttöönsä tyhjän kaksiulotteisen kuvaaja-alueen, johon hän piirtää käyttökokemuksen kehittymistä kuvaavan käyrän vapaalla kädellä. Käyrän muutoskohtiin liittyvät syyt pyydetään kirjoittamaan vastaaviin kohtiin käyrää.

Käyränpiirtotehtävä retrospektiivisenä käyttökokemuksen kuvaajana vaikuttaa toimivalta ratkaisulta, koska sillä on vahva teoreettinen pohja, ja edellä esitellyissä tutkimuksissa metodin avulla saatiin kerättyä suunnitelmien mukaisesti käyttökokemuskuvauksia. Myös UX Curve -pilottitutkimuksessa kaikki käyttäjät pystyivät käyttämään menetelmää ja pitivät sitä jopa hauskana, paitsi muutosten syiden kirjoittamista, joka koettiin uuvuttavana. iScale-tutkimuksessa erityisesti aikajärjestyksessä etenevä rakentava iScale auttoi käyttäjiä muistamaan vapaata käsin piirtämistä enemmän kokemuksia ja yksityiskohtia [5], kun taas UX Curve -tutkimuksessa käyränpiirto ylipäätään auttoi vastaajia muistamaan tapahtumia paremmin kuin CORPUS-menetelmä. UX Curve todettiin hyödylliseksi pitkän aikavälin käyttökokemuksen analysointiin, sillä sen avulla saatiin kerättyä rikasta yksityiskohtaista tietoa käyttökokemuksesta. [4]

UX Curvella saatava data ei ole yhtä syvällistä kuin kenttätutkimuksilla saatava tieto, mutta UX Curve -dataa on helpompi analysoida, se voi kattaa pidemmän aikavälin, ja vastaajajoukkoa saadaan helpommin suurennettua, jolloin tulosten yleistettävyys paranee. Menetelmän vahvuutena on myös se, että UX Curve sai käyttäjät kertomaan merkittävistä ja muistettavista kokemuksistaan, joista on hyötyä suunnittelijoille jotka haluavat luoda positiivisia ja välttää negatiivisia käyttäjäkokemuksia, voiden näin vaikuttaa asiakasuskollisuuteen [4]. UX Curve osoittautui myös kohtuullisen kustannustehokkaaksi pitkän aikavälin käyttäjäkokemuksen tutkimustavaksi verrattuna pitkittäistutkimuksiin, mutta DrawUX:n kaltainen web-järjestelmä luonnollisesti lisää kustannustehokkuutta edelleen.

Käyränpiirron ongelmakohdaksi havaittiin muun muassa vastaajien lisäämien annotaatioiden epäyhdenmukaisuus [4]. Osa vastaajista merkitsi kuvaajan huippukohtaan negatiivisen kokemuksen kuvauksen (kokemus joka aloitti kohdasta alkavan laskun), osa taas kuvasi korkeimmassa käyrän kohdassa positiivisia käyttökokemukseen liittyviä asioita. Käyränpiirto ei myöskään sovellu jokaiseen mahdolliseen tutkimukseen. Tehtävän todettiin sopivan parhaiten päivittäin tai toistuvasti käytettävien, jo markkinoilla olevien tuotteiden tutkimiseen. Sitä vastoin käyränpiirto ei välttämättä sovi vasta kehitysvaiheessa olevien ja vain silloin tällöin käytettävien tuotteiden tutkimiseen eikä tilanteisiin, joissa käyttäjä on lopettanut tuotteen käytön ensimmäisen negatiivisen kokemuksen jälkeen. Käyränpiirto ei myöskään korvaa muita menetelmiä, kuten käytettävyystestausta.

Näiden aiempien työkalujen ja tutkimusten pohjalta kehitettiin DrawUX:n oma versio käyränpiirtotehtävästä, joka on iScalen kaltainen verkkopohjainen työkalu, mutta piirtämisen vapauden suhteen enemmän UX Curven kaltainen [1]. DrawUX:n käyränpiirtotehtävä antaa vastaajalle enemmän vapautta käyränpiirron suhteen kuin kumpikaan iScalen versio, sillä lisättyään ensimmäisen pisteen käyttäjä saa vapaasti jatkaa käyrää haluamallaan tavalla. Tämä tekee käyrän helpommaksi muokata, mutta herättää kysymyksen siitä, menetetäänkö näin rakentavan iScalen tutkimuksissa havaittu lisätuki muistikuvien rakentamiseen. DrawUX-käyrissä aikajana-annotaatio on automaattisesti osa kuvaajan x-akselia, eli käyttäjän ei itse tarvitse merkitä tapahtumien ajankohtia, ainoastaan asettaa piste mahdollisimman oikean x-koordinaatin kohdalle. Valmiin aika-akselin on tarkoitus auttaa vastaajaa mahdollisimman tarkan käyrän piirtämisessä. Kuten edellä esitellyissä työkaluissa, myös DrawUX:ssa lisättyihin pisteisiin voidaan liittää käyttötilanteita kuvaavia kommentteja. Lähtökohtaisesti DrawUX-käyränpiirto on hyvin vapaata, mutta tutkija voi halutessaan asettaa rajoitteita kommenttien minimimäärästä ja siitä, pitääkö käyrä piirtää loppuun.

2.2 Web-kyselyt

Internet tarjoaa tutkijoille ainutlaatuisen mahdollisuuden päästä tutkimaan suuria ihmismääriä, koska maantieteelliset tekijät eivät rajoita tutkimukseen osallistumista, tutkimuksen julkaisu ei vie juurikaan resursseja, eikä suuri vastaajamäärä lisää tutkimuksen toteutustyötä samalla tavalla kuin kasvokkain haastattelut ja muut perinteiset tutkimuksen muodot. Internet-kyselyn kustannukset muodostuvat Internet-yhteydestä, joka yrityksillä on lähes aina joka tapauksessa käytössään, palvelimesta jolla kysely julkaistaan, sekä kyselyohjelmiston kehittämisestä. Nämä kustannukset ovat yleensä huomattavasti pienemmät kuin paperilla julkaistavan kyselytutkimuksen kustannukset, joihin kuuluvat mm. kyselyn fyysinen julkaisu, jakelu, vastaajien muistuttaminen, vastausten keruu, vastausten syöttö tietojärjestelmään, sekä palautteen julkaisu ja jakelu. [20]

Myös Duffy et al. [21] mainitsevat web-kyselyjen vahvuudeksi kustannustehokkuuden, mutta huomauttaa kuitenkin että vastaajapaneelin kokoaminen vie alussa resursseja.

Kustannussäästöt alkavat muodostua vasta vastailun yhteydessä, kun uusiin vastauskertoihin ei enää liity lisämenoja.

Suuri vastaajamäärä on tutkimukselle hyödyksi, sillä se parantaa tulosten yleistettävyyttä ja lisäksi parantaa todennäköisyyttä päästä tutkimaan myös harvinaisia ja epätavallisia kokemuksia tutkittavasta aiheesta. Vastaajien mahdollisen suuren määrän lisäksi web-kyselyjen etuna on myös kohtalaisen helppo tapa löytää juuri halutun kaltaisia vastaajia. Ihmisiä voidaan houkutella kyselyn aiheeseen liittyviltä verkkosivuilta ja foorumeilta. Näin löydetään ihmisiä, joilla on kokemusta tietyistä tutkimuksen kohteena olevista asioista, ja tämän vuoksi web-kysely soveltuukin erityisen hyvin tutkimuksiin, joiden aihe on kapea tai harvinainen [20]. Nykyään myös sosiaalisen median sivustot, kuten Twitter, Facebook ja Google+, nähdään kiinnostavina mahdollisuuksina tavoittaa vastaajia (ks. esim. [22]). Myös satunnaisotosta lähelle pääsevä otanta on mahdollinen esimerkiksi RAWI-tekniikalla (Random Web Interviewing), jossa kyselyyn houkutellaan vastaajia sijoittamalla popup-mainoksia satunnaisille verkkosivuille [23].

Jos vastauksia halutaan analysoida tietokoneella, perinteisessä haastattelu- tai kenttätutkimuksessa vastausten keräämisen ja vastausten analysoinnin väliin tulee ylimääräinen ja työläs askel, datan syöttäminen tietojärjestelmään. Web-kyselyissä tämä vaihe jää kokonaan pois, sillä vastaajan lähettäessä vastauksensa palvelimelle ohjelma tallentaa vastaukset automaattisesti helposti käsiteltävään muotoon, yleensä tietokantaan. Näin säästetään jälleen resursseja. Muihin web-kyselyjen hyviin puoliin kuuluu mahdollisuus antaa vastaajalle interaktiivisesti palautetta, kuten suorittaa automaattisia virhetarkastuksia tai esittää kokonaan uusia kysymyksiä edellisten vastausten pohjalta. Vastaaja voi myös olla kiinnostunut kaikista vastauksista kootuista yhteenvedoista. Hyvin toteutettuna kyselyjen dynaamisuus ja interaktiivisuus lisäävät vastaajan motivaatiota. [20]

Web-kyselyillä voidaan tietyissä tapauksissa myös saada todenmukaisempaa tietoa kuin kasvokkain tapahtuvissa haastatteluissa, sillä haastattelijan läsnäolo saattaa vaikuttaa vastaajaan [21]. Vastaajalla voi tällöin olla paine vastata sosiaalisesti hyväksyttävällä tavalla. Esimerkiksi sairauksien yleisyyttä tutkittaessa web-kyselyillä saatu data vastaa paremmin tunnettua levinneisyyttä kuin puhelimessa tai fyysisessä haastattelutilanteessa saatu data. Sosiaalisesti ei-hyväksyttävä käytös myönnetään todennäköisemmin verkkokyselyyn vastatessa kuin haastattelutilanteessa, ja verkkokyselyvastauksilla on myös tapana olla kriittisempiä esimerkiksi tuotteita arvioitaessa [23].

Web-kyselyihin sisältyy myös riskejä, kuten turvallisuusriskit, puuttuva tai virheellinen data ja saman vastaussetin tallentuminen moneen kertaan, mutta näihin tilanteisiin pystytään yleensä varautumaan ohjelmaa suunniteltaessa. Suunnittelussa pitää luonnollisesti huomioida myös kaikille web-sivustoille yhteiset huolenaiheet, kuten sivustojen saavutettavuus ja erilainen ulkoasu ja toiminnallisuus eri selaimilla. Schmidt [20] huomauttaa myös, että tyypillinen Internetin käyttäjä ei välttämättä edusta tyypillistä ihmistä, sillä tyypillisellä Internetin käyttäjillä on keskimääräistä enemmän tuloja ja koulutusta.

Kuitenkin Internet-yhteyden omaavien ihmisten joukko kasvaa jatkuvasti. Olennaista on huomioida mahdolliset vääristymät tulosten analysointivaiheessa ja tehdä harkiten päätelmät siitä, mihin ihmisjoukkoihin tulokset voidaan vastaajajoukon perusteella yleistää. Tämän vuoksi kyselyissä kannattaa kysyä riittävä määrä demografisia tekijöitä ja muita tarkkaa profilointia auttavia kysymyksiä.

Mobiililaitteiden kasvava käyttö on web-kyselyjen kannalta huomionarvoinen trendi. Kyselyihin vastaaminen mobiililaitteella on eri hypoteesien mukaan nähty joko hyötynä tai haittana. Erään hypoteesin mukaan mobiililaite on hyödyksi osallistujien kutsumisen kannalta, sillä mobiililaitteella sähköposti tavoittaa vastaajan nopeammin. Toisaalta yleensä pöytäkoneille suunniteltujen kyselyjen käytettävyys mobiililaitteella on herättänyt kysymyksiä. Cunninghamin et al. [24] tutkimuksessa molemmat oletukset saivat tukea kun havaittiin, että mobiililaitteella kutsun saaneet vastaajat avasivat kyselyn nopeammin, mutta täyttivät kyselyn loppuun pienemmällä todennäköisyydellä kuin pöytäkonetta käyttävät vastaajat. Vastaajista 29,1 % käytti vastaamiseen mobiililaitetta (yleensä älypuhelinta, jonkin verran myös tabletteja). Mobiililaitteiden huomiointi moderneissa web-kyselyissä on siis selvästi tarpeen.

Web-kysely pohjautuu enimmäkseen usein jopa ilmaiseksi saatavilla olevaan teknologiaan, kuten avoimen lähdekoodin tietokantoihin ja palvelinohjelmistoihin. Kyselyn toteuttamisessa hankalinta on varsinaisen kyselyohjelman toteutus. Tätä kynnystä madaltamaan on luotu useita web-kyselytyökaluja, joilla tutkija voi luoda ja julkaista kyselynsä automaattisesti joutumatta itse toteuttamaan työkalun ohjelmointivaihetta. DrawUX on eräs tällainen työkalu.

3. DRAWUX-JÄRJESTELMÄ

Diplomityössä jatkokehitettiin DrawUX-järjestelmää, joka on keskikokoinen ohjelmisto web-pohjaisten kyselytutkimusten järjestämiseen. Järjestelmän kehityksen lisäksi työhön kuuluu nykyisen toteutuksen pienimuotoinen asiantuntija-arviointi, jonka tarkoituksena on olennaisimpien puutteiden tunnistaminen sekä parannusehdotusten esittäminen ja priorisoiminen.

DrawUX on web-pohjainen kyselytyökalu, joka on erityisesti tarkoitettu pitkän aikavälin käyttäjäkokemuksen tutkimiseen [1]. DrawUX:n erikoisuus on käyränpiirtotehtävä, jossa vastaaja voi piirtää käyttökokemustaan kullakin ajanhetkellä vastaavan käyrän. Kuvaajan x-akseli kuvaa aikaa ja y-akseli jotakin käyttökokemuksen laatua kuvaavaa attribuuttia. Lisäksi kyselyyn voi kuulua lomakesivuja, jotka sisältävät perinteisiä web-kyselyjen kysymystyyppejä.

Työkalulla voidaan tehdä kahdenlaisia kyselyjä, jotka näyttävät erilaisilta vastaajalle, mutta joiden tuloksia voidaan tarkastella samalla tavalla. Kyselytyypit ovat:

- *retrospektiivinen kysely*, johon käyttäjä vastaa kerran. Käyränpiirtotehtävässä vastaaja piirtää koko käyrän kerralla antaen valitsemansa määrän eri ajanhetkiä kuvaavia pisteitä. Retrospektiiviseen kyselyyn ei voi vastata mobiililaitteella, sillä Highcharts-kirjastolla toteutetun käyränpiirtotehtävän sujuvaan käyttöön tarvitaan suuri näyttö ja kohtuullisesti suorituskykyä.
- *päiväkirjatyyppinen kysely*, johon käyttäjä vastaa useamman kerran ja antaa kullakin vastauskerralla arvion senhetkisestä käyttökokemuksestaan. Yhteensä näistä arvioista muodostuu lopulta vastaavanlainen käyrä kuin retrospektiivisessä kyselyssä. Päiväkirjatyyppisestä kyselystä on toteutettu erikseen HTML5-mobiiliversio, jonka käyttöliittymä on paremmin pienelle näytölle sopiva.

Työkalu ei ota kantaa siihen, millä tavalla tutkimus järjestetään. Työkalun kautta on mahdollista kutsua kyselyyn vastaajia, riittää kun tutkija tietää mahdollisten vastaajien sähköpostiosoitteet. Kuitenkin tutkimuksen suunnittelussa kannattaa ottaa huomioon osallistujien motivointi. Yksittäinen sähköposti ei välttämättä ole tarpeeksi saamaan vastaajia osallistumaan tutkimukseen. Tämän vuoksi järjestelmää olisi suositeltavaa käyttää vasta varsinaisen osallistujien rekrytoinnin jälkeen, jolloin vastaajat ovat mukana vapaaehtoisesti eikä lähetettyjä kutsuja ja muistutuksia koeta häiritsevinä.

Seuraavissa luvuissa esitellään järjestelmää tarkemmin. Luku 3.1 esittelee lyhyesti tärkeimmät käyttäjäryhmät ja käyttötapaukset. Luvussa 3.2 esitellään järjestelmän tärkeimmät näkymät ja niihin liittyvät toiminnot.

3.1 Käyttäjät ja käyttötapaukset

DrawUX:n tärkeimmät käyttäjäryhmät ovat tutkija ja vastaaja. Järjestelmä on kehitetty vastaamaan näiden kahden ryhmän tarpeisiin. Lisäksi järjestelmää voivat käyttää kehittäjät, ylläpitäjät ja testaajat, mutta järjestelmä ei suoraan tue heidän työtään tarjoamalla mitään erityisiä toimintoja. Kuva 1 esittää oleelliset käyttötapaukset.

Kuva 1. *Käyttötapauskaavio DrawUX-järjestelmän oleellisimmista toiminnoista.*

Tutkija on ainoa käyttäjäryhmä jonka tulee ensin rekisteröityä palveluun. Tutkijan olennaisin tehtävä palvelussa on luoda kyselyjä. Kyselyn luonti alkaa valitsemalla luotavan kyselyn tyyppi (retrospektiivinen vai päiväkirjatyyppinen). Kyselyeditorissa tutkija voi lisätä ja poistaa kyselysivuja ja muuttaa sivujen järjestystä. Sivuille voidaan lisätä erilaista sisältöä niiden tyypistä riippuen. Käyränpiirtotehtävää edustavalle sivulle annetaan tehtävään liittyvät asetukset, kuten piirtotehtävän aihe ja akselien nimet. Tavallista lomakesivua edustavalle välilehdelle puolestaan voi lisätä vapaasti valittavan määrän erilaisia kysymyksiä, joiden tyypit esitellään tarkemmin luvussa 3.2. Tutkija voi myös

muuttaa kyselyyn liittyviä yleisiä asetuksia. Kun kysely on tallennettu järjestelmään, tutkija voi esikatsella, muokata, kopioida ja poistaa kyselyjä. Tutkijoilla on oikeus hallinnoida ja tarkastella vain itse luomiaan kyselyjä, muiden tutkijoiden luomat kyselyt eivät ole näkyvillä tai muuten saavutettavissa. Kyselyt ovat aluksi suljettuja eli halutessaan julkaista kyselyn tutkijan tulee ensin avata kysely vastauksille, minkä jälkeen tutkija voi kutsua vastaajia syöttämällä järjestelmään näiden sähköpostiosoitteet. Vastaajia voidaan myös muistuttaa vastaamisesta. Tutkija voi tarkastella omien kyselyjensä vastauksia joko tähän tarkoitetun Data Viewer -sivun kautta tai lataamalla tulokset Excel-taulukkona. Saadut vastaukset voi kääntää toiselle kielelle. Tutkija voi myös tarkastella ja tarvittaessa poistaa eri vastaajien vastauksia ja vastauskertoja.

Vastaaja käy vastaamassa kyselyyn yhden tai useamman kerran riippuen kyselyn tyypistä. Vastaaminen on yksinkertaista, vastaajan tulee vain syöttää vastauksensa lomakeelementteihin ja mahdollisesti piirtää käyttökokemuskäyriä. Jos kyselysivulla on elementtejä, jotka tutkija on valinnut pakollisiksi, vastaajan tulee vastata vähintään näihin kysymyksiin ennen kuin pääsee siirtymään seuraavalle sivulle. Vastaaja on joko kutsuttu erikseen sähköpostissa lähetetyllä henkilökohtaisella linkillä, joka yksilöi vastaajan, tai vastaaja saattaa päätyä kyselysivulle julkisen linkin kautta, jolloin vastaaminen tapahtuu anonyymisti.

Järjestelmän *testaaja* voi suorittaa järjestelmätestausta käyttämällä palvelua käyttöliittymän kautta joko tutkijan tai vastaajan roolin mukaisesti. Jos testataan tutkijan näkymiä, tulee testaajan rekisteröityä palveluun samalla tavalla kuin tutkijankin. DrawUX-järjestelmä ei tällä hetkellä tarjoa erityistä tukea testaajalle, joten käytännössä testaaja voi suorittaa vain normaaliin järjestelmän käyttöön perustuvaa musta laatikko -testausta. Muunlainen testaus vaatii järjestelmään kuulumattomia erityistyökaluja tai lähdekoodin muokkausta. Tähän asti kehityksessä ei ole ollut erikseen mukana testaaja-roolissa toimivaa henkilöä, vaan toiminnan varmistuksen on suorittanut kehittäjä. Jatkokehitystä ajatellen testaajan roolin erottaminen kehittäjästä sekä testauksen parempi huomioiminen sekä järjestelmässä että kehitysprosessissa olisi ehdottomasti tarpeellista.

Kehittäjä voi jatkokehittää palvelua ja tiedottaa muutoksista. Järjestelmään oli myös suunnitteilla *ylläpitäjä*-rooli, jolle tarjottaisiin oma admin-näkymä käyttäjien ja kyselyjen hallinnointiin sekä muutoksista tiedottamiseen. Järjestelmässä on uutissivu, joka sisältää tiedotuksia järjestelmään kulloinkin tehdyistä muokkauksista. Toistaiseksi kehittäjä tai ylläpitäjä joutuu kuitenkin syöttämään nämä tiedotukset suoraan tietokantaan, sillä DrawUX-järjestelmän käyttöliittymä ei vielä sisällä toimintoa uutisten lisäykseen. Käytännössä tähän asti ylläpitäjä- ja kehittäjärooleissa ovat toimineet samat henkilöt, joita on ollut vain yksi kerrallaan.

3.2 Näkymät ja toiminnallisuus

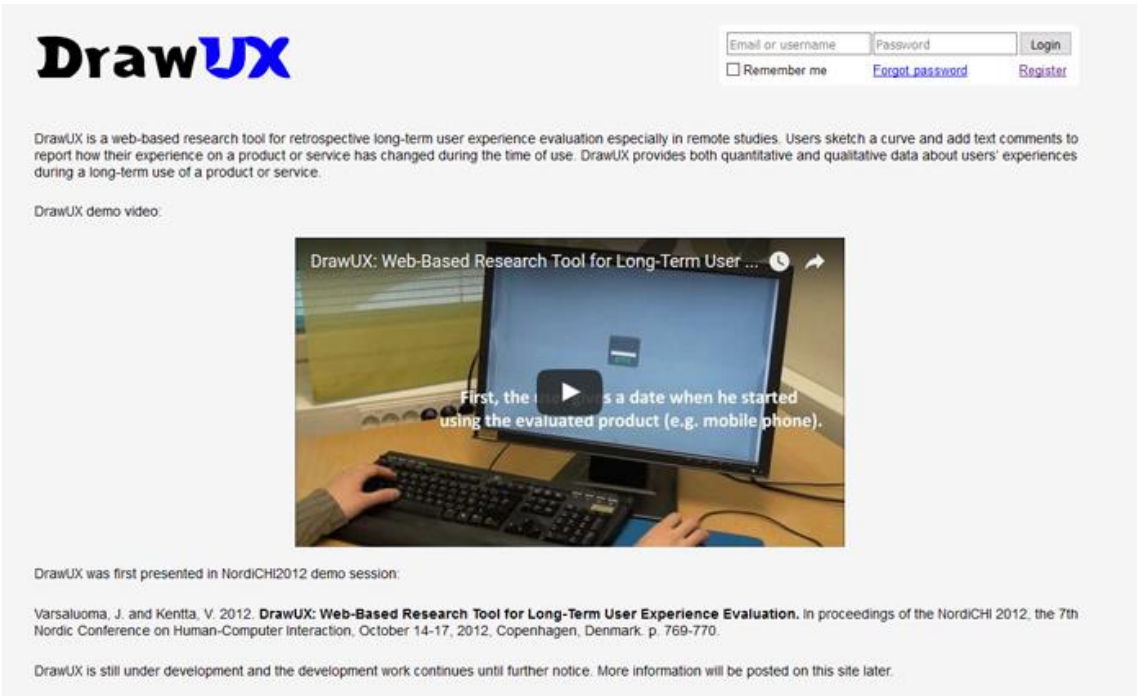
Alun perin DrawUX:n suunniteltiin koostuvan kolmesta olennaisesta komponentista:

1. verkkopohjainen kyselyeditori tutkijoille
2. verkkopohjainen työkalu kyselyihin vastaamiseen ja käyrien piirtoon vastaajille
3. tietokanta tietojen tallennukseen

[1]. Nämä ovat järjestelmän olennaisimmat osat. Lisäksi järjestelmään on lisätty tämän jälkeen uusia näkymiä ja työkaluja.

Jokainen tutkijalle näkyvä DrawUX-järjestelmän sivu sisältää yläreunaan sijoitetun *navigaatiopalkin*, josta pääsee pääsivulle, omiin kyselyihin, uutissivulle, palautteen antoon sekä omiin asetuksiin. Vastaajalle tarkoitettu vastausnäkymä ei luonnollisesti sisällä navigaatiopalkkia, sillä muut sivut on tarkoitettu vain tutkijoiden käyttöön.

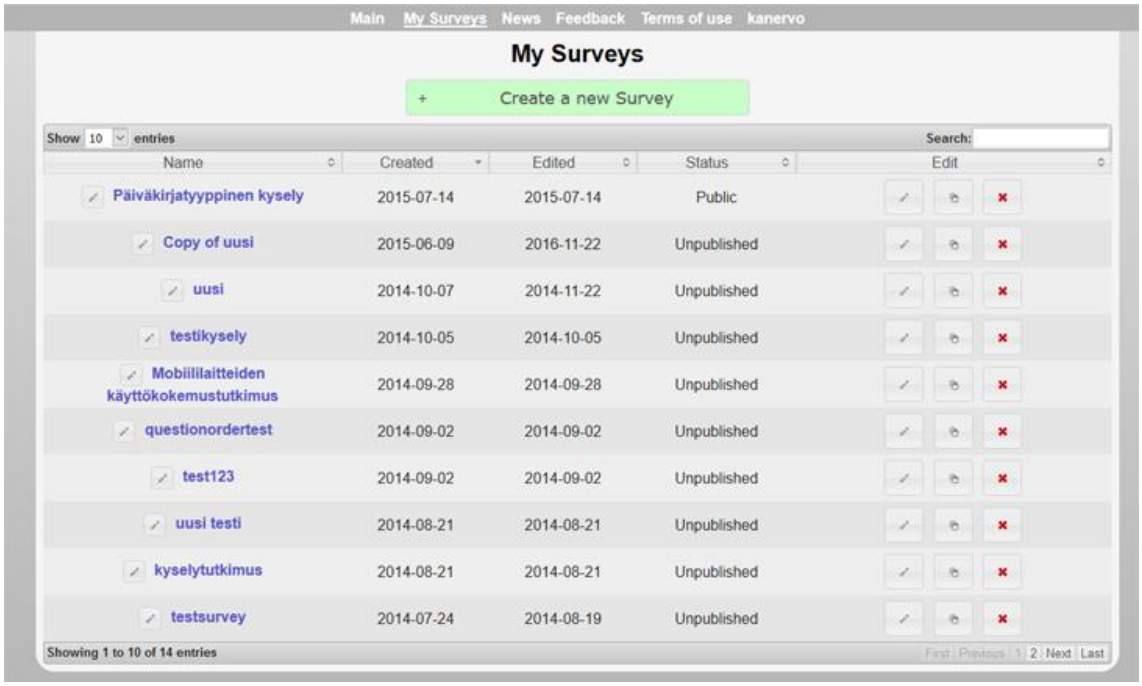
DrawUX-järjestelmän *etusivu* (kuva 2) on hyvin yksinkertainen ja sisältää staattista informaatiota sivustosta sekä kyselytyökalun esittelyvideon. Sivun yläreunassa on sisäänkirjautumiskenttä.



Kuva 2. *Osa DrawUX-etusivua.*

Omat kyselyt -sivulla (kuva 3) tutkija voi tarkastella listana kaikkia luomiaan kyselyjä. Listan voi järjestää kyselyn nimen, luontiajan, muokkausajan ja tilan (julkinen/suljettu) mukaan. Näiden kenttien perusteella kyselyjä voi myös hakea. Kunkin kyselyn nimen vieressä on editointipainike, josta klikkaamalla tutkija voi muokata kyselyn nimeä suoraan kyselylistauksessa. Kyselytaulukon viimeisessä sarakkeessa on muokkauspainik-

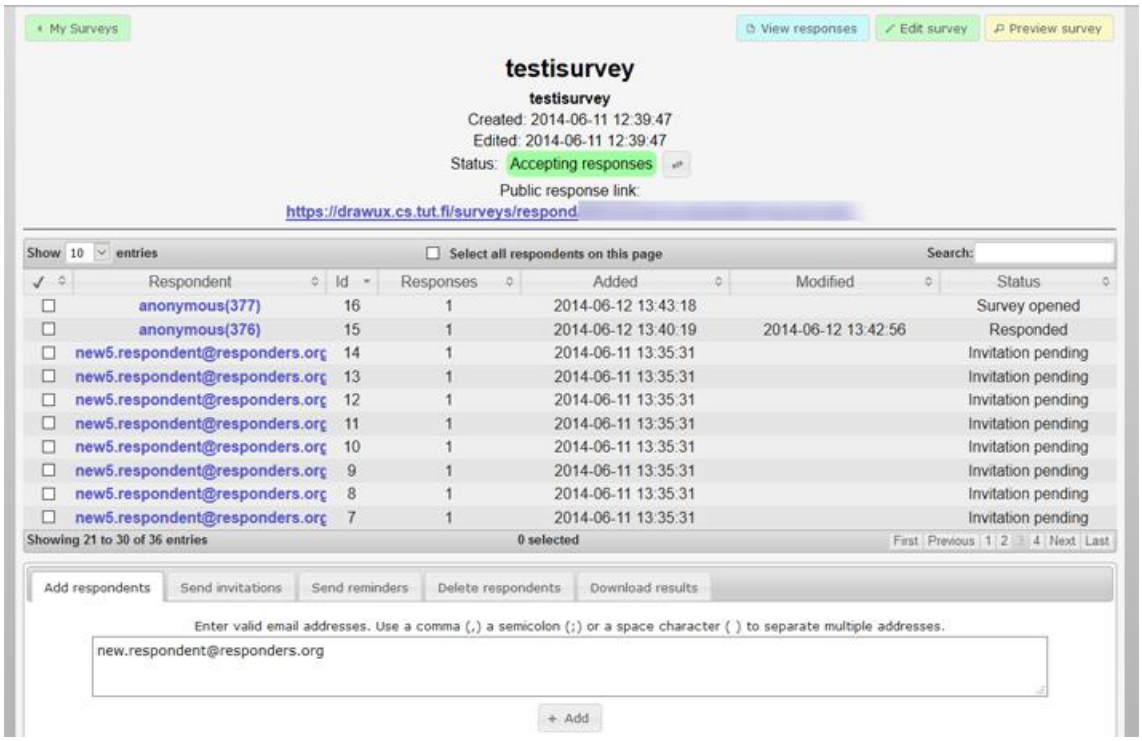
keita, joiden avulla kyselyn voi avata kyselyeditorissa muokattavaksi, kopioida tai poistaa. Sivun ylälaidassa on painike, josta pääsee kyselyeditoriin luomaan kokonaan uuden kyselyn.



Kuva 3. *Omien kyselyjen listaus.*

Kyselyn otsikkoa klikkaamalla käyttäjä pääsee kyselylistauksesta *yksittäisen kyselyn sivulle* (kuva 4), jossa voi tarkastella kyselyyn saatuja vastauksia taulukkolistauksessa. Sähköpostin kautta kutsutut tai kutsuttavat vastaajat listataan sähköpostiosoitteen mukaan. Julkisiin kyselyihin voi kuitenkin vastata myös ilman erikseen lähetettyä linkkiä, jolloin vastaukset ovat anonyymeja. Tällöin ’respondent’-kentässä lukee ’anonymous’ ja vastauksen yksilöivä tunniste.

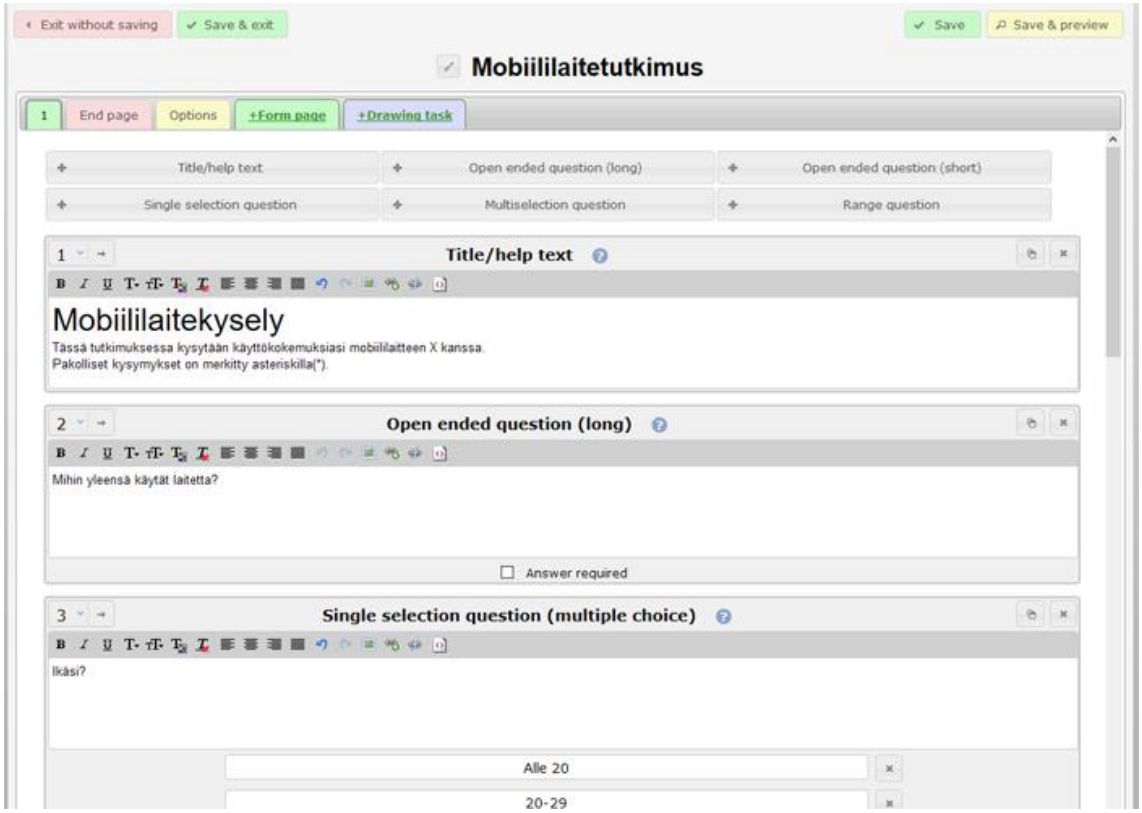
Näkymän alareunassa on kenttä, jonka eri välilehtien kautta tutkija voi lisätä vastaajia sähköpostiosoitteen perusteella vastaajataulukkoon, lähettää kutsuja tai muistutuksia valituille vastaajille, poistaa vastaajia (jolloin myös heidän tallennetut vastauksensa poistuvat tietokannasta), tai ladata Excel-muotoisen yhteenvedon kaikista kyselyyn saaduista vastauksista. Sivulla voi myös muuttaa kyselyn tilaa. Aluksi jokainen luotu kysely on ’Not accepting responses’-tilassa, jolloin vain tutkija itse pääsee käsiksi kyselyyn. Jos joku muu yrittää avata vastausnäkymän, sivulla näkyy vain virheilmoitus. Jos kysely taas on auki, tila on ’Accepting responses’ ja kyselyyn voi vastata normaalisti.



Kuva 4. *Yksittäisen kyselyn sivu.*

Kyselyeditorissa tutkija luo ja muokkaa kyselyjä. Ensimmäiseksi uutta kyselyä luodessaan tutkijan tulee valita kyselyn tyyppi, eli onko kyseessä retrospektiivinen vai päiväkirjatyyppinen kysely. Editori sisältää neljäntyyppisiä välilehtiä. Olennaisimmat ovat lomakesivu- ja piirtotehtävätyyppiset välilehdet, joista kukin edustaa yhtä vastaajalle kokonaisuutena esitettävää kyselysivua. Käyttäjä voi lisätä haluamansa määrän kumpaakin sivutyyppiä.

Lomakesivu (kuva 5) koostuu vapaavalintaisesta määrästä tavallisia kyselyelementtejä. Kuhunkin elementtiin liittyy tekstikenttä, johon tutkija voi kirjoittaa kysymyksen tekstin. Kenttään voi liittää muotoiluja (esimerkiksi fontin tyyppi, väri, koko tai kappaleen tasaus). Kenttään voi myös liittää kuvia tai hyperlinkkejä. Sivulle lisätään elementtejä ylä- ja alareunojen painikeriveistä.

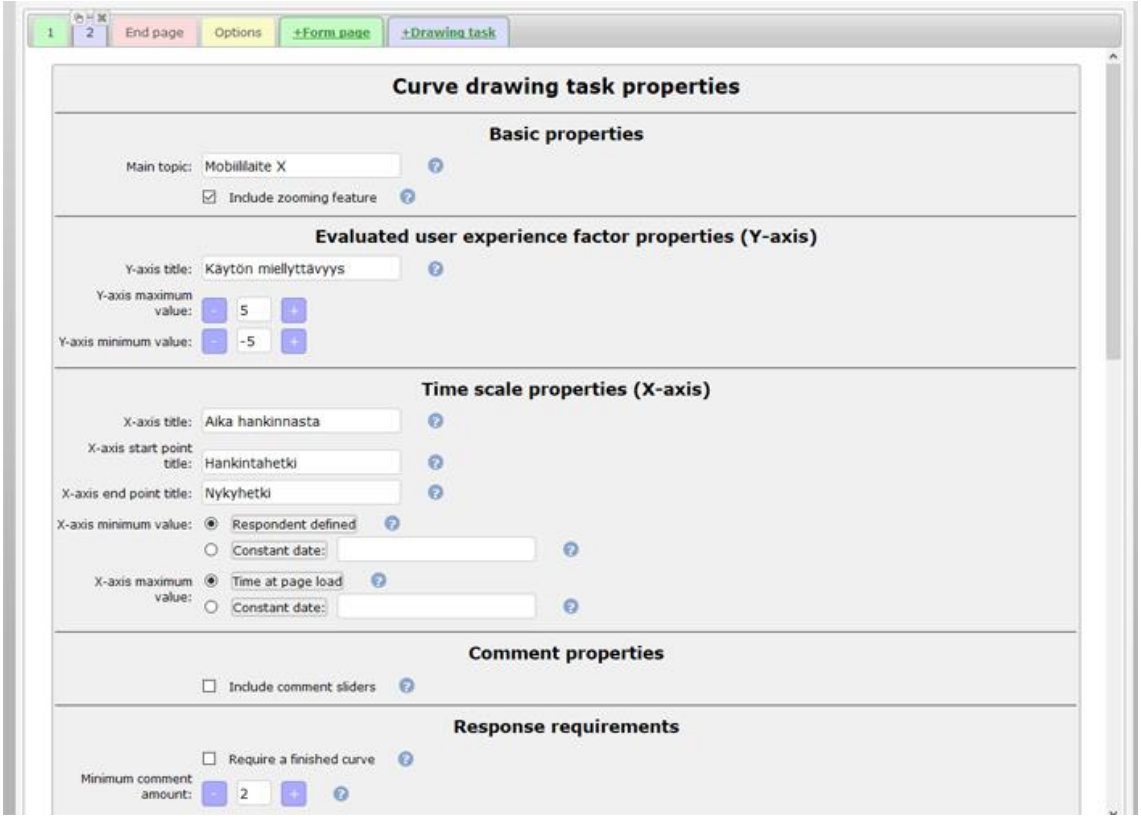


Kuva 5. *Kyselyeditorinäkymä, jossa muokataan lomakesivua.*

DrawUX tarjoaa tällä hetkellä kuusi kysymystyyppiä. Staattinen tekstikenttä on tarkoitettu vastaajan ohjeistukseksi tai kyselyn otsikoksi. Vapaa tekstikenttä on tarkoitettu tekstimuotoisia vastauksia varten. Tekstikentästä on myös pitkä versio, jonka tekstialueeseen mahtuu usean kappaleen verran tekstiä, kun taas lyhyeen versioon sopii muutaman sanan mittainen vastaus. Editori sisältää myös kahdenlaisia monivalintakysymyksiä: radionappitehtäviä, joissa vastaaja voi valita vain yhden annetuista vaihtoehdoista, ja valintaruututehtäviä, joissa voi valita vapaasti haluamansa määrän vaihtoehtoja. Jos kysymys on merkitty pakolliseksi, vastaajan täytyy valita vähintään yksi vastausvaihtoehto. Kumpaankin monivalintakysymystyyppiin voi editorissa lisätä haluamansa määrän vastausvaihtoehtoja, mukaan lukien ’Muu’-vastausvaihtoehdon, johon vastaaja voi itse kirjoittaa annetuista vaihtoehdoista puuttuvan vastauksen. Viimeinen kysymystyyppi on skaalakysymys, jossa käyttäjän tulee asettaa vastauksensa sopivaan kohtaan annettua arvoväliä. Kukin skaalakysymys voi sisältää yhden tai useampia skaaloja, joista jokaisella on oma otsikkonsa. Esimerkiksi tehtävänä voisi olla arvioida samaa tuotetta useiden eri ominaisuuksien suhteen, jolloin kaikki nämä arviot voisivat kuulua saman kysymyksen alle omina skaaloinaan. Kukin skaala voidaan esittää vastaajalle joko viiden tai seitsemän vaihtoehdon radionappirivinä tai vaihtoehtoisesti liukusäätimenä, jonka aseman voi valita vapaasti.

Kunkin kysymyksen voi merkitä haluttaessa pakolliseksi, jolloin vastaajan tulee vastata kysymykseen ennen kuin voi siirtyä kyselyssä eteenpäin. Jos kyseessä on päiväkirjatyyppinen kysely, jokaiseen lomakesivuun voidaan liittää ehto siitä, näytetäänkö sivu joka kerta kun vastaaja avaa kyselyn, vai vain *N* ensimmäisellä vastauskerralla. Tarkoituksena on, että käyttäjää voitaisiin alussa opastaa enemmän (näyttämällä staattisia tekstejä joita ei tarvitse näyttää joka kerta) ja että ensimmäisellä vastauskerralla voitaisiin kerätä vastaajan taustatietoja, jotka eivät muutu vastauskertojen välillä.

Piirtotehtävä on aina yksin omalla sivullaan. Piirtotehtävää edustaa editorissa ’drawing task’-välilehti (kuva 6), jossa piirtotehtävän ominaisuudet voidaan asettaa. Tehtävälle voi määrittää otsikon sekä akselien nimet ja skaalat. Piirtotehtävän aikana käyttäjä saa neuvoja ja palautetta. Näiden palautelaatikkojen tekstit voidaan kustomoida editorissa. Kentillä on kuitenkin englanninkieliset selkeät oletustekstit, jotta tutkijan ei tarvitsisi laatia niitä erikseen joka kyselyä varten.



Kuva 6. *Piirtotehtävän muokkaus editorissa.*

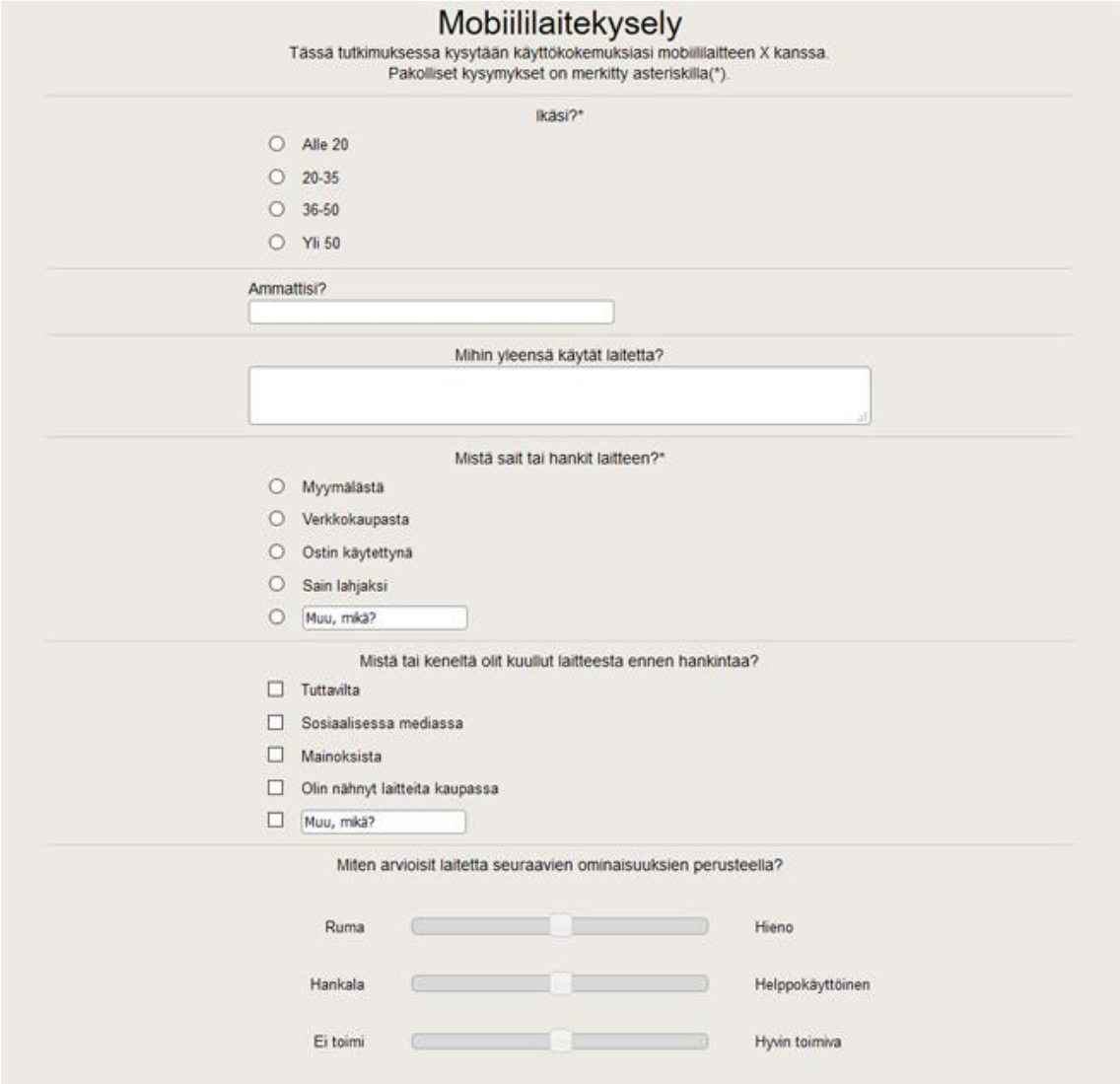
Lisäksi piirtotehtävä voi sisältää vapaavalintaisia ominaisuuksia, jotka voidaan ottaa käyttöön editorissa. Valinnaisiin ominaisuuksiin kuuluvat esimerkiksi zoomaustoiminto, kommenttikenttien liukusäätimet sekä mahdollisuus vaatia minimimäärä ei-tyhjiä kommentteja. Tutkija voi myös määrittää, millä tavalla x-akselin aikaskaala toimii. Oletuksena halutaan tutkia käyttökokemuksen kehittymistä tuotteen hankinnasta nykyhetkeen asti, joten käyttäjän annetaan itse valita käyrän alkuhetki päivän tarkkuudella, ja x-

akselin loppupisteeksi asetetaan sivun lataushetki. Akselille voidaan kuitenkin tarvittaessa asettaa myös kiinteät alku- ja loppupisteet.

Jokaiseen kyselyyn kuuluu automaattisesti loppusivuvälilehti, jossa tutkija voi asettaa kyselyn lopuksi käyttäjälle näkyvän viestin. Täällä tutkija voi myös määrittää verkkosivun, jolle vastaaja uudelleenohjataan kyselyn vastausten tallennuksen jälkeen. Asetukset-välilehti sisältää yleisiä koko kyselyyn liittyviä asetuksia, kuten eri käyttöliittymäelementteihin liittyvät tekstit, kyselyn etenemisen esittämistapa, ja valinta siitä, voiko käyttäjä navigoida edelliselle sivulle vai eteneekö kysely ainoastaan eteenpäin. Myös kyselyn taustan väriä voidaan vaihtaa asetusvälilehdellä.

Lisättyään haluamansa määrän sivuja ja kysymyksiä tutkija voi tallentaa kyselyn ja esikatsella sitä. *Esikatselunäkymässä* tutkija näkee, miltä kysely näyttää vastaajalle. Esikatselunäkymässä kyselyyn ei kuitenkaan voi vielä vastata, eli jos tutkija lähettää vastauksensa esikatselunäkymän kautta, mitään ei tallenneta tietokantaan. Näkymä myös tarjoaa enemmän navigointimahdollisuuksia kuin vastausnäkymä, jotta tutkija pääsisi helposti tarkastelemaan haluamaansa sivua joutamatta esimerkiksi vastaamaan kaikkiin pakollisiin kysymyksiin pitkässä kyselyssä.

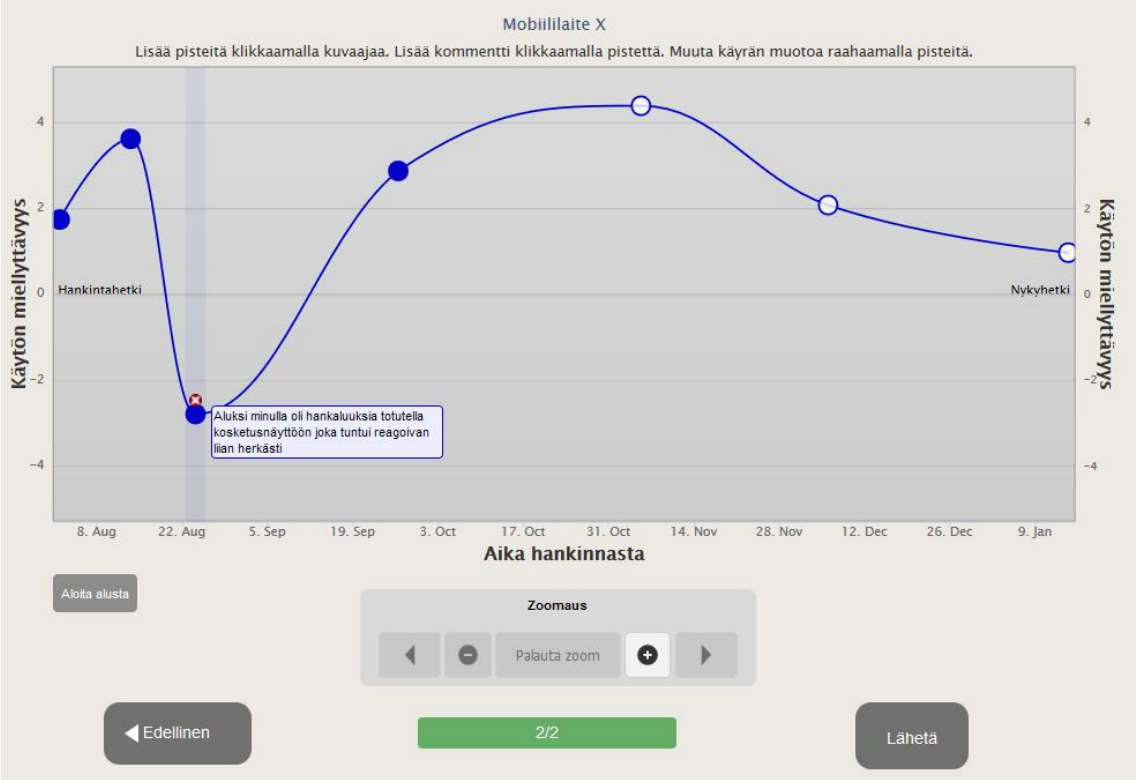
Kyselyn varsinainen *vastausnäkymä* on ainoa muille kuin tutkijoille näkyvä näkymä. Kysely koostuu tavallisista kyselylomakesivuista ja piirtotehtävistä. Lomakesivun elementit näyttävät kuvan 7 mukaisilta. Pakolliset kysymykset merkitään automaattisesti asteriskilla, tutkijan ei siis itse tarvitse lisätä merkintää pakollisten kysymysten perään. Jos käyttäjä yrittää siirtyä seuraavalle sivulle ennen kaikkiin pakollisiin kysymyksiin vastaamista, puuttuvien kenttien alla näytetään punainen virheilmoitus.



Kuva 7. *Eri kysymystyyppejä lomakesivulla vastausnäkymässä.*

Käyränpiirtotehtävä (kuva 8) alkaa yleensä käyttökokemuksen aloituspäivän valinnalla. Vastaajalle näytetään kalenterielementti, josta vastaaja valitsee sopivan päivän. Tämän jälkeen käyttäjälle näytetään ensimmäinen opastusteksti, jossa käyttäjää neuvotaan aloittamaan käyränpiirto vasemmasta reunasta eli käyttökokemuksen alusta. Vastaaja lisää pisteitä kuvaajaan klikkaamalla tyhjää kohtaa kuvaajassa. Kun vastaaja klikkaa lisättyä pistettä, näkymään aukeaa kommenttilaatikko, johon vastaaja voi kirjoittaa tarkemmin pistettä vastaavasta käyttökokemuksesta. Näin saadaan rikasta kvalitatiivista tietoa käyttökokemuksen nousujen ja laskujen syistä itse käyrän muodon tarjoaman kvantitatiivisen tiedon lisäksi [1]. Lisätyn pisteen voi poistaa rastisymbolista, joka näkyy kun osoitin on pisteen päällä. Pistettä voi myös helposti siirtää raahaamalla. Jos tutkija on sallinut käyrän zoomaustoiminnon, vastaaja voi lähentää ja loitontaa kuvaajaa joko kuvaajan alla olevasta zoom controls -painikkeista tai valitsemalla kuvaajasta tietyn aikavälin hiirellä raahaamalla. Valitun välin alku- ja päätepisteet muodostavat lähennetyn osan x-akselin rajat. Toiminnosta on hyötyä erityisen pitkän aikavälin käyttö-

kokemuksen tutkimuksessa, jos vastaaja haluaa ajoittaa kokemuksensa tarkasti. Käyrän piirron voi myös tarvittaessa aloittaa kokonaan alusta.

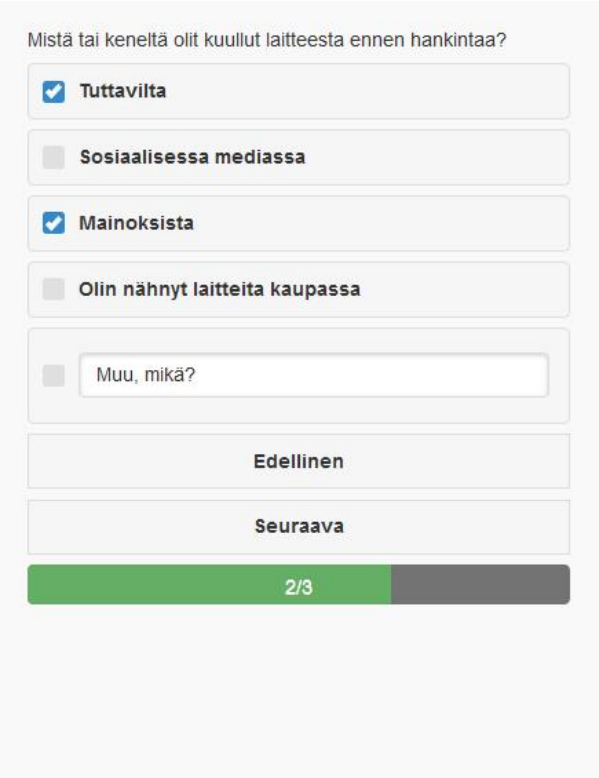


Kuva 8. *Käyränpiirtotehtävä vastausnäkymässä. Pisteelle lisätty kommentti näkyy, kun osoitin viedään pisteen päälle.*

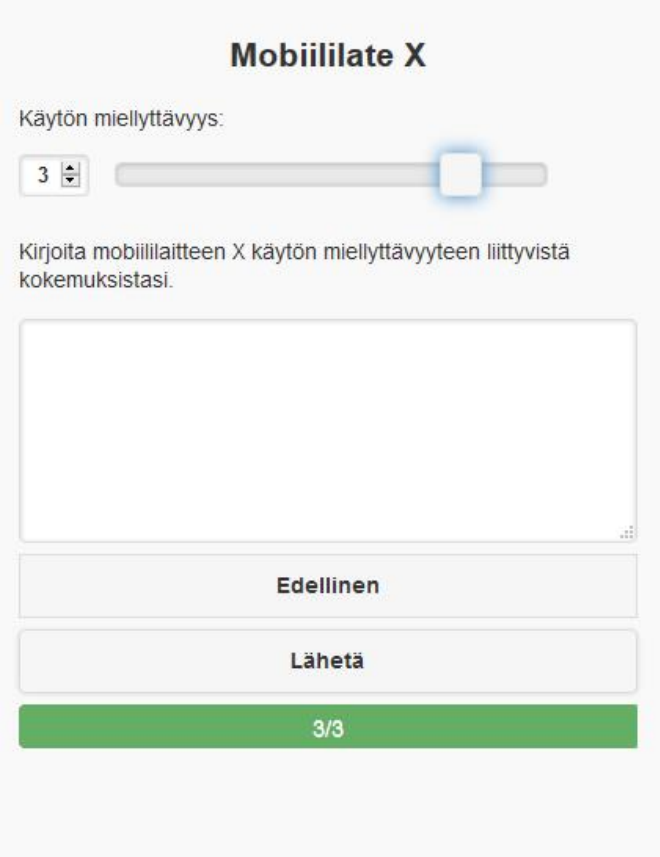
Jos kyseessä on päiväkirjatyyppinen kysely, käyränpiirtotehtävän tilalla näkyy vain liukusäädin, jolla vastaaja voi valita tämänhetkisen käyttäjäkokemuksensa tason, sekä tekstikenttä, johon hän voi kirjoittaa vapaan kuvauksen. Yksi tällainen vastaus vastaa yhtä piirtotehtävään lisättyä pistettä. Liukusäätimelle valittu arvo vastaa piirtotehtävän y-akselin arvoa, kun taas x-akseli eli aika asetetaan automaattisesti vastaamaan vastausten tallennusaikaa. Tarkoituksena on, että vastaaja käy säännöllisesti vastaamassa kyselyyn. Osa kysymyksistä saatetaan kysyä vain valitulla määrällä ensimmäisiä vastauskertoja. Päiväkirjatyyppiseen kyselyyn ei voi vastata julkisen linkin kautta anonyymisti, sillä vastaaja täytyy voida yksilöidä, jotta eri vastauskerrat voidaan liittää samaan kokonaisuuteen.

Päiväkirjatyyppiseen kyselyyn voi vastata myös mobiilisti. Järjestelmä tunnistaa automaattisesti mobiililaitteen ja tarjoaa vastaajalle *mobiilikäyttöliittymän*. Kyselyelementit ovat suunnilleen samanlaisia kuin pöytäkoneella vastattaessa, mutta jokainen kysymys on omalla sivullaan. Näin sivut saadaan pienemmiksi ja vastaaja välttyy turhalta näkymän vierittämiseltä. Kuvassa 9 on esimerkki lomakekysymyksestä (monivalintakysymys) mobiilikäyttöliittymässä ja kuvasta 10 nähdään piirtotehtävän korvaava liukusää-

dinkysymys. Retrospektiiviseen kyselyyn ei toistaiseksi voida vastata mobiililaitteella, sillä käyränpiirtotehtävä vaatii suurempaa näyttöä ollakseen selkeä ja helposti käytettävä.

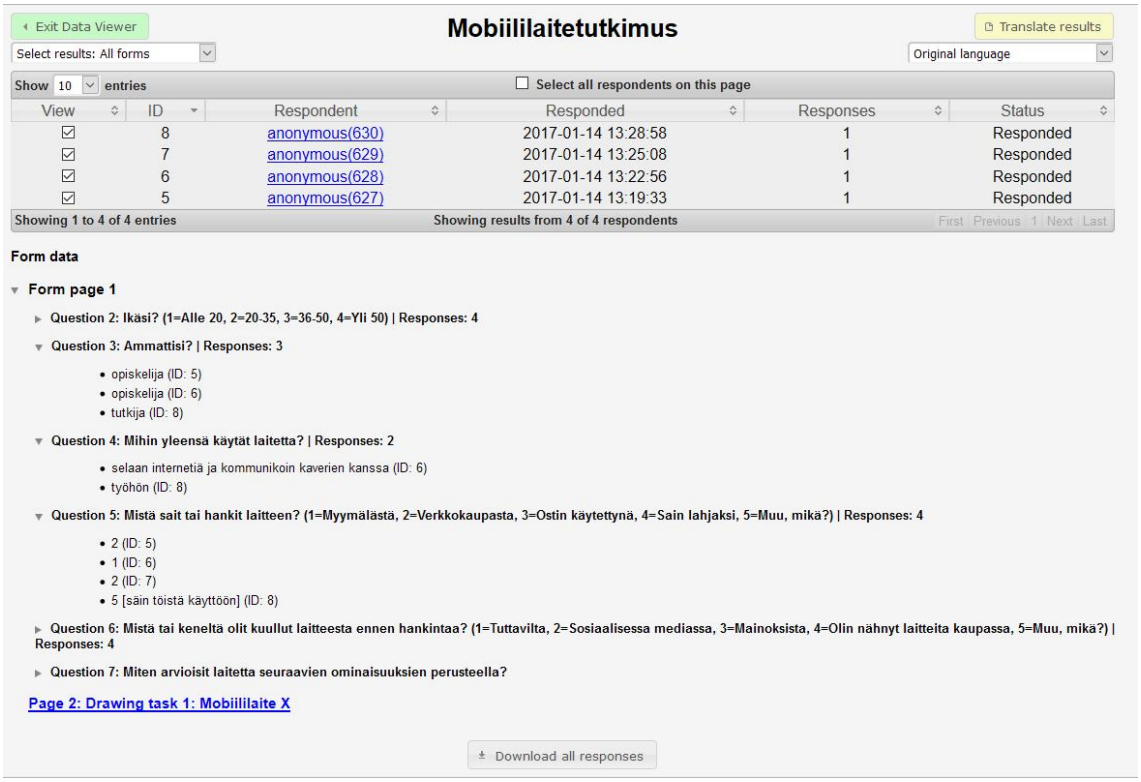


Kuva 9. *Monivalintatehtävä mobiilinäkymässä.*



Kuva 10. *Mobiilinäkymässä päiväkirjatyyppisen kyselyn tehtävä, joka korvaa käyränpiirron. Yksi vastauskerta tehtävään vastaa yhtä käyrälle asetettua pistettä.*

Data Viewer -näkymässä tutkija voi tarkastella kerralla kaikkia kyselyyn saatuja vastauksia. Tarkoituksena on antaa kokonaiskuva vastauksista. Näkymässä kootaan kaikkien kyselyn lomaketyyppisten sivujen kysymysten vastaukset omalle sivulleen, kun taas jokaiselle piirtotehtävälle on oma sivunsa. Lomakekysymysten yhteenvedossa (kuva 11) vastaukset on ryhmitelty lomakesivun ja kysymyksen mukaan. Jokaisen ryhmän voi näyttää tai piilottaa tarpeen mukaan. Näin on helppo pitää näkymä selkeänä jos halutaan tarkastella vain yhtä kysymystä kerrallaan. Kustakin vastauksesta näytetään vastaajan ja vastauskerran tunnisteet sekä itse vastauksen teksti tai valittu arvo. Sivun yläreunassa on taulukkomuotoinen esitys vastauksista. Vastaajista voi valita, keiden vastausten halutaan näkyvän tarkastelussa. Koska vastauksia voi olla paljon, oletuksena vastaajista on valittuna vain viisi uusinta.

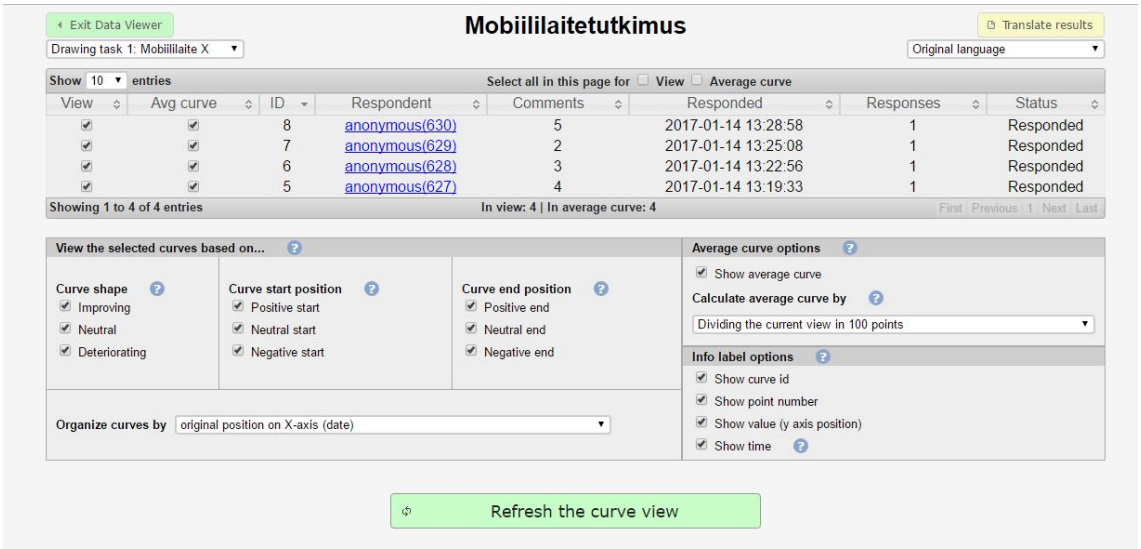


Kuva 11. *Data Viewer -sivulla voi tarkastella kyselyyn saatuja vastauksia.*

Käyränpiirtotehtävää vastaavalla Data Viewer -sivulla nähdään valittujen vastaajien piirtämät tai päiväkirjatyyppisen kyselyn vastauskertojen pisteistä muodostetut käyrät kerralla samassa kuvaajassa. Näin käyriä voidaan helposti verrata toisiinsa ja nähdä silmämääräisesti trendejä. Näkymä koostuu kolmesta alueesta. Ylhäällä on lukuisia asetuksia, joilla kuvaajaa voidaan säätää (kuva 12). Keskellä sivua on kuvaaja-alue, johon käyrät piirtyvät (kuva 13). Alue muistuttaa vastausnäkymän kuvaajakenttää, mutta tutkija ei voi lisätä pisteitä tai muutenkaan muutella käyrien muotoa. Kenttään liittyy päivitystpainike, jonka painaminen ottaa käyttöön muutokset asetuksissa. Sivun alareunassa on tietolaatikko, jossa oletuksensa listataan kaikkien käyrien pisteet. Tällä tavalla pisteiden kommentteja ja muita yksityiskohtia voidaan tarkastella kerralla, ja niiden joukosta voidaan etsiä kommentteja hakutoiminnolla.

Kuvassa 12 näkyvien säätimien avulla käyriä voidaan suodattaa muodon, alkupisteen paikan ja loppupisteen paikan pohjalta. Esimerkiksi tutkija voi haluta tarkastella ainoastaan nousevia käyriä tai käyriä, joiden alkupiste on negatiivinen. Suotimet vaikuttavat ainoastaan vastaajiin, jotka on valittu näkyviksi. Asetuksissa on myös valinta sille, millä tavalla käyrät piirretään toisiinsa nähden. Käyrät voidaan asettaa yhteiselle aikaasteikolle tai ne voidaan siirtää alkamaan samasta hetkestä tai lopettaa samaan hetkeen. Kuitenkin käyrien muoto pysyy joka tapauksessa samana.

Käyristä näytetään oletuksena myös keskiarvokäyrä, joka laskee käyrien pisteistä keskiarvon ja näyttää tuloksen paksuna punaisena käyränä kuvaajassa. Keskiarvokäyrän tarkoituksena on auttaa tutkijaa havaitsemaan yleisiä trendejä vastauksissa. Keskiarvokäyrän voi kuitenkin piilottaa ja sen laskutapaa voi muuttaa asetuksista. Vastauslistauksesta pystyy valitsemaan myös erikseen, mitkä vastaukset vaikuttavat keskiarvokäyrään. Tästä voi olla hyötyä, jos vastauksia on hyvin paljon eikä kaikkia haluta näyttää kuvaajassa kerralla, jolloin voi olla selkeämpää näyttää ainoastaan keskiarvokäyrä, johon kaikki käyrät on laskettu mukaan.

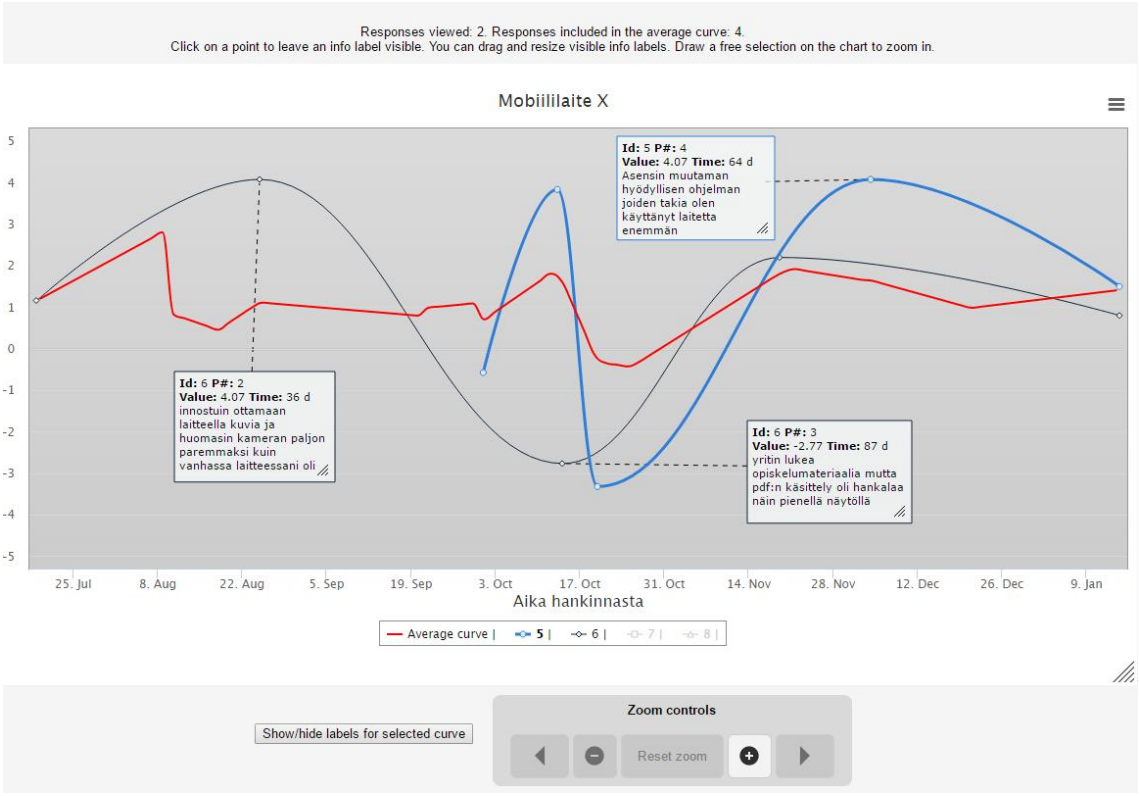


Kuva 12. *Data Viewerin asetuksia käyränäkymälle.*

Kuvan 13 mukaisessa kuvaaja-aluueessa käyrän voi valita joko klikkaamalla itse käyrää tai käyrän symbolia kuvaajan alla sijaitsevasta kuvatekstilaatikosta. Valittu käyrä tuodaan etualalle, näytetään paksumpana ja sen tiedot näytetään alareunan tietolaatikossa. Jokaiseen käyräpisteeseen liittyy myös tietoruutu, jossa näkyy oletuksena käyrän tunniste, pisteen järjestysnumero käyrällä, pisteen y-arvo sekä pisteeseen liittyvä aika (kuinka kauan tuotetta oli käytetty kyseiseen ajankohtaan mennessä). Nämä tiedot voi ottaa käyttöön tai piilottaa kuvan 12 asetuksista. Lisäksi jokaisessa tietoruudussa näytetään pisteeseen liittyvä kommentti. Pisteeseen liittyvä tietoruutu näytetään, kun käyttäjä vie osoittimen pisteen päälle. Kun tutkija klikkaa käyrän pistettä, pisteen tietoruutu jää näkyviin. Kuvan selkeyttämiseksi ruutuja voi myös siirrellä kuvaajassa ja niiden kokoa voi säätää vapaasti, kuten kuvassa 13 on tehty. Valittuna olevan käyrän kaikki ruudut voidaan näyttää tai piilottaa ’show/hide labels for selected curve’-painikkeella.

Kuvaajan oikeasta ylänurkasta aukeaa valikko, josta sillä hetkellä näkyvän kuvaajan voi tulostaa tai tallentaa eri muodoissa (.png, .jpg, .pdf, .svg). Data Viewerin oikeassa yläreunassa on kielivalikko, jonka kautta voidaan valita, näytetäänkö kommentit ja muut tekstit alkuperäiskielellä vai käännöksinä. Kielen vaihto vaikuttaa kaikkiin piirtotehtä-

viin ja lomakesivujen yhteenvetoon kerralla. ’Translate results’ -painikkeella voidaan siirtyä kääntönäkymään.



Kuva 13. *Käyränäkymä Data Viewerissä. Punaiseen keskiarvokäyrään on laskettu mukaan myös piilotetut käyrät. Joitakin tietoruutuja on tuotu näkyviin.*

Kääntönäkymän tarkoituksena on antaa käyttäjälle mahdollisuus kääntää kyselyyn saatuja vastauksia toiselle kielelle. Toistaiseksi järjestelmä tukee vain yhtä käännöstä. Käännetyt tulokset voidaan nähdä Data Viewer -sivulla sekä ladattavassa Excel-yhteenvedossa. Kääntönäkymä oli asiakkaan toive, sillä asiakkaalla oli suunnitteilla tutkimuksia Virossa ja Espanjassa.

Uutiset-sivulla käyttäjä voi lukea lyhyitä uutisia järjestelmän päivityksistä. Kehittäjän on tarkoitus julkaista näitä uutisia aina kun järjestelmään on lisätty uusia ominaisuuksia tai merkittäviä virheitä on korjattu. Näkymä on kuitenkin sama kaikille käyttäjille, joten toistaiseksi kehittäjän täytyy syöttää uutistekstit manuaalisesti tietokantaan.

Palautesivu sisältää lomakkeen, jonka kautta käyttäjä voi ottaa yhteyttä järjestelmästä vastaavaan henkilöön. Lomaketta on tarkoitus käyttää kaikenlaisen palautteen antoon, kuten löytyneistä virheistä kertomiseen ja jatkokehitystoiveisiin. Kun lomake lähetetään, siihen kirjoitettu teksti lähetetään sähköpostilla järjestelmästä vastaavalle henkilölle.

Näiden sivujen lisäksi DrawUX sisältää joitakin lähes kaikille web-järjestelmille yhteisiä sivuja, jotka liittyvät käyttäjätilien hallintaan. Tällaisia näkymiä ovat mm. rekisteröitymislomake, sisäänkirjautumissivu, tunnuksen vaihto ja unohtuneesta salasanasta ilmoittaminen. Nämä yksinkertaiset sivut eivät ole mielenkiintoisia DrawUX:n ydintoiminnallisuuden kannalta.

4. TEKNINEN TOTEUTUS

DrawUX on tyypillinen web-sovellus, jossa on käytetty useita eri selain- ja palvelinohjelmoinnin teknologioita. Eräs tärkeimmistä toteutusta ohjanneista periaatteista oli, että järjestelmää tulee voida kehittää nopeasti. Teknologiavalinnat olivat siis avainasemassa, sillä valittujen ohjelmointikielten ja kolmannen osapuolen komponenttien tulisi tukea nopeaa kehitystä tarjoamalla valmiina mahdollisimman paljon tarvittavia ominaisuuksia sekä sopivia kustomointi- ja laajennusmahdollisuuksia. Lisäksi teknologioiden tulisi olla syntaksiltaan ja käyttöönotoltaan mahdollisimman helppoja. Myös tarkoituksenmukainen arkkitehtuuri on keskeisessä asemassa, jotta keskisuuresta web-järjestelmästä saadaan järkevästi hallittava ja toimiva kokonaisuus.

Tässä luvussa kuvataan, miten DrawUX on toteutettu. Luku 4.1 esittelee käytetyt teknologiat ja luvussa 4.2 käsitellään järjestelmän arkkitehtuuria. Luvussa 4.3 kerrotaan, mitkä järjestelmän osat ja toiminnot kuuluivat tähän jatkokehitysprojektiin, ja luku 4.4 kuvaa käytännön toteutusprosessia tarkemmin.

4.1 Teknologiavalinnat

Teknologiavalinnat teki enimmäkseen projektin alkuperäinen kehittäjä. Valinnat tehtiin sillä perusteella, että työkalu on web-käyttöinen, joten kielet ja tekniikat ovat yleisiä web-teknologioita. Lisäksi teknologioiden tuttuus vaikutti päätöksiin.

Järjestelmän pääasiallinen ohjelmointikieli on *PHP (PHP: Hypertext Preprocessor)*. Ensimmäisen kerran vuonna 1995 julkaistu PHP on komentosarjakieli, joka soveltuu erityisesti web-ohjelmointiin [25]. Kielen hyviä puolia ovat muun muassa joustavuus ja yksinkertaisuus sekä tuki tärkeimmille käyttöjärjestelmille ja tietokannoille. PHP kuuluu edelleen suosituimpiin ohjelmointikieliin, ks. esim. [26], [27]. PHP:n kaltaisen yleisessä käytössä olevan kielen valinta on kannattavaa, koska tällöin projektiin on helppo löytää osaavia kehittäjiä.

Näkymät ovat tavallisia HTML-sivuja, paitsi mobiilinäkymä, joka on toteutettu kokonaan HTML5:lla. Näkymissä käytetään *jQuery*-kehystä [28]. jQuery on ilmainen avoimen lähdekoodin JavaScript-kirjasto, joka sisältää lukuisia selainohjelmointia helpottavia ominaisuuksia. jQuery helpottaa muun muassa HTML-elementtien valintaa ja käsittelyä, tapahtumankäsittelyä ja Ajax-kommunikointia. jQuery on hyvin dokumentoitu ja lisäksi niin yleisessä käytössä, että sen syntaksiin ja käyttöön löytyy helposti apua tarvittaessa.

Palvelinpuolella DrawUX:ssa on käytetty *CodeIgniter*-ohjelmistokehystä (versio 2.1.3). EllisLabin kehittämä, nykyään British Columbia Institute of Technologyn omistama CodeIgniter on kevyt PHP-kehys web-ohjelmointiin [29]. CodeIgniteria voidaan kuvata avoimen lähdekoodin kehykseksi ja se pyrkii olemaan erityisesti kevyt, pienen jalanjäljen ohjelmistokehys, joka panostaa myös erityisen hyvälaatuiseen dokumentointiin [30]. Lisäksi CodeIgniter on ilmainen käyttää, hyvin tuettu, helposti laajennettavissa, ja tarjoaa useita hyödyllisiä apukirjastoja ja luokkia sekä parantaa tietoturvaa esimerkiksi suojaamalla automaattisesti SQL-injektiolta [31]. Ohjelmistokehyksen käytöstä ylipäätään on runsaasti hyötyä erityisesti nopean kehityksen kannalta, joka oli tässä projektissa avainasemassa. Kun ohjelmistokehys tarjoaa web-sovelluksille yhteiset perusasiat ja hyvän arkkitehtuurin, ohjelmoija säästää paljon aikaa ja voi keskittyä olennaiseen sovelluslogiikkaan [31]. Lisäksi valmiin kehyksen voi luottaa olevan toimiva ja kunnolla testattu, joten myös testausaikaa säästyy. Sovelluskehyksissä on hyvä rakenne ja hyväksi havaittuja ohjelmointimalleja [31], jotka auttavat välttämään monia yleisiä web-antipatterneja [32]. CodeIgniterin lisäksi muita PHP-MVC-kehyksiä olisivat esimerkiksi Zend ja CakePHP. Pittin [30] mukaan CodeIgniter on näitä yksinkertaisempi, mutta tarjoaa oletuksena vähemmän toimintoja. Kehystä onkin DrawUX:n kehityksessä jouduttu laajentamaan useilla kolmansien osapuolten kirjastoilla. Matalan käyttöönottokynnyksensä vuoksi vain olennaisuuksiin keskittyvä CodeIgniter oli kuitenkin varmasti aikoinaan järkevin ratkaisu DrawUX:n, sillä kehitys haluttiin saada nopeasti käyntiin, joten helposti opittava kehys oli tarpeen. Nykyään CodeIgniterin versio 2 on jo legacy-ohjelmisto ja uudempi versio 3 olisi myös tarjolla [29].

Tietojen tallennusta varten käytössä on *PostgreSQL*-tietokanta. PostgreSQL on luotettavana tunnettu avoimen lähdekoodin oliorelaatiotietokannan hallintajärjestelmä [33]. PostgreSQL tukee kaikkia olennaisia tietokantaominaisuuksia, kuten ACID-toimintoja (atomisuus, eheys, eristyneisyys, pysyvyys), vierasavaimia, liitoksia, herättimiä ja tallennettuja proseduureja. PostgreSQL on myös helppo ottaa käyttöön eri alustoilla, mistä on ollut hyötyä DrawUX-järjestelmää rakennettaessa ja siirrettäessä.

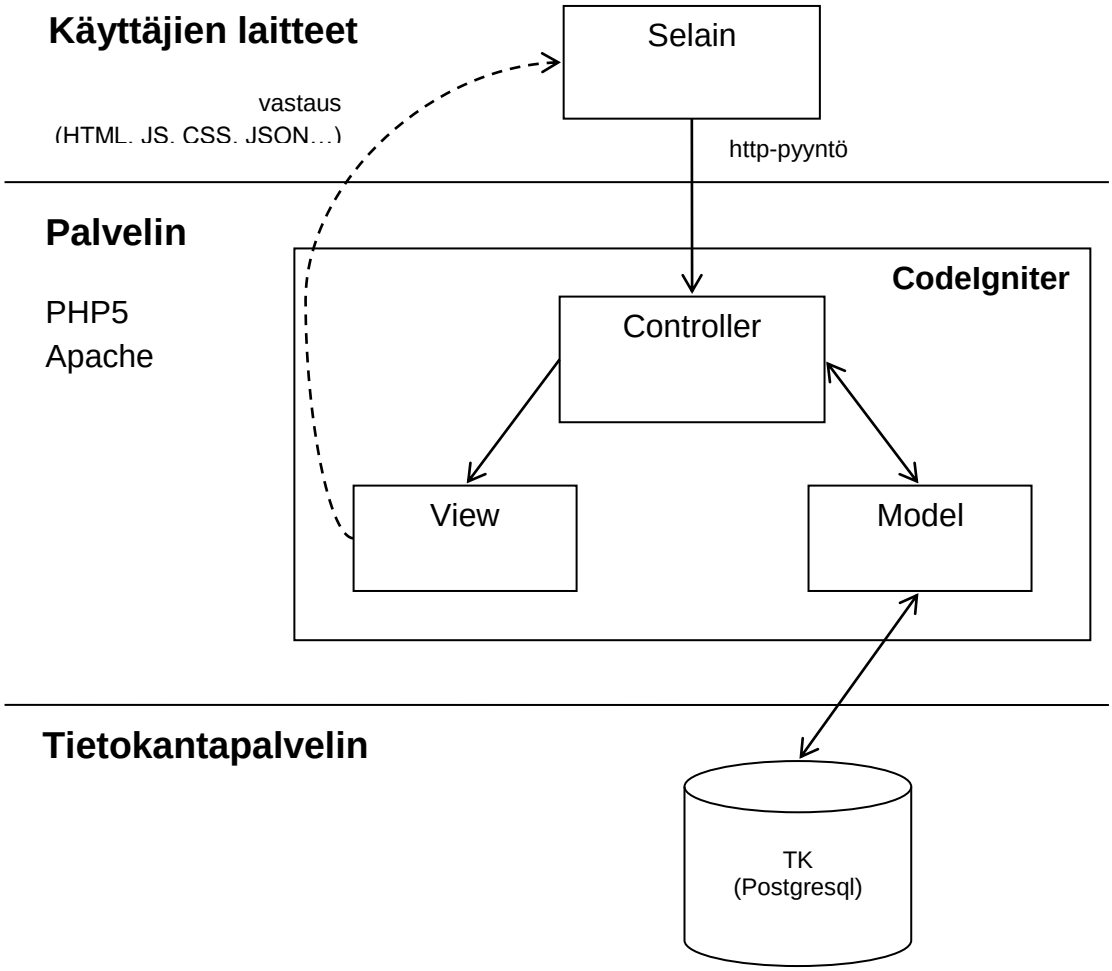
Piirtotehtävässä ja Data Viewer -näkymässä käytettävät kaaviot on toteutettu Highsoftin *Highcharts*-kirjastolla. Vuonna 2009 julkaistu Highcharts on puhtaalla JavaScriptilla kirjoitettu kirjasto interaktiivisten diagrammien piirtoon selaimessa. Highcharts on julkaisustaan asti ollut kaavionpiirtotyökalujen kärkeä. Kirjasto on hyvin dokumentoitu ja tarjoaa lukuisia kustomointi- ja laajennusmahdollisuuksia, mistä on DrawUX-projektin aikana ollut erityistä hyötyä. [34]

Palvelimen ja selaimen väliseen kommunikointiin on joissakin näkymissä käytetty *JSON*-formaattia. JSON (JavaScript Object Notation) on kevyt ja helposti luettava tiedonsiirtoformaatti, jossa data kuvataan lähinnä objekteina ja taulukoina. Muunnokset JSONin ja JavaScript-objektien välillä käyvät erittäin helposti. JSON käyttää useista ohjelmointikielistä tuttua syntaksia, minkä vuoksi formaatti on helppo käyttää ja ideaalinen valinta tiedonsiirtoon DrawUX-järjestelmässä. [35]

Lisäksi käytössä on useita pienempiä komponentteja. Ladattavan kyselytulostaulukon rakentamiseen käytetään *PHPExcel*-lisäosaa [36]. Syötteen tarkistukseen on käytetty *HTML Purifier* -kirjastoa [37]. Näkymät jotka käyttävät taulukoita on useimmiten toteutettu *DataTables*-lisäosan [38] avulla. Kyselyeditorin tekstikenttiin on käytetty *CLEditor*-lisäosaa [39] ja taustavärinvalintakomponenttina toimii *colpick Color Picker 2.0.2* [40].

4.2 Arkkitehtuuri

DrawUX on toteutettu perinteisenä web-sovelluksena, joten kyseessä on asiakas-palvelin-arkkitehtuuri. DrawUX on rakennettu pitkälti CodeIgniterin tarjoaman MVC-arkkitehtuurimallin pohjalta. DrawUX on sijoitettu webhotelliin, jossa on käytössä Linux-käyttöjärjestelmä, Apache-palvelinohjelmisto ja PHP 5. Tietokanta sijaitsee omalla tietokantapalvelimellaan, mikä auttaa skaalautuvuutta, sillä palvelimen ei tarvitse yhtä aikaa käsitellä käyttäjien kutsuja ja suorittaa tietokantaoperaatioita. Selain-palvelin-kommunikointi pohjautuu enimmäkseen lomakkeenlähetykseen tai Ajaxiin. Nämä tekniikat ja mallit on esitelty tarkemmin seuraavissa alaluvuissa.



Kuva 14. *DrawUX-järjestelmän arkkitehtuuri yleisellä tasolla. Todellisuudessa järjestelmässä on useita käsittelijöitä, malleja ja näkymiä.*

Kuva 14 esittelee järjestelmän korkean tason arkkitehtuurin. Kuvasta nähdään selvä MVC-rakenne, joka esitellään tarkemmin luvussa 4.2.2.

4.2.1 Web-pohjainen asiakas-palvelin-arkkitehtuuri

Järjestelmän korkean tason arkkitehtuurimallina on asiakas-palvelin-arkkitehtuuri. Arkkitehtuurin toimintaperiaatteena on yksittäisen käyttäjän tietokoneella pyörivä asiakassovellus, tässä tapauksessa Internet-selain, ja palvelimella sijaitseva sovellus, joka kuuntelee asiakaskomponenttien palvelupyyntöjä ja vastaa niihin. Asiakassovellus lähettää palvelupyynnön palvelimelle jotakin väylää pitkin, internet-sovelluksessa HTTP-protokollaa noudattaen Internetin kautta. Palvelin hyväksyy tai hylkää kutsun ja lähettää vastauksen. Palvelin tarjoaa yleensä palveluja useammalle kuin yhdelle asiakkaalle. [41]

Periaatteessa muitakin vaihtoehtoja olisi kuin web-sovellus. DrawUX voisi toimia omalle tietokoneelle asennettavana ohjelmana, jolloin vastaajille ja tutkijoille tarvittai-

siin omat sovellukset. Tässä kuitenkin huonona puolena olisivat käyttäjien erilaiset ympäristöt, eli sovelluksesta tulisi tarjota tärkeimpiin käyttöjärjestelmiin sopivat versiot. Lisäksi sovelluksen käytöstä ja käyttöönotosta tulisi huomattavasti hankalampaa. Eräs web-sovelluksen hyvistä puolista on helppo käyttöönotto – mitään ei tarvitse asentaa, vain web-selain ja internet-yhteys tarvitaan [41]. Käyttäjän, eritysesti vastaajan, kannalta on helpoin vain seurata lähetettyä linkkiä ja käyttää web-sivua vastaamiseen. Myös kehittäjän kannalta on helpointa ylläpitää ja jatkokehittää yksittäistä web-sivua sen sijaan, että ohjelmasovelluksiin pitäisi ohjelmoida päivitystenhaku. Oli siis perusteltua toteuttaa sovellus web-pohjaisena asiakas-palvelin-arkkitehtuurina.

4.2.2 MVC

MVC-arkkitehtuuri on ohjelmointimalli, jossa tietojen hallinta, esitystapa ja sovellusaluelogiikka pidetään erossa toisistaan pitämällä ne omissa loogisissa komponenteissaan [31]. Näitä kolmea pääkomponenttityyppiä kutsutaan vastaavasti *malleiksi* (*model*), *näkymiksi* (*view*) ja *käsittelijöiksi* (*controller*). MVC-arkkitehtuuri sopii hyvin web-sovellukseen, sillä tyypillisessä web-sovelluksessa tarvitaan kaikkia MVC-mallin peruskomponentteja: näkymiä web-sivujen esittämiseen, käsittelijöitä sovellusaluelogiikkaan eli käyttäjän pyyntöjen ja datan käsittelyyn, ja malleja tietokannanhallintaan. Moderneissa ohjelmistokehyksissä, kuten CodeIgniterissa, MVC-arkkitehtuuri toteutetaan yleensä tiedostoihin perustuen, eli kukin malli, käsittelijä ja näkymä on oma tiedostonsa.

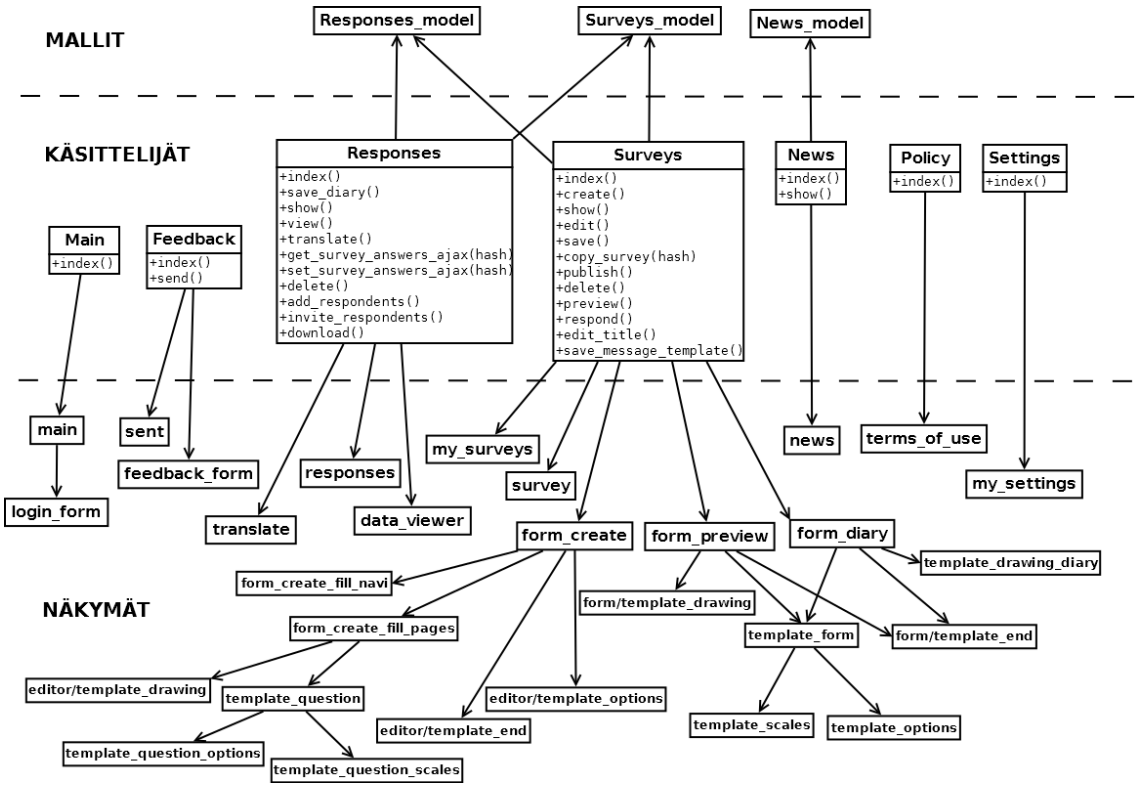
Malli kapseloi datan tallennukseen liittyvät asiat ja abstrahoi tietokannan. DrawUX:ssa on omat mallinsa muun muassa kyselyille ja vastauksille. Kukin DrawUX:n malli on luokka, joka tarjoaa rajapinnan johonkin loogisesti yhtenäiseen kokonaisuuteen liittyvien tietojen käsittelyyn. Muualla kuin malleissa ei käytetä tietokantaa suoraan.

Näkymä vastaa käyttöliittymän ulkoasusta ja tietojen esityksestä käyttäjälle. Näkymiin voi kuulua mm. HTML-merkkausta, CSS-muotoilua ja JavaScript-koodia. DrawUX:n näkymät eivät ole staattisia tiedostoja, vaan CodeIgniterissa näkymät rakennetaan PHP:n avulla dynaamisesti. Useat järjestelmän tärkeistä toiminnoista, kuten käyränpiirto ja kyselyn rakennus, suoritetaan kokonaan selaimessa ja tallennetaan vasta toiminnon lopussa kokonaisuutena palvelimelle.

Käsittelijä yhdistää näkymät ja mallit. Kaikki kommunikointi näkymän ja mallin välillä kulkee käsittelijän kautta. Käsittelijä reagoi käyttäjän näkymän kautta lähettämiin pyyntöihin ja kutsuu malleja tarvittaessa esimerkiksi tietojen tallentamiseen. Kaikki laskenta, joka ei kuulu varsinaiseen ydintoiminnallisuuteen, pyritään suorittamaan käsittelijöissä, jotta mallien funktiot voitaisiin pitää mahdollisimman yleiskäyttöisinä. Tällaisia käsittelijöihin sijoitettavia toimintoja ovat muun muassa tietokannasta saatujen tietojen muotoilu näkymään sopiviksi ja vastaavasti käyttäjältä saatujen tietojen muotoilu tietokantaan sopiviksi.

MVC-arkkitehtuuri antaa sovellukselle loogisen rakenteen, jossa eri osat pidetään tiiviisti erillään. Näin muutokset on helppo rajata yhteen komponenttiin, esimerkiksi tietokannan muutokset rajoittuvat vain malleihin. Rakenteettomassa sovelluksessa pienetkin muutokset saattavat vaikuttaa koko sovellukseen ja tarvitaan enemmän työvoimaa varmistamaan, etteivät muutokset yhteen osaan riko muuta toiminnallisuutta. Kannattaa kuitenkin huomata, että MVC-arkkitehtuurin hyödyt saadaan käyttöön vain, jos eristämisen periaatetta noudatetaan tarkasti ja koodin osat asetetaan malleihin, käsittelijöihin ja näkymiin harkiten. Pelkän MVC-kehyksen käyttö ei yksinään takaa hyvää arkkitehtuuria, jos MVC:n periaatteissa ei pysytä. [30]

Kuva 15 esittelee DrawUX:n sisältämät mallit, näkymät ja käsittelijät ja niiden väliset yhteydet. Nuoli käsittelijästä malliin tarkoittaa sitä, että käsittelijä käyttää mallia sen tarjoaman rajapinnan kautta. Malleja ei voi käyttää muualta kuin käsittelijöistä käsin. Nuoli käsittelijästä näkymään tarkoittaa, että käsittelijä voi ladata kyseisen näkymän ja palauttaa sen käyttäjän selaimelle vastauksena HTTP-pyyntöön. Nuoli näkymästä toiseen näkymään kertoo, että kutsuva näkymä sisällyttää itseensä alinäkymän, joka upotetaan kutsuvan näkymän koodiin.



Kuva 15. *DrawUX:n MVC-rakenne tiedostotasolla.*

Olennaisia sovelluskohtaisia malleja on vain kolme. *Surveys_model* sisältää kyselyihin liittyvien tietojen, kuten kyselyn kysymysten, otsikon ja muiden ominaisuuksien, käsittelyyn tarvittavia funktioita. *Responses_model* puolestaan sisältää vastauksiin ja vastauskertoihin liittyviä funktioita. Kun käyttäjä vastaa kyselyyn, annetut vastaukset tallennetaan responses_modelin kautta. *News_model* on melko triviaali malli, joka sisältää funktion ainoastaan uutisten hakemiseen tietokannasta. Olennaisia käsittelijöitä on seitsemän. *Main*, *Feedback*, *News*, *Policy* ja *Settings* ovat yksinkertaisia käsittelijöitä, jotka on tarkoitettu lähinnä etusivun, palautesivun, uutissivun, käyttöehdot-sivun sekä asetussivun lataamiseen. Järjestelmän suurimmat ja olennaisimmat käsittelijät ovat *Responses* ja *Surveys*, jotka kokoavat yhteen vastaavasti vastausten ja kyselyjen käsittelyyn tarvittavia operaatioita. Monimutkaisin tiedostorakenne on näkymillä, joissa koodi on pilkottu lukuisiin pienempiin template-tiedostoihin. Tällä tavalla on helpompi hallita suuria kokonaisuuksia, kuten editorinäkymää ja vastausnäkymää, jotka koostetaan useista erityyppisistä kysymys- ja tehtäväkentistä.

Kuvasta 15 on selkeyden vuoksi jätetty joitakin vähemmän tärkeitä yksityiskohtia pois. Kuvaan merkittyjen näkymien lisäksi lähes kaikissa näkymissä on sisällytettynä myös *header*- ja *footer*- alinäkymät. Kirjautumiseen, rekisteröintiin ja muuhun tunnusten- ja sessionhallintaan liittyvät osat jätettiin kuvaamatta, sillä näitä tiedostoja on paljon ja ne on toteutettu enimmäkseen valmiin *Auth*-kirjaston avulla. Pois on jätetty myös Surveys-käsittelijän lataama mobile/form-näkymä, joka vastaa form_diary-näkymää hyödyntäen samankaltaista alitiedostorakennetta (mobile/template_drawing, mobile/template_end jne.).

CodeIgniterissa jokaista sivun URL-osoitetta vastaa jokin näkymä ja näkymän sisällä yksittäinen funktio. Lisäksi osoitteen loppuun voi liittää funktiolle annettavia parametreja. Esimerkiksi kun selain ohjataan osoitteeseen *drawux.tut.fi/surveys/respond/<hash>* , kutsu ohjautuu Surveys-käsittelijälle, joka on käytännössä *Surveys*-niminen PHP-luokka. Luokan sisällä kutsutaan *respond*-funktiota, jolle URL:n loppuosa, eli kyselyn yksilöivä tiiviste, annetaan parametrina. Jokaisella käsittelijällä on myös *index()*-funktio eli oletusfunktio, jota kutsutaan, kun URL-osoitteeseen ei sisälly alisivua. Esimerkiksi selaimen ohjaaminen osoitteeseen https://drawux.cs.tut.fi/surveys vastaa Surveys-käsittelijän index()-funktion kutsumista. Tämä CodeIgniterin toteutus osoitteen ja funktion väliselle reititykselle on eräs syy siihen, miksi suuria käsittelijöitä ei ole pilkottu pieniksi osiksi kuten näkymiä. Tätä automaattista reititystä voisi myös säätää omien reitityssääntöjen avulla, mutta käytännössä tämä hämärtäisi osoitteen ja tiedoston välistä yhteyttä tehden ylläpidosta hankalampaa.

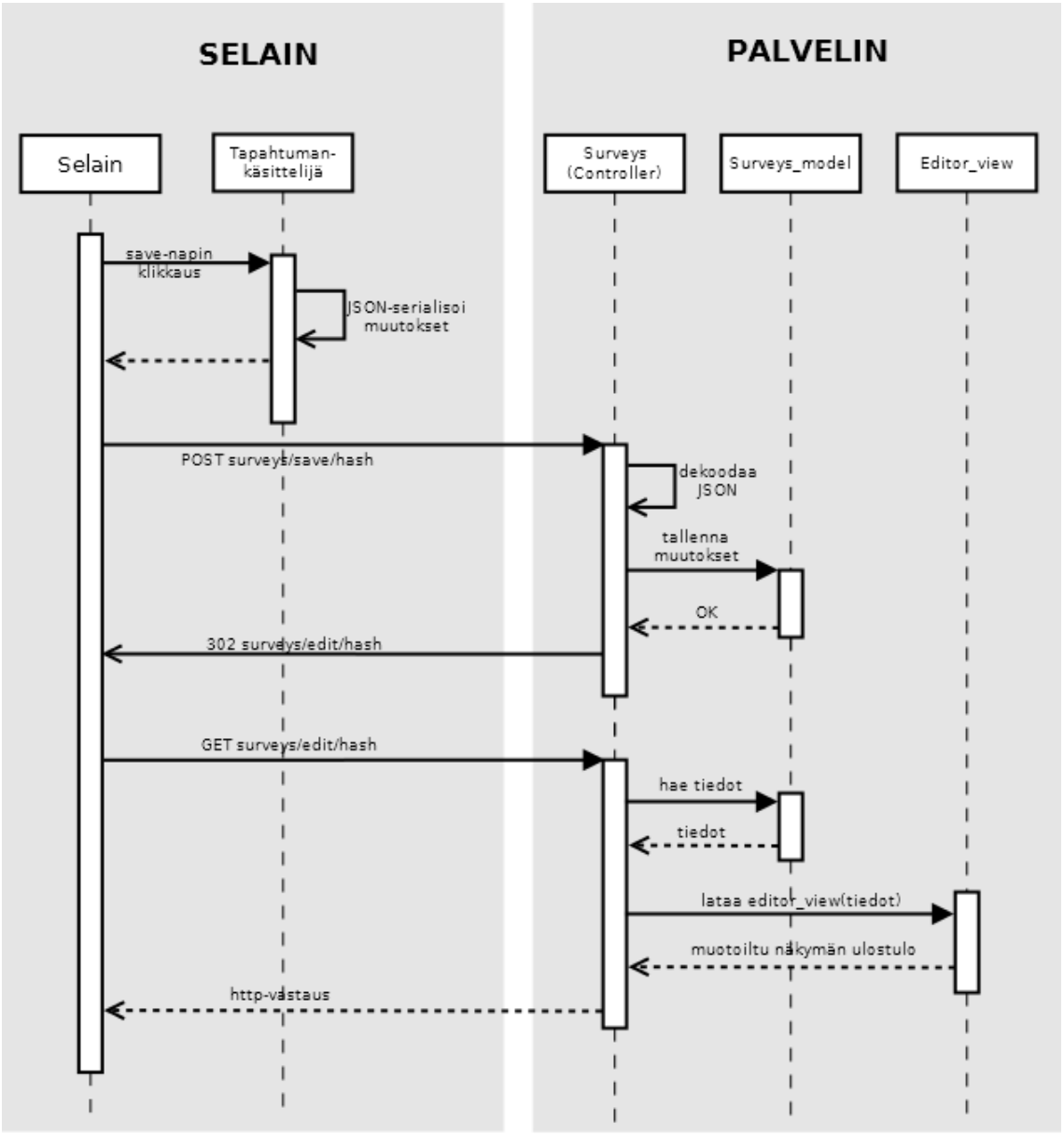
4.2.3 Kommunikointi selaimen ja palvelimen välillä

Kun selain ohjataan johonkin osoitteeseen sivustolla, palvelin hakee tarvittavat tiedot tietokannasta ja palauttaa HTML-sivun. Monimutkaisempaa käsittelyä tarvitaan tilanteisiin, joissa selaimen kautta tarvitsee lähettää tietoja palvelimelle, sillä tavalliseen HTTP

GET -kutsuun olisi epäkäytännöllistä liittää suurta määrää välitettävää dataa. Näissä tapauksissa tiedonvälitys palvelimen ja selaimen välillä hoidetaan pääasiallisesti kahdella tavalla, lomakkeenlähetyksellä ja Ajaxilla.

Osa tiedonvälityksestä käyttäjän selaimen ja DrawUX-palvelimen kanssa hoidetaan HTTP POST -kutsua käyttäen. HTML-sivu sisältää lomakkeen, johon täytetyt tiedot lähetetään lomake-elementin *action*-attribuuttiin merkitylle käsittelijälle. Palvelimella täytetyn lomakkeen tiedot saadaan luettua PHP:n *$_POST*-taulukosta tai CodeIgniterin omalla funktiolla *$this->input->post()*. Käsittelijässä täytyy myös ladata seuraava näkymä tai vaihtoehtoisesti ohjata selain toiseen osoitteeseen *redirect*-komennolla, jotta käyttäjä saisi selkeän palautteen toimintaansa.

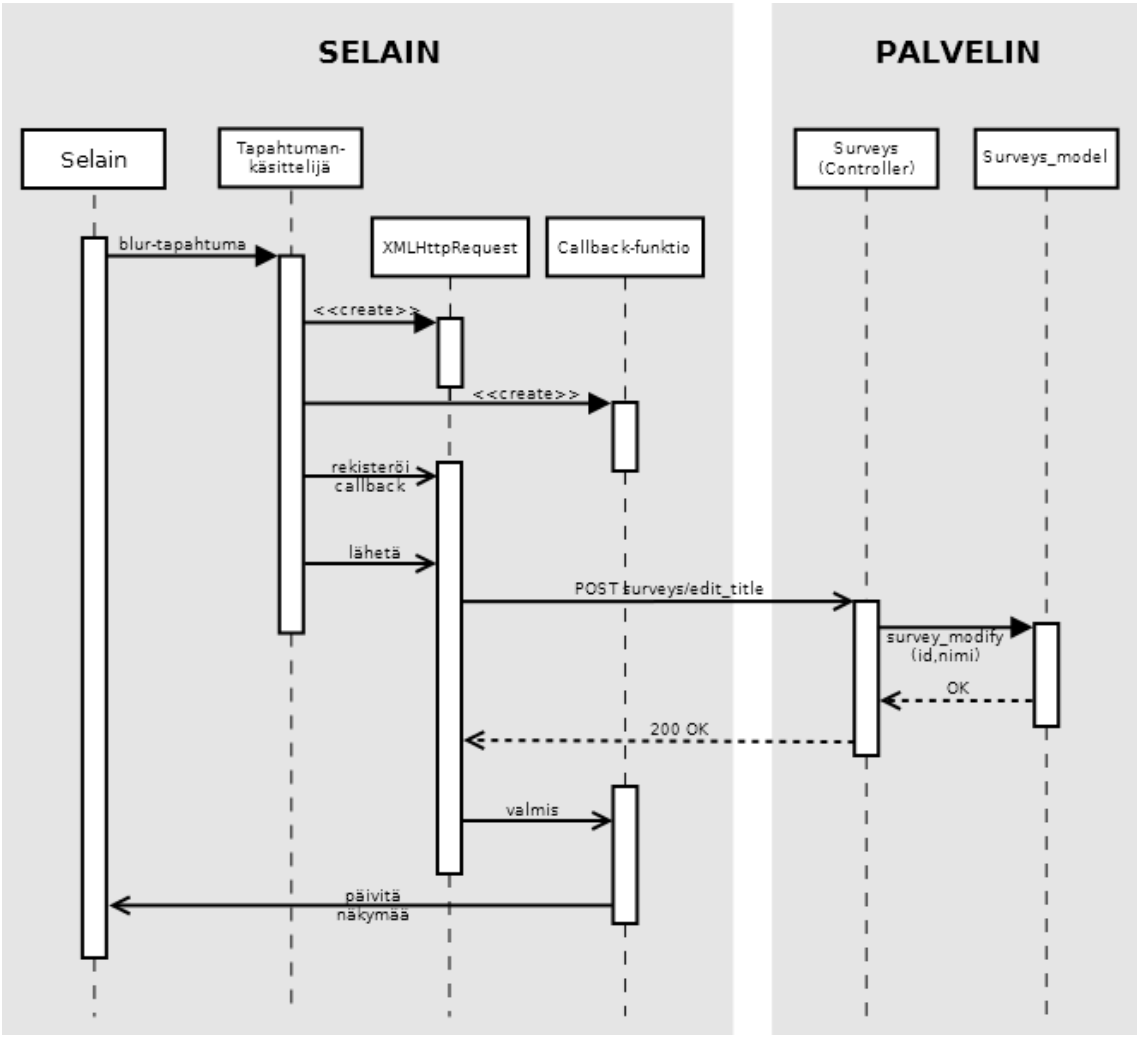
Lomakkeenlähetystä käytetään DrawUX:ssa niissä paikoissa, joissa merkittävä osa näkymää joudutaan päivittämään tai joissa tietojen tallennusta seuraa kokonaan uuteen näkymään siirtyminen. Esimerkiksi rekisteröinti- ja sisäänkirjautumislomakkeet toimivat perinteisen lomakkeenlähetysmekanismin kautta. Näissä on hyödynnetty CodeIgniterin valmista lomakekirjastoa ja sen takaisinkutsumekanismia, joka tekee kenttien tarkistuksesta ja mahdollisten virheilmoitusten esittämisestä helppoa. Myös kyselyeditorissa on käytetty lomakkeenlähetysmekanismia muutosten tallennukseen, joskin käytössä oleva lomake on erikoinen. Editorinäkymässä ei voida suoraan käyttää lomaketta, jossa jokaista kyselyn kysymystä vastaisi oma lomakekenttänsä, sillä käsiteltävien kyselyelementtien määrää ja muutosten tyyppiä ja lukumäärää ei voida tietää etukäteen. Tässä tilanteessa palvelimelle lähetettävä lomake koostuu ainoastaan kolmesta lomakeelementistä: piilotettu kenttä *changes* sisältää JSON-sarjallistettuina tiedot kaikista muutoksista, piilotettu kenttä nimeltä *hash* sisältää käsiteltävän kyselyn tunnisteen, ja *submit*-painike lähettää lomakkeen. Painikkeella on tapahtumankäsittelijä, joka kirjoittaa *changes*-kenttään tiedot kyselyyn tehdyistä muutoksista JSON-muodossa ennen lomakkeen lähetystä. Kun muutokset on tallennettu tietokantaan, käyttäjän selain uudelleenohjataan lataamaan joko editorinäkymä tai esikatselunäkymä painetusta napista riippuen. Kuva 16 esittelee tapahtumasekvenssin, jossa käyttäjä tallentaa muutokset editorissa.



Kuva 16. *Muutosten tallennus editorissa, kun tallennus tapahtuu save-napin kautta. Tallennuksen jälkeen käyttäjä ohjataan lataamaan editorinäkymä uudelleen.*

Suurin osa DrawUX:n palvelin-selain-kommunikoinnissa on kuitenkin toteutettu Ajax-teknologian avulla. *Ajax* (*Asynchronous JavaScript and XML*) on interaktiivisten web-sovellusten kehityksessä käytettävä joukko tekniikoita, joiden avulla selain ja palvelin voivat vaihtaa pieniä datamääriä kerralla. Ajaxin toimintaperiaate on vain osittainen uudelleenlataus, eli koko HTML-sivua ei tarvitse ladata kokonaan uusiksi kun siitä vain osa halutaan päivittää. Näin nopeutetaan sivuston toimintaa ja parannetaan sivuston skaalautuvuutta, kun osa palvelimen työmäärästä on annettu selaimen Ajax-moottorille. Tällainen asynkroninen kommunikointi ei myöskään keskeytä käyttäjän toimintaa sivulla. Käyttäjä voi tehdä näkymässä muuta samalla kun palvelin käsittelee pyyntöä, mikä johtaa joustavampaan käyttäjäkokemukseen. [42]

Ajaxia käytetään DrawUX:ssa kaikissa niissä tilanteissa, kun tiedonvälityksen jälkeenkin halutaan pysyä samalla sivulla ja mahdollisesti päivittää vain pieni osa sivusta. Esimerkiksi tutkijan kyselylistauksessa voidaan poistaa tai kopioida kysely tai muuttaa kyselyn otsikkoa. Kaikki näistä toiminnoista aiheuttavat vain pieniä muutoksia näkymään: yhden rivin poisto taulukosta, uuden rivin lisäys tai yhden taulukon kentän päivitys. Ajax käy siis tarkoitukseen hyvin. Selaimella lähetetään tiedot halutusta muutoksesta palvelimelle JSON-muodossa, ja kun palvelin on suorittanut halutut tietokantamuutokset, palvelimen vastaus käsitellään Ajaxin post-funktioon liitetyssä takaisinkutsukäsittelijässä. Ajaxiin liitetään usein omat käsittelijänsä onnistuneelle pyynnölle sekä virhetilanteille. Ero virhetilanteen ja onnistuneen suorituksen välillä voidaan kertoa selaimelle vastaukseen liitettävällä HTTP-otsikkokoodilla.



Kuva 17. *Ajax-kommunikointi kyselylistausnäkymässä. Fokuksen siirtäminen otsikkokentästä toiseen elementtiin (blur-tapahtuma) käynnistää tapahtumasekvenssin, jossa uusi otsikko tallennetaan Ajax-kommunikoinnin kautta.*

Kuva 17 esittää tapahtumasekvenssiä, jossa käyttäjä muokkaa kyselyn otsikkoa kyselylistauksessa. Kun fokus siirtyy kyselyn muokatusta otsikosta muualle näkymään, otsikkokentän blur-tapahtuma laukaisee tapahtumaketjun. Ensin luodaan XMLHttpRequest-olio, jolle rekisteröidään takaisinkutsufunktio. Tämän jälkeen muokkaukseen tarvittavat tiedot (kyselyn tunniste ja uusi nimi) lähetetään Surveys-käsittelijän *edit_title*-funktiolle, josta tiedot edelleen välitetään Surveys_modelille. Kun malli on tallentanut muutokset, suoritus palaa käsittelijään joka kuittaa onnistumisen XMLHttpRequest-oliolle, jonka takaisinkutsufunktiosta käsin näkymää päivitetään vastaamaan muuttuneita tietoja. Muokattu otsikkorivi, jossa on vielä esillä tekstinmuokkauskenttä, vaihdetaan siis uuden otsikon mukaiseksi staattiseksi tekstiksi.

4.3 Projektin laajuus

DrawUX-järjestelmää on kehitetty osissa. Ennen tätä diplomityöprojektia järjestelmästä oli jo toteutettu ensimmäinen versio. Projektin aikana järjestelmään lisättiin kokonaan uusina ominaisuuksina Data Viewer -näkymä, tulosten kääntämisnäkymä sekä päiväkirjatyylinen vastausnäkymä mobiiliversioineen. Suureksi osaksi projektissa keskityttiin kuitenkin jo olemassa olevien näkymien korjailuun ja laajentamiseen.

Etusivu, Data Viewer, kääntönäkymä, palautesivu, käyttöehtosivu ja uutissivu toteutettiin kokonaan osana projektia. Alla on listattu näkymäkohtaisesti, mitkä osat muista näkymistä toteutettiin tämän projektin aikana.

Editori/vastausnäkymä:

- käyränpiirtoon liittyvien käyttöliittymätekstien kustomoitavuus
- hymiöliukusäätimen valinnaisuus
- mahdollisuus vaatia käyrä piirrettäväksi loppuun
- mahdollisuus asettaa kysymyksiä pakollisiksi
- kustomointimahdollisuus pakollisiin kysymyksiin liittyville virheilmoituksille
- ’Asetukset’-välilehden lisäys
- ulkoasumuutoksia
- kysymyspohjiin liittyvät työkaluvihjetekstit tutkijan avuksi
- edistymispalkin valinnaisuus ja sen esitystavan valinta
- ’Muu’-vastauskenttien kustomoitavuus
- kyselyn taustan värin valinta
- kielivalintaan lisätty vaihtoehtoja
- lyhyt tekstimuotoinen vastaus kysymystyyppi
- ilmoitukset istunnon umpeutumisesta
- päiväkirjatyyppiset kyselyt

Vastausnäkymä:

- ensimmäinen ohjetekstilaatikko pakolliseksi kuitata ennen piirron jatkamista
- käyrän resetointimahdollisuus
- ’Edellinen sivu’-painike
- zoomaustoiminto
- mobiilivastausnäkymä

Muuta:

- kyselyn kopiointi
- kyselykohtaisen kutsu- tai muistutusviestin tallennus
- mahdollisuus ladata myös käännetty versio Excel-taulukosta

Kuva 18. *Projektin työt jaettuna näkymiin. Tummansinisellä pohjalla olevat näkymät on toteutettu projektissa kokonaan. Keskisinisiä näkymiä on muokattu runsaasti, vaaleansinisiä on muokattu vain vähän. Harmaalla pohjalla esitettäviä näkymiä ei ole muokattu tämän projektin aikana.*

Kuva 18 näyttää jaon projektin aikana toteutettuihin DrawUX:n osiin ja siinä jo valmiiksi olleisiin ominaisuuksiin. Suurin osa työstä liittyi kyselyjen editointiin ja niihin vastaamiseen. Projektissa alusta asti toteutetuista näkymistä Data Viewer oli suuritöinen, muut hyvin yksinkertaisia. Uusien ominaisuuksien lisäksi työ sisälsi ohjelmointivirheiden korjailua, muutoksia ulkoasuun ja muita sekalaisia töitä.

4.4 Toteutusprosessi

DrawUX julkistettiin ensimmäisen kerran NordiCHI2012-konferenssin demosessiossa [1]. Tarve DrawUX-työkalulle todettiin Tampereen teknillisen yliopiston Ihmiskeskeisen teknologian yksikön DELUX (Delightful Long-Term User Experience)-projektissa. Projektissa tutkittiin rahoittajien eri tuotteiden pitkäaikaista käyttäjäkokemusta. Tutkimukseen käytettiin sähköistä iScale-työkalua ja paperipohjaista UX Curvea, mutta projektin aikana todettiin tarve UX Curven kaltaiselle sähköiselle työkalulle, joka olisi samalla iScalea helppokäyttöisempi. Projektipäällikkö Jari Varsaluoma laati alustavan suunnitelman käyttöliittymästä ja tarvittavista toiminnoista keväällä 2012 ja varsinainen toteutus alkoi kesäkuussa 2012.

Jo alusta asti DrawUX:n kehityksen oli tarkoitus olla iteratiivista ja tapahtua sykleissä, joihin liittyisi useita käytettävyystutkimuksia ja vertailututkimuksia muiden pitkän ajan käyttökokemuksen tutkimustapojen kanssa [1]. DrawUX onkin toteutettu erillisissä jaksoissa. Ohjelman perustoiminnallisuuden kehitti projektin alkuperäinen tutkimusapulainen aikavälillä kesäkuu 2012–joulukuu 2013. Tärkeimmät näkymät kyselyjen rakentamiseen ja niihin vastailuun olivat siis jo olemassa ja toimivia tämän diplomityöprojektin alkaessa, ja perustavanlaatuiset arkkitehtuuriratkaisut oli tehty. Kesäkuussa 2014 kehitys jatkui, kun aloitin työt tutkimusapulaisena CCD MobiLe (Cross-Cultural Design for Mobile Learning) -projektissa. Kehitysjakson alkuun kuului useita pieniä korjauksia ja lisäyksiä sekä luvussa 4.3 esiteltyjä ominaisuuksia jo olemassa oleviin näkymiin. Loppuvuodesta 2014 toteutettiin kokonaan Data Viewer -näkymä. Tämän kehitysjakson aikana ohjelmistoa kehitettiin ainoastaan TTY:n omaan käyttöön.

Alkuvuodesta 2015 työkalua testattiin ensimmäistä kertaa asiakkaalla mobiilisovelluksen käyttäjäkokemusta tutkivassa pilottitutkimuksessa. Tutkimuksen suorittaneilta kehittäjiltä saatiin palautetta työkalun käytöstä.

Keväällä ja kesällä 2015 projektia jatkettiin siten, että mukana oli asiakkaana eräs pohjoismainen peliviihdeyritys. Data Viewer -näkymään lisättiin siirrettävät tietoruudut ja ohjelmaan kehitettiin käännösnäkymä. Muita olennaisia uusia lisäyksiä olivat mobiilivastaaminen sekä päiväkirjatyylinen vastaaminen. Jakson lopussa työhön kuului myös palvelun jäädytetyn version kopiointi asiakkaan palvelimelle. Ratkaisuun päädyttiin, koska Tampereen teknillinen yliopisto ei voinut taata pysyvää palvelun tarjoamista omilta palvelimiltaan.

5. JÄRJESTELMÄN ARVIOINTI

Tässä luvussa kuvataan DrawUX-järjestelmän tämänhetkisen toteutuksen laatua. Arviointi tapahtui laatukriteeripuun (liite 1) pohjalta. Jokaista alimman tason ominaisuutta tarkasteltiin erikseen. Arviointi kokonaisuudessaan löytyy liitteestä 2. Tässä luvussa ei käsitellä kaikkia arvioinnin kohtia, vaan käydään läpi tärkeimmät arvioinnissa esiin tulleet huomiot. Tarkoituksena on antaa yhteenveto olennaisista arvioinnin tuloksista.

Ensin luvussa 5.1 esitellään ohjelmistojen arvioinnin teoriaa ja arviointiin valittuja laatuominaisuuksia. Varsinainen arviointi alkaa luvusta 5.2, jossa tarkastellaan yleisesti järjestelmän sopivuutta käyttöön, eli miten hyvin se vastaa tarpeita, joita varten se alun perin kehitettiin. Luvussa 5.3 tarkasteluun otetaan suorituskyky ja vasteajat eli järjestelmän tehokkuus, ja luvussa 5.4 työkalua arvioidaan sen ylläpidettävyyden näkökulmasta. Lopuksi luku 5.5 toimii yhteenvetona arvioinnin olennaisimmista löydöksistä.

5.1 Arvioinnin pohja

Ohjelmistoja voidaan arvioida lukemattomien erilaisten laatuominaisuuksien kannalta. Esimerkiksi Precheltin [43] mukaan web-ohjelmistoissa olennaisia laatuominaisuuksia ovat tuotteliaisuus eli se että ohjelmisto ja sen teknologiavalinnat tukevat nopeaa kehitystä, robustius, virheenkäsittely ja turvallisuus, suorituskyky sekä ylläpidettävyys. Offutt [44] listaa tärkeiksi laatutekijöiksi luotettavuuden, turvallisuuden, käytettävyyden, saatavuuden, skaalautuvuuden, ylläpidettävyyden ja julkaisunopeuden. Kansainvälinen standardi ISO/IEC 9126-1 kuvaa ohjelmistojen laadun kuutena piirteenä ja 27 alipiirteenä [45]. Kuusi ylemmän tason laatuominaisuutta ovat toimivuus, tehokkuus, käytettävyys, luotettavuus, ylläpidettävyys ja siirrettävyys. Sittemmin ISO/IEC 9126-1 on korvattu standardilla ISO/IEC 25010, jossa on kahdeksan ylemmän tason laatuominaisuutta, joista uusia lisäyksiä ovat yhteensopivuus ja turvallisuus [46]. Myös alipiirteitä on nimetty uudelleen ja lisätty.

Seuraavissa aliluvuissa tutustutaan tarkemmin arvioinnin pohjaan. Luku 5.1.1 esittelee rajoitteet ja vaatimukset, jotka vaikuttivat kehitykseen. Luvuissa 5.1.2–5.1.4 käydään läpi arviointiin valitut laatuominaisuudet ja luvussa 5.1.5 käsitellään arvioinnin toteutukseen liittyvää teoriaa.

5.1.1 Toteutusta ohjanneet rajoitteet ja vaatimukset

Ennen laadun analysointia on tärkeä tietää, millaisia laatuvaatimuksia työhön kohdistui ja millaiset periaatteet ja käytännön rajoitteet työtä ohjasivat. DrawUX:n toteutukseen

liittyi erityisesti kriittisiä aika- ja resurssirajoitteita. Kyseessä oli pieni, vain yhden ohjelmoijan kerrallaan toteuttama projekti. Toteuttajan tuli toimia yhtä aikaa palvelinohjelmoijana, selainohjelmoijana, käyttöliittymäsuunnittelijana, tietokantavastaavana ja testaajana, sillä mihinkään näistä rooleista ei ollut erikseen palkattua työvoimaa. Erityisesti järjestelmällisen testauksen puute prosessissa oli suuri rajoite, sillä kulloinkin työtä tekevän kehittäjän tuli tasapainottaa ohjelmointi toiminnan varmistamisen kanssa, jolloin testaus oli lähinnä ohjelmoinnin lomassa tapahtuvaa improvisoitua ad hoc -kokeilua. Testeistä ei myöskään ole mitään dokumentointia.

Aikataulu- ja resurssirajoitteista huolimatta järjestelmä oli kooltaan ja ominaisuusvalikoimaltaan melko suuri. DrawUX:n kuuluu lukuisia näkymiä ja haluttuja toimintoja oli runsaasti, joten kehityksen täytyi olla hyvin nopeaa. Kehityksen nopeus olikin tärkein ohjelmistoon kohdistuva paine. Avainasemassa olivat harkittu valmiiden ilmaiskomponenttien käyttö oikeissa paikoissa, vain olennaiseen keskittyminen, sekä koodin rakenteen selkeys ja tulkittavuus. Näiden avulla kehitys pysyisi jatkuvasti nopeana ja tehokkaana. Nopean kehityksen katsottiin olevan sidoksissa ohjelmiston ylläpidettävyyteen. Koodin arkkitehtuurin ja yleisen selkeyden haluttiin olevan sellaisella tasolla, että kehityksen jatkaminen olisi mahdollisimman helppoa niin samalle kuin uudellekin kehittäjälle.

DrawUX-järjestelmään kohdistuu yleisellä tasolla myös useita muita laatuvaatimuksia. Järjestelmän tulee olla tarpeeksi nopea ollakseen käyttökelpoinen. Suoritus- ja latausaikoihin ei asetettu tarkkoja rajoja, mutta latausaikojen toivottiin olevan ’mahdollisimman lyhyet’, erityisesti vastausnäkymässä ja varsinkin sen mobiiliversiossa. Latausajoille esitettiin tavoitteeksi, mutta ei kuitenkaan tiukaksi rajoitukseksi, mielellään 1–2 sekunnin latautuminen. 5 sekunnin latausaika ajateltiin vielä siedettäväksi suurissa kyselyissä.

Muidenkaan sivujen latausaikojen ei tulisi olla kohtuuttomia. Aikoja ei missään projektin vaiheessa kuitenkaan mitattu. Ajonaikaisen tehokkuuden suhteen kriittisin näkymä on Data Viewer, jonka tulee pystyä käsittelemään normaalin kyselyn kaikkia vastauksia kerralla. Kriteerinä oli lähinnä että realistisen suuruista käyrämäärää tulisi voida käsitellä interaktiivisesti (esim. suodatus, zoomaus) ilman että selain menee jumiin tai kaatuu. Kuitenkin myös tämä kriteeri jäi auki siinä mielessä, että mitään tavoitemäärää yhtäaikaisesti käsiteltäville käyrille ei projektin johdon puolesta asetettu eikä asiaa myöskään tutkittu kuormitustestein. Realistinen koko käsiteltävälle datalle on myös jätetty avoimeksi kysymykseksi, mutta jotain suuntaviittaa saadaan siitä, että yhteen kyselyyn suositellaan lisättävän 1–2, maksimissaan 3, käyränpiirtotehtävää. Muita kysymyksiä voi olla huomattavasti enemmän. Käyttäjäkokemustutkimuksissa sata vastaajaa on jo suuri otos, käytännössä DrawUX-tutkimusten vastaajakokojen ajatellaan olevan kymmenien vastaajien luokkaa. Yksi vastaaja sijoittaa käyrälle keskimäärin noin 5–10 pistettä, päiväkirjatyyppisissä kyselyissä enemmän. Käytännössä ei kuitenkaan tarvitse varautua esimerkiksi kymmenientuhansien pisteiden käsittelyyn. Suurella datamäärällä tutkijan

ajatellaan voivan odottaa Data Viewerin latautumista huomattavasti kauemmin kuin vastaaja voi odottaa vastausnäkymän latautumista, joten näkymässä ei välttämättä tarvitse pyrkiä muutaman sekunnin latausaikaan, mutta kuitenkin tutkijankin näkymien tulisi pysyä käyttökelpoisina.

Järjestelmän tulisi olla mahdollisimman helposti käytettävä. Erityisesti vastaajalle tarkoitettujen kyselynäkymien käytettävyys on keskeistä, sillä vastaajilla ei oleteta olevan erityistä tietotekniikkaosaamista ja heidän on tärkeää pystyä vastaamaan kyselyihin tästä huolimatta. Koko työkalun ensisijainen tavoite olikin tarjota vastaajalle mahdollisimman helppokäyttöinen ja vaivaton tapa raportoida pitkän aikavälin käyttökokemuksia. Tutkijoiden näkymien ei tulisi sisältää merkittäviä käytettävyysongelmia, mutta käytettävyyden hiomista näissä näkymissä ei koettu tarkoituksenmukaiseksi, sillä järjestelmää tutkija-roolissa käyttävillä henkilöillä oletetaan olevan vähintään kohtalaiset tietotekniikkataidot. Varsinaisen toimintovalikoiman oletetaan olevan tälle roolille tärkeämpää. Kuitenkin kehitystyön edetessä pidemmälle myös tutkijoiden käyttäjäkokemukseen alettiin kiinnittää enemmän huomiota.

Laatuvaatimuksista saavutettavuus on toistaiseksi asetettu taka-alalle, järjestelmän toimivuus tietyllä pöytäkoneselaimella (Google Chrome) sekä mobiilivastausnäkymän toimivuus yleisillä mobiililaitteilla koetaan riittäväksi. Lisäksi järjestelmän tulee luonnollisesti olla luotettava, turvallinen ja saatavilla aina tai lähes aina. Ylipäätään voisi sanoa, että järjestelmässä toiminnallisuus on ensisijainen asia, jotta ohjelmisto saataisiin toimimaan tavalla josta olisi mahdollisimman paljon hyötyä käyttökokemuksen tutkimisessa. Suuremmassa ja paremmin resursoidussa projektissa myös ei-toiminnallisiin vaatimuksiin kiinnitettäisiin luonnollisesti enemmän huomiota.

Merkityksellisen arvioinnin toteuttamiseksi arviointi tulisi keskittää vain muutamaan laatuominaisuuteen. Tähän projektiin on valittu arvioitaviksi ominaisuuksiksi järjestelmän yleinen sopivuus käyttöön, tehokkuus sekä ylläpidettävyys.

5.1.2 Sopivuus käyttöön

ISO 9126-1 -standardin mukaan ohjelmistojen tärkeä ominaisuus on että se vastaa käyttäjän tarpeisiin. Standardin mukaan ohjelmistojärjestelmän laadukkuus muodostuu sisäisestä laadukkuudesta, ulkoisesta laadukkuudesta ja laadusta käytössä, joka on loppukäyttäjän näkökulma laatuun [45]. Laatu käytössä on ergonomista käytettävyyttä laajempi käsite ja seuraus sisäisestä ja ulkoisesta laadukkuudesta. Covellan ja Olsinan [47] mukaan laatua käytössä voidaan mitata ja arvioida sen suhteen, miten käyttäjät saavuttavat tietyt päämäärät tehokkaasti, tuottavasti, turvallisesti ja tyytyväisesti tietyissä käyttötilanteissa. Käyttölaadun mittauksessa on aina mukana todellisia käyttäjiä, kun taas asiantuntija-arvioilla mitataan ulkoista laadukkuutta. Kunnolliseen käyttöön sopivuuden arviointiin tarvittaisiin siis käyttäjätestejä, joita ei voida tämän diplomityön puitteissa järjestää. Saatavilla on kuitenkin palautetta projektin keskellä järjestetyistä pilottitutki-

muksista, joista saadaan jonkinlainen näkökulma siihen, miten hyvin työkalu on palvellut asiakasta kehityksen keskivaiheessa.

Kitschenham ja Pfleeger [48] esittelevät viisi eri näkökulmaa ohjelmistojen laatuun: transsendenttinen näkökulma (laatua voidaan havaita mutta ei määrittää), käyttäjän näkökulma (ohjelmiston sopivuus käyttöön), tuotannon näkökulma (spesifikaatioiden noudatus ohjelmiston tuotannossa), tuotenäkökulma (laatu on sidoksissa ohjelmistotuotteen sisäisiin ominaisuuksiin) ja arvonäkökulma (laatu on sidoksissa siihen kuinka paljon asiakas on valmis maksamaan tuotteesta). Metriikkapohjainen arviointi perustuu yleensä tuotenäkökulmaan, eli oletetaan että ohjelmiston sisäisiä ominaisuuksia, kuten vasteaikaa ja koodin kompleksisuutta, mittaamalla saadaan tietoa tuotteen laadusta. Cherfi et al. [6] suosittelevat kuitenkin myös globaalia laadunarviointia eikä vain yksittäisiin ominaisuuksiin keskittymistä, sillä laatuominaisuudet eivät ole riippumattomia muuttujia. Olennaisia asioita saattaa jäädä huomaamatta, jos keskitytään vain yhteen laatutekijään kerrallaan. DrawUX-järjestelmän arvioinnissa otetaan tämän vuoksi ensin yleinen näkökulma, joka vastaa suunnilleen lähteen [48] mainitsemaa käyttäjän näkökulmaa. Tarkoituksena on arvioida yleisellä tasolla, kuinka hyvin järjestelmä soveltuu käyttötarkoitukseensa. Tämä arviointi yhdistelee ohjelmiston funktionaalisia ja laadullisia ominaisuuksia, sillä ne yhdessä määrittävät kuinka hyvin tuote palvelee käyttötarkoitustaan. Arvioinnin tässä osassa käyttäjäryhmistä huomioidaan tutkija ja vastaaja.

5.1.3 Tehokkuus

Yksi ISO-standardin laatumääreistä on tehokkuus, joka mittaa ohjelman vaatimaa ajan ja resurssien määrää [45]. Suorituskykyinen ohjelma vastaa käyttäjän pyyntöihin nopeasti. Suorituskykyä voidaan mitata esimerkiksi tarkastelemalla sivun latausaikaa tai jonkin ohjelman osan keskimääräistä vasteaikaa. Suoritusaikaan liittyviä metriikoita on kehitetty vähän suhteessa muihin laatuvaatimuksiin [6]. Syynä voi olla se, että pitkään web-ohjelmien suoritusajan pullonkaulana oli tiedonsiirtoyhteyksien hitaus. Nykyään kun nopeat laajakaistayhteydet ovat yhä useampien käyttäjien saatavilla, ohjelman optimoinnilla pystytään vaikuttamaan merkittävämmin myös suoritus- ja vasteaikaan.

Web-ohjelmoinnissa ohjelman suorituskyky voidaan jakaa kahteen olennaiseen osaan, web-tehokkuuteen ja suoritusaikatehokkuuteen [49]. Web-tehokkuudella tarkoitetaan nopeutta, jolla ohjelma voidaan esittää käyttäjälle, eli esimerkiksi pyydetyn sivun latausaikaa. Suoritusaikatehokkuudessa taas on kyse siitä, miten nopeasti ohjelma vastaa käyttäjän toimiin suorituksen aikana. Suorituskyvyllä on väliä, koska verkkosivun nopeus vaikuttaa merkittävästi käyttäjän käyttökokemukseen, joka taas tekee käyttäjistä tyytyväisiä ja saa heidät palaamaan sivustolle. Barkerin [49] mukaan on tärkeää että käyttäjä pääsee sivustolle nopeasti, sillä jos palvelimelta kestää liian kauan vastata kutsuun, käyttäjä hylkää istunnon.

5.1.4 Ylläpidettävyys

Web-ohjelmistokehitys eroaa perinteisestä sovellusohjelmoinnista muun muassa siten, että web-kehitys tehdään usein nopeissa sykleissä verrattuna perinteiseen sovellukseen ja web-ohjelmistolla on suuri päivitystiheys [44]. Kompleksisella, vaikeasti ylläpidettävällä ohjelmistolla on suuret päivityskustannukset [50]. Näin ollen web-ohjelmiston ylläpidettävyys on erityisen tärkeä osa ohjelmiston laatua. Bhattin et al. [7] mukaan ylläpidettävyys on sitä, kuinka helposti ohjelmistojärjestelmää tai sen osaa voidaan muokata virheiden korjaamiseksi, sen laatuominaisuuksien parantamiseksi tai muuttuneeseen ympäristöön sovittamiseksi.

Ylläpidettävyyteen vaikuttavat mm. ohjelmiston koko, selkeys, kommentointi ja modulaarisuus [43]. Ghosheh et al. [50] listaavat ylläpidettävyyden olennaisiksi aliominaisuuksiksi ymmärrettävyyden, analysoitavuuden, muokattavuuden ja testattavuuden. Lähde myös esittelee useita kvantitatiivisia metriikoita web-ohjelmistojen ylläpidettävyyden mittaamiseen. Myös Cherfi et al. [6] esittelevät kokoon, monimutkaisuuteen ja kytkentöihin liittyviä metriikoita. Hegedűs et al. [51] tutkivat ylläpidettävyysmetriikoiden suhdetta subjektiivisiin arvioihin ylläpidettävyydestä. Testeissä ei löydetty tilastollisesti merkittäviä suhteita minkään yksittäisen ylläpidettävyyteen oletetusti vaikuttavan metriikan ja ylläpidettävyyden subjektiivisen arvion väliltä, mutta yhdessä joillakin metriikoilla oli ennustusvoimaa.

Ylläpidettävyyttä pidetään olennaisena web-ohjelmistojen ominaisuutena ja kaikista web-ohjelmistojen laatua mittaamaan kehitetyistä metriikoista suuri osa keskittyy nimenomaan ylläpidettävyyden arviointiin [6]. Tämän valossa ylläpidettävyys kannattaa valita yhdeksi erikseen arvioitavaksi ominaisuudeksi. Arvioinnin tässä osassa sidosryhmistä keskitytään erityisesti kehittäjiin/ylläpitäjiin.

5.1.5 Arvioinnin toteutus

Laadunarviointitavat voidaan jakaa asiantuntijalähtöisiin ja käyttäjälähtöisiin menetelmiin sekä laadullisiin ja määrällisiin menetelmiin. Esim. [6] listaa useita erilaisia metriikoita joilla erilaisia laatuominaisuuksia voidaan mitata kvantitatiivisesti. Tässä työssä on tarkoitus suorittaa kvalitatiivinen asiantuntijalähtöinen arviointi. Tarkoituksena ei ole verrata ohjelmistoa mittarien avulla muihin ohjelmistoihin, joten kvantitatiivista metriikoiden käyttöä ei katsota tarpeelliseksi. Ennemminkin halutaan verrata DrawUX:n nykyistä toteutusta siihen, millainen se olosuhteet huomioiden voisi parhaimmillaan olla. Tarkoituksena on löytää realistisia parannusehdotuksia sekä poimia toteutuksesta erityisen hyviä ratkaisuja, jotta ne voidaan tarkoituksella säilyttää järjestelmässä. Käyttäjätutkimuksesta olisi suurta hyötyä käyttökelpoisuuden arvioinnissa, mutta todellisten käyttäjien mukaanottoon ei tämän projektin puitteissa ole resursseja. Näihin tavoitteisiin ja resursseihin nähden laadullinen asiantuntija-arviointi palvelee tarkoitusta parhaiten.

Vaikka arviointi on laadullinen, se pohjautuu muunneltuun versioon kvantitatiivisesta WebQEM-menetelmästä.

Web Quality Evaluation Method eli WebQEM on tunnettu kvantitatiivinen asiantuntijalähtöinen laadunarviointistrategia, joka käyttää laatumallinaan ISO/IEC 9126-1-standardiin pohjautuvaa laatuvaatimuspuuta. Menetelmässä arvioidaan web-palvelun laatuun vaikuttavia ominaisuuksia järjestelmällisesti jakamalla ISO/IEC 9126-1 -standardin laatuominaisuudet yksityiskohtaisemmiksi aliominaisuuksiksi, kunnes päästään niin matalalle tasolle, että alimman tason laatuominaisuutta tai vaatimusta voidaan suoraan mitata. WebQEM-arviointistrategia alkaa vaatimusten määrittelyllä, jonka jälkeen suoritetaan alustava ja globaali arviointi. Viimeisenä vaiheena on suositusten tekeminen arvioinnin tulosten pohjalta. Alustavassa arviointivaiheessa jokaiselle mitattavalle ominaisuudelle määritetään kriteerifunktio, joka mittaa kriteerin täyttymisen astetta välillä 0–1. Funktioiden ei tarvitse olla jatkuvia. Metriikka voi tässä tapauksessa tarkoittaa myös esimerkiksi halutun ominaisuuden olemassaolon tai puuttumisen toteamista, jolloin laatupuuhun voidaan yhdistää toiminnallisia ja ei-toiminnallisia vaatimuksia. Kun mittaukset on suoritettu, tehdään globaali arviointi. Tässä vaiheessa laaditaan koostamiskriteerit ja pisteytysmalli, joiden avulla alimman tason tulokset kootaan yhteen ylemmän tason laatuarvioiksi aina puun juureen saakka. Pisteytysmalleissa käytetään painokertoimia, joilla laatuattribuutin aliominaisuuksille asetetaan suhteelliset painoarvot. Pisteytysmallit voivat olla lineaarisia tai epälineaarisia. Kun pisteytysmallit on laadittu, kunkin ylemmän laatuvaatimuksen pisteet lasketaan alhaalta ylös edeten aina laatupuun juureen saakka, josta saadaan koko palvelun yleiset laatupisteet. Analysoinnin pohjalta sidosryhmät voivat ymmärtää arvioidun palvelun vahvuuksia ja heikkouksia asetettuihin vaatimuksiin nähden. [8]

Tässä työssä tehdään WebQEM-strategiaa vapaasti mukaileva arviointi. Kullekin valitulle ylemmän tason laatuominaisuudelle (sopivuus käyttöön, tehokkuus ja ylläpidettävyys) on rakennettu oma alikriteeripuunsa, jota edelleen on jaettu pienempiin kokonaisuuksiin kunnes on saavutettu arvioitavissa olevia kokonaisuuksia. Kuitenkin WebQEM-menetelmästä poiketen tässä arvioinnissa alikriteereihin ei liitetä metriikoita eikä tulosten koostamiseen käytetä pisteytysfunktioita, vaan kutakin alimman tason laatukriteeriä tarkastellaan laadullisesti. Puutteellisten ratkaisujen kohdalla tarjotaan myös idean tasolla vaihtoehtoisia ratkaisuja, mutta parannusehdotusten toteutuksen käsittelyyn ei mennä kovin yksityiskohtaisesti.

Kuten Olsina ja Rossi [8] huomauttavat, on hankalaa kehittää täsmällinen laatuvaatimuspuu, jossa käytetyt attribuutit korreloisivat vahvasti haluttujen laatuominaisuuksien kanssa. Laatukriteeripuun laatiminen onkin syytä tehdä erityisen huolella. Tässä työssä kriteeristön laatimiseen on käytetty lähteinä kirjallisuudesta löytyvää teoreettista tietoa kustakin laatuominaisuudesta, pilottitestauksen jälkeisessä haastattelussa asiakkaalta saatua palautetta, projektin aikaista kirjanpitoa suoritetuista ja jäljelläolevista tehtävistä, jatkokehitysprojektin toiseen vaiheeseen liittyvää aloituspalaverin pöytäkirjaa, jossa

määriteltiin tämän jakson aikana toteutettavat toiminnalliset vaatimukset, sekä projektin aikaista sähköpostienvaihtoa projektipäällikön kanssa. Yhdessä nämä tietolähteet antanevat tarpeeksi kattavan ja realistisen kuvan kuhunkin laatuominaisuuteen vaikuttavista alemman tason kriteereistä sekä niiden täyttymisen tasosta DrawUX-järjestelmässä.

Arviointia varten laadittu laatupuu löytyy kokonaisuudessaan liitteestä 1. Lehtiattribuutit, joista kustakin tehdään erillinen tarkastelu, on kursivoitu. Liitteessä 2 on varsinainen arvioinnin toteutus. Varsinainen arviointi käydään läpi seuraavissa luvuissa 5.2–5.4. Eri laatukriteerien rakentaminen tiukaksi puurakenteeksi on jossain määrin keinotekoista, sillä laatuattribuutit ovat vuorovaikutuksessa toistensa kanssa ja sama asia vaikuttaa yleensä useampaan kuin yhteen laatuominaisuuteen. Tämän vuoksi esimerkiksi teknologiavalintoja käsitellään useissa eri luvuissa. Jotkin arvioitavat ratkaisut tulivat ilmi vain yhdessä kohtaa puuta, mutta näilläkin ratkaisuilla on usein merkittäviä vaikutuksia myös muihin arvioitaviin laatuominaisuuksiin. Näissä tapauksissa myös tällaiset sivuvaikutukset on mainittu siinä kriteeripuun kohdassa, jossa ne löydettiin.

5.2 Sopivuus käyttöön

Alun perin DrawUX-projektin päämäärä on ollut kehittää ”tehokas tutkimustyökalu etätutkimuksia varten helppokäyttöisellä käyttöliittymällä” [1]. Työkalulle ei ole asetettu tämän tarkempia tavoitteita, joten tässä luvussa arvioidaan laadullisesti, palveleeko DrawUX:n tämänhetkinen toteutus tätä päämäärää ja miten hyvin (helposti, tehokkaasti, tarkoituksenmukaisesti) sillä pystyy suorittamaan niitä tehtäviä, joihin se on tarkoitettu. Luvussa etsitään vastauksia mm. seuraaviin kysymyksiin: tarjoaako työkalu kaiken tarpeellisen toiminnallisuuden pitkän aikavälin etätutkimuksien suorittamiseen? Miten hyvin työkalun tarjoamat toiminnot on toteutettu? Miten niissä on otettu huomioon helppokäyttöisyys? Mitä työkalun toimintoja ja näkymiä voisi parantaa ja miten, jotta ne sopisivat paremmin tutkijan tai vastaajan käyttöön?

Suuri osa käyttöön sopivuuden arvioinnista liittyy toiminnallisiin vaatimuksiin. Lisäksi tarkastellaan käyttöliittymiä, ohjeistusta sekä sitä, voidaanko järjestelmän toimivuuteen luottaa.

5.2.1 Ohjeistus

DrawUX-sivustolla ei ole lainkaan ohjesivua. Käyttäjän oletetaan tietävän mitä tehdä missäkin näkymässä ja tämän vuoksi näkymät on pyritty tekemään yksinkertaisiksi ja selkeiksi käyttää. Käyttöä on helpotettu asettamalla työkaluvihjetekstejä olennaisiin paikkoihin erityisesti editorinäkymässä ja Data Viewer -näkymässä. Editorissa useimpiin asetuskenttiin liittyy työkaluvihje, joka kuvaa mitä kenttään on tarkoitus kirjoittaa, ja Data Viewerissä kaikkiin eri käyrien suodatustoimintoihin ja muihin asetuksiin liittyy oma työkaluvihjeensä.

Käyränpiirtonäkymässä vastaajaa ohjeistetaan kustomoitavilla ohjetekstilaatikoilla lähes joka vaiheessa käyränpiirtoa. Näin tarjotaan aina kontekstiin liittyviä ohjeita. Ensimmäinen ohjetekstilaatikko tulee kuitata luetuksi, jotta vastaajan voidaan olettaa pääsevän piirrossa alkuun luettuaan ensimmäiset ohjeet.

Olisi kuitenkin hyvä että käyttäjällä olisi jokin paikka, josta lähteä etsimään vastausta ongelmatilanteissa. Suurilla ja keskikokoisilla verkkosivustoilla on perinteisesti Ohje- tai Usein kysytyt kysymykset -osio, johon on koottu yleisimpiä ongelmia ja kysymyksiä vastauksineen. DrawUX:n kehityksessä on tullut vastaan tapauksia, joissa käyttäjä (tutkija) on ymmärtänyt jotakin väärin, ja ongelmaa on jouduttu selvittämään yhdessä kehittäjän kanssa. Vastaavia ongelmia voitaisiin ehkäistä kattavalla dokumentoinnilla.

5.2.2 Toiminnallisuuden kattavuus

Olennaisinta käyttöön sopivuuden kannalta on luonnollisesti se, että ohjelmalla voidaan suorittaa ne korkeamman tason työt ja tehtävät, joita varten ohjelma on olemassa. DrawUX on olemassa pitkän ajan käyttökokemuksen etätutkimusten toteutusta sekä tulosten analyysia varten. Liitteen 2 kohta 1.2 tarjoaa yksityiskohtaisen erittelyn toiminnallisuuden kattavuudesta. Arvioinnista nähdään, että haluttuja ominaisuuksia on toteutettu runsaasti. DrawUX kattaa halutut ydintoiminnot. Kyselyjen luontiin ja vastaajien hallinnointiin liittyvät peruskäyttötarpeet täyttyvät hyvin. Myös projektipäällikön mukaan alkuperäiset tavoitteet toiminnallisuuden suhteen saavutettiin.

Kuitenkin myös jatkokehitystarpeita löytyy, sillä järjestelmään oli suunniteltu ominaisuuksia, joista kaikkia ei projektin aikana ehditty toteuttaa. Osa suunnitelluista ominaisuuksista oli kokonaan uusia toimintoja, osa taas vain pieniä parannuksia esimerkiksi käytettävyyteen tai ulkoasuun.

Editoriin kaivattaisiin lisää asetuksia. Asiakas toi esiin tarpeen asetukselle, jolla voitaisiin vaatia kaikkiin käyrän pisteisiin ei-tyhjä kommentti. Asiakkaalla olisi ollut tarvetta myös erilaisille käyrälle liitettäville kontekstitiedoille, kuten kuville ja käyttökokemuksen paikan valinnalle, joita ei projektin aikana ehditty toteuttaa. Projektipäällikkö puolestaan toivoisi parempaa tukea mobiilikäyttöliittymille, erityisesti skaalautuvuutta. Tarvetta olisi myös erilaisille kysymystyypeille.

Päiväkirjatyylinen vastaaminen oli eräs olennaisimmista projektin aikana lisätyistä ominaisuuksista, joten sen toteuttaminen voidaan lukea suureksi positiiviseksi saavutukseksi toiminnallisuuden kattavuuden kannalta. Kuitenkin päiväkirjatyyliseen vastaamiseen oli alun perin suunnitteilla joustavampaa sivujen esityskertojen valintaa, joten toiminto ei ole vielä aivan loppuun asti hiottu.

Työkalu sai asiakkaalta kiitosta siitä, että sen avulla saatiin paljon mielenkiintoista dataa kerättyä innovatiivisella tavalla. Kuitenkin datan analysointiin liittyi vielä muutamia

ongelmia. Data Viewer on olennaisessa osassa sen suhteen, mitä tuloksia tutkimuksesta saadaan irti. Asiakkaan mukaan suuresta datamäärästä oli toisinaan vaikea erottaa olennaista, joten jatkokehityksessä tähän ongelmaan tulisi puuttua lisäämällä Data Vieweriin hyödyllisiä yhteenveto- ja analysointiominaisuuksia. Näkymään oli myös suunnitteilla vielä monipuolisempia suodatusmahdollisuuksia ja mm. asiakkaan toiveiden mukainen siirreltyjen tietoruutujen paikkojen sessiokohtainen tallennus, jolloin tietoruudut säilyttäisivät paikkansa vaikka näkymän lataisi uudelleen. Kokonaisuutena kuitenkin Data Vieweriin saatiin sisällytettyä paljon toivottuja ominaisuuksia, joten toimintojen suhteen näkymää voidaan pitää melko onnistuneena osana projektia.

Puutteita on myös lokalisoinnissa, jota tuetaan vain rajoitetusti. Asiakas toivoi parannusta kääntötoimintoon. Nykyisessä toteutuksessa kysymysten ja vastausten kääntäminen kieleltä toiselle tapahtuu omassa näkymässään, jolloin käytännössä kääntäjästä tulee oma käyttäjäroolinsa järjestelmään. Kuitenkin asiakas toivoi, että käännökset voitaisiin tehdä täysin erillään DrawUX:n käytöstä esimerkiksi käännöstoimistossa ja viedä DrawUX:n tietokantaan automaattisesti lukemalla käännöstiedot sisältävä tiedosto.

Näiden vielä puuttuvien ominaisuuksien lisääminen auttaisi järjestelmää vastaamaan paremmin erilaisiin käyttötilanteisiin. Järjestelmän toteutuksessa on siis edelleen kattavuuspuutteita verrattuna alkuperäisiin suunnitelmiin ja jatkokehitykselle olisi tarvetta. Kuitenkin alusta asti oli selvää, että tässä jatkokehitysprojekstissa ei millään voitaisi lisätä kaikkia suunnitteilla olevia ominaisuuksia järjestelmään, joten niitä priorisoitiin ja aikataulutettiin realistisella tavalla. Projektin mitalle aikataulutetut ominaisuudet saatiin lähes kokonaan toteutettua projektin aikana. Nykyinen toteutus kattaa selvästi olennaisimmat tarpeet ja työkalua voi nykyisellään käyttää tarkoitetulla tavalla tutkimuksissa.

5.2.3 Käyttöliittymät

Käyttöliittymien esteettiseen ilmeeseen ei ole juuri panostettu. Asiakas kritisoi kehityksen keskivaiheilla käyttöliittymiä kömpelöiksi ja toivoi visuaaliseen ilmeeseen hienosäätöä. Yhtenäisempää ja siistimpää ja modernimpaa ilmettä voisi tuoda jonkin käyttöliittymäkirjaston tai selainkehyksen käyttö. Esimerkiksi Bootstrap sisältää hyödyllisiä käyttöliittymäelementtipohjia, joiden käyttö koko järjestelmässä toisi sivustolle siistin, yhtenäisen ulkoasun ja samalla helpottaisi käyttöliittymän ylläpitoa. Vaihtoehtoisesti voitaisiin hyödyntää enemmän jQuery UI:n käyttöliittymäteemoja, joita on jo käytetty joissakin osissa järjestelmää, kuten mobiilisovelluksessa ja joissakin interaktiivisissa käyttöliittymäelementeissä.

DrawUX:ssa käyttöliittymiä ei voi toistaiseksi personoida muuten kuin valitsemalla kyselylle taustavärin ja hyödyntämällä rikkaan HTML-kysymyseditorin tarjoamia rajattuja tyylimahdollisuuksia (esim. tekstin koon ja värin valinta). Nämä ovat kuitenkin vain tutkijan työkaluja säätää vastaajan näkymää, eikä kumpikaan käyttäjäryhmä voi personoida omia käyttöliittymiään millään tavalla. Kyseessä on ns. *Tight-Skin* -

antipattern [32]. Käyttäjän kokemusta palvelusta voisi parantaa lisäämällä personointi- ja kustomointimahdollisuuksia.

Järjestelmän eri näkymät on pyritty saamaan ulkoasultaan yhtenäisiksi mm. sisällyttämällä sama pohjatyylitiedosto (*base.css*) kaikkiin tutkijan näkymiin. Kuitenkin näkymät poikkeavat toisistaan ainakin hieman. Jokaisella näkymällä on käytössä yksi tai useampi oma CSS-tiedosto. Tutkijan näkymiä yhdistää ylhäällä oleva navigointipalkki, mutta muuten näidenkin näkymien välillä on eroja mm. väreissä ja muissa tyyleissä. Osa näkymistä käyttää voimakkaita perusvärejä (esim. painikkeiden värit tunnustenhallintasivuilla, jossain määrin myös vastaajan näkymässä) kun taas osa näkymistä käyttää painikkeissa ja muissa elementeissä vaaleita pastellivärejä (esim. editorinäkymä ja Data Viewer). Erityisesti muista näkymistä poikkeaa vastaajan näkymä, jossa värimaailma on aivan erilainen ja jossa ei ole samaa globaalia navigointia kuin tutkijoilla. Tämä on kuitenkin ehkä tarkoituksenmukaista, sillä näkymää käyttää eri käyttäjäryhmä. Joissakin käyttöliittymissä on käytetty joitakin jQuery UI:n ominaisuuksia, erityisesti painikkeiden kuvakkeissa. Kuitenkaan jQuery UI ei ole kokonaisvaltaisesti käytössä.

Epäyhtenäisyyttä on myös käyttäjälle annettavan tilannekohtaisen palautteen esittämisessä. Esimerkiksi editorissa session umpeutumisesta ilmoittava teksti esitetään CSS:llä tyylitellyssä elementissä, joka muistuttaa tyyliltään vastaajalle annettavia vastausohjeita käyränpiirtonäkymässä ja estää muun näkymän käsittelyn ennen kuin ilmoitukseen on reagoitu klikkaamalla jotakin ilmoituksen painikkeista. Rekisteröinti- ja muilla lomakesivuilla, editorissa ja vastausnäkymässä on myös omat erilaiset tapansa esittää syötteeseen liittyviä pituus- ja sisältövirheitä, yleensä punaisina teksteinä mahdollisimman lähellä virheen lähdettä. Vastaajien hallintanäkymässä puolestaan ilmoitus vastaajan kutsumisen tai muun toiminnon onnistumisesta annetaan JavaScriptin *alert*-funktiolla eli täysin tyylittelemättömänä ponnahdusikkunana. Tästä on haittaa niin esteettiseltä kannalta kuin toiminnallisestikin, sillä käyttäjä saattaa asettaa selaimen kieltämään ponnahdusikkunat, jolloin palautetta ei saada. Käyttäjä saa siis eri näkymissä loogisesti samantyyppistä palautetta yhteensä ainakin kolmella täysin visuaalisesti erilaisella tavalla.

Työkalun käytettävyyttä tulisi tutkia tarkemmin, mutta jonkin verran palautetta on jo saatu asiakkaan pilottitutkimuksista. Palaute oli melko ristiriitaista, sillä toisaalta työkalu koettiin helpoksi ja sen käyttö onnistui lähes kaikilta vaivatta, mutta kuitenkin yksityiskohdista löytyi useita pieniä käytettävyysongelmia. Esimerkiksi samaa tuotetta useilla käyränpiirtotehtävillä arvioitaessa joillakin vastaajilla oli hankaluuksia hahmottaa y-akselin attribuutin merkitystä, eli mitä ominaisuutta tuli milloinkin arvioida. Asiakas arvioi, että vastaajien läsnäolo paikan päällä tehdyssä työkalun ja tutkimuksen esittelysessiossa oli tarpeellista, eikä työkalu näin ollen välttämättä soveltuisi etätutkimuksiin, joita varten se on tarkoitettu. Vastaajan näkymän käytettävyyteen tulisi siis panostaa vielä enemmän.

Myös tutkijoiden näkymistä löytyi käytettävyyspuutteita. Esimerkiksi Data Viewerissä käyräkuvaajan pisteiden korostus ei aina toimi intuitiivisesti järkevällä tavalla, vaan on hankala saada haluttu piste korostettua. Vapaalla suorakaidevalinnalla tapahtuva zoomaus saattaa toisinaan tapahtua vahingossa, eli elementti on liian herkkä lukemaan käyttäjän liikkeet zoomausyritykseksi. Kokonaisarvio työkalusta kuitenkin oli ”suoraviivainen ja helppokäyttöinen”.

5.2.4 Testauksen puute

Kuten luvussa 5.1.1 tuli ilmi, prosessi itsessään oli puutteellinen, sillä siitä puuttui varsinainen testaus kokonaan. Koska järjestelmällä oli vain yksi kehittäjä kunkin kehitysjakson aikana, kehittäjä itse joutui työnsä lomassa tekemään myös toiminnan varmistusta. Kuitenkaan testaukselle ei ollut varattu aikaa aikataulussa ja kunkin kehitysjakson tavoitteita määritettäessä, ja toteutettavaa toiminnallisuutta oli paljon, joten systemaattiselle testaukselle ei yksinkertaisesti ollut aikaa.

Sen sijaan kehityksen lomassa tehtiin kokeiluluontoista ad hoc -testausta aina kun jotain uutta oli kehitetty järjestelmään. Tällaisen testauksen riittävyyteen ei voida juurikaan luottaa, joten on luultavaa että toteutuksessa on edelleen joitakin virheitä jotka tulisivat vasta pitkäaikaisemman käytön tai järjestelmällisemmän testauksen yhteydessä ilmi.

5.3 Tehokkuus

Tässä luvussa käsitellään järjestelmän tehokkuuteen vaikuttavia ratkaisuja. Kaksi tehokkuuteen liittyvää olennaista kokonaisuutta ovat sivujen latausajat, joita käsitellään luvussa 5.3.1, ja selainkoodin suorituskyky, jota arvioidaan luvussa 5.3.2. Lisäksi luvussa 5.3.3 tarkastellaan, onko turhan palvelinlaskennan vähentäminen otettu tarpeeksi hyvin huomioon suunnittelussa.

5.3.1 Sivujen latausajat

Näkymistä olennaisin latausajan suhteen on vastaajalle esitettävä vastausnäkymä. Näkymä vaikuttaa käytössä nopealta, mutta latausaika riippuu kyselyn rakenteesta, eli paljonko dataa tietokannasta pitää lukea näkymän rakennusta varten. Pitkä kysely latautuu luonnollisesti hitaammin kuin lyhyempi kysely. Piirtotehtäviin liittyy erityisen paljon dataa verrattuna muihin tehtävätyyppeihin. Kun näkymä on kerran latautunut, sivulta toiselle siirtyminen on erittäin nopeaa, sillä vastausnäkymän sivujen tietoja ei tarvitse enää erikseen hakea tietokannasta, vaan jokaisen kyselysivun HTML on jo olemassa näkymässä piilotettuna.

Taulukkoon 1 on koottu joitakin vastausnäkymän kokoja ja latausaikoja 100/100 Mbit/s laajakaistayhteydellä testattuna. Testissä käytettiin kyselyä, jossa oli yksi vakiona pidettävä keskipituinen lomakesivu (7 kysymystä, kaikkia eri kysymystyyppejä) sekä vaihtu-

va määrä piirtotehtäviä. Taulukosta nähdään sivun koko kilotavuissa sekä DOM-sisällön valmistumisajat ja kokonaan ladatun sivun valmistumisajat. Erona näillä kahdella on se, että DOM-sisällön valmistumisaikaan ei ole laskettu mukaan kuvien, tyylitiedostojen ja alikehysten valmistumista. Käytännössä tällä sivulla ero on minimaalinen.

Taulukko 1. *Vastausnäkymän latausaikoja eri määrällä piirtotehtäviä.*

Piirtotehtävien määrä	Sivun koko (kt)	DOM-sisältö ladattu (s)	Kokonaan ladattu (s)
1	651	1,01	1,03
2	654	1,02	1,04
3	657	1,17	1,20
4	660	1,56	1,59

Taulukosta 1 havaitaan, että vastausnäkymä latautuu erittäin nopeasti realistisen kokoisella kyselyllä. Edes piirtotehtävien lisääminen ei hidasta latautumista merkittävästi.

Lähes kaikkien muidenkin järjestelmän sivujen latausajat ovat samaa luokkaa, mutta Data Viewer vaikuttaa muita sivuja selvästi hitaammalta. Tämä käy järkeen kun huomioidaan, että näkymään pitää lukea kaikki vastausdata ja samalla rakentaa mahdollisesti useita käyräkuvaajia joista kuhunkin alustetaan kaikki vastauskäyrät. Taulukko 2 kuvaa joitakin Data Viewerin latausaikoja. Testeissä käytetyssä kyselyssä on yksi lomakesivu ja yksi käyränpiirtotehtävä. Kuhunkin käyrään lisättiin noin viisi pistettä.

Taulukko 2. *Data Viewerin latausaikoja eri vastaajamäärillä.*

Vastaajia	Sivun koko (Mt)	DOM-sisältö ladattu (s)	Kokonaan ladattu (s)
10	1,3	5,12	5,21
20	1,4	10,15	10,21
30	1,5	17,41	17,47
40	1,6	28,83	28,88

Taulukon perusteella voidaan sanoa, että sivulla on latausaikaongelmia. Pienillä vastausmäärillä latausaika pysyy siedettävällä tasolla, mutta jo muutama kymmenen vastaajaa aiheuttaa yli 20 sekunnin latausajan, joka yleensä koetaan liian pitkäksi. Data Viewerin optimointi olisi selvästi tarpeen. Pullonkaulana on todennäköisesti kuvaajan alustus ja keskiarvokäyrän laskenta. Alustusta voitaisiin nopeuttaa esimerkiksi ’laiska alustus’-mallin avulla, eli jättämällä osa käyristä kokonaan alustamatta siihen saakka, kunnes tutkija valitsee ne näytettäviksi. Tällöin alustukseen liittyvä kuormitus jakautuisi tasaisemmin ja tutkija pääsisi nopeammin tarkastelemaan ainakin joitakin tuloksia. Lisää tutkimista vaatii kuitenkin keskiarvokäyrän laskennan vaikutus hitauteen. Jokaiselle käyrälle pitää ensin ekstrapoloida tietty määrä pisteitä, joita laskentaan käytetään, ja tämän jälkeen kaikista samaan x-kohtaan lasketuista pisteistä otetaan vielä keskiarvo. Tämä yleensä 100 pistettä eli runsaasati muita käyriä enemmän pisteitä sisältävä keskiarvokäyrä pitää myös lisätä kuvaajaan. Jos käyriä on paljon, nämä operaatiot saattavat

olla hitaita. Mikäli keskiarvokäyrän laskennalla havaitaan olevan merkittävä osuus näkymän hitauteen, keskiarvokäyräkin voitaisiin piilottaa oletuksena ja alustaa vasta, kun se halutaan näyttää.

Web-suorituskykyyn vaikuttaa myös palvelimelta selaimelle lähetettävien tiedostojen koko. DrawUX:n näkymät ovat melko suurikokoisia tavumäärältään. Esimerkiksi taulukoista 1 ja 2 nähdään, että Data Viewer (CSS, JavaScript, kuvat ja muut tiedostot mukaan luettuna) sisältää yli megatavun verran siirrettävää dataa ja vastaajan näkymä tästä noin puolet. Muut näkymät ovat noin 100–800 kilotavun suuruisia. Näkymien sekä niihin liitettävien skriptien ja CSS:n tavumäärää pienentämällä palvelin pystyisi lähettämään tiedostot nopeammin selaimelle. Tätä olisi kuitenkin vaikea tehdä ilman että karsittaisiin liikaa näkymien ulkoasua tai toiminnallisuutta. Vaikka HTML-koodia tuntuu olevan runsaasti, siirrettävästä datasta vain pieni osa on HTML:ää. Ylivoimaisesti suurin osa siirrettävästä datasta on JavaScriptia.

Latausaikoihin voidaan vaikuttaa myös palvelinpuolen ratkaisuilla, erityisesti tietokannan tasolla. Eräässä yksittäisessä tilanteessa tarvittavien tietojen hakemisen helpottamiseksi tietokantaan on lisätty yksi näkymä. Muuten tietokannassa ei ole hyödynnetty juurikaan tietokantaohjelmointia, kuten tallennettuja proseduureja ja herättimiä. Kaikki tietokantakäsittely tehdään malleista käsin, mikä on usein huomattavasti hitaampaa kuin antaa tietokannanhallintajärjestelmän hoitaa tällaiset asiat herättimien ja tallennettujen proseduurien avulla, sillä tietokannanhallintajärjestelmät on optimoitu suorituskykyä ajatellen. Suorituskykyä parantaisi se, että kaikkien tietokantakutsujen ei tarvitsisi kulkea CodeIgniterin *Active Record* -kerroksen läpi, vaan tapahtuisi suoraan tietokannan tasolla. Nykyinen menetelmä lisää myös virhealttiutta ja vaarantaa tietokannan eheyden, sillä ohjelmoijan tulee muistaa kirjoittaa kaikki operaation yhteyteen vaadittavat käsittelyaskeleet. Tästäkin haitasta päästäisiin järkevällä tallennettujen proseduurien ja herättimien käytöllä.

Esimerkiksi kyselyn muokkauksen yhteydessä mallissa kutsutaan erikseen funktiota, joka päivittää kyselyn muokkausajankohdan (Ohjelma 1). Tämän voisi hoitaa herättimenä, jolloin käsittely olisi nopeampaa (ei useaa erillistä tietokantakutsua) ja funktiota ei tarvitse muistaa kutsua aina, kun jokin koodin osa muokkaa kyselyn tietoja. [52]

```
public function survey_modify_update_time($surveyId) {
      $data['time_updated'] = date("Y-m-d H:i:s");
      $this -> db -> where('id', $surveyId);
      if ($this -> db -> update('surveys', $data)) {
            return true;
      }
      return false;
}
```

Ohjelma 1. *Funktio kyselyn muokkausajankohdan päivittämiseen. Tämä funktio voitaisiin korvata tietokantahallintajärjestelmässä sijaitsevalla herättimellä.*

Paremmasta indeksoinnista voisi olla apua latausaikojen parantamisessa. Tietokannanhallintajärjestelmä luo automaattisesti indeksit pääavaimista, mutta muita indeksejä ei ole itse lisätty. Indeksien käyttö parantaa hakuaikaa mutta hidastaa rivien lisäystä, poistoa ja päivitystä. Käytettävät indeksit tulisi siis valita huolellisesti ottaen huomioon, mitä tietoja kyselyissä haetaan. Indekseihin kannattaa valita sarakkeita, joilla on suuri valikoivuus eli vain vähän rivejä joilla on sama indeksoitavan sarakkeen arvo. Tärkeät vierasavainsarakkeet kannattaa indeksoida, samoin taululiitoksiin tai tulosten järjestämiseen tai ryhmittelyyn käytettävät sarakkeet. [53]

Kompromissi hakuaikojen parantamiseksi tallennusaikojen kustannuksella tässä järjestlemässä kannattaa, sillä tietoja tallennetaan harvemmin kuin luetaan. Tietojen tallennusvaiheessa on usein jo saatu jonkin kriittinen työvaihe tehtyä, jolloin ei ole niin suuri ongelma, jos käyttäjä joutuu odottamaan vastausta pidempään. Tietojen haku sen sijaan liittyy usein näkymän lataamiseen tai päivitykseen, jolloin olennainen käsittely on vasta alussa ja on kriittistä, että käyttäjä ei keskeytä sessiota hitaan latauksen takia.

5.3.2 Selainkoodin suorituskyky

DrawUX:n selainkoodissa Highcharts-käyränäkymät ovat suorituskyvyltään avainasemassa. Muiden näkymien selainkoodi on suhteellisen yksinkertaista eikä yhtä kovan rasituksen alla. Highcharts-käyräkuvaajat sen sijaan joutuvat käsittelemään monimutkaisia objekteja ja muokkaamaan käyttöliittymää ja Data Viewerissä myös käsittelemään useita eri käyriä yhtä aikaa ja reagoimaan jatkuvasti käyttäjän muuttamiin asetuksiin.

Käyränpiirto normaalissa tapauksessa vastaajan näkymässä ei näytä kuormittavan selainta, vaan näkymän käyttö pysyy sujuvana vaikka kyselyssä olisi useita piirtotehtäviä. Data Viewer tarvitsisi lisää kuormitustestausta, sillä vaikka se pienissä kokeiluluontoisissa testeissä on suoriutunut hyvin eikä ole esimerkiksi saattanut selainta ’ei vastaa’-tilaan. Samalla käyrämäärällä, joka auheutti merkittävän latausaikaongelman, Data Viewerin ajonaikainen suorituskyky pysyy edelleen hyvänä. 40 käyrän kuvaajaa voi zoomata, suodattaa ja muutenkin käsitellä melko lyhyillä vasteajoilla. Joidenkin sekuntien viiveitä esiintyy, jos kaikkia käyriä käsitellään kuvaajassa yhtä aikaa, mutta tämä ei muutenkaan ole suositeltavaa, sillä kuvaajasta tulee väkisinkin sekava. Näissä testeissä ei kuitenkaan ole käsitelty tarpeeksi suurta käyrämäärää, jotta testien perusteella voitaisiin tehdä lopullisia päätelmiä Data Viewerin suorituskyvystä. Alustavasti kuitenkin näyttäisi siltä, että Data Viewer -kuvaajien ajonaikainen suorituskyky on riittävän hyvä, jotta osan alustukseen liittyvää kuormitusta voisi siirtää ajonaikaiseen suoritukseen.

Kuten luvussa 5.3.1 mainittiin, eri näkymissä kuormitusta voitaisiin vähentää lisäämällä käyttöliittymään elementtejä vasta kun niitä todella tarvitaan. Esimerkiksi Data Viewerin käyttöliittymässä näytetään ensin oletuksena vain viisi uusinta vastauskertaa ja tutkija voi halutessaan valita useampia vastauksia näytettäväksi. Kuitenkin myös loput

vastaukset ja käyrät ovat olemassa näkymässä piilotettuina. Kyseessä on siis vain käyttöliittymän käytettävyyttä ja selkeyttä parantava ratkaisu, joka ei paranna suorituskykyä. Jos käyrät ja vastaukset todella lisättäisiin näkymään vasta, jos tutkija haluaa tarkastella näitä vastauksia, säästettäisiin vähintäänkin alkulataukseen kuluvaa aikaa ja mahdollisesti myös kokonaissuoritusaikaa, sillä tutkija ei välttämättä tarkastele kaikkia vastauksia.

Selaimen suorituskykyä voitaisiin lisätä hyödyntämällä enemmän funktioiden ketjutusta. Kun muutokset käyttöliittymäelementtiin tehdään jQueryn avulla ketjutettuina, täytyy käsiteltävää elementtiä muuttaa vain kerran, jolloin voidaan saada huomattavia suoritusaikaparannuksia verrattuna jokaisen muutoksen tekemiseen erikseen [49]. Kuitenkin testaus- ja ylläpitosyistä useimmissa DrawUX:n JavaScript-tiedostoissa on nimenomaan pyritty erottamaan jokainen elementteihin tehtävä muutos omalle rivilleen, mikä auttaa jäljittämään esimerkiksi millä rivillä jokin virhe tapahtuu, ja selkeyttää koodia hieman. Ratkaisu on kyseenalainen ja kenties ohjelmointivirheen löytymisen jälkeen koodi voitaisiin muuttaa ketjutetuksi. Kuitenkaan kaikkien jo koodissa olevien eri riveille erotettujen muutosten etsiminen ja ketjuttaminen ei luultavasti olisi siihen kuluvan ajan arvoinen työ, paitsi ehkä suoritusaikakriittisessä Data Viewerissä.

Muusta koodista poiketen Highcharts-kuvaajien yhteydessä ketjutusta on hyödynnetty aina kuin mahdollista. Aina, kun jonkin Highcharts-kuvaajan tietoja päivitetään, päivitysfunktioille voidaan antaa parametrina tieto siitä, halutaanko kuvaaja piirtää uudelleen vastaamaan muutoksia. Oletuksena tämä arvo on positiivinen, mutta jos kuvaajaan halutaan tehdä kerralla paljon muutoksia, parametrille voi antaa arvon *False* ja vasta muutosten jälkeen kutsua erikseen funktiota *Chart.redraw()*, joka piirtää kuvaajan kerralla uudelleen [54]. Näin vältytään lukuisilta turhilta ja raskailta uudelleenpiirroilta. Tätä menetelmää kaaviomuutosten ketjuttamiseen on hyödynnetty useissa kohdissa koodia, mikä on olennainen suorituskykyparannus, sillä DOM-muutokset ja SVG:n piirtäminen ovat kriittisiä osia selainkoodin suorituskyvyn kannalta.

Olennainen osa selainkoodin tehokkuutta on luonnollisesti se, että turhaa laskentaa vältetään aina kuin mahdollista. Enimmäkseen DrawUX:ssa tämä toteutuu, ylimääräiseen koodiin ei ole projektissa ollut edes aikaa, joten kaikki toteutettu toiminnallisuus on selvästi ollut haluttua ja tarpeellista. Data Viewerissä on erityisesti yritetty vähentää kuormitusta käyttämällä *Päivitä*-nappia automaattisen muutosten käyttöönoton sijaan. Kun käyttäjä muuttaa asetuksia, kuten valitsee erilaisia suotimia tai kuvaajaan sisällytettäviä vastauksia, muutokset eivät heti näy käyräkuvaajassa, vaan käyttäjän pitää painaa nappia päivittääkseen näkymän. Tämä pieni hidastus käytettävyyteen on suureksi hyödyksi suorituskyvylle, sillä näin voidaan ryhmittää muutoksia. Jos kaikki muutokset tulisivat automaattisesti heti asetusten muuttamisen jälkeen voimaan, jouduttaisiin tekemään paljon ylimääräistä raskasta laskentaa kun kuvaajaa päivitettäisiin jatkuvasti. Napista on erityisesti hyötyä tilanteissa, joissa käyttäjä valitsee jonkin asetuksen vahin-

gossa ja poistaa valinnan heti. Kun nappia ei paineta tässä välissä, ei turhaa laskentaa tarvitse tehdä.

Ohjelmasta löytyy myös koodia, joka voitaisiin mahdollisesti korvata tehokkaammalla vaihtoehdolla. Hyvä esimerkki on muutostaulukon käyttö editorissa ja Data Viewerissä. Editorinäkymässä selainkoodin tulee jollakin tavalla merkitä muistiin, mitä muutoksia käyttäjä on tehnyt kyselyyn, jotta muutosten tiedot voidaan käsittelyn lopuksi lähettää palvelimelle tietokantaan tallennettavaksi. Muutostietojen merkintään olisi useita vaihtoehtoja. DrawUX:ssa on käytössä muutostaulukko, eli JavaScriptissa on julkinen taulukko, johon merkitään kunkin muutoksen tyyppi, elementti johon muutos kohdistuu, ja muut yksityiskohdat muutoksesta. Kuhunkin editorissa näytettävään kyselyelementtiin liittyy tapahtumankäsittelijä, joka reagoi jokaiseen käyttäjän tekemään muutokseen lisäämällä sen tiedot muutostaulukkoon. Esimerkiksi kun jotakin kysymyksen tekstille tarkoitettua kenttää muokataan, muokkaus kutsuu kentälle lisättyä takaisinkutsufunktiota, joka puolestaan kutsuu *changes_question*-funktiota. Tämän funktion muutosten tallennukseen liittyvä osa on esitetty hieman yksinkertaistettuna ohjelmassa 2.

```
function changes_question(…) {
      //käyttöliittymästä haetaan muutoksen tiedot
      var questionId = parseInt(question.attr('data-question-id'));
      var questionType = question.attr('data-question-type');
      var pageId = parseInt(question.parent().parent().attr('data-page-id'));
      var pageIndex = changes_find_page_index(pageId, 'form');
      var reqAnswer = 0;
      if ($(question).find('#requireanswer').is(':checked')) {
            reqAnswer = 1;
      }
      //muutoksen tyyppi saadaan parametrina, tässä esitetty vain muokkaus
      if (operation === 'modify') {
            var questionIndex = changes_find_question_index(questionId, ques-
            tionType, pageIndex);
            //changes on globaali muutostaulukko
            //tekstejä täytyy muokata käytännön syistä
            changes.pages[pageIndex].questions[questionIndex].text =
            question.find('textarea').val();
            changes.pages[pageIndex].questions[questionIndex].number  =  par-
            seInt(question.attr('data-question-number'));
            changes.pages[pageIndex].questions[questionIndex].reqAnswer     =
            reqAnswer;
      }
}
```

Ohjelma 2. *Kysymyksen muutostietojen lisääminen muutostaulukkoon changes_question-funktiossa.*

Ensin haetaan editorin DOM:sta kaikki tarvittavat tiedot. HTML-elementeillä on käytetty useita data-alkuisia attribuutteja tällaisten tietojen keräämistä varten. Muutostaulukosta pitää löytää oikea päivitettävä kohta, jota etsitään erillisissä funktioissa *changes_find_page_index* ja *changes_find_question_index*. Näitä indeksejä käytetään päivittämään kysymykseen liittyvät tiedot eli kysymyksen teksti, kysymyksen numero ja se,

onko kysymykseen vastaaminen pakollista. Vaihtoehtoja tai skaaloja sisältävät kysymystyypit sisältävät enemmän käsittelyä, ohjelmassa 2 on esitetty vain tavalliseen tekstikenttätyyppiseen kysymykseen liittyvät askeleet. Koska editorissa on paljon erilaisia kyselyelementtejä, joihin voidaan kohdistaa useita erilaisia muutoksia, muutostaulukon päivittämiseen tarvitaan runsaasti koodia. Data Viewerissä on lähes samanlainen tilanne, jossa muutostaulukkoa käytetään pitämään kirjaa muutetuista käyräkuvaajan asetuksista (mm. suotimet ja käyttäjien valinnat).

Taulukon läpikäyminen on hidasta kun sitä joudutaan tekemään usein. Kuten ohjelmasta 2 nähdään, jo pienten muutosten lisäämiseen käytetään suhteessa suuri määrä koodia. Vaihtoehtoinen tapa olisi merkitä muutetut elementit suoraan HTML:n. Jokaisella elementillä tulisi edelleen olla tapahtumankäsittelijä, mutta käsittelijät olisivat yksinkertaisempia, sillä ne vain muokkaisivat tapahtuman laukaissutta elementtiä itseään sen sijaan että niiden tulisi käydä läpi suurta muutostaulukkoa. Kun käyttäjä haluaa tallentaa vastauksensa, koko editorin html voitaisiin käydä läpi välilehti kerrallaan ja etsiä HTML-merkkauksesta muutoksia vastaavat kohdat ja lähettää niiden tiedot palvelimelle. Suurin osa esimerkiksi ohjelman 2 käsittelystä voitaisiin jättää kokonaan pois, sillä kysymyksen teksti ja muut asetukset olisivat suoraan luettavissa HTML:stä. Eli sen sijaan että joka muokkauksen yhteydessä nämä tiedot luetaan käyttöliittymästä, ne tarvitsisi lukea vain kerran siinä vaiheessa, kun tiedot halutaan lähettää palvelimelle.

Lisäksi vaihtoehtoista ratkaisua puoltaa se, että kaikki käyttäjän tekemät muutokset eivät välttämättä ole pysyviä. Jos käyttäjä esimerkiksi lisää sivulle uuden kysymyksen, mutta muuttaa heti mielensä ja poistaa kysymyselementin, kysymyksen tietojen lisäys muutostaulukkoon ja sitten haku ja poisto taulukosta vievät runsaasti aikaa suhteessa vaihtoehtoon jossa vain lisättäisiin uusi HTML-elementti sopivalla uutta lisäystä kuvaavalla tunnisteella. Elementin poiston yhteydessä ei tarvitsisi tehdä mitään lisätoimenpiteitä, vaan lisätty elementti voitaisiin yksinkertaisesti poistaa HTML-koodista. Ratkaisun huonona puolena olisi kuitenkin se, että HTML-merkkauksen tietojen muunto JSON-dataksi ei olisi yhtä suoraviivaista kuin jo valmiiksi kootun JavaScript-taulukon muunto JSON-muotoon. Tämä lisäaskel saattaisi kumota osan ratkaisulla saavutettavista suoritusaikaeduista, mutta ei kuitenkaan kaikkia, sillä vaikka muutostiedoista jouduttaisiinkin kokoamaan taulukko, tämä tapahtuisi kerralla eikä taulukko olisi jatkuvan päivityksen alaisena.

Edellä esitetty DOM:n käsittelyyn perustuva ratkaisu parantaisi tehokkuuden lisäksi myös ylläpidettävyyttä, sillä editori on eräs muutosaltteimmista osista ohjelmaa. Erilaisia kysymystyyppejä ja kyselyyn liittyviä asetuksia on jatkuvasti tarvetta lisätä. Muutostaulukko hankaloittaa ohjelman ylläpitoa, sillä aina kun halutaan lisätä editoriin jokin uusi asetus, elementille tulee kirjoittaa tapahtumankäsittelijä joka lisää, poistaa tai muokkaa tietoja muutostaulukossa.

Edelleen käsittelyä voitaisiin helpottaa käyttämällä jotakin sopivaa selainkehystä, joka pitäisi yllä DOM-elementtien ja niitä vastaavan datan välistä yhtenäisyyttä. Esimerkiksi Googlen AngularJS:n datansidontamekanismi pystyisi tähän [55]. Tällaisen valmiin kehyksen käyttö parantaisi melko varmasti ylläpidettävyyttä, mutta todennäköisesti heikentäisi tehokkuutta jos kehyksessä on oletuksena mukana myös ei-tarpeellista laskentaa.

Muutenkin selainkehyksen eli DrawUX:n tapauksessa jQueryn käyttö voi vaikuttaa negatiivisesti suorituskykyyn puhtaaseen JavaScriptiin verrattuna [49]. Jos selainkehys jätettäisiin pois ja ohjelmoitaisiin puhtaalla JavaScriptilla, voitaisiin silti toteuttaa kaikki mitä jQueryllakin voidaan, ja voitaisiin jättää myös paljon jQueryn tarpeettomia ominaisuuksia pois ja koodata vain olennainen. Esimerkiksi jQueryn sisäänrakennettu useimpien selainten tukeminen ei välttämättä ole tarpeen DrawUX-projektille, jossa on alusta asti painotettu järjestelmän käyttöä Google Chrome -selaimella. Barkerin [49] testeissä todettiin jQueryn *each*-funktio huomattavasti hitaammaksi kuin vastaava silmukka puhtaalla JavaScriptilla. Samoin DOM-elementtien käsittely oli merkittävästi hitaampaa.

Jos kehystä kuitenkin halutaan käyttää, jQuery on suorituskyvyltään hyvä vaihtoehto [56]. Lähteen suorituskykytesteissä jQuery 1.7.0 pärjäsi hyvin verrattuna muihin vastaaviin selainkehyksiin (Dojo 1.7.2, MooTools 1.4.4, YUI 2.9.0, ExtJS 4.0.7 ja Protoype 1.7). DrawUX on ensisijaisesti suunniteltu käytettäväksi Google Chrome-selaimella, joten suorituskyky Chromella on olennaisin. Näissä testeissä ainoastaan Prototype ja ExtJS pärjäsivät paremmin. Kaikki havaitut erot olivat kuitenkin pieniä.

jQueryn lisäksi muillakin teknologiavalinnoilla on vaikutusta suorituskykyyn. DrawUX käyttää JSON-koodausta sekä Ajax- että lomakkeenlähetyspohjaisessa kommunikoinnissa. Lähteen [57] mukaan JSON on perinteistä XML:ää parempi vaihtoehto datansiirtoon selaimen ja palvelimen välillä. Tiedonsiirrossa XML säilyttää datan hierarkkisen puurakenteen, joten sen parsiminen on raskasta. Yksinkertainen JSON on helpompi generoida ja parsia. JSON-muodossa tiedonsiirto on nopeampaa riippumatta lähetettävän datan määrästä, tosin suurilla datamäärillä erot ovat merkittävämpiä. Lisäksi palvelimen lähettämä data pitää deserialisoida. JavaScriptin *eval*-funktio tukee suoraan helppoa parsimista: JSON-taulukoista saadaan suoraan JavaScript-taulukkoja ja JSON-objekteista JavaScript-objekteja. XML vaatii enemmän käsittelyä jotta puumuoto saadaan tulkittua selaimessa, joten JSONin tulkinta selaimessa on paljon tehokkaampaa kuin XML:n. XML:n tulkinta-aika riippuu myös voimakkaasti datan määrästä, JSONilla samaa ei ole juuri havaittavissa, joten erityisesti suurella datamäärällä JSON on suositeltava.

Highcharts on olennainen osa suorituskykyä, sillä SVG-pohjainen piirtäminen kuormittaa selainta. Highchartsin vakuutetaan kuitenkin toimivan nopeasti ja luotettavasti kohtuullisella datamäärällä useisiin tuhansiin datapisteisiin asti. Tämän pitäisi riittää Dra-

wUX:n tarpeisiin. Ongelmia alkaa tulla jos pisteitä on kymmeniä tuhansia. Highcharts-optimointiin tarkoitettuja moduuleja on myös olemassa, kuten kehitteillä parhaillaan oleva boost.js. [58]

5.3.3 Turhan palvelinlaskennan välttäminen

DrawUX:n kehityksessä on pyritty jakamaan mahdollisimman paljon laskentaa selaimelle, jotta yhteisestä palvelimesta ei muodostuisi suorituksessa pullonkaulaa. Tämän vuoksi esimerkiksi monet validointiaskelet pyritään tekemään selaimessa aina kuin mahdollista, jotta turhilta palvelinkutsuilta vältytään.

Turhan palvelinlaskennan välttämisen periaate toteutuu myös siinä, että ohjelmassa on hyödynnetty runsaasti Ajax-kommunikointia, jossa päivitetään vain pieni osa näkymää sen sijaan että koko näkymä tarvitsisi pienten muutosten jälkeen ladata uudelleen. Joissakin näkymissä olisi mahdollista korvata vielä lisää palvelinkutsuja osittaisella Ajax-päivityksellä. Esimerkiksi kuvan 16 käyttötapaus on tehokkuuden suhteen epäideaali. Näkymä pysyy lähes samana, mutta silti se ladataan kokonaan uudelleen *redirect*-vastauskoodin seurauksena. Tämä tapahtumasekvenssi kannattaisi suunnitella uudelleen siten, että päivitettävät tiedot lähetettäisiin Ajax-kutsuna palvelimelle ja näkymään voitaisiin tehdä pieniä muutoksia takaisinkutsufunktiossa. Näin kuvasta jäisi kokonaan jälkimmäinen kutsu pois, ja käsittely nopeutuisi huomattavasti.

5.4 Ylläpidettävyys

DrawUX:n ylläpidettävyyteen ja nopeaan kehitykseen vaikuttavat monet asiat teknologiavalinnoista koodin määrään ja rakenteeseen. Kuten luvussa 5.1.1 todettiin, toteutusprojektia rajoittivat merkittävät aika- ja työvoimarajoitukset ja näin tulisi todennäköisesti olemaan jatkokehitysprojektissakin. Näin ollen ohjelman ylläpidettävyyden tulisi olla sellaisella tasolla, että se tukee helppoa, nopeaa jatkokehitystä. DrawUX-järjestelmässä näin ei kuitenkaan aina ole ollut. Syynä ylläpidettävyyttä heikentäviin ominaisuuksiin on ollut lyhyen tähtäimen ajattelu, eli koodia on kehitetty mahdollisimman nopeasti kiinnittämättä huomiota tyyliin tai muihin ylläpidettävyyttä tukeviin ominaisuuksiin, keskittyen ainoastaan siihen että haluttu toiminto saadaan toteutettua mahdollisimman tehokkaasti. Näin säästyy aikaa vain lyhyellä tähtäimellä, sillä huonosti ylläpidettävä koodi vie aikaa siinä vaiheessa, kun samaa kohtaa tarvitsisi muokata uudelleen. Seuraavissa aliluvuissa eritellään joitakin ylläpidettävyyteen negatiivisesti tai positiivisesti vaikuttavia ratkaisuja, joita ohjelmassa on käytetty.

5.4.1 Teknologiavalinnat

Nopean kehityksen kannalta ratkaiseva päätös oli käyttää projektissa mahdollisimman paljon valmiita ilmaisohjelmistokomponentteja. Näitä on käytetty sekä ohjelmiston pe-

rusrakenteeseen (CodeIgniter) että erikoistuneeseen toiminnallisuuteen (Highcharts, PHPExcel). Nämä ratkaisut nopeuttivat kehitystä huomattavasti, kun kaikkea ei tarvinnut toteuttaa alusta asti. Käytetyt teknologiat on valittu tarkoituksenmukaisesti, sillä ne ovat hyvin muokattavissa ja laajennettavissa ja näin ollen palvelevat projektin tarpeita. Erityisesti Highcharts on kätevästi laajennettavissa omilla tapahtumankäsittelijöillä ja takaisinkutsufunktioilla. Myös CodeIgniteriin on ollut helppo liittää muita kolmannen osapuolen komponentteja, kuten PHPExcel. Käytetyt teknologiat on myös dokumentoitu kattavasti, mistä oli suurta hyötyä projektin aikana.

Ohjelmistokehyksen käyttö parantaa ylläpidettävyyttä huomattavasti tarjoamalla selkeän, hyväksi todettua arkkitehtuuria noudattavan rakenteen, joka on tässä tapauksessa MVC-malli. CodeIgniter myös abstrahoi tietokannan, mikä tekee tietokannan mahdollisesta vaihtamisesta erittäin helppoa – muutoksia tarvitaan ainoastaan tietokanta-asetukset sisältävään tiedostoon.

jQueryn käyttö parantaa ylläpidettävyyttä tekemällä koodista luettavampaa tavalliseen JavaScriptiin verrattuna. Esimerkiksi elementtien haku CSS-tyyppisillä valitsimilla on helpompaa kuin vastaavien JavaScript-funktioiden käyttö. Kun koodi on helpommin ymmärrettävää, sitä on helpompi ylläpitää. Samoin JSONin käyttö XML:n sijasta tiedonvälitykseen on ymmärrettävyyden välityksellä ylläpidettävyyttä parantava päätös.

5.4.2 Dokumentointi

Suuri haitta ylläpidettävyyden kannalta on se, että kehitysprosessi ja järjestelmän toteutus on alkuvaiheen suunnitelmien jälkeen ollut heikosti dokumentoitu. Eri ominaisuuksien toteutusta seurattiin Azendoo-projektinhallintajärjestelmällä, mitä samalla voidaan pitää dokumentaationa siitä, mitä järjestelmään on toteutettu ja mitä vielä puuttuu. Lisäksi käytetyt kolmannen osapuolen teknologiat versionumeroineen kirjattiin erilliseen listaan omalle web-sivulleen. Tämän jatkokehitysprojektin ensimmäisen osan päätyttyä kirjoitettiin myös hyvin suppea, mutta olennaisimmat asiat käsittelevä dokumentti perusarkkitehtuurista ja tiedostorakenteesta avuksi mahdolliselle jatkokehittäjälle. Lisäksi versionhallinnan kommenteista sekä DrawUX-järjestelmän uutissivulta löytyy tietoja toteutuksen etenemisestä. Myös tietokannan skeemasta on olemassa kuva, mutta sitä ei ole päivitetty muutosten myötä. Dokumentteja järjestelmän suunnittelemisesta on olemassa, mutta niihinkään ei ole kirjattu mitään todellisesta kehityksestä.

Tarpeelliset tiedot ovat siis hajallaan eri paikoissa, ja mitään varsinaista kattavampaa dokumentaatiota toteutuksesta ei ole. Esim. Horchin [59] mukaan dokumentit ovat ylläpitäjän tärkein työkalu, joita ilman ylläpitäjän tulee keksiä uudelleen tai päätellä tiedot, joiden pohjalta he voivat tehdä päätöksensä.

Dokumentaation vähyys oli kehitysresurssien rajallisuuteen pohjautuva tietoinen päätös. Työkalua on ollut toteuttamassa vain yksi ohjelmoija kerrallaan, joten ajateltiin, ettei

ohjelmoijan työaikaa kannata käyttää dokumentteihin, joita lukisi vain kehittäjä itse. Näin pienessä projektissa oletettiin, että kehittäjä voi muistaa toteutuksesta kaiken tarvittavan joko ulkoa tai koodin kommenttien avulla.

Koodi itsessään on melko monimutkaista. Sitä on kommentoitu jonkin verran – joka funktion alussa on kommentti joka kuvaa mitä funktio tekee – mutta kommentit kirjoitettiin usein vasta kauan koodin kirjoittamisen jälkeen ja kiireellä, joten niistä voi puuttua olennaisia tietoja eivätkä ne kuvaa tarpeeksi yksityiskohtia. Kehittäjä saattaa joutua miettimään esimerkiksi miksi tietty tietorakenne on käytössä, mitä jokin silmukka tekee tai miksi jokin aiempi koodin kohta on kommentoitu pois ja korvattu toisella ratkaisulla. Tällaisissa tilanteissa olisi hyvä, jos kaikki hylätytkin ratkaisut olisi dokumentoitu ja hylkäyspäätökset perusteltu, jotta kehittäjä ei tuhlaisi aikaa tekemällä parannuksia, jotka on jo aiemmin todettu epäkäytännöllisiksi. Erityisen suuri dokumentoinnin tarve olisi tiedostojen niille kohdille, joissa Highchartsia laajennetaan omalla toiminnallisuudella. Nämä funktiot ovat monimutkaisia ja niiden kokonaisuutta voi olla vaikea hahmottaa. Samoin mallien funktiot, joissa tehdään monimutkaisia tietokantataulujen yhdistelyjä, voivat olla vaikeasti tulkittavia, vaikka CodeIgniterissa mallin koodi onkin luettavampaa kuin puhdas SQL olisi.

Positiivista on, että koodissa käytetyt muuttujat ja funktiot on nimetty järkevästi. Nimistä on melko helppo päätellä, mitä muuttuja sisältää tai mitä funktio tekee. Esimerkiksi funktiot *toggleSelection(series)* vaihtaa annetun käyrän tilan valituksi, jos käyrä ei ollut valittuna, ja poistaa valinnan jos käyrä oli valittuna. Muuttuja *activeDrawing* sisältää sillä hetkellä aktiivisena (näkyvissä) olevan Highcharts-kuvaajan. Muutkin muuttujat ja funktiot on pyritty nimeämään tällä tavalla intuitiivisesti. Pieniä epäyhtenäisyyksiä nimeämisessä käytetyssä terminologiassa kuitenkin on. Esimerkiksi Data Viewer -näkymässä keskiarvokäyrään viitataan sekä nimellä *mean curve* että *average curve* ja käyräkuvaajista käytetään vaihdellen muun muassa nimityksiä *chart* ja *drawing*. Tämä saattaa hankaloittaa koodin tulkintaa hieman, mutta tuskin kuitenkaan kriittisellä tavalla.

5.4.3 Kooditiedostojen laatu

DrawUX on projektin työvoimaan nähden suurikokoinen ohjelma. Koodin suuri määrä vaikeuttaa ylläpitoa (esim. [7], [6], [51], [60]), mutta koodia on hankala karsia jos ei haluta luopua joistakin jo toteutetuista ominaisuuksista. Sen sijaan koodista voisi tehdä paremmin hallittavaa jakamalla sitä selkeämpiin pieniin osiin. Tällä hetkellä koodia on liikaa yhdessä tiedostossa. Ylipitkät tiedostot vaikeuttavat ylläpitoa, koska ohjelmoijalla kuluu turhaa aikaa jo halutun koodin kohdan hakemiseen, varsinkin jos muutoksia pitää tehdä jatkuvasti eri puolille tiedostoa.

Taulukko 3. *Tärkeimpien JavaScript-tiedostojen pituudet.*

Tiedoston nimi	**Pituus (koodiriviä)**
Kyselyn editointi	
form_edit_changes.js	502
form_edit.js	2064
Vastaaminen	
chart_jquery.js	940
chart.js	542
form_diary.js	372
form_retrospective.js	509
form.js	844
Data Viewer	
data_viewer_filters.js	591
data_viewer.js	1721
drawing_init.js	307
tooltip_labels.js	823

Taulukko 3 esittää olennaisimpien JavaScript-tiedostojen pituudet koodiriveissä. Taulukko 4 puolestaan sisältää olennaisimpien PHP-tiedostojen koot.

Taulukko 4. *Tärkeimpien PHP-tiedostojen pituudet.*

Tiedoston nimi	**Pituus (koodiriviä)**
Ohjaimet	
responses.php	764
surveys.php	843
auth.php	594
Mallit	
responses_model.php	1241
surveys_model.php	1305
Näkymät	
form/form_diary.php	178
form/form_preview.php	299
form/template_drawing_diary.php	9
form/template_drawing.php	51
form/template_end.php	5
form/template_form.php	51
form/template_options.php	21
form/template_scales.php	25
form_editor/form_create_fill_pages.php	85
form_editor/form_create.php	182
form_editor/template_drawing.php	368
form_editor/template_end.php	23
form_editor/template_options.php	333
form_editor/template_question_options.php	39
form_editor/template_question_scales.php	38
form_editor/template_question.php	94
data_viewer.php	375
my_surveys.php	57
responses.php	28
survey.php	102
translate.php	61

Taulukoista nähdään, että JavaScript-tiedostot ja palvelinpuolella ohjaimet ja mallit ovat pitkiä (usein jopa yli 1000 riviä). Sen sijaan näkymätiedostot ovat enimmäkseen pienikokoisia (alle 100 koodiriviä) ja pisimmälläänkin alle 400 koodiriviä. Tämä ero johtuu siitä, että näkymät koostetaan useita pieniä template-tiedostoja hyödyntäen, joten mikään yksittäinen näkymätiedosto ei veny liian pitkäksi.

Joihinkin tiedostokokoihin on helpompi vaikuttaa kuin toisiin. Palvelimella käsittelijöiden on lähes pakko olla näin suuria, jotta tietyn näkymän www-osoite pysyisi haluttuna, mutta mallit voitaisiin mahdollisesti jakaa pienempiin selkeämpiin osiin. Erityisen suuri ongelma ovat JavaScript-tiedostot, jotka ovat paisuneet pahimmillaan yli 2000 koodirivin mittaisiksi.

Koodin jakaminen useampiin pienempiin tiedostoihin tekisi kokonaisuudesta helpommin hallittavan. Ohjelmoijan tarvitsisi avata vain tietyt tiedostot ja järkevän kokoisesta tiedostosta etsitty kohta löytyisi helpommin kuin massiivisesta tiedostosta. Kuitenkin

tiedostojako tulisi tehdä hyvin harkitusti ja loogisin perustein, jotta ohjelmoijan tai jatkokehittäjän olisi helppo hahmottaa jo tiedostonimen perusteella, mitkä funktiot mistäkin tiedostosta löytyvät. Jos sama loogisesti yhteenkuuluva toiminnallisuus hajautetaan satunnaisesti useisiin tiedostoihin yhden tiedoston koon pienentämiseksi, ylläpidettävyys saattaa vain kärsiä [60]. Tämän vuoksi palvelinpuolen luokkia ei välttämättä kannata lähteä pilkkomaan, vaikka ne ovatkin rivimäärältään suuria. Näkymätiedostojen jo olemassa olevaa hierarkkista rakennetta ei missään nimessä kannata enää syventää. Sen sijaan selainpuolella yksittäinen JavaScript-tiedosto saattaa sisältää vain löyhästi toisiinsa liittyviä tapahtumankäsittelijöitä ja muuta koodia, joten tiedostojen pilkkominen pienempiin loogisiin osiin voi tulla kyseeseen.

Mitään tarkkaa rajaa sopivalle tiedostopituudelle ei ole, mutta esimerkiksi tutkimuksessa, jossa todettiin tiedoston koon vaikuttavan ylläpitotyömäärään enemmän kuin tunnettujen antipatternien, tutkituissa järjestelmissä oli keskimäärin 89–189 koodiriviä tiedostoa kohti [60]. Tämä on selvästi vähemmän kuin DrawUX:n skripteissä, vaikka mukaan laskettaisiin yllä lueteltujen tärkeimpien tiedostojen lisäksi myös pienemmät tiedostot.

Koodin pituuden lisäksi sen ylläpidettävyyteen vaikuttaa koodin selkeys. DrawUX:n koodissa on melko sekava rakenne, erityisesti selainskripteissä, joissa on sekaisin jQuery- ja JavaScript-koodia ja samat asiat on toteutettu useiden eri syntaksien kautta. Suurin osa koodista on tapahtumankäsittelijöitä erilaisille DOM-elementeille. Jo oikean tapahtumankäsittelijän löytäminen voi olla hankalaa, sillä tapahtumankäsittelijän ja käyttöliittymäelementin yhteen liittämiseen käytetään ainakin kolmea eri syntaksia. Osa triviaaleista käsittelijöistä on kirjoitettu kokonaisuudessaan *$(document).ready()*-osioon, osa tapahtumankäsittelijöistä liitetään siellä oikeisiin elementteihin mutta käsittelijän runko on muualla, kun taas osa tapahtumankäsittelijöistä on kirjoitettu kokonaan *$(document).ready()*:n ulkopuolelle omana erillisenä koodinaan. Lisäksi liittämiseen on käytetty kahta erilaista syntaksia: osa käsittelijöistä liitetään tapahtuman nimeä kuvaavalla funktiolla, esim. *click()*, jolle annetaan parametrina haluttu tapahtumankäsittelijä. Osa käsittelijöistä taas liitetään yleisellä *.on(events [, selector] [, data], handler)*-funktiolla, jolle voidaan antaa parametrina myös halutun kohde-elementin valitsin. Näin voidaan käyttää myös halutun kohde-elementin yläelementtejä liittämään kohteisiin haluttuja käsittelijöitä. Tällaisten delegoitujen tapahtumien käytöllä voidaan liittää käsittelijöitä esimerkiksi dynaamiseen sisältöön, jota ei vielä käsittelijöiden liittämisen aikaan ole olemassa [61]. Osa koodin sisällä alustetuista monimutkaisemmista olioista myös ottaa alustuksen yhteydessä jäsenmuuttujien arvona eri tapahtumien käsittelijöitä, mikä on jälleen yksi erilainen tapa liittää tapahtumia käsittelijöihin. Kaikkiaan koodissa on siis käytetty sekaisin vähintään viittä eri tapaa liittää DOM-elementille tapahtumankäsittelijä. Ohjelmat 3 ja 4 havainnollistavat näitä erilaisia tapoja. Tällaisesta syntaksien ja käytäntöjen paljoudesta on vaikea ottaa selvää ja löytää halutun käsittelijän ydinkoodi nopeasti. Erilaiset käytetyt syntaksit vaikeuttavat myös editorin hakutoiminnon käyttöä halutun tapahtumankäsittelijän löytämiseen.

```
$(function() {

	//käsittelijän liittäminen .click()-funktiolla
	$(".save_exit").click(function(){
		//käsittelyä
	});

	//startSelection-nimisen käsittelijän liittäminen .on()-funktiolla
	$(".drawing_container").on("mousedown", startSelection);

	//käsittelijä olion jäsenmuuttujan (activate) arvona
	$('#respondent_controls').tabs({
		activate : function(event, ui) {
			//käsittelyä
		}
	});
});
```

Ohjelma 3. *Esimerkkejä tapahtumankäsittelijöiden liittämisestä $(document).ready():n sisällä click()-funktion kautta, on()-funktion avulla, sekä olion jäsenmuuttujan arvona. Yksityiskohdat on jätetty pois listauksesta.*

```
$(function() {
	…
});

$("#download_img").click(function(){
	//käsittelyä
});

$("body").on("click", ".reset_zoom", function() {
	//käsittelyä
});
```

Ohjelma 4. *Tapahtumankäsittelijöitä, jotka on liitetty elementteihin $(document).ready():n ulkopuolella click()-funktion ja on()-funktion avulla.*

Funktioiden järjestys tiedostoissa ei myöskään noudata mitään yhtenäistä käytäntöä. Useimmissa tiedostoissa ensimmäisenä on *$(document).ready()*, mutta ainakin yhdessä tiedostossa se on vasta tiedoston lopussa. Osassa tiedostoista funktiot on asetettu aakkosjärjestykseen, osassa tapahtumankäsittelijät ovat ensimmäisenä ja muut funktiot alla. Joissakin tiedostoissa suunnilleen samaan kokonaisuuteen liittyvät funktiot ovat toisiaan lähellä koodissa, kun taas toisista tiedostoista on vaikea havaita minkäänlaista loogista järjestystä. Tämä yhdistettynä tiedostojen suureen pituuteen tekee kokonaisuudesta hankalasti hahmotettavan.

Sekavuuden tunnetta koodiin lisäävät myös epäyhtenäiset kirjoitusasukäytännöt nimeämisessä. Samasta tiedostosta löytyvät esimerkiksi funktiot nimeltä *find_drawing_task(number)* ja *findDrawingByPageNumber(number)*. Nimeäminen ei siis noudata mitään yhtä tyyliä vaan vaihtelee sekalaisesti eri tyylien välillä, lähinnä

edellä esiteltyjen tyylien, ns. käärmetyylin ja karavaanityylin, välillä. Spinellis [62] väittää koodissa käytettyjen tyyliseikkojen (formaattien, nimeämisen, sisennysten ja kommenttien) yhtenäisyyden olevan merkittävin koodin luettavuuteen vaikuttava yksittäinen tekijä, sillä yhtenäinen tyyli auttaa lukijan aivoja huomaamaan olennaisia kuvioita koodissa. Ohjelman koodin tulisi olla kirjoitusasultaan vähintään sisäisesti yhtenäinen, eli DrawUX:n sisällä käytettäisiin yhtä valittua tyyliä. Vielä parempi olisi jos käytössä olisi jokin yleinen tyyliopas, jolloin koodista tulisi myös ulkoisesti yhtenäistä ja se olisi helposti tulkittavissa myös jatkokehittäjille, jotka saattavat olla tutustuneet samaan vakiintuneeseen tyylikäytäntöön toisessa yhteydessä. Myös Hegedűs et al. [51] esittävät, että koodin tyyliin liittyvät seikat, kuten sisennykset, kommentointi ja nimeämiskäytännöt, auttavat selittämään subjektiivista tunneta koodin muokattavuudesta. Näihin asioihin tulisi siis kiinnittää enemmän huomiota tässäkin projektissa.

Lisäksi yksittäiset funktiot ovat usein pitkiä, mikä usein lisää koodaussääntörikkomuksia [51]. Edellä käsiteltiin jo koodin jakoa useampiin tiedostoihin, mutta sen lisäksi suurimpia funktioita kannattaisi mahdollisesti pilkkoa pienempiin osiin. Jos suurikokoinen funktio korvattaisiin kompaktilla funktiolla joka kutsuu selkeästi nimettyjä apufunktioita, jotka voitaisiin mahdollisesti jopa erottaa omaksi kirjastokseen, koodin toiminnasta saisi yhdellä silmäyksellä paremman käsityksen. Toisaalta jos funktioiden kutsuhierarkia menee liian syväksi, tämä puolestaan hankaloittaa ohjelman suorituksen kulun hahmottamista, joten pitkien funktioiden pilkkomisessa ei kannata mennä liiallisuuksiin.

Sekavuutta ja pituutta tiedostoille lisää myös melko suuri määrä poiskommentoitua koodia, joka on jostain syystä jätetty mukaan, yleensä siksi että kokeiltuun ratkaisuun olisi helppo palata jos vaihtoehtoinen ratkaisu ei toimi. Kuitenkin lopullisen päätöksen jälkeen nämä ylimääräiset koodit olisi hyvä poistaa kokonaan tiedostoista selkeyden vuoksi. Positiivinen asia koodin siistiydessä on, että koodi mahtuu lähes aina vaakasuunnassa näytölle, eli käytössä on sopivan pituiset rivit. Tästä poikkeuksena on muutama näkymätiedosto, joissa on toisinaan kirjoitettu yhteen kaikki visuaalisesti samalle tasolle tuleva merkkaus. Tyhjiä rivejä on käytetty sopivasti erottelemaan funktiot ja tuomaan rakennetta koodille.

Kun järjestelmä toisen kehitysjakson lopussa kopioitiin asiakkaan palvelimelle, huomattiin, että joitakin kohtia varsinaisista kooditiedostoista tuli muuttaa käsin niihin kovakoodattujen asetusten takia. Näitä kohtia oli vain muutamia, kuten sähköpostiosoite josta vastaajille lähetetään kutsuja ja muistutuksia, sekä osoite johon palautelomakkeen kautta otetaan yhteyttä. Kuitenkin se, että kovakoodausta on käytössä edes vähän, tarkoittaa että kaikki tiedostot pitää tarkistaa huolella ennen kuin järjestelmää voidaan siirtää. Tähän kuluu turhaa aikaa. Huomattavasti parempi käytäntö olisi siirtää kaikki asetukset konfigurointitiedostoon, jollaisten käyttöä CodeIgniter tukee hyvin.

Valitettavasti erääseen Highcharts-lähdekooditiedostoon jouduttiin koodaamaan omia muutoksia, jotta Highcharts saatiin tukemaan haluttua toiminnallisuutta siirreltäviin

tietoruutuihin liittyen. Tämä tarkoittaa, ettei Highchartsia ainakaan kaikissa näkymissä pystytä päivittämään uudempiin versioihin ilman, että manuaaliset muutokset lisättäisiin myös uuteen lähdekoodiin. Koska projekti oli heikosti dokumentoitu, muutosten kopiointi oikein voi osoittautua hankalaksi. Lähdetiedostojen muokkaus on tämän vuoksi ylläpidettävyyttä heikentävä asia, toisaalta tässä siltä ei voitu välttyä jos haluttiin saada tietty toiminto osaksi ohjelmaa. Kuitenkin jatkossa vastaavat muutokset pitäisi ehdottomasti dokumentoida paremmin.

5.4.4 Moduulien roolijako

Ylläpidettävyyttä auttaa suuresti se, että järjestelmällä on järkevä ylemmän tason arkkitehtuuri. Pyörää ei ole keksitty uudelleen vaan järjestelmässä on hyödynnetty tuttua, hyväksi todettua MVC-arkkitehtuuria, jonka moduulien rooleissa on lähes kaikissa tapauksissa pysytty tiukasti. Esimerkiksi tietokantaa ei käsitellä muualta kuin mallitiedostoista, ja kukin ohjaimista ja malleista kokoaa yhteen järkevän loogisen kokonaisuuden, esimerkiksi kyselyjen vastauksiin liittyvät funktiot. Tämän vuoksi on nopeaa ja helppoa löytää erityisesti palvelinpuolen koodista haluttu funktio. Poikkeuksena selkeissä rooleissa pysymisestä on datankäsittely, jota käsitellään myöhemmin tässä luvussa tarkemmin.

Tiedonvälityksen ja näkymien lataamisen tulisi olla mahdollisimman suoraviivaista ja loogista, jotta ylläpitäjän on helppo ymmärtää millä tavalla suoritus osien välillä etenee. Luvussa 4.2 esitelty palvelin-selain-kommunikointi täyttää tämän kriteerin suurelta osin. Kuitenkin joissakin näkymissä voitaisiin monimutkaisempi tapahtumaketju korvata yksinkertaisemmalla. Luvussa 5.3.3 kuvattiin jo, millä tavalla Ajaxin käyttö editorinäkymässä tallennukseen poistaisi ylimääräisen HTTP-kutsun, mutta lisäksi tämä korjaus helpottaisi editorinäkymän hahmottamista. Näkymässä tallennuspainikkeisiin liittyy tapahtumankäsittelijöitä, jotka keskeyttävät tai estävät napin luonnollisen toiminnan kirjoittaakseen tässä välissä piilotettuun muuttujaan tallennukseen liittyvät muutokset. Tällainen piilotettu toiminnallisuus tuo mukaan epäintuitiivista, hankalasti koodista löydettävää toiminnallisuutta, jota ylläpitäjä ei välttämättä tule ajatelleeksi. Toisaalta myös Ajax tuo mukanaan omat monimutkaiset puolensa, kuten takaisinkutsufunktiot, mutta kyseessä on kuitenkin laajassa käytössä oleva selkeä ohjelmointimalli.

Suurempi tiedonvälitykseen liittyvä ylläpidettävyysongelma löytyy kuitenkin vastaajan näkymästä, jossa on käytetty erikoista tapaa tiedonsiirtoon palvelimelta selaimelle. Tiedot kyselyyn kuuluvista käyränpiirtotehtävistä enkoodataan JSON-muotoon näkymässä, ja koko datamassa kirjoitetaan piilotetun muuttujan arvoksi ohjelmassa 5 kuvatulla tavalla.

```
<input id="json_post" type="hidden" name="form" value='<?php echo
str_replace('\\\\', '\\',json_encode($drawing_tasks)); ?>'>
```

Ohjelma 5. *Näkymässä lisätään HTML:n sekaan piilotettu muuttuja, joka sisältää piirtotehtävien yksityiskohdat.*

Tämän jälkeen JavaScript-koodi lukee muuttujan arvon, parsii sen JavaScript-taulukoksi ja alustaa kunkin piirtotehtävän saatujen tietojen perusteella ohjelman 6 mukaisesti. Toisin kuin tavalliset kyselylomakesivut, joiden alustaminen voidaan tehdä suoraan näkymässä, piirtotehtävät tulee alustaa JavaScriptin avulla, sillä piirtotehtävissä on käytetty Highcharts-komponenttia, jonka ulkoasun ja toiminnallisuuden alustus vaatii asetusten säätöä koodista käsin. Palvelimelta saatujen piirtotehtävätietojen käsittely JavaScriptissa on siis perusteltua.

```
var json_post = $("#json_post").val();
form_json = $.parseJSON(json_post);
for (var i = 0; i < form_json.length; i++) {
      //käsittelyä, esim:
      x_start = form_json[i].properties.x_start_title;
}
```

Ohjelma 6. *JavaScript tulkitsee piilotetun muuttujan tiedot ja käsittelee ne piirtotehtävä kerrallaan.*

Kuitenkin piilotetun muuttujan rooli selain-palvelin-kommunikoinnissa on kyseenalainen. Selain pääsisi suoremmin käsiksi palvelimelta saatuihin tietoihin, jos piilotettuun muuttujaan kirjoittaminen jätettäisiin kokonaan välistä ja käytettäisiin sen sijaan Ajax-kommunikointia. Näin prosessi suoraviivaistuisi ja merkistökoodaus, joka on aiheuttanut hankaluuksia järjestelmän tässä osassa, helpottuisi huomattavasti. JSONin parsiminen JavaScriptin *parseJSON()*-komennolla ei useinkaan kohtaa virheitä. Sen sijaan piilotettuun muuttujaan kirjoittamisessa täytyy lisäksi HTML-enkoodata koko data, jotta HTML-erikoismerkit, kuten lainausmerkki (”) ja suuruusvertailumerkit (<, >) eivät rikkoisi piilotettua muuttujaa. Piilotettua muuttujaa käytettäessä JavaScript joutuu purkamaan useita eri merkistökoodauksia ennen kuin varsinaiseen sisältöön päästään käsiksi, ja tämä hidastaa käsittelyä, tekee siitä virhealttiimpaa ja monimutkaisempaa ja siten myös vaikeuttaa koodin osan ymmärrettävyyttä ja virheiden etsintää.

Piilotetun muuttujan käyttöön on syynä luultavasti se, että on haluttu tuoda kaikki data kerralla palvelimelta näkymään. Jos ratkaisu korvattaisiin Ajaxilla, jouduttaisiin hakemaan ensin muiden tehtävien tiedot, rakentamaan niiden pohjalta alustava näkymä, ja tämän jälkeen jouduttaisiin tekemään uusi tietokantakutsu Ajaxin välityksellä ja rakentamaan näkymä loppuun näin saatavien piirtotehtävätietojen perusteella. Kuitenkin tämä korjaus auttaisi ylläpidettävyyttä ja vähentäisi virheitä, joten kokonaislaadun kannalta pieni kompromissi suoritusajassa olisi perusteltu, ja koska ylimääräisiltä enkoodaus- ja dekoodausaskelilta vältyttäisiin, suoritusaikaa voisi jopa säästyä.

Datankäsittely ylipäätään on aiheuttanut järjestelmässä ongelmia. Ensimmäinen ongelmista on yhtenäisten käytäntöjen puute mm. validoinnissa. Carnellin ja Harropin [32] mukaan epäyhdenmukaiset datanvalidointimenetelmät hankaloittavat ylläpitoa ja tekevät ohjelmoijille vaikeaksi tukea toistensa koodia. Kun ohjelmassa ei ole käytössä yhdenmukaista virheidenkäsittelyä vaan validointi on sekalaisesti jaettuna selaimen ja palvelimen välille, kyseessä on *Validation confusion* -antipattern, joka tekee koodin ylläpidon hankalaksi, sillä ohjelmoijan on vaikea tietää ja muistaa missä jokin tietty validointisääntö on. Validoinnin tulisi olla erillään liiketoimintalogiikasta, koska se on altis muutoksille ja riippuu voimakkaasti näkymästä.

DrawUX:ssa on useita lomakkeita joissa kerätään käyttäjän syöttämää tietoa, joka pitää validoida. Validointi kohdistuu esimerkiksi syötteen pituuteen ja sisältöön. Tietokannassa kentillä on maksimipituudet, joten tietokantavirheitä halutaan välttää ottamalla ylipitkät syötteet kiinni ennen tietokantaan kirjoitusta. Validoinnin lisäksi suoritetaan paljon muutakin syötteenkäsittelyä. Osa, mutta ei kuitenkaan kaikki, syötteestä käy läpi syötteenpuhdistuksen, jossa karsitaan kaikki HTML-merkkaus joitakin vaarattomina pidettyjä elementtejä lukuun ottamatta. Se, mikä syöte puhdistetaan ja mitä ei, ei aina ole loogista. Tutkijan syötteitä puhdistetaan enemmän kuin vastaajan, vaikka voitaisiin olettaa että työkalulla työtään tekevään tutkijaan voitaisiin luottaa enemmän kuin satunnaiseen vastaajaan. Syötettä myös enkoodataan hieman eri tavalla eri näkymien yhteydessä, mikä on aiheuttanut vaikeasti ratkaistavia virheitä siinä vaiheessa, kun eri näkymistä tulevia tietoja tulisi yhdistellä esimerkiksi Data Viewerissä ja dekoodata kerralla.

Erilaista validointia ja syötteenkäsittelyä suoritetaan siis paljon, mutta silti järjestelmässä ei ole mitään yhtenäistä tapaa jolla nämä toiminnot tehtäisiin. Rekisteröintilomake käyttää CodeIgniterin omia lomakkeenvalidointimetodeja callback-patternin kanssa, kun taas editorissa osa validoinneista ja käsittelyistä tehdään selaimessa ja osa palvelimella. Vastaajan näkymässä syötteenpituustarkastelut ja pakollisten kenttien tarkastukset tehdään selaimessa, kaikki muu käsittely palvelimella. Osittain epäyhdenmukaiset validointitavat johtuvat epäyhdenmukaisista kommunikointitavoista selaimen ja palvelimen välillä, mikä on perusteltavissa eri näkymiin liittyvillä erilaisilla vaatimuksilla; rekisteröintilomakesivulla ei ole muuta kuin kyseinen lomake joten ei jouduta lataamaan suurta datamäärää uudestaan vaikka käytetäänkin lomakkeenlähetystä. Carnellin ja Harropin [32] mukaan Validation confusion -antipatternin eräs oire on epäyhdenmukaiset virheilmoitukset, joita DrawUX:ssa esiintyy, kuten luvussa 5.2.3 tuli ilmi.

Ratkaisuna DrawUX:n pitäisi määritellä yhdenmukainen joukko palveluja, joilla syötteet validoidaan ja käsitellään. Näitä palveluja tulisi kutsua ennen liiketoimintalogiikkaa. Virheet saadaan heti kiinni ja käyttäjälle tiedotetaan yhdenmukaisella tavalla. Näin syötteen validointi ja muu sopivaan muotoon valmistelu muodostaa oman kerroksensa arkkitehtuuriin. Haasteena on rakentaa kerros, joka toimisi hyvin yhteen sekä Ajax- että lomakekutsujen kanssa.

5.5 Arvioinnin yhteenveto

Edellisissä luvuissa 5.2–5.4 tarkasteltiin DrawUX:n sopivuutta käyttöön, tehokkuutta ja ylläpidettävyyttä yksityiskohtaisesti. Taulukko 5 kokoaa yhteen arvioinnin olennaisimmat löydökset. Jokaisen päätöksen tai ominaisuuden kohdalla on arvioitu niiden vaikutuksen suunta ja suuruus kuhunkin käsiteltyyn laatuominaisuuteen.

Taulukko 5. *Yhteenveto arvioinnin tuloksista.*

Löydös tai ratkaisu	**Sopivuus käyttöön**	**Tehokkuus**	**Ylläpidettävyys**
Käyttäjän ohjeistuksen puute	--		+
Puuttuvat ominaisuudet	---	+	++
Käyttöliittymät	--		-
Testauksen puute	---		
Hidas latautuminen (Data Viewer)	(--)	---	
Tietokannanhallintajärjestelmän vähäinen hyödyntäminen		--	-
Päivitysnappi (Data Viewer)	-	+++	
Muutostaulukko		-	-
jQuery		+	+++
JSON		+	+
Highcharts	+++		
CodeIgniter	+++	(+)	+++
AJAX	++	+++	
Editorinäkymän tallennussekvenssi		--	-
Dokumentoinnin puute			---
Suuret tiedostot		--	---
JavaScript-tiedostojen sekava rakenne			---
Piilotettu muuttuja		(-)	--
Epäyhtenäinen validointi ja datankäsittely	-	-	---

Käyttäjälle tarkoitetun ohjesivun tai muun selkeän ohjeistuksen puute voi olla suureksi haitaksi käyttöön sopivuudelle, jos käyttäjä kohtaa tilanteen jossa tarvitsisi ohjeita. Ohjesivua tulisi kuitenkin päivittää jatkuvasti kun ohjelmaa kehitetään, joten se lisäisi ylläpidollista taakka hieman.

Joidenkin suunniteltujen ominaisuuksien puuttuminen luonnollisesti huonontaa ohjelmiston sopivuutta aiottuun käyttöön. Kuitenkin ylimääräisten ominaisuuksien puuttuminen on hyväksi tehokkuudelle (kevyempi ohjelmisto) ja ylläpidettävyydelle (vähemmän ylläpidettävää koodia).

Käyttöliittymissä on hiomista, sillä palautteen mukaan niihin liittyy käytettävyysongelmia ja esteettisiä ongelmia, erityisesti vastaajan näkymään sekä joihinkin Data Viewerin yksityiskohtiin. Tämä huonontaa käyttöön sopivuutta. Käyttöliittymien nykyinen toteu-

tus on myös ylläpidettävyyden kannalta epäideaali, sillä epäyhtenäisiin, lähes kokonaan itse ilman valmismalleja rakennettuihin näkymiin liittyy runsaasti erilaisia CSS-tiedostoja, joiden muokkaus voi olla hankalaa kun näkymien ulkoasua halutaan muuttaa. Tämä ylläpidettävyyshaitta on kuitenkin merkitykseltään vähäinen.

Myös prosessista itsestään löytyi puutteita. Testauksen puute huonontaa kokonaislaatua ja ohjelmaan jäävä virheiden riski vaikuttaa negatiivisesti käyttöön sopivuuteen. Toteutuksen dokumentoinnin puute puolestaan haittaa ohjelman ylläpitoa ja jatkokehitystä.

Testeissä Data Viewerin latautuminen todettiin turhan hitaaksi jo muutamilla kymmenillä vastaajilla. Tämä johtuu enimmäkseen siitä, että suuri määrä käyriä alustetaan kuvaajaan kerralla ja käyristä lasketaan keskiarvokäyrä. Tämä selvä tehokkuushaitta madaltaa luonnollisesti myös sopivuutta käyttöön, sillä latautumisen odottelu on turhauttavaa.

Tietokannanhallintajärjestelmän tarjoamien apujen, kuten tallennettujen proseduurien ja herättimien sekä indeksien, vähäinen hyödyntäminen on tekninen päätös, joka ei vaikuta millään tavalla ohjelman käyttöön sopivuuteen. Sen sijaan ratkaisu saattaa johtaa epäideaaliin tehokkuuteen ja vähäisessä määrin myös ylläpidettävyyteen, jos ohjelman koodiin lisätään ylimääräisiä toimenpiteitä, jotka voitaisiin antaa tietokannanhallintajärjestelmän vastuulle.

Yksittäisistä matalan tason koodiratkaisuista lähinnä positiivisia vaikutuksia on Data Viewerin päivitysnapilla, joka tekee käytöstä kenties hieman kömpelömpää kuin automaattinen päivitys olisi, mutta säästää järjestelmää liian suurelta kuormitukselta. Sen sijaan muutostaulukon keräys editorissa vaikuttaa negatiivisesti tehokkuuteen ja ylläpidettävyyteen, joskin erot vaihtoehtoisiin ratkaisuihin nähden ovat pieniä.

Teknologiavalinnat vaikuttavat enimmäkseen hyviltä: jQueryn helppo syntaksi, JSONin keveys ja luettavuus sekä CodeIgniterin tarjoama toiminnallisuus ja looginen rakenne vaikuttavat laatuominaisuuksiin positiivisesti. Highcharts tukee useimpia halutuista kaaviotoiminnallisuuksista ja antaa mahdollisuuden oman koodin liittämiseen silloin kun valmista Highcharts-ratkaisua ei ole. Ajax-mallin käyttö kommunikointiin tekee käytöstä mukavampaa ja nopeampaa, mutta sitä ei ole käytetty kaikkialla missä olisi kannattanut; erityisesti editorinäkymän monimutkainen tallennussekvenssi hidastaa käyttöä ja hankaloittaa vähäisessä määrin myös ylläpidettävyyttä.

Tiedostojen (erityisesti JavaScript) suuri koko aiheuttaa suuria hankaluuksia ylläpidettävyydelle sekä hidastaa järjestelmää välittömästi (vaikutus sivun latausaikaan) ja välillisesti (suuri tiedosto sisältää paljon suoritettavaa koodia). Myös tiedostojen epäyhtenäinen ja hankalasti hahmotettava rakenne hankaloittaa ylläpitoa huomattavasti.

Datankäsittelyyn ja tiedonvälitykseen liittyen moduulien roolijako ei ole aina selvä. Yksittäisistä ratkaisuista erityisesti vastausnäkymän piilotetun muuttujan käyttö selain-

palvelin-kommunikointiin vaikuttaa kyseenalaiselta, sillä se sekavoittaa koodia ja näin hankaloittaa ylläpidettävyyttä. Lisäksi muuttujalla on marginaalisia negatiivisia vaikutuksia tehokkuuteen (vaihtoehtoinen ratkaisu vaatisi toisen palvelinkutsun alkuperäisen latauksen lisäksi, mutta jos alkuperäiseen lataukseen liittyvää tietokantakyselyä voidaan optimoida siten, ettei samaa dataa jouduta hakemaan kahdesti, säästetään ainakin piilotetun muuttujan tulkitsemiseen ja erilaisiin en- ja dekoodauksiin kuluva suoritusaika).

Epäyhtenäinen datankäsittely aiheuttaa negatiivisia vaikutuksia kaikkiin kolmeen ominaisuuteen. Suurin vaikutus on ylläpidettävyyteen, sillä ylläpitäjän on vaikea jäljittää ja hahmottaa minkä käsittelyaskelten läpi data kulkee missäkin osassa ohjelmaa. Jonkin verran haittaa on myös tehokkuudelle, jos data käy samat tarkastukset tarpeettomasti läpi kahdessa eri kohdassa tai turhan monta käsittelyaskelta kun vähempikin riittäisi. Myös sopivuus käyttöön kärsii joidenkin askelten sivuvaikutusten vuoksi, esimerkiksi merkistökoodausongelmat saattavat toisinaan välittyä käyttöliittymään asti näkyvinä virheinä.

6. TULOKSET

Edellisessä luvussa suoritettu arviointi tuotti hyvin yksityiskohtaista tietoa DrawUX-järjestelmän laadusta, mutta näistä yksityiskohdista on todellista hyötyä vasta, kun niiden pohjalta voidaan suorittaa konkreettisia parannustoimia. Tämän vuoksi arvioinnissa esiin tulleet ongelmat tulisi priorisoida ja muuntaa toimenpidesuosituksiksi.

Luvussa 6.1 arvioidaan, mitkä löytyneistä ongelmista kannattaisi korjata ja missä järjestyksessä. Tuloksena esitetään priorisoitu lista toimenpidesuosituksia jatkokehitystä varten. Luku 6.2 käsittelee vielä itse arvioinnin hyödyllisyttä ja onnistumista.

6.1 Suositukset jatkokehitystä varten

On selvää, että kaikkea ei jo olemassa olevasta järjestelmästä kannata alkaa muuttamaan. Korjausehdotuksen tulee parantaa järjestelmän laatua tarpeeksi suhteessa korjaukseen vaadittavaan työmäärään. Useat järjestelmää mahdollisesti hieman parantavat korjaukset, kuten alla olevan ohjelmointikielen vaihto, vaatisivat niin runsaasti työtä, ettei niiden toteuttaminen olisi järkevää. Käytännössä on siis rajoituttava tarkastelemaan pienempiä yksittäisiä korjauksia, jotka voivat parantaa laatua suhteellisen paljon verrattuna työmäärään.

Arvioinnissa löytyi runsaasti sekä hyviä että huonoja ratkaisuja. Erityisen hyviä ratkaisuja ovat teknologiavalinnat ja päivitysnapin käyttö. Negatiivisia puolia löytyi runsaasti sekä prosessista itsestään että yksittäisistä toteutusratkaisuista. Seuraavassa on yritetty arvioida korjausten realistisuutta ja priorisoida realistisista korjauksista tärkeimmät.

Taulukosta 5 nähdään, että epäyhtenäinen validointi ja datankäsittely erottuu erityisen negatiivisena, sillä se vaikuttaa kaikkiin arvioituihin laatuominaisuuksiin negatiivisesti ja sen vaikutukset ylläpidettävyyteen ovat erityisen pahat. Myös piilotettuun muuttujaan kirjoittaminen selain-palvelin-tiedonvälityksen mekanismina vaikuttaisi kaikin puolin negatiiviselta ratkaisulta verrattuna vaihtoehtoihin. Näihin toisiinsa jossain määrin sidoksissa oleviin ongelmiin vaadittaisiin kohtalaisesti työaikaa, mutta korjaus kannattaisi, sillä se tekisi jatkokehityksestä paljon helpompaa ja suorituksesta vähemmän virhealtista. Tämän korjauksen tulisikin mielestäni olla prioriteetiltaan korkealla.

Toinen olennainen korjattava asia on Data Viewerin hidas latautuminen. Tässä suositeltavaa olisi etsiä keinoja jakaa kuormitusta muualle kuin alun lataukseen, eli esimerkiksi jättää osa käyristä kokonaan alustamatta kunnes käyttäjä haluaa näyttää käyrät ensimmäisen kerran. Tämä korjaus vaikuttaisi negatiivisesti ylläpidettävyyteen monimutkais-

tamalla koodia, mutta tehokkuudessa saatava etu olisi kriittinen, joten korjaus kannattaisi. Korjausta ei myöskään olisi kohtuuttoman vaikea toteuttaa. Myös keskiarvokäyrän laskennan osuus alustamisen hitauteen vaatii lisää tutkimista. Data Viewer on olennainen osa ohjelmaa joten se olisi syytä saada toimimaan kohtuullisella nopeudella.

Haluttujen lisäominaisuuksien puuttuminen on luonnollisesti suuri puute ja jatkokehityksen ensisijainen tarve. Testauksen ja dokumentoinnin puutteet ovat suuria riskejä projekteissa, joten niiden korjaaminen olisi nähdäkseni tärkeää. Sen sijaan käyttäjälle suunnatun ohjesivun puuttumisen kanssa voidaan mahdollisesti pärjätä varmistamalla, että työkaluvihjeet ja vastaajalle suunnatut ohjelaatikot ovat selkeitä. Editorinäkymän epäideaali tallennussekvenssi toimii näinkin, joten ellei editorin tallennuksessa ala ilmetä merkittäviä viiveitä tai muita ongelmia, tämän hienosäädön voi toistaiseksi jättää tekemättä.

JavaScript-tiedostojen suuri koko ja epämääräinen rakenne ovat suuria ylläpidettävyysongelmia, jotka tulisi korjata. Käyttöliittymien käytettävyyttä tulisi myös hioa, jotta asiakkaan mainitsemasta kömpelyydestä päästäisiin eroon, mutta esimerkiksi CSS-pohjan vaihtaminen kokonaan vaikkapa selaintyylikehykseen olisi epäkäytännöllisen työlästä, joten pienet muutokset omaan CSS:n ja HTML:n riittävät. Tietokannanhallintajärjestelmän puutteellisen hyödyntämisen kanssa voidaan jatkaa nykyisellä toteutuksella, ellei käytössä tule vastaan merkittäviä tietokantaan ja palvelimeen liittyviä latausaikaongelmia. Muutostietojen keruu editorinäkymässä on jossain määrin epäideaali ratkaisu, mutta sen korjaaminen vaatisi koko näkymän selainkoodin vaihtamista toiseksi, joten muutosten työläys tekee korjauksen hyödyttömäksi suhteessa siitä saataviin pieniin etuihin.

Suosittelen siis, että jatkokehityksessä keskityttäisiin seuraaviin korjauksiin:

1. Kunnollinen järjestelmä- sekä yksikkötestaus tulisi ottaa osaksi mahdollista jatkokehitysprosessia ja nykyinen toteutus testata huolellisesti.
2. Data Viewerin latausaikaa tulisi pyrkiä pienentämään laiskalla alustuksella (alustamalla vain osa käyräobjekteista heti latauksen yhteydessä, loput vasta kun niitä tarvitaan näkymässä). Muitakin optimointikeinoja suositellaan tutkittaviksi.
3. Datan kulku ja käsittely järjestelmän läpi tulisi suunnitella uudelleen siten, että vältytään epäyhtenäiseltä, hankalalta, vaikeasti hahmotettavalta käsittelyltä validoinnissa ja syötteenpuhdistuksessa sekä enkoodauksessa ja dekoodauksessa.
4. Vastausnäkymässä piilotettuun muuttujaan kirjoittaminen tulisi korvata Ajax-kommunikoinnilla tai muulla paremmalla ratkaisulla.
5. Käyttöliittymien käytettävyyttä tulisi hioa asiakaspalautteen ja mahdollisuuksien mukaan myös käytettävyystestien pohjalta. Erityistä huomiota tulisi kiinnittää vastaajan näkymään, jotta voidaan varmistua käytön olevan niin helppoa, että työkalua voidaan ongelmitta käyttää etätutkimuksiin. Tutkijan näkymistä erityisesti tiedon analysointivaihe (Data Viewer) kaipaa säätöä.
6. Järjestelmään kannattaa lisätä puuttuvia ominaisuuksia asiakkaiden tarpeiden mukaan.

7. Ainakin suurimmat (yli 500 rivin) JavaScript-tiedostot tulisi pilkkoa suuremmaksi määräksi pienempiä tiedostoja. Jako tulisi tehdä loogisesti ja samalla koodin rakennetta tulisi muutenkin selkeyttää ja yhtenäistää (esim. järkevä funktiojärjestys jota noudatettaisiin kaikissa tiedostoissa).
8. Järjestelmän toteutus tulisi dokumentoida paremmin.
9. Jos jatkokehitysprojektin resurssit riittävät, myös muita arvioinnissa esille tuotuja negatiivisia ratkaisuja kannattaa pyrkiä korjaamaan

Näiden suositusten noudattaminen parantaisi järjestelmän laatua useiden laatuominaisuuksien suhteen ja on mahdollista toteuttaa kohtuullisella budjetilla. Olennaista olisi varata näille toimenpiteille erikseen aikaa jatkokehityksen aikataulusta sen sijaan, että varataan aikaa ainoastaan uusien toimintojen kehittämiseen. Yllä esitettyjen toimenpiteiden noudattaminen maksanee itsensä takaisin kun ylläpidettävyys, tehokkuus ja yleinen käyttöön soveltuvuus paranevat.

6.2 Omien tulosten arviointi

Luvussa 5 suoritettu laatukriteeripuuta hyödyntävä asiantuntija-arviointi pohjautui kirjallisuudesta löytyvään teoriatietoon sekä omiin kokemuksiin ongelmista ja asioista jotka tulisivat jatkokehityksessä vastaan seuraavallekin ohjelmoijalle. Lisäksi arvioinnissa ja laatukriteeripuun rakentamisessa hyödynnettiin asiakkaalta ja projektipäälliköltä saatua palautetta. Kuitenkin järjestelmän systemaattisempi arviointi olisi tarpeen, metriikoilla ja käyttäjätesteillä saataisiin erilaista näkökulmaa eri laatuominaisuuksien arviointiin.

Tässä suoritettu arviointi pohjautui löyhästi WebQEM-arvioinnin [8] ideaan. Noudattamalla WebQEMiä tarkemmin saataisiin eksaktimpi kuva järjestelmän laadusta. Tällöin tässä käytettyyn kriteeripuuhun tulisi lisätä jokaiselle lehtikriteerille metriikka sekä päättää jokaiselle ylemmän tason attribuutille funktio, jonka avulla alikriteerien tulokset kootaan yhteen. Käytännössä alikriteereille tulisi siis valita painoarvot ja tämän valinnan tulisi mielellään tehdä asiakas tai projektipäällikkö. Tällainen kvantitatiivinen versio kriteeripuun käytöstä olisi hyödyllinen, jos järjestelmää halutaan verrata tarkasti muihin vastaaviin järjestelmiin. Numeerisia laatuarvioita olisi helpompi verrata toisiinsa kuin kvalitatiivisia tuloksia. Yksittäisen palvelun kohdalla sen sijaan kvantitatiivisesta arvioinnista saatava kriteerien täyttymisaste ei välttämättä olisi erityisen mielekäs tulos ilman luonnollista skaalaa tai vertailukohtaa.

Osaan arvioiduista ominaisuuksista tässä käytetty asiantuntijalähtöinen lähestymistapa sopii paremmin kuin toisiin. Ylläpidettävyys on pitkälti ohjelmoijan subjektiivinen kokemus siitä miten vaivatonta ohjelmistoa on ylläpitää, joten tämäntyyppinen arviointi voi tavoittaa ylläpitokokemukseen vaikuttavat olennaiset asiat. Kuitenkin ylläpidettä-

vyyteen liittyy myös runsaasti hyödyllisiä metriikoita, joilla arviointia olisi voinut täydentää.

Suorituskyvystä saataisiin tarkempaa tietoa erilaisten suorituskykymetriikoiden kautta, jolloin sivustoa voitaisiin mahdollisesti vertailla myös muihin samantyyppisiin järjestelmiin. Kuitenkin kirjallisuudesta löytyvät perustellut parhaat käytännöt ja yleiset virheet sekä kilpailevien teknologien vertailut antavat teoreettista suuntaviittaa siitä miten suorituskykyyn voidaan vaikuttaa, joten niiden kautta tapahtunut arviointi antaa myös ainakin jonkinlaisen kuvan suorituskykyyn vaikuttavista asioista. Arvioinnissa löytynyt yksittäisen näkymän suoritusaikaongelma vaatii tarkempien työkalujen käyttöä, jotta voidaan saada selville, mitkä kohdat koodia erityisesti hidastavat näkymän alustusprosessia. Tässä arvioinnissa ei vielä päästy suoritusaikaongelman ytimeen, vaikka valistuneita arvauksia ongelman syystä voitiinkin esittää.

Sopivuus käyttöön on tarkoituksella jätetty määritelmältään hieman epämääräiseksi. Se, miten hyvin järjestelmä vastaa käyttäjien tarpeisiin, vaatisi ehdottomasti käyttäjien haastatteluja. Tällaiseen ei tämän projektin puitteissa ollut mahdollisuutta, mutta jonkin verran käyttöön sopivuutta pystyttiin arvioimaan teoreettisesti, rajallisen asiakaspalautteen kautta ja omaan kokemukseen pohjautuen. Projektipäällikön sekä asiakkaan asettamien toiminnallisten tavoitteiden vertailu todellisuudessa toteutettuihin toimintoihin antoi myös tarkan kuvan siitä, miten kattavasti järjestelmä vastaa alkuperäisiin tarpeisiin.

Ylipäätään siis voidaan sanoa, että valittu lähestymistapa sopi riittävän hyvin tämän kokoiseen työhön, mutta kuitenkin hieman suuremmilla resursseilla suoritettavat jatkotutkimukset metriikoiden ja käyttäjätestien avulla olisivat suositeltavia. Esimerkiksi [51] listaa hyviä metriikoita ylläpidettävyyden arvioimiseen lähdekoodista ja [50] kuvaa tapoja mitata järjestelmän arkkitehtuurin ylläpidettävyyttä UML-mallinnuksen avulla.

Sekä arviointi että kriteeripuun laadinta suoritettiin puolueettomasti ja monipuoliseen materiaaliin pohjalta. Kriteeristöön ei pyritty valitsemaan tarkoituksella asioita, joissa järjestelmä pärjää hyvin. Pikemminkin kriteeristöstä tuli lähes liiankin vaativa johtuen siitä, että käytetyssä materiaalissa (omat muistiinpanot mm. korjattavista asioista, kesken projektia saatu asiakaspalaute yms.) usein korostuvat ongelmakohdat. Onnistuneet toteutukset sen sijaan jäävät usein ilman erityisiä mainintoja, sillä ne mielletään itsestään selviksi. Jonkin ominaisuuden toimivuuteen, käytettävyyteen, hyödyllisyyteen ja muuhun laatuun kiinnitetään luonnollisesti enemmän huomiota, kun siinä on ongelmia, kuin jos kaikki sujuu odotusten mukaisesti.

Arviointi rajattiin tarkoituksella vain kolmeen laatuominaisuuteen, joten siinä mielessä tämä arviointi on vain pintaraapaisu kokonaislaatuun. Muitakin laatuominaisuuksia kannattaisi tutkia. Web-järjestelmien luotettavuus ja turvallisuus on äärimmäisen olennaista. Myös saavutettavuudessa eri selaimilla ja päätelaitteilla olisi mielenkiintoista

tutkittavaa, sillä käyttäjän sitominen tiettyyn selaimeen (Google Chrome) ei välttämättä ole järkevin ratkaisu, jos käytön halutaan olevan mahdollisimman vaivatonta. Lisää tietoa erilaisista mahdollisesti tutkimisen arvoisista laatuominaisuuksista löytyy esimerkiksi julkaisuista [44], [48], [6].

Arvioinnin tavoitteena oli poimia järjestelmän toteutuksessa käytettyjä yksittäisiä ratkaisuja, joista on erityisen paljon hyötyä tai haittaa kullekin laatuominaisuudelle. Tarkoituksena oli oppia hyviä ja huonoja käytäntöjä web-järjestelmien toteutuksesta sekä nostaa esille tärkeiksi arvioituja seikkoja, joiden korjaamisesta olisi hyötyä järjestelmän laadulle. Näin arviointi voisi toimia muistilistana DrawUX-järjestelmän jatkokehittäjälle. Arvioinnin tulokset vastaavat näitä asetettuja tavoitteita ja antavat nähdäkseni hyvän yleiskuvan järjestelmän laadusta, joten arviointi voidaan katsoa onnistuneeksi.

7. YHTEENVETO

Tähän diplomityöprojektiin liittyi kaksi erillistä osa-aluetta, web-työkalu DrawUX:n jatkokehitys ja sen arviointi. DrawUX on tarkoitettu retrospektiivisen pitkän ajan käyttäjäkokemustiedon keräämiseen. Käytännössä retrospektiivinen data kerätään käyränpiirtotehtävällä, jossa kuvaajan x-akseli kuvaa aikaa ja y-akseli jotakin käytön laatuun liittyvää ominaisuutta, kuten tuotteen miellyttävyyttä tai helppokäyttöisyyttä. Työkalu on verkkopohjainen perinteinen hypermediajärjestelmä, joka on toteutettu PHP-ohjelmointikielellä CodeIgniter-sovelluskehystä käyttäen. Lisäksi toteutukseen on käytetty runsaasti muitakin ohjelmointikieliä ja teknologioita, kuten HTML, JavaScript, jQuery, JSON sekä erilaisia sovellusaluekohtaisia lisäosia. Näistä tärkein oli kaaviokirjasto Highcharts, jonka avulla voidaan toteuttaa selainpohjaisia interaktiivisia kuvaajia.

Työkalusta oli olemassa jo tärkeimmät perustoiminnot sisältävä versio, joten jatkokehitykselle oli hyvä pohja. Diplomityöprojektin ensimmäisen osan tavoitteena oli lisätä työkaluun toiminnallisuutta, joka parantaisi sen hyödyllisyyttä. Kehitykseen ei kohdistunut selkeitä laadullisia tavoitteita tai vaatimuksia, mutta vastaajan näkymän käytettävyyttä pidettiin olennaisena. Kyseessä oli resursseiltaan pieni projekti, jossa oli vain yksi kehittäjä kerrallaan, joten aika ja työvoima olivat suurimmat rajoitteet. Näiden käytännön rajoitteiden vuoksi nopeaa kehitystä ja tätä kautta myös ylläpidettävyyttä pidettiin erityisen tärkeänä, sillä työkaluun oli suunniteltu paljon kehitettävää toiminnallisuutta. Työkaluun haluttiin muun muassa näkymä tulosten monipuoliseen tarkasteluun (Data Viewer), kääntönäkymä tuloksille, mahdollisuus mobiilivastaamiseen, mahdollisuus pitkittäistutkimuksille (päiväkirjatyylinen vastaaminen), sekä useita yksittäisiä lisätoimintoja, kuten käyräkuvaajan zoomaus ja kyselyeditoriin lukuisia uusia asetuksia. Käytännössä jatkokehitys tapahtui kahdessa ajallisesti erillisessä osassa, joista toisessa oli mukana asiakas, jolle siirrettiin projektin lopuksi jäädytetty versio senhetkisestä toteutuksesta.

Lähes kaikki halutut ominaisuudet saatiin projektin aikana toteutettua. Erityisesti kaikki tärkeimmät jatkokehityksen tavoitteet, kuten mobiilivastaaminen, päiväkirjatyylinen vastaaminen, Data Viewer ja kääntöominaisuus, saatiin valmiiksi. Jotkin ominaisuudet toteutettiin hieman eri tavalla kuin oli tarkoitus. Esimerkiksi kääntötoimintoon asiakkaalla olisi ollut erilainen toivomus, mutta projektin mitoituksen puitteissa se olisi ollut epärealistinen toteuttaa. Jatkokehitysprojekti voidaan kuitenkin katsoa onnistuneeksi ainakin toiminnallisuuden toteutuksen suhteen, sillä haluttuja toimintoja on saatu toteutettua yllättävänkin paljon verrattuna käytettyyn aikaan ja työvoimaan. Kuitenkin arviointiosuudessa tuli ilmi joitakin laatuongelmia, joista useimpiin on vaikuttanut jokaisen kehitykseen osallistuneen kehittäjän osuus, joten selvästi tässäkin jatkokehitysprojektis-

sa olisi ollut vielä parannettavaa ei-toiminnallisten ominaisuuksien suhteen. Laatuominaisuuksiin enemmän panostamalla olisi kuitenkin menetetty aikaa varsinaiselta kehittämiseltä, joten viime kädessä kyse oli projektipäällikön suorittamasta harkitusta priorisoinnista. Tässä projektissa haluttiin tietoisesti keskittyä enemmän toiminnallisuuteen, mutta tämä saattaa aiheuttaa ongelmia myöhemmin, jos esimerkiksi ylläpidettävyyshaitat hidastavat mahdollista jatkokehitystä huomattavasti.

Toinen osuus työtä oli nykyisen toteutuksen arviointi, joka oli tyypiltään laadullinen asiantuntija-arviointi. Arvioinnissa tarkasteluun otettiin kolme laatuominaisuutta, sopivuus käyttöön, tehokkuus ja ylläpidettävyys. Tavoitteena oli löytää järjestelmästä positiivisia ja negatiivisia ratkaisuja ja toteutusyksityiskohtia sekä priorisoida löydöksistä olennaisimmat listaksi realistisia parannusehdotuksia. Arvioinnin pohjana käytettiin laadullista sovellusta WebQEM-menetelmästä, jossa laatuvaatimusten täyttymisen järjestelmälliseen läpikäyntiin käytetään vaatimuksista rakennettua kriteeripuuta. Arvioinnissa järjestelmästä löytyi runsaasti sekä positiivisia että negatiivisia ominaisuuksia. Positiivisisia puolia olivat erityisesti järjestelmän perusarkkitehtuuri ja teknologiavalinnat. Järjestelmä oli alusta asti toteutettu hyödyntäen mahdollisimman paljon valmiita komponentteja, mikä osoittautui sekä ajankäytöllisesti että ylläpidettävyyden kannalta järkeväksi ratkaisuksi. Valmiskomponentit, kuten CodeIgniter-kehys, toivat järjestelmään loogisen, hyvänä pidettyjen ohjelmointimallien mukaisen rakenteen. Myös korjattavaa löytyi kaikista kolmesta arvioidusta ominaisuudesta. Vielä puuttuvat ominaisuudet ja testauksen puute madalsivat DrawUX:n sopivuutta käyttöön. Sivuista Data Viewer todettiin hitaaksi. Järjestelmän muuhun tehokkuuteen negatiivisesti vaikuttivat monet yksityiskohdat, kuten tietokannanhallintajärjestelmän vähäinen käyttö optimointiin sekä jotkin epäideaalisti toteutetut suoritussekvenssit. Ylläpidettävyyttä haittasivat merkittävästi suuret ja sekavat kooditiedostot, dokumentoinnin puute sekä epäyhtenäinen datankäsittely.

Löydösten pohjalta tehtiin suosituksia jatkokehitystä varten. Tärkeimmät suositukset olivat testauksen ottaminen osaksi prosessia, Data Viewerin latausajan pienentäminen jakamalla objektien alustuksen taakkaa muuallekin kuin alkulatauksen yhteyteen, syötteenkäsittelyn uudelleensuunnittelu omaksi selkeäksi kerroksekseen, erään ongelmia aiheuttaneen yksittäisen selain-palvelin-kommunikointiratkaisun korvaaminen toisella, sekä käytettävyyden hiominen asiakaspalautteen pohjalta.

Arvioinnin tuloksia voidaan soveltaa myös muihin web-järjestelmiin. Työssä kävi ilmi ainakin se, että projektissa, jossa kaikki työaika käytetään toiminnallisten vaatimusten toteuttamiseen dokumentoinnin ja testauksen kustannuksella, ohjelmiston laatu kärsii. Näin ollen pienestäkin projektista kannattaisi löytää edes jonkin verran aikaa järjestelmälliseen laadunvarmistukseen. Voidaan myös olettaa, etteivät muutkaan negatiiviset löydökset ole yksilöllisiä tälle järjestelmälle, vaan kaikkiin tässä löydettyihin asioihin, kuten syötteenkäsittelyyn ja koodin loogiseen rakenteeseen, kannattaa kiinnittää huomiota web-järjestelmiä ohjelmoitaessa ja suunniteltaessa. Positiivisina asioina voidaan

oppia mm. käytettyjen teknologioiden sopivuus tämän kaltaiseen järjestelmään. Esimerkiksi CodeIgniter soveltuu varsin hyvin MVC-järjestelmän pohjaksi, ja Highcharts on hyödyllinen apu sovelluksiin, joissa tarvitaan erilaisia interaktiivisia kuvaajia. Käytettyjen teknologioiden, erityisesti Highchartsin, laajentaminen omalla koodilla todettiin käteväksi.

Arviointi katsottiin onnistuneeksi, sillä sen avulla saatiin monipuolisesti melko yksityiskohtaista tietoa sovelluksen laadusta. Arviointi huomioi useita eri sidosryhmänäkökulmia (vastaajan, tutkijan ja kehittäjän näkökulma) ja laatuominaisuuksia (sopivuus käyttöön, tehokkuus, ylläpidettävyys) ja käytetty kriteeripuu oli muutenkin kattava, erityisesti toiminnallisten vaatimuksien suhteen. WebQEM ja siitä käytetty laadullinen muunnos soveltuivat arvioinnin pohjaksi hyvin, sillä kriteeripuu auttoi jäsentämään järjestelmää ja keskittymään arvioinnissa olennaiseen. Jatkotarpeita arvioinnille kuitenkin nähtiin. Arvioinnin laajentaminen numeeriseksi WebQEM-arvioinniksi metriikoiden avulla tuottaisi järjestelmästä lisää tietoa muodossa, jonka avulla DrawUX:ia olisi helpompi verrata numeerisesti muihin vastaaviin järjestelmiin. Kriteeripuun laajentaminen entisestään kattamaan lisää laatuominaisuuksia ja alikriteerejä lisäisi edelleen arvioinnilla saatavaa tietoa. Lisäksi nähtiin tarve kattavalle käyttäjätestaukselle, mitä ei tässä työssä voitu toteuttaa.

Vaikka käsitellyssä järjestelmässä on vielä parannettavaa, se on kuitenkin toteutukseltaan tarpeeksi toimiva ja käyttökelpoinen jotta sitä pystyy käyttämään aiottuun tarkoitukseen, retrospektiivisten pitkän aikavälin käyttäjäkokemustutkimusten järjestämiseen. Työkalulla voi kerätä tietoa tavoilla joita ei juurikaan ole tarjolla muissa kyselytyökaluissa, mutta joille on tarvetta HCI-alalla, joten selvästi kyseessä on perusajatukseltaan hyödyllinen ja innovatiivinen työkalu. Toteutuksen ongelmakohdat ovat web-järjestelmille yleisiä, mutta niistä monet olisivat kohtuullisella työmäärällä korjattavissa. Uskon että pienillä parannuksilla työkalusta voisi olla suurta hyötyä käyttäjäkokemustutkimuksen alalla.

LÄHTEET

[1] J. Varsaluoma, V. Kentta, DrawUX: Web-based Research Tool for Long-term User Experience Evaluation, Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design, ACM, New York, NY, USA, pp. 769-770.

[2] B. Edvardsson, I. Roos, Critical incident techniques: Towards a framework for analysing the criticality of critical incidents, Int J of Service Industry Mgmt, Vol. 12, No. 3, 2001, pp. 251-268.

[3] M. von Wilamowitz-Moellendorff, M. Hassenzahl, A. Platz, Dynamics of user experience: how the perceived quality of mobile phones changes over time, User Experience – Towards a Unified View, Workshop at the 4th Nordic Conference on Human-Computer Interaction, pp. 74-78.

[4] S. Kujala, V. Roto, K. Väänänen-Vainio-Mattila, E. Karapanos, A. Sinnelä, UX Curve: A method for evaluating long-term user experience, Interacting with Computers, Vol. 23, No. 5, 2011, pp. 473-483.

[5] E. Karapanos, J. Martens, M. Hassenzahl, Reconstructing experiences with iScale, International Journal of Human-Computer Studies, Vol. 70, No. 11, 2012, pp. 849-865.

[6] S.S. Cherfi, A. Do Tuan, I. Comyn-Wattiau, An Exploratory Study on Websites Quality Assessment, in: J. Parsons, D. Chiu (ed.), Advances in Conceptual Modeling: ER 2013 Workshops, LSAWM, MoBiD, RIGiM, SeCoGIS, WISM, DaSeM, SCME, and PhD Symposium, Hong Kong, China, November 11-13, 2013, Revised Selected Papers, Springer International Publishing, Cham, 2014, pp. 170-179.

[7] P. Bhatt, G. Shroff, A.K. Misra, Dynamics of Software Maintenance, SIGSOFT Softw.Eng.Notes, Vol. 29, No. 5, 2004, pp. 1-5.

[8] L. Olsina, G. Rossi, Measuring Web application quality with WebQEM, IEEE Multimedia, Vol. 9, No. 4, 2002, pp. 20-29.

[9] ISO 9241-210:2010, Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems, International Organization for Standardization, 2010, 32 p.

[10] D. Norman, J. Nielsen, The Definition of User Experience (UX), Nielsen Norman Group, verkkosivu. Saatavissa (viitattu 12.03.2017): https://www.nngroup.com/articles/definition-user-experience/.

[11] N. McNamara, J. Kirakowski, Functionality, Usability, and User Experience: Three Areas of Concern, interactions, Vol. 13, No. 6, 2006, pp. 26-28.

[12] H. van der Heijden, User Acceptance of Hedonic Information Systems, MIS Quarterly, Vol. 28, No. 4, 2004, pp. 695-704.

[13] D. Kahneman, A.B. Krueger, D.A. Schkade, N. Schwarz, A.A. Stone, A Survey Method for Characterizing Daily Life Experience: The Day Reconstruction Method, Science, Vol. 306, No. 5702, 2004, pp. 1776.

[14] J.C. Flanagan, The critical incident technique, Psychological bulletin, Vol. 51, No. 4, 1954, pp. 327-358.

[15] D.A. Norman, THE WAY I SEE IT: Memory is More Important Than Actuality, interactions, Vol. 16, No. 2, 2009, pp. 24-26.

[16] S. Oishi, H.W. Sullivan, The predictive value of daily vs. retrospective well-being judgments in relationship stability, Journal of experimental social psychology, Vol. 42, No. 4, 2006, pp. 460-470.

[17] G. Goldschmidt, The Dialectics of Sketching, Creativity Research Journal, Vol. 4, No. 2, 1991, pp. 123-143.

[18] J. Sonnemans, N.H. Frijda, The structure of subjective emotional intensity, Cognition and Emotion, Vol. 8, No. 4, 1994, pp. 329-350.

[19] T. Betsch, H. Plessner, C. Schwieren, R. Gütig, I Like it but I Don't Know Why: A Value-Account Approach to Implicit Attitude Formation, Personality and Social Psychology Bulletin, Vol. 27, No. 2, 2001, pp. 242-253.

[20] W.C. Schmidt, World-Wide Web survey research: Benefits, potential problems, and solutions, Behavior Research Methods, Instruments, {\&} Computers, Vol. 29, No. 2, 1997, pp. 274-279.

[21] B. Duffy, K. Smith, G. Terhanian, J. Bremer, Comparing data from online and face-to-face surveys, International Journal of Market Research, Vol. 47, No. 6, 2005, pp. 615.

[22] D. Kim, J. Zhong, M. Lee, D. Li, A.O. Tokuta, Efficient Respondents Selection for Biased Survey Using Online Social Networks, in: Z. Cai, A. Zelikovsky, A. Bourgeois (ed.), Computing and Combinatorics: 20th International Conference,

COCOON 2014, Atlanta, GA, USA, August 4-6, 2014. Proceedings, Springer International Publishing, Cham, 2014, pp. 608-615.

[23] P. Comley, Innovation in Online Research - Who Needs Online Panels? MRS Research Conference Paper 36, Research 2003 Conference Papers.

[24] J.A. Cunningham, C. Neighbors, N. Bertholet, C.S. Hendershot, Use of mobile devices to answer online surveys: implications for research, BMC Research Notes, Vol. 6, No. 1, 2013, pp. 258.

[25] PHP: Hypertext Preprocessor, The PHP Group, verkkosivu. Saatavissa (viitattu 14.02.2017): http://php.net/.

[26] T. F. Bissyandé, F. Thung, D. Lo, L. Jiang, L. Réveillère, Popularity, Interoperability, and Impact of Programming Languages in 100,000 Open Source Projects, Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual, pp. 303-312.

[27] The PHP Programming Language, TIOBE software BV, verkkosivu. Saatavissa (viitattu 08.12.2016): http://www.tiobe.com/tiobe-index/php/.

[28] jQuery, The jQuery Foundation, verkkosivu. Saatavissa (viitattu 19.11.2016): https://jquery.com/.

[29] CodeIgniter Web Framework, British Columbia Institute of Technology, verkkosivu. Saatavissa (viitattu 09.02.2017): https://www.codeigniter.com/.

[30] C. Pitt, Pro PHP MVC, Apress, Berkeley, CA, 2012, 471 p.

[31] I. P. Vuksanovic, B. Sudarevic, Use of web application frameworks in the development of small applications, MIPRO, 2011 Proceedings of the 34th International Convention, pp. 458-462.

[32] J. Carnell, R. Harrop, What We Do Wrong: Web Antipatterns Explained, in: K. Mittal (ed.), Pro Apache Struts with Ajax, Apress, Berkeley, CA, 2007, pp. 1-30.

[33] PostgreSQL: About, The PostgreSQL Global Development Group, verkkosivu. Saatavissa (viitattu 28.01.2017): https://www.postgresql.org/about/.

[34] About Us | Highcharts, Highsoft, verkkosivu. Saatavissa (viitattu 21.02.2017): http://www.highcharts.com/about.

[35] ECMA-404, The JSON Data Interchange Format, Ecma International, 2013, 7 p.

[36] PHPOffice/PHPExcel: A pure PHP library for reading and writing spreadsheet files Github, Inc., verkkosivu. Saatavissa (viitattu 13.03.2017): https://github.com/PHPOffice/PHPExcel.

[37] HTML Purifier, verkkosivu. Saatavissa (viitattu 09.02.2017): http://htmlpurifier.org/.

[38] DataTables | Table plug-in for jQuery, SpryMedia Ltd, verkkosivu. Saatavissa (viitattu 09.02.2017): https://datatables.net/.

[39] About CLEditor, Premium Software, verkkosivu. Saatavissa (viitattu 09.02.2017): http://premiumsoftware.net/cleditor.

[40] J. Vargas, colpick Color Picker, GitHub, Inc., verkkosivu. Saatavissa (viitattu 09.02.2017): https://github.com/josedvq/colpick-jQuery-Color-Picker.

[41] R.T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, 2000.

[42] Y. Liao, Z. Zhang, Y. Yang, Web Applications Based on Ajax Technology and Its Framework, in: M. Zhao, J. Sha (ed.), Communications and Information Processing: International Conference, ICCIP 2012 Aveiro, Portugal, March 7-11, 2012 Revised Selected Papers, Part I, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 320-326.

[43] L. Prechelt, Plat_Forms: A Web Development Platform Comparison by an Exploratory Experiment Searching for Emergent Platform Properties, IEEE Transactions on Software Engineering, Vol. 37, No. 1, 2011, pp. 95-108.

[44] J. Offutt, Quality attributes of Web software applications, IEEE Software, Vol. 19, No. 2, 2002, pp. 25-32.

[45] ISO/IEC 9126-1:2001, Software engineering -- Product quality -- Part 1: Quality model, International Organization for Standardization, 2001.

[46] ISO/IEC 25010:2011, Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models, International Organization for Standardization, 2011, 34 p.

[47] G.J. Covella, L.A. Olsina, Assessing Quality in Use in a Consistent Way, Proceedings of the 6th International Conference on Web Engineering, ACM, New York, NY, USA, pp. 1-8.

[48] B. Kitchenham, S. L. Pfleeger, Software quality: the elusive target [special issues section], IEEE Software, Vol. 13, No. 1, 1996, pp. 12-21.

[49] T. Barker, Pro JavaScript Performance: Monitoring and Visualization, Apress, Berkeley, CA, 2012, 207 p.

[50] E. Ghosheh, S. Black, J. Qaddour, Design metrics for web application maintainability measurement, 2008 IEEE/ACS International Conference on Computer Systems and Applications, pp. 778-784.

[51] P. Hegedűs, T. Bakota, L. Illés, G. Ladányi, R. Ferenc, T. Gyimóthy, Source Code Metrics and Maintainability: A Case Study, in: T. Kim, H. Adeli, H. Kim, H. Kang, K.J. Kim, A. Kiumi, B. Kang (ed.), Software Engineering, Business Continuity, and Education: International Conferences ASEA, DRBC and EL 2011, Held as Part of the Future Generation Information Technology Conference, FGIT 2011, in Conjunction with GDC 2011, Jeju Island, Korea, December 8-10, 2011. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 272-284.

[52] M. Kruckenberg, J. Pipes, Triggers, in: Pro MySQL, Apress, Berkeley, CA, 2005, pp. 443-465.

[53] S.R. Alapati, Improving Database Performance: SQL Query Optimization, in: Expert Oracle Database 11g Administration, Apress, Berkeley, CA, 2009, pp. 1041-1127.

[54] Chart.redraw(), Highsoft, verkkosivu. Saatavissa (viitattu 10.02.2017): http://api.highcharts.com/highcharts/Chart.redraw.

[55] AngularJS, Google, verkkosivu. Saatavissa (viitattu 16.12.2016): https://angularjs.org/.

[56] A. Gizas, S. Christodoulou, T. Papatheodorou, Comparative Evaluation of Javascript Frameworks, Proceedings of the 21st International Conference on World Wide Web, ACM, New York, NY, USA, pp. 513-514.

[57] B. Lin, Y. Chen, X. Chen, Y. Yu, Comparison between JSON and XML in Applications Based on AJAX, Computer Science & Service System (CSSS), 2012 International Conference on, pp. 1174-1177.

[58] Highcharts Performance Boost, Highsoft, verkkosivu. Saatavissa (viitattu 15.03.2017): http://www.highcharts.com/blog/news/175-highcharts-performance-boost/.

[59] J.W. Horch, Practical Guide to Software Quality Management, 1st ed. Artech House, Inc, Norwood, MA, USA, 1996.

[60] D. I. K. Sjøberg, A. Yamashita, B. C. D. Anda, A. Mockus, T. Dybå, Quantifying the Effect of Code Smells on Maintenance Effort, IEEE Transactions on Software Engineering, Vol. 39, No. 8, 2013, pp. 1144-1156.

[61] .on() | jQuery API Documentation, The jQuery Foundation, verkkosivu. Saatavissa (viitattu 15.03.2017): http://api.jquery.com/on/.

[62] D. Spinellis, Code Quality: The Open Source Perspective (Effective Software Development Series), Addison-Wesley Professional, 2006.

LIITE A: LAATUKRITEERIPUU

1. Sopivuus käyttöön

1.1. Ohjeistus

1.1.1. Järjestelmässä on erillinen ohje-/UKK-sivu

1.1.2. Kontekstuaaliset ohjeet

1.1.2.1. Käyränpiirrossa vastaaja saa tarpeeksi kontekstuaalisia vihjeitä, jotta hän osaa piirtää käyttäjäkokemuskäyrän

1.1.2.2. Tutkijan näkymissä on tarpeeksi käyttöä helpottavia kontekstuaalisia opastustekstejä

1.2. Toiminnallisuuden kattavuus

1.2.1. Kyselyjen luonti

1.2.1.1. Kyselyjä voi luoda

1.2.1.2. Kyselyjä voi muokata

1.2.1.3. Kokonaisen kyselyn kysymyksineen voi kopioida

1.2.1.4. Kyselyn voi poistaa vastauksineen

1.2.1.5. Kyselyä voi esikatsella

1.2.1.6. Kyselyn tilaa (auki/suljettu) voi muuttaa

1.2.1.7. Tutkija voi tarkastella omia kyselyjään listana

1.2.1.8. Kyselyn perustietoja voi muokata nopeasti myös kyselylistauksessa

1.2.1.9. Editorissa on tarjolla kattava valikoima kysymystyyppejä

1.2.1.10. Kyselyn voi lokalisoida helposti

1.2.1.11. Yksittäisen kysymyksen voi asettaa pakolliseksi

1.2.1.12. Piirtotehtävässä voidaan vaatia loppuun jatkettu kuvaaja

1.2.1.13. Piirtotehtävässä voidaan vaatia, että jokaiseen lisättyyn pisteeseen tulee liittää kommentti

1.2.1.14. Vastausnäkymän ohje- ja muita käyttöliittymätekstejä voi kustomoida

1.2.1.15. Kyselyn vastausnäkymän ulkoasua ja tyyliä voi kustomoida

1.2.1.16. Päiväkirjatyylisissä kyselyissä kunkin kysymyksen esityskerrat voi asettaa erikseen

1.2.1.17. Kysymyksen voi piilottaa poistamatta sitä kokonaan

1.2.2. Vastaajien hallinnointi

1.2.2.1. Kyselyyn voi kutsua vastaajia

1.2.2.2. Vastaajia voi muistuttaa vastaamisesta

1.2.2.3. Vastaajia ja vastauskertoja voi poistaa järjestelmästä

1.2.2.4. Kyselylle voi tallentaa kyselykohtaiset kutsu- ja muistutusviestit

1.2.3. Vastaaminen

1.2.3.1. Vastaaja voi piirtää käyttäjäkokemustaan kuvaavan käyrän

1.2.3.2. Käyrää voi lähentää ja loitontaa

1.2.3.3. Käyrän voi aloittaa uudelleen alusta

1.2.3.4. Kommentteihin ja kysymyskenttiin voi kirjoittaa tarpeeksi pitkiä vastauksia

1.2.3.5. Käyrän pisteisiin voi liittää valokuvia/kuvakaappauksia

1.2.3.6. Käyrän pisteisiin voi liittää riittävästi kontekstitietoa

1.2.3.7. Järjestelmä tukee myös päiväkirjatyylisiä kyselyjä, joihin voi vastata useita kertoja

1.2.3.8. Päiväkirjatyylisiin kyselyihin voi vastata myös julkisen linkin kautta hyödyntäen järjestelmän generoimaa tunnistetta

1.2.3.9. Vastaajan voi ohjata tietylle kyselyn sivulle toisen sivun vastausten perusteella

1.2.4. Mobiilivastaaminen

1.2.4.1. Kyselyyn on mahdollista vastata mobiilisti oman mobiilinäkymän kautta

1.2.4.2. Mobiilinäkymä skaalautuu hyvin erikokoisille näytöille

1.2.4.3. Mobiili-DrawUX on oma sovelluksensa käyttäjän laitteessa

1.2.5. Vastausten tarkastelu

1.2.5.1. Saatavilla on yhteenveto kaikista kyselyyn tulleista vastauksista

1.2.5.2. Yhteenveto tarjoaa tarpeeksi syvällistä tietoa, jotta tuloksista on helppo löytää olennaiset asiat

1.2.5.3. Tulosten yhteenvedossa näkyvät vastaukset voi valita vastaajakohtaisesti

1.2.5.4. Käyrien yhteenvedossa näkyviä käyriä voi suodattaa monipuolisesti

1.2.5.5. Yhteenvedossa näkyviä käyräkommentteja voi suodattaa monipuolisesti

1.2.5.6. Kaikkia tietyn käyrän tai kaikkien käyrien kommentteja voi tarkastella helposti kerralla

1.2.5.7. Yksittäisen käyräpisteen tietoruudun saa näkyviin yhteenvetokuvaajassa

1.2.5.8. Tietoruutuja voi siirtää kuvaajassa

1.2.5.9. Tietoruutujen kokoa voi muuttaa

1.2.5.10. Kuvaajaan sisältyy hyödyllisiä toimintoja sisältävä kontekstivalikko

1.2.5.11. Muokatun käyräyhteenvetokuvan voi viedä eri muodoissa
1.2.5.12. Siirrettyjen tietoruutujen asemat voi tallentaa
1.2.5.13. Tuloksista voi ladata Excel-muotoisen yhteenvedon
1.2.5.14. Vain valittujen vastaajien vastauksista voi ladata Excel-muotoisen yhteenvedon
1.2.5.15. Tuloksista voi ladata automaattisesti generoidun lomakevastaukset ja käyräyhteenvedot sisältävän tiedoston
1.2.5.16. Suodatetut kommentit voi viedä esim. Excel- tai tekstitiedostona

1.2.6. Kääntömahdollisuus
1.2.6.1. Kysymykset voi kääntää toiselle kielelle
1.2.6.2. Vastaukset voi kääntää toiselle kielelle
1.2.6.3. Kyselyä voi tarkastella käännettynä
1.2.6.4. Kyselystä voi tehdä useita eri käännöksiä
1.2.6.5. Kääntäminen tukee valmiin käännöstiedoston lukemista

1.2.7. Muuta
1.2.7.1. Rekisteröityminen
1.2.7.2. Sisäänkirjautuminen
1.2.7.3. Muu tunnuksenhallinta
1.2.7.4. Järjestelmässä on etusivu
1.2.7.5. Järjestelmässä on uutissivu, jossa kehittäjä voi tiedottaa muutoksista
1.2.7.6. Järjestelmässä on palautesivu, jossa käyttäjä voi lähettää ylläpitäjälle sähköpostia
1.2.7.7. Järjestelmässä on admin-sivu käyttäjien ja oikeuksien hallinnointiin ja uutisten syöttöön
1.2.7.8. Järjestelmässä on tutkijoille suunnattu käyttöehto/yksityisyyspolitiikkasivu
1.2.7.9. Järjestelmässä on erillinen vastaajille näkyvä yksityisyyspolitiikkasivu

1.3. Käyttöliittymät

1.3.1. Käyttöliittymien esteettinen miellyttävyys
1.3.1.1. Käyttöliittymistä saatu subjektiivisiin mielipiteisiin perustuva palaute on positiivista
1.3.1.2. Käyttöliittymien ulkoasuun on panostettu käyttämällä riittävästi valmiita templateja
1.3.1.3. Käyttöliittymät ovat personoitavissa

1.3.2. Käyttöliittymien yhdenmukaisuus
1.3.2.1. Järjestelmän eri sivut ovat ulkoasultaan selvästi yhdenmukaisia
1.3.2.2. Järjestelmän antama käyttäjäpalaute on eri näkymissä ja eri tilanteissa yhdenmukaista

1.3.3. Käyttöliittymien helppokäyttöisyys
1.3.3.1. Vastaajan käyttöliittymä koetaan niin käytettäväksi, että työkalua voidaan käyttää etätutkimuksiin
1.3.3.2. Tutkija kokee tutkijalle tarkoitetut käyttöliittymät helppokäyttöisiksi

1.4. Luottamus järjestelmän toimivuuteen
1.4.1. Nykyinen toteutus on perusteellisesti testattu
1.4.2. Testaus on järjestelmällisesti osa kehitysprosessia

2. Tehokkuus

2.1. Sivujen latausaikojen optimointi
2.1.1. Käytännön latausajat ovat tarpeeksi lyhyitä
2.1.2. Sivujen koko (tavuissa) on pieni

2.1.3. Tietokannanhallintajärjestelmän riittävä hyödyntäminen
2.1.3.1. Herättimet ja tallennetut proseduurit korvaavat malleista käsin tapahtuvan käsittelyn aina kun mahdollista
2.1.3.2. Tietokannassa on hyödynnetty indeksejä

2.2. Selainkoodin suorituskyky

2.2.1. Käyränäkymien kuormitus
2.2.1.1. Vastausnäkymän käyränpiirto ei kuormita selainta liikaa
2.2.1.2. Selain pystyy käsittelemään käyräyhteenvetokuvaajaa suurellakin käyrämäärällä

2.2.2. Selainkoodin suorituskyvyn optimointi
2.2.2.1. Kuormitusta pyritään keventämään sopivien oletusasetusten avulla
2.2.2.2. Muutokset DOM-elementteihin ketjutetaan aina kuin mahdollista
2.2.2.3. Turhaa laskentaa vältetään aina kuin mahdollista
2.2.2.4. Selainkoodissa käytetään puhdasta JavaScriptia
2.2.2.5. Käytetyt teknologiat ovat vaihtoehtoisiin teknologioihin nähden tehokkaita

2.3. Laskennasta mahdollisimman suuri osa tehdään selaimessa

3. Ylläpidettävyys

3.1. Teknologiavalinnat
3.1.1. Järjestelmän arkkitehtuurin pohjana on käytetty valmista MVC-kehystä
3.1.2. Järjestelmässä on hyödynnetty mahdollisimman paljon valmiita komponentteja

3.1.3. Käytetyt teknologiat ovat helposti laajennettavia ja toimivat hyvin yhteen DrawUX:n oman koodin kanssa
3.1.4. Käytettyjen teknologioiden syntaksi on helposti ymmärrettävää
3.2. Dokumentointi
3.2.1. Toteutus ja kriittiset ratkaisut on dokumentoitu riittävällä tasolla
3.2.2. Dokumentointi on osa prosessia ja sille on selkeät käytännöt
3.2.3. Koodin kommentointi
3.2.3.1. Koodia on kommentoitu tarpeeksi
3.2.3.2. Koodin kommentit ovat sisällöltään hyödyllisiä
3.2.4. Koodissa käytetty nimeäminen
3.2.4.1. Muuttujien ja funktioiden nimet ovat intuitiivisia
3.2.4.2. Muuttujien ja funktioiden nimissä käytetään yhdenmukaista terminologiaa
3.3. Kooditiedostot
3.3.1. Tiedostot ovat rivimäärältään tarpeeksi pieniä
3.3.2. Koodin selkeys
3.3.2.1. Saman asian toteutukseen käytetään samaa vakiintunutta tapaa/syntaksia aina kun mahdollista
3.3.2.2. Funktioiden järjestys tiedostoissa on looginen
3.3.2.3. Muuttujien ja funktioiden kirjoitusasut noudattavat valittua yhdenmukaista käytäntöä
3.3.2.4. Funktiot ovat pituudeltaan tarpeeksi lyhyitä
3.3.2.5. Koodi on siistiä
3.3.3. Tiedostot eivät sisällä kovakoodausta
3.4. Moduulien roolijako
3.4.1. Järjestelmä noudattaa selkeää arkkitehtuuria, jossa ohjelman osilla on selvä roolijako
3.4.2. Palvelin-selain-kommunikointi on mahdollisimman suoraviivaista
3.4.3. Datankäsittely
3.4.3.1. Datankäsittely näkymien välillä on yhdenmukaista
3.4.3.2. Datankäsittely muodostaa oman loogisen kerroksensa

LIITE B: ARVIOINTI

1. Sopivuus käyttöön

1.1. Ohjeistus

1.1.1. Järjestelmässä on erillinen ohje-/UKK-sivu

Toiminto puuttuu.

1.1.2. Kontekstuaaliset ohjeet

1.1.2.1. Käyränpiirrossa vastaaja saa tarpeeksi kontekstuaalisia vihjeitä, jotta hän osaa piirtää käyttäjäkokemuskäyrän

Ohjelaatikoita esitetään runsaasti. Ainakin ensimmäisten ohjeiden lukeminen varmistetaan (ohjeet tulee kuitata luetuiksi klikkaamalla ennen kuin käyränpiirron voi aloittaa).

1.1.2.2. Tutkijan näkymissä on tarpeeksi käyttöä helpottavia kontekstuaalisia opastustekstejä

Monimutkaisemmissa näkymissä (Data Viewer, editori) on runsaasti työkaluvihjeitä selittämässä kunkin elementin tarkoitusta.

1.2. Toiminnallisuuden kattavuus

1.2.1. Kyselyjen luonti

1.2.1.1. Kyselyjä voi luoda

Toiminto löytyy.

1.2.1.2. Kyselyjä voi muokata

Toiminto löytyy.

1.2.1.3. Kokonaisen kyselyn kysymyksineen voi kopioida

Toiminto löytyy, mutta toimintoon liittyy pieni ohjelmointivirhe (retrospektiivisen kyselyn kopion kyselysivujen yhteydessä näytetään päiväkirjatyylisiin kyselyihin liittyvä sivun esityskertojen valinta). Virhe on käytettävyys- /esteettinen ongelma, mutta ei estä toiminnon käyttämistä.

1.2.1.4. Kyselyn voi poistaa vastauksineen

Toiminto löytyy.

1.2.1.5. Kyselyä voi esikatsella

Toiminto löytyy.

1.2.1.6. Kyselyn tilaa (auki/suljettu) voi muuttaa

Toiminto löytyy.

1.2.1.7. Tutkija voi tarkastella omia kyselyjään listana

Toiminto löytyy.

1.2.1.8. Kyselyn perustietoja voi muokata nopeasti myös kyselylistauksessa

Toiminto löytyy (perustiedoista vain otsikkoa voi muokata listausnäkymässä). Kyselyn voi myös poistaa tai kopioida listausnäkymässä nopeasti.

1.2.1.9. Editorissa on tarjolla kattava valikoima kysymystyyppejä

Kysymystyyppejä on kuusi. Peruskyselyihin liittyvät kysymystyypit löytyvät. Asiakas koki kysymystyyppivalikoiman kattavaksi. Kuitenkin jatkokehitysideoista keskusteltaessa projektipäällikkö on tuonut esiin tarpeen lisäkysymystyypeille.

1.2.1.10. Kyselyn voi lokalisoida helposti

Työkalu tukee lokalisointia rajoitetussa määrin. Aloituspäivänvalintaan käytettävän komponentin kieli sekä piirrosalueen aika-akselin kieli valitaan asetusvälilehdeltä, kielivaihtoehtoja on tällä hetkellä neljä, valittu asiakkaan tarpeiden mukaan (suomi, englanti, eesti, espanja). Tämän kielivalinnan alaiset käyttöliittymätekstit löytyvät omasta kielitiedostostaan. Suunnitteilla oli, että lokalisoinnin alaiset tekstit voitaisiin säilyttää tietokannassa. Tutkija voi itse asettaa muut yleiset käyttöliittymätekstit haluamikseen asetusvälilehdellä. Piirrostehtäväkohtaiset käyttöliittymätekstit asetetaan piirtotehtävän omalta välilehdeltä. Tämä käytäntö tukee kuitenkin vain yhtä kieliversiota kerrallaan, joten jos sama kysely halutaan toteuttaa eri maissa, kysely tulee ensin kopioida ja kääntää asetukset käsin. Suunnitteilla oli päivämääräformaatin vapaa valinta, mutta toimintoa ei toistaiseksi ole toteutettu.

1.2.1.11. Yksittäisen kysymyksen voi asettaa pakolliseksi

Toiminto löytyy.

1.2.1.12. Piirtotehtävässä voidaan vaatia loppuun jatkettu kuvaaja

Toiminto löytyy.

1.2.1.13. Piirtotehtävässä voidaan vaatia, että jokaiseen lisättyyn pisteeseen tulee liittää kommentti

Toiminto puuttuu. Piirtotehtävässä voidaan vaatia valittu minimimäärä pisteitä, mutta toistaiseksi ei löydy asetusta, joka vaatisi jokaiseen piirrettyyn pisteeseen kommentin. Asiakas toi haastatteluissa ilmi tarpeen toiminnolle.

1.2.1.14. Vastausnäkymän ohje- ja muita käyttöliittymätekstejä voi kustomoida

Toiminto löytyy.

1.2.1.15. Kyselyn vastausnäkymän ulkoasua ja tyyliä voi kustomoida

Toiminto löytyy rajoitetussa määrin. Kysymysten tyyliä voidaan muokata editorissa rikkaan HTML-editorin avulla (tekstin fonttia, väriä ja muuta esitystapaa voidaan muuttaa ja kysymykseen voidaan upottaa esimerkiksi kuvia). Koko kyselyn taustan väri voidaan valita vapaasti. Kuitenkin työkalulle on suunniteltu myös mahdollisuutta valita tyyli suoraan erilaisista vaihtoehdoista. Oli myös suunnitteilla, että tutkija voisi säätää kyselykohtaisesti mm. mitkä apuviivat kuvaaja-alueella näytetään sekä kustomoida vapaasti painikkeiden yms. elementtien värejä. Kustomoitavuudessa on siis vielä puutteita alkuperäisiin suunnitelmiin nähden.

1.2.1.16. Päiväkirjatyylisissä kyselyissä kunkin kysymyksen esityskerrat voi asettaa erikseen

Toiminto löytyy rajoitetusti. Kysymyssivun voi asettaa näkymään aina tai ensimmäisillä *N* vastauskerralla. Suunnitteilla oli myös vastausajankohtaan tai kysymyksen aikaisempiin vastauskertoihin perustuva ehdollinen kysymyssivun esittäminen.

1.2.1.17. Kysymyksen voi piilottaa poistamatta sitä kokonaan

Toiminto puuttuu.

1.2.2. Vastaajien hallinnointi

1.2.2.1. Kyselyyn voi kutsua vastaajia

Toiminto löytyy.

1.2.2.2. Vastaajia voi muistuttaa vastaamisesta

Toiminto löytyy.

1.2.2.3. Vastaajia ja vastauskertoja voi poistaa järjestelmästä

Toiminto löytyy.

1.2.2.4. Kyselylle voi tallentaa kyselykohtaiset kutsu- ja muistutusviestit

Toiminto löytyy.

1.2.3. Vastaaminen

1.2.3.1. Vastaaja voi piirtää käyttäjäkokemustaan kuvaavan käyrän

Toiminto löytyy.

1.2.3.2. Käyrää voi lähentää ja loitontaa

Toiminto löytyy.

1.2.3.3. Käyrän voi aloittaa uudelleen alusta

Toiminto löytyy.

1.2.3.4. Kommentteihin ja kysymyskenttiin voi kirjoittaa tarpeeksi pitkiä vastauksia

Kenttiä on projektin aikana pidennetty useaan kertaan. Vastauskenttien pituudeksi haluttiin projektin aikana väh. 2000 merkkiä, mikä toteutettiin.

1.2.3.5. Käyrän pisteisiin voi liittää valokuvia/kuvakaappauksia

Toiminto puuttuu. Asiakas on esittänyt toiveen kuvien lisäysmahdollisuudesta käyrälle.

1.2.3.6. Käyrän pisteisiin voi liittää riittävästi kontekstitietoa

Toiminnossa puutteita. Tutkija voi halutessaan ottaa käyttöön pisteeseen liittyvän liukusäätimen, jonka kautta vastaaja voi kertoa liittyykö kommentti positiiviseen vai negatiiviseen kokemukseen. Asiakas on esittänyt lisäksi toiveen muun kontekstitiedon keräämisestä, esim. alasvetovalikko tapahtuman paikan valintaan.

1.2.3.7. Järjestelmä tukee myös päiväkirjatyylisiä kyselyjä, joihin voi vastata useita kertoja

Toiminto löytyy.

1.2.3.8. Päiväkirjatyylisiin kyselyihin voi vastata myös julkisen linkin kautta hyödyntäen järjestelmän generoimaa tunnistetta

Toiminto puuttuu.

1.2.3.9. Vastaajan voi ohjata tietylle kyselyn sivulle toisen sivun vastausten perusteella

Toiminto puuttuu.

1.2.4. Mobiilivastaaminen

1.2.4.1. Kyselyyn on mahdollista vastata mobiilisti oman mobiilinäkymän kautta

Toiminto löytyy.

1.2.4.2. Mobiilinäkymä skaalautuu hyvin erikokoisille näytöille

Ei juurikaan testattu. Projektipäällikkö on toivonut parempaa skaalautuvuutta erikokoisille näytöille.

1.2.4.3. Mobiili-DrawUX on oma sovelluksensa käyttäjän laitteessa

Toiminto puuttuu. Mobiilikäyttöliittymä on toteutettu HTML-sivuna, joten vastaajan tulee mobiilistikin käyttää vastaamiseen selainta. Asiakas on esittänyt toiveen siitä, että mobiilivastaaminen tapahtuisi mieluummin erillisen mobiilisovelluksen kautta.

1.2.5. Vastausten tarkastelu

1.2.5.1. Saatavilla on yhteenveto kaikista kyselyyn tulleista vastauksista

Toiminto löytyy.

1.2.5.2. Yhteenveto tarjoaa tarpeeksi syvällistä tietoa, jotta tuloksista on helppo löytää olennaiset asiat

Toiminnossa mahdollisesti puutteita. Asiakas toi haastattelussa esiin, että dataa saadaan paljon, mutta analyysivaiheessa hankala poimia olennainen esiin. Toisaalta kuitenkin käyrät saivat positiivista palautetta (havainnollinen tapa tarkastella käyttäjäkokemusdataa).

1.2.5.3. Tulosten yhteenvedossa näkyvät vastaukset voi valita vastaajakohtaisesti

Toiminto löytyy.

1.2.5.4. Käyrien yhteenvedossa näkyviä käyriä voi suodattaa monipuolisesti

Käyriä voidaan suodattaa käyrän muodon perusteella (nouseva/laskeva/tasainen) sekä alku- ja loppupisteen y-koordinaattiasemien perusteella. Lisäksi oli suunnitteilla mahdollisuus suodattaa näkyviä käyriä esim. tietyn taustatekijän (tietyn kysymyksen vastauksen) perusteella, mitä ei toistaiseksi ole toteutettu.

1.2.5.5. Yhteenvedossa näkyviä käyräkommentteja voi suodattaa monipuolisesti

Toiminnossa puutteita. Käyräkommentteihin voi käyttää suodattavaa sanahakua, mutta lisäksi on suunniteltu, että kommentteja voitaisiin suodattaa myös esim. paikan perusteella (näytä vain käyrän huippu- tai laaksokohtiin liittyvät käyrän kommentit, näytä vain tiettyyn x-akselin aikaväliin liittyvät kommentit). Lisäksi tällä hetkellä sanahakusuodatus kohdistuu vain kommenttiyhteenvetotaulukkoon eikä vaikuta mitenkään käyränäkymään.

1.2.5.6. Kaikkia tietyn käyrän tai kaikkien käyrien kommentteja voi tarkastella helposti kerralla

Toiminto löytyy. Tietyn käyrän kommentit saadaan näkyviin taulukkomuodossa valitsemalla käyrän kuvaajasta. Kuvaajaan kommentit saadaan näkymään kerralla valitsemalla ’näytä kaikki tietoruudut’-asetus. Kaikkien käyrien kommentteja voi tarkastella, kun valittuna on keskiarvokäyrä tai ei yhtään käyrää.

1.2.5.7. Yksittäisen käyräpisteen tietoruudun saa näkyviin yhteenvetokuvaajassa

Toiminto löytyy.

1.2.5.8. Tietoruutuja voi siirtää kuvaajassa

Toiminto löytyy.

1.2.5.9. Tietoruutujen kokoa voi muuttaa

Toiminto löytyy.

1.2.5.10. Kuvaajaan sisältyy hyödyllisiä toimintoja sisältävä kontekstivalikko

Toiminto puuttuu. Suunnitteilla oli kontekstivalikko, johon liittyisi toimintoja kuten valitun käyrän piilotus tai sen tietoruutujen näyttäminen.

1.2.5.11. Muokatun käyräyhteenvetokuvan voi viedä eri muodoissa

Toiminto löytyy. Kaikkia suunniteltuja vientiformaatteja (.png, .jpg, .svg, .pdf) tuetaan.

1.2.5.12. Siirrettyjen tietoruutujen asemat voi tallentaa

Toiminto puuttuu. Asiakas toi ilmi tarpeen voida tallentaa siirreltyjen tietoruutujen asemat, jotta tutkija ei menetä huolelliseen näkymän siistimiseen käytettyä työtä.

1.2.5.13. Tuloksista voi ladata Excel-muotoisen yhteenvedon

Toiminto löytyy.

1.2.5.14. Vain valittujen vastaajien vastauksista voi ladata Excel-muotoisen yhteenvedon

Toiminto puuttuu.

1.2.5.15. Tuloksista voi ladata automaattisesti generoidun lomakevastaukset ja käyräyhteenvedot sisältävän tiedoston

Toiminto puuttuu. Oli jossain määrin suunnitteilla/keskustelun alaisena projektin aikana, mutta aikarajoitteiden vuoksi päädyttiin toteuttamaan vain kuvaajan vienti.

1.2.5.16. Suodatetut kommentit voi viedä esim. Excel- tai tekstitiedostona

Toiminto puuttuu. Haluttiin tapa viedä suodatetut kommentit automaattisesti generoitavan tiedostona, tai vähintäänkin helppo tapa kopioida suodatetut kommentit selainikkunasta. Jälkimmäinen onnistuu (Data Viewerin kommenttitaulukosta), mutta tiedoston vienti olisi parempi.

1.2.6. Kääntömahdollisuus

1.2.6.1. Kysymykset voi kääntää toiselle kielelle

Toiminto löytyy. Käännös kuitenkin vaikuttaa vain Data Vieweriin, eli ei auta vastaajan näkymän esittämisessä eri kielillä.

1.2.6.2. Vastaukset voi kääntää toiselle kielelle

Toiminto löytyy.

1.2.6.3. Kyselyä voi tarkastella käännettynä

Toiminto löytyy.

1.2.6.4. Kyselystä voi tehdä useita eri käännöksiä

Toiminto puuttuu, kyselystä on olemassa vain yksi käännös kerrallaan.

1.2.6.5. Kääntäminen tukee valmiin käännöstiedoston lukemista

Toiminto puuttuu. Asiakashaastattelussa tuli ilmi tarve helpommalle tavalle lukea järjestelmään eri kieliversioita, esim. valmiin käännöstiedoston lukeminen järjestelmään napin avulla. Tällöin kääntäjän ei tarvitsisi käyttää itse DrawUX-palvelua, vaan käännöstoimisto voisi tehdä työnsä erillään omaan tiedostoonsa, joka tutkijan olisi helppo ladata järjestelmään. Toiminto ei kuitenkaan ehditty toteuttaa.

1.2.7. Muuta

1.2.7.1. Rekisteröityminen

Toiminto löytyy.

1.2.7.2. Sisäänkirjautuminen

Toiminto löytyy.

1.2.7.3. Muu tunnuksenhallinta

Tarvittava tunnuksenhallinta löytyy. Salasanan voi vaihtaa ja unohtuneen salasanan tilalle voi pyytää uutta salasanaa sähköpostiin. Sähköpostiosoitteen voi muuttaa. Järjestelmästä voi myös poistaa tunnuksen.

1.2.7.4. Järjestelmässä on etusivu

Toiminto löytyy.

1.2.7.5. Järjestelmässä on uutissivu, jossa kehittäjä voi tiedottaa muutoksista

Toiminto löytyy.

1.2.7.6. Järjestelmässä on palautesivu, jossa käyttäjä voi lähettää ylläpitäjälle sähköpostia

Toiminto löytyy.

1.2.7.7. Järjestelmässä on admin-sivu käyttäjien ja oikeuksien hallinnointiin ja uutisten syöttöön

Toiminto puuttuu.

1.2.7.8. Järjestelmässä on tutkijoille suunnattu käyttöehto/yksityisyyspolitiikkasivu

Toiminto löytyy.

1.2.7.9. Järjestelmässä on erillinen vastaajille näkyvä yksityisyyspolitiikkasivu

Toiminto puuttuu.

1.3. Käyttöliittymät

1.3.1. Käyttöliittymien esteettinen miellyttävyys

1.3.1.1. Käyttöliittymistä saatu subjektiivisiin mielipiteisiin perustuva palaute on positiivista

Käyttöliittymistä saatu asiakaspalaute on ollut jossain määrin negatiivista. Käyttöliittymiä on kuvailtu kömpelöiksi, erityisesti kysymysten ulkoasu vastaajalle.

1.3.1.2. Käyttöliittymien ulkoasuun on panostettu käyttämällä riittävästi valmiita templateja

Varsinkaan tutkijan käyttöliittymiin ei ole juurikaan panostettu. jQuery UI -tyylejä on käytetty jonkin verran, mutta sekalaisesti. Joka näkymälle yhtenäistä käyttöliittymäkirjastoa ei ole käytössä.

1.3.1.3. Käyttöliittymät ovat personoitavissa

Käyttöliittymiä ei voi juurikaan personoida. Tutkija voi personoida vastaajan näkymää hyvin rajoitetusti, mutta vastaaja tai tutkija ei voi muokata omaa käyttöliittymäänsä millään tavalla.

1.3.2. Käyttöliittymien yhdenmukaisuus

1.3.2.1. Järjestelmän eri sivut ovat ulkoasultaan selvästi yhdenmukaisia

Eri sivut ovat jossain määrin erinäköisiä. Tutkijan näkymiä yhdistää globaali navigointi ja pohjatyyli. Erityisesti uutissivu poikkeaa tyyliltään ja väreiltään muista. Osa näkymistä (editori, Data Viewer, kyselylistaus) käyttää yhdenmukaista painiketyyliä (pastellivärit yhden pikselin reunalla), kun taas tunnustenhallintaan liittyvät sivut käyttävät erilaista painiketyyliä (voimakkaat perusvärit, ei reunoja). Myös vastaajan näkymä on väreiltään ja tyyliltään erilainen kuin tutkijan näkymät.

1.3.2.2. Järjestelmän antama käyttäjäpalaute on eri näkymissä ja eri tilanteissa yhdenmukaista

Eri näkymissä palaute annetaan eri tavoilla.

1.3.3. Käyttöliittymien helppokäyttöisyys

1.3.3.1. Vastaajan käyttöliittymä koetaan niin käytettäväksi, että työkalua voidaan käyttää etätutkimuksiin

Asiakaspalaute ristiriitaista. Toisaalta työkalun käyttö vastaamiseen koettiin helpoksi ja sen käyttö onnistui lähes kaikilta vastaajilta vaivatta (yksi saattoi käyttää työkalua väärin). Toisaalta vastaajien läsnäolo demosessiossa koettiin olennaiseksi ja asiakas arvioi, ettei näin erilainen tiedonkeruumenetelmä välttämättä toimisi etätutkimuksissa ilman paikan päällä tapahtuvaa opastussessiota. Y-akselin arvioitava attribuutti haluttiin selkeämmin esille ja sille mahdollinen määritelmä. Joidenkin vastaajien oli hankala hahmottaa vaihtuvia attribuutteja (kun samaa tuotetta arvioitiin useilla piirtotehtävillä eri ominaisuuksien suhteen).

1.3.3.2. Tutkija kokee tutkijalle tarkoitetut käyttöliittymät helppokäyttöisiksi

Asiakaspalautteen mukaan työkalu oli ’suoraviivainen ja helppokäyttöinen’. Joitakin yksityiskohtia voisi parantaa. Data Viewerissä käyräkuvaajan pisteiden korostus on saanut kritiikkiä (ei aina korosteta pistettä joka tuntuisi olevan intuitiivisesti lähimpänä osoitinta, vaikea saada haluttua pistettä korostettua). Vapaalla suorakaidevalinnalla tapahtuva zoomaus saattaa olla liian herkkä, tapahtuu joskus vahingossa. Tietoruudun lukitseminen näkyviin voisi antaa tilan muutoksesta selkeämmän palautteen käyttäjille, esim. taustavärin välähdys.

1.4. Luottamus järjestelmän toimivuuteen

1.4.1. Nykyinen toteutus on perusteellisesti testattu

Toteutusta on testattu lähinnä ad hoc -kokeilun tasolla.

1.4.2. Testaus on järjestelmällisesti osa kehitysprosessia

Testaus ei ollut oma osansa kehitysprosessia eikä sille ollut varattu omaa henkilökuntaa.

2. Tehokkuus

2.1. Sivujen latausaikojen optimointi

2.1.1. Käytännön latausajat ovat tarpeeksi lyhyitä

Data Viewer on ainoa sivu jonka latautumisajat ovat ongelma. Pienellä vastaajamäärällä (alle 20) latausaika on kohtuullinen, mutta jo 40 vastaajalla hidas (yli 20s latausaika nopealla 100/100 Mbit/s yhteydellä). Johtuu käyräkuvaajien kaikkien käyrien alustamisesta kerralla ja mahdollisesti keskiarvokäyrästä. Data Viewerіä lukuun ottamatta sivut latautuvat nopeasti. Vastausnäkymän latausajat hyvillä yhteyksillä 1–2 sekunnin luokkaa, vaikka piirtotehtäviä olisi useita.

2.1.2. Sivujen koko (tavuissa) on pieni

Näkymät ovat melko suurikokoisia, muut sivut n. 100kt-800kt, Data Viewer suurin, usein yli 1Mt. HTML:n merkitys kokonaisuudessa on vähäinen. Suurin osa ladattavasta datasta on JavaScriptia, loput lähinnä CSS. Sivujen koon ei pitäisi olla ongelma moderneilla yhteyksillä.

2.1.3. Tietokannanhallintajärjestelmän riittävä hyödyntäminen

2.1.3.1. Herättimet ja tallennetut proseduurit korvaavat malleista käsin tapahtuvan käsittelyn aina kun mahdollista

Herättimiä ja tallennettuja proseduureja ei ole käytössä, kaikki vastaava käsittely tehdään malleista käsin.

2.1.3.2. Tietokannassa on hyödynnetty indeksejä

TKHJ luo automaattisesti indeksit pääavaimista, mutta muuten indeksointia ei ole käytetty suorituskykyoptimointiin.

2.2. Selainkoodin suorituskyky

2.2.1. Käyränäkymien kuormitus

2.2.1.1. Vastausnäkymän käyränpiirto ei kuormita selainta liikaa

Näyttäisi olevan OK. Käyränpiirto normaalitapauksessa ei tunnu kuormittavan selainta liikaa, vaan piirtäminen onnistuu sujuvasti vaikka kyselyssä olisi useita piirtotehtäviä. Omassa käytössä ja asiakkaan pilottitesteissä eikä muussakaan kehityksen vaiheessa ole tullut vastaan tilanteita, joissa selain ei olisi pystynyt käsittelemään piirtotehtävää sujuvasti.

2.2.1.2. Selain pystyy käsittelemään käyräyhteenvetokuvaajaa suurellakin käyrämäärällä

Ei testattu tarpeeksi. Omissa testeissä pienellä käyrämäärällä (max n. 40 käyrää) selain suoriutuu alkulatauksen jälkeisestä käsittelystä melko sujuvasti, mutta vaaditaan kunnollista kuormitustestausta ennen kuin Data Viewerin käyräkuvaajan suorituskyky voidaan todeta riittäväksi.

2.2.2. Selainkoodin suorituskyvyn optimointi

2.2.2.1. Kuormitusta pyritään keventämään sopivien oletusasetusten avulla

Ei käytetty. Data Viewerissä oletuksena näytetään vain 5 vastaajan vastaukset, mutta loput vastauksista on silti näkymässä mukana piilotettuna. Auttaisi myös suorituskykyä, jos nämä todella lisättäisiin näkymään vasta tarvittaessa.

2.2.2.2. Muutokset DOM-elementteihin ketjutetaan aina kuin mahdollista

Ei käytetty tarpeeksi. Muutokset tehdään usein yksi kerrallaan jokainen omalla rivillään. Toisaalta hyödynnetty hyvin Highcharts-kuvaajien kanssa (kuvaajamuutosten ketjutus + Chart.redraw():n kutsuminen jälkeenpäin).

2.2.2.3. Turhaa laskentaa vältetään aina kuin mahdollista

Data Viewerissä käytössä päivitysnappi, jonka avulla vältetään turhaa laskentaa kuormituskriittisessä kuvaajanäkymässä. Muutostaulukon käyttö editorissa saattaa lisätä turhaa laskentaa hieman, mutta vaihtoehtoisilla ratkaisuilla olisi myös heikkoutensa.

2.2.2.4. Selainkoodissa käytetään puhdasta JavaScriptia

Jotkin Highchartsia käyttävät osat on toteutettu puhtaalla JavaScriptilla, mutta suurimmaksi osaksi selainkoodi käyttää jQuery-kehystä.

2.2.2.5. Käytetyt teknologiat ovat vaihtoehtoisiin teknologioihin nähden tehokkaita

jQuery pärjää tehokkuusvertailussa muille vastaaville kehyksille. JSON on tehokas tiedonsiirtoformaatti. (Myös palvelinpuolella CodeIgniter on kevyt kehys jossa vain vähän valmista oletustoiminnallisuutta mukana, mikä tekee kehyksestä tehokkaan vaihtoehdon.)

2.3. Laskennasta mahdollisimman suuri osa tehdään selaimessa

Useissa näkymissä käytetään osittaiseen päivitykseen Ajaxia sen sijaan että koko näkymä rakennettaisiin uudelleen. Voitaisiin mahdollisesti hyödyntää vielä enemmän, esim. kyselyn tallennus editorissa kun pysytään editorinäkymässä. Monet validoinnit pyritään tekemään selaimessa jo ennen datan lähettämistä palvelimelle.

3. Ylläpidettävyys

3.1. Teknologiavalinnat

3.1.1. Järjestelmän arkkitehtuurin pohjana on käytetty valmista MVC-kehystä

Kyllä, käytössä CodeIgniter.

3.1.2. Järjestelmässä on hyödynnetty mahdollisimman paljon valmiita komponentteja

Aina kuin mahdollista. Käytössä mm. Highcharts, PHPExcel, HTMLPurifier, CLEditor, DataTables.

3.1.3. Käytetyt teknologiat ovat helposti laajennettavia ja toimivat hyvin yhteen DrawUX:n oman koodin kanssa

Kyllä. Valittujen teknologioiden kanssa ei ole tullut ongelmia, vaan ne helpottivat ja tukivat hyvin DrawUX:n kehitystä.

3.1.4. Käytettyjen teknologioiden syntaksi on helposti ymmärrettävää

Kyllä. Erityisesti jQueryn syntaksi helpottaa selainkoodin ymmärrettävyyttä suuresti verrattuna puhtaaseen JavaScriptiin. CodeIgniter selkeyttää PHP-koodia. Highchartsissa asetuksilla ja funktioilla on intuitiiviset nimet. JSON on luettavampaa kuin XML.

3.2. Dokumentointi

3.2.1. Toteutus ja kriittiset ratkaisut on dokumentoitu riittävällä tasolla

Ei tarpeeksi dokumentointia.

3.2.2. Dokumentointi on osa prosessia ja sille on selkeät käytännöt

Dokumentointi ei ole vakiintunut osa prosessia. Dokumentointi sekalaisissa muodoissa (alustavat suunnitelmat, koodin kommentointi, teknologialistaus, Azendoo, omat muistiinpanot), ei yhtenäistä käytäntöä.

3.2.3. Koodin kommentointi

3.2.3.1. Koodia on kommentoitu tarpeeksi

Koodissa on kohtalainen määrä kommentteja, ei ehkä kuitenkaan tarpeeksi, kaikkia hankalimpia kohtia ei ole kommentoitu. Joka funktion alussa kommentti.

3.2.3.2. Koodin kommentit ovat sisällöltään hyödyllisiä

Kommentit ovat enimmäkseen hyödyllisiä. Kuitenkin monissa kohtaa ratkaisuun päätymisen tai hylkäämisen perusteita on jätetty selittämättä. Kommenttien näennäistä määrää kasvattaa suuri määrä poiskommentoitua koodia. Funktion alussa käytetyt kommentit eivät noudata tiettyä yhtenäistä käytäntöä eivätkä aina kuvaa esim. missä tilanteessa funktiota kutsutaan ja millä esi- ja jälkiehdoilla.

3.2.4. Koodissa käytetty nimeäminen

3.2.4.1. Muuttujien ja funktioiden nimet ovat intuitiivisia

Muuttujien ja funktioiden nimet kuvaavat selkeästi sitä, mitä muuttujaan tallennetaan tai mitä funktio tekee.

3.2.4.2. Muuttujien ja funktioiden nimissä käytetään yhdenmukaista terminologiaa

Ei aina. Joihinkin asioihin viitataan koodin kommenteissa (ja suunnitelmissa ja muussa dokumentaatiossa) useilla eri nimillä.

3.3. Kooditiedostot

3.3.1. Tiedostot ovat rivimäärältään tarpeeksi pieniä

Tiedostojen rivimäärät ovat suuria, usein n. 500–2000 riviä.

3.3.2. Koodin selkeys

3.3.2.1. Saman asian toteutukseen käytetään samaa vakiintunutta tapaa/syntaksia aina kun mahdollista

Enimmäkseen kyllä, mutta esim. tapahtumankäsittelijöiden liittämiseen käytetään samassakin tiedostossa useaa syntaksiltaan erilaista tapaa. Sekaisin jQuerya ja puhdasta JavaScriptia.

3.3.2.2. Funktioiden järjestys tiedostoissa on looginen

Tiedostoilla on erilaisia funktiojärjestyksiä, monissa tiedostoissa ei mitään ilmeistä funktiojärjestystä.

3.3.2.3. Muuttujien ja funktioiden kirjoitusasut noudattavat valittua yhdenmukaista käytäntöä

Ei yhtä yhdenmukaista kirjoitusasukäytäntöä. Nimeäminen vaihtelee lähinnä ns. karavaanityylin (camelCase) ja käärmetyylin (snake_case) välillä.

3.3.2.4. Funktiot ovat pituudeltaan tarpeeksi lyhyitä

Funktioiden pituudessa suuri vaihteluväli. Osa funktioista varsinkin palvelinpuolella on liian pitkiä mahtuakseen kerralla näytölle.

3.3.2.5. Koodi on siistiä

Koodissa on käytetty sopivasti tyhjiä rivejä kokonaisuuksien (esim. funktioiden) erotteluun. Rivit ovat sopivan pitkiä, ei liikaa merkkejä yhdellä rivillä, tästä poikkeuksena näkymätiedostot, joissa HTML-merkkaus venyy usein vaakasuunnassa pitkäksi. Tiedostojen siistiyttä madaltaa suuri määrä ylimääräistä poiskommentoitua koodia.

3.3.3. Tiedostot eivät sisällä kovakoodausta

Hyvin vähän kovakoodausta (muutama sähköpostiosoite). Erääseen Highcharts-tiedostoon jouduttiin koodaamaan suoraan.

3.4. Moduulien roolijako

3.4.1. Järjestelmä noudattaa selkeää arkkitehtuuria, jossa ohjelman osilla on selvä roolijako

Toteutuu enimmäkseen hyvin (MVC-mallin mukaisesti). Poikkeuksena datankäsittely, jossa roolijako epäselvä, ks. kohta 3.4.3.

3.4.2. Palvelin-selain-kommunikointi on mahdollisimman suoraviivaista

Enimmäkseen toteutuu. Ajaxia voisi hyödyntää enemmän, esim. editorissa tiedon tallennuksessa kun pysytään editorissa. Vastaajan näkymässä käytetty piilotetun muuttujan lukeminen lisää käsittelyyn sekavuutta ja ylimääräisiä askelia.

3.4.3. Datankäsittely

3.4.3.1. Datankäsittely näkymien välillä on yhdenmukaista

Ei toteudu. Näkymistä tuleva syöte käy läpi näkymäkohtaisesti erilaisia käsittelyaskelia.

3.4.3.2. Datankäsittely muodostaa oman loogisen kerroksensa

Ei toteudu. Syötteenkäsittely on hajautettu sekalaisesti selainkoodiin, käsittelijöihin ja malleihin. Myös näkymissä erilaisia en- ja dekoodausaskelia.