TUIMA –projektin loppuraportti

Opetus- ja kulttuuriministeriö
Yleissivistävän koulutuksen ja varhaiskasvatuksen osasto
Kehittämispäällikkö Jarkko Moilanen

16.12.2016

Sisällysluettelo

Johdanto ja projektin tavoitteet

Valtiovarainministeriö päätti rahoittaa 3.6.2015 TUIMA-projektia 498 750,00 €:lla (VM/988/02.02.03.09/2015) hyväksyttyään esitetyn projektisuunnitelman.

Suunnitelmassa eritellyt lopputulokset ovat:

1. Toteutussuunnitelma (Liite 1)
2. Helposti useisiin palveluihin integroitavissa oleva sosiaalisen median tunnuksia hyväksikäyttävä tunnistusratkaisu ja kevyt roolienhallinta ratkaisu. Kokonaisuus on integroitavissa kansalliseen tunnistuksenohjaukseen ja tukeutuu sopivilta osin rooli- ja valtuutuspalveluun. (Liite 2)
3. X-Roadin/palveluväylä osalta eri osapuolten yhteiskäytössä oleva Github varasto (repository), jossa integrointimallit ja avoimen lähdekoodin koodiesimerkit (kuvattu jäljempänä)
4. Viro-Suomi akselilla tapahtuvan avoimen lähdekoodin päälle rakentuvan yhteistyön kuvaus ja alustavat toimintamallit. Kuvaus koskee käytännön läheistä työtä, ei sopimuksellisia seikkoja. (Liite 3)
5. Selvitys ohjelmistokomponenttien lisenssien hallinnan nykytarpeista ja toimintaehdotukset jatkotoimia varten yhteensopivuuden maksimoimiseksi ja riskien minimoimiseksi. Selvitys tehdään yhteistyössä VTT:n kanssa, jotta voidaan tarkastella tämän prosessin sovittamista osaksi avoimen tuotteen hallintamallia. (Liite 4)
6. Loppuraportti (tämä dokumentti)

Projektin organisointi

Koska hanke kosketti monia organisaatioita ja tietojärjestelmien kehittämisprojekteja ja linkittyy osaksi kansallisen arkkitehtuurin kehittämistä, erityistä huomiota kohdistettiin sujuvaan tiedonvaihtoon. Hankkeen osaprojektien päälliköt vastasivat siitä, että sopivat sidosryhmien kanssa tiedonvaihtotavoista ja säännöllisistä tapaamisista. Hankkeen vastuullinen vetäjä edellytti osaprojektien vetäjiltä kuukausittaisia yhteenvetoja saavutetuista ja suunnitelluista töistä. Iso osa hankkeen viestintää tapahtui Digipalvelutehdas ja Joint X-Road Slack työtiloissa.

TUIMA kokonaisuuden koordinointi:

1. Kehittämispäällikkö Jarkko Moilanen (OKM)

Tunnistautumisen kehittäminen:

2. Projektipäällikkö: Manne Miettinen (CSC)

Integraatiomallien kehittäminen ja yhteistoiminnan kuvaaminen ja mallintaminen:

3. Projektipäällikkö: Taija Björklund (Sampo Software Oy)

Yhteentoimivuusmallin kehittäminen:

4. Projektipäällikkö: Salum Abdul-Rahman (Gofore Oy)

Koko hankkeen osalta pidettiin kaksi ohjausryhmätapaamista, joissa oli suunnitelman mukaisesti mukana VRK:n ja VM:n henkilöstöä OKM:n virkamiesten lisäksi. Varsinaista ohjausryhmää ei perustettu, koska sitä ei nähty tarpeellisesti. Projektin seuranta tehtiin kuukausittaisilla raportoinneilla hankesalkkuun.

Yhteenveto toteutuksesta ja saavutetut tavoitteet

TUIMA-projekti eteni suunnitelman mukaisesti ilman merkittäviä riskien toteutumista tai muutoksia. Vain yllä mainittu kohta 3 (X-Roadin/palveluväylä osalta eri osapuolten yhteiskäytössä oleva Github varasto) jäi sisällöltään ohueksi, koska eri osapuolten välisessä sujuvassa yhteistyössä

todettiin haasteita ja iso osa energiasta jouduttiin kohdentamaan aluksi yhteisten toimintamallien löytämiseen sen sijaan että olisi päästy heti käsiksi tuottavaan työhön. Toisaalta tehty työ edisti merkittävästi Suomen ja Viron välistä yhteistyötä Palveluväylän / X-Roadin suhteen. Samalla pohjustettiin palveluväylän ympärille syntymässä oleva yhteisö ja sen roolia.

Projekti alitti budjettinsa reilusti (liite 5). Tähän lienee kaksi syytä: epäonnistunut budjetointi ja onnistuneet hankinnat. Alla yhteenveto alaluvuittain tuloksista. Lisäksi osaprojekteista on omat seikkaperäiset raportit ja selvitykset liitteinä.

Tunnistusratkaisu - MPASSid

Suunnitelmissa ollut sosiaalisen median tunnuksia hyväksikäyttävä tunnistusratkaisu ja kevyt roolienhallinta ratkaisu eteni jopa hieman pidemmälle kuin suunniteltiin. Ratkaisusta käytetään nyt nimitystä MPASSid, jolla on myös promootiosivut verkossa http://mpass.fi. Ratkaisun lähdekoodit löytyvät Githubista: https://github.com/Digipalvelutehdas/MPASSid-proxy. Osana kehitystä on tehty VM/VRK:n pyynnöstä Open ID Connect proof of concept testi, joka ei kuulunut suunnitelmaan mutta oli tarpeellista sekä MPASS että suomi.fi-tunnistamisen palvelun kehityksen kannalta.

MPASSid eteni beta-ohjelma vaiheeseen tammikuussa 2016. Beta-ohjelman tarkoitus oli integroida noin 100 000 loppukäyttäjää ja 5-10 palveluntarjoajaa ratkaisun piiriin ja siten saada laajapohjainen ja kattava asiakaskokemukseen perustuva palaute ratkaisun toimivuudesta. Maaliskuussa ratkaisun piirissä oli noin 50 000 käyttäjää, jotka olivat lähes kokonaan yleissivistävän opetuksen piirissä olevia oppilaita ja opettajia. Nyt saatujen kokemusten perusteella voidaan todeta että MPASS integrointi kuntien järjestelmiin on kustannustehokasta ja nopeaa vieden parhaimmillaan vain tunnin ja pahimmillaan viikon. Pitkittynyt integraatioaika johtui muusta kuin tekniikasta (esim palomuurin avaus kunnissa).

MPASSid –ratkaisu on saanut erittäin suuren kannatuksen niin opetuksenjärjestäjien ja palveluiden tuottajien keskuudessa. Ei ole perusteetonta arvioida että avoimeen lähdekoodin perustuvasta MPASSid:sta voi tulla palvelu, joka helpottaisi noin 1,5 miljoonan kansalaisen elämää opetussektorilla ja laajemminkin mikäli MPASSid tai sen komponentteja integroidaan osaksi kansallista Suomi.fi-tunnistuspalveluun.

Projektin aikana on tullut ilmi, että suomi.fi –tunnistuspalveluun ollaan mahdollisesti ottamassa MPASSid ratkaisusta sosiaalisen median tunnuksien avulla tapahtuvan tunnistamisen mahdollistavat komponentit (”Suomi-Some”). VRK:n kanssa käydyissä keskusteluissa on tullut ilmi ettei MPASS ratkaisua ja suomi.fi-tunnistusta voida liittää yhdeksi kokonaisuudeksi, kuten ehkä aiemmin oli ajatus. Syyt edellä mainittuun eivät ole teknisiä. HAKA-verkostossa on noussut mielenkiintoa MPASS Suomi-Some komponentteihin. Sen sijaan kevyt roolienhallinta ei ole käyttökelpoinen suomi.fi –tunnistuspalvelussa vaan se tulee erottaa omaksi osuudekseen, jonka hallinnasta ja kuluista vastaa erikseen hallintamallissa määritelty taho. Roolienhallinta palvelee lähinnä opetussektoria. Kokonaisuuden hallintaa varten tulee laatia avoimen tuotteen hallintamalli. Osittain tästä syystä MPASSid jatkokehitys irrotettiin TUIMA-projektista maaliskuussa 2016 ja kehityksen kulut ovat sen jälkeen menneet OKM:n resursseista.

X-Road integrointimallit ja koodiesimerkit

X-Roadin/palveluväylä osalta eri osapuolten yhteiskäytössä oleva Github varasto (repository) on tehty ja josta löytyy integrointimallit (5 kpl) ja muutama avoimen lähdekoodin koodiesimerkki niiden soveltamisesta. Osana tätä työtä on käynnistetty myös Suomen ja Viron yhteinen avoin Joint X-Road yhteisö, jolle on tehty myös portaali: http://jointxroad.github.io/ (community.xroad.eu). Vähäinen määrä integraatiomallien käyttöä havainnollistavia koodiesimerkkejä johtuu osittain siitä

syystä että muiden organisaatioiden palveluväylään liittymistä ohjaavat toimijat (VRK) ja yhteisö (Joint X-Road community) eivät ole löytäneet toisiaan. Toisin sanoen, kun organisaatiot ovat integroineet palveluitaan palveluväylään, ei ole osattu hyödyntää yhteisön tämän projektin puitteissa tekemää työtä. Yhteistyötä eri osapuolten välillä ja yhteiskäytössä olevien resurssien hyödyntämistä tulisi ja on mahdollista parantaa paremmalla viestinnällä ja organisaatioiden sekä yhteisön törmäyttämisellä.

Viro – Suomi yhteistyön kuvaaminen ja toimintamallit

Viro-Suomi akselilla tapahtuvan avoimen lähdekoodin päälle rakentuvan yhteistyön kuvaus ja alustavat toimintamallit tehtiin suunnitellusti.

Keskeiset toimenpide-ehdotukset

1. yhteisö tarvitsee yhteisömanagerin,
2. lisää mielenkiintoista julkista materiaalia X-Road integraatioista,
3. sujuvampaa viestintää tehdyistä päätöksistä ja roolijakoa ns virallisten kehitystiimien ja yhteisön välillä

Tarkempi raportti liitteenä 3.

Ohjelmistokomponenttien lisenssien hallinta

Projektisuunnitelmassa oleva selvityksen tekeminen ohjelmistokomponenttien lisenssien hallinnan nykytarpeista ja toimintaehdotukset jatkotoimia varten teki Gofore Oy. Selvitystä hankittiin Gofore Oy:ltä siksi, että siellä tiedettiin olevan Suomen paras asiantuntija tämän selvittämisen tekemiseen.

Keskeiset tulokset

Haastattelujen perusteella sekä asiakkaan, että toimittajien puolella on epäselvää, mitä vaikutuksia ohjelmistosta löytyvästä tekijänoikeusloukkauksesta olisi. Pahimmat seuraukset olisivat palvelun tai ohjelmiston poistuminen käytöstä oikeuden tai palveluntarjoajan toimesta. Toiseksi suurimmat vaikutukset olisivat oikeuskäsittelyn tai korjauksen vaatimat työmäärät, sekä näiden vaikutukset asiakkaan ja toimittajan toimintakykyyn. Suomalaisessa oikeuskäsittelyssä määrätyt korvaukset tekijänoikeusloukkauksesta ovat pienet, ellei kyseessä ei ole komponentin levittäminen, mitä julkishallinnon hallinnon palveluissa ei yleensä tehdä. Poikkeuksena on tietysti selaimessa ajettava koodi, mutta selaimessa ajettava koodi on yleisesti lisensoitu sallivilla lisensseillä, eikä komponenteilla tai kehyksillä ei ole erillisiä kaupallisia versioita, joiden perusteella komponentin hinnan voisi määrittää.

Haastatteluissa kävi ilmi, että monet haastatelluista pelkäävät mahdollisen rikkomuksen vaikuttavan negatiivisesti maineeseen. Mainevaikutukset eivät kuitenkaan ole julkishallinnon palvelukehityksessä vakava seuraamus, sillä suurimmalla osalla palveluista ei ole kilpailijoita. Tällöin asiakkailla ei ole mahdollisuutta vaihtaa kilpailevaan tuotteeseen. Lisäksi myöskään avoimen lähdekoodin yhteisöihin ei ole yleensä ole suhteita, joita huono maine voisi pilata.

Korkeammalla tasolla ongelmana julkishallinnossa on, että asiakas ei aina osaa ottaa omistajuutta lisenssien hallinnasta. Lisenssien hallinnassa ensimmäinen kulmakivi on se, millä lisenssillä oman palvelun lähdekoodi julkaistaan vai julkaistaanko olleenkaan. Asiakas on aina liiketoiminnan asiantuntija ja liiketoiminnan asiantuntijaa tarvitaan, koska oman koodin julkaisuun käytettävän lisenssin pitää sopia liiketoiminnan tavoitteisiin. Mikäli julkaisuun käytettävää lisenssiä valittaessa on tiedossa, että jokin komponentti tai kehys on olennainen osa tuotettua palvelua, on myös näiden lisenssien yhteensopivuus otettava huomioon.

Mikäli asiakas ei osaa määrittää projektille lisenssien hallinnan tarpeita on täysin kehittäjien oman aktiivisuuden varassa, miten asiat hoidetaan. Vaikka toimittaja haluaisikin hoitaa lisenssien hallinnan kunnolla ja sitä sopimuksessa vaaditaan, ei jokaiselle kehittäjällä ole kompetenssia ottaa avoimen lähdekoodin lisenssien ehtoja huomioon työssään ilman ohjausta. Haastatteluissa kävi myös ilmi, että asiakkaat epäilevät, että tekijänoikeusloukkauksesta kärsisi sopimuksen ansiosta vain toimittaja, joka joutuisi tekemään korjaukset, maksamaan sopimussakkoja ja saattaisi menettää sopimuksen. Tosiasiassa nämä vaikutukset tulisivat voimaan vasta sen jälkeen kun seuraamukset olisivat tulleet asiakkaalle.

Lisensseihin liittyvien tekijänoikeusloukkauksen suurimmat riskit liittyvät oikeuskäsittelyehtoon, nimeämisehtoihin ja SaaS-ehtoon. Mikäli lisenssiloukkauksen oikeuskäsittely tapahtuu jossain muualla kuin Suomessa, oikeuskäsittelyn kustannuksia sekä mahdollisen korvauksen määrää on melkein mahdotonta arvioida. Osa nimeämisehdoista kieltää tekijänoikeudenhaltijan tietojen julkaisemisen palvelussa, kun taas osaa vaatii sitä. Esimerkiksi BSD-lisenssin eri versioissa ovat nämä vastakkaiset ehdot. Tämän vuoksi nimeämisehdot on käytävä tarkasti läpi ja tekijänoikeudenhaltijoiden nimet on julkaistava sopivassa paikassa niin vaadittaessa. Suurin osa julkishallinnon ohjelmistokehityksestä on verkkopalvelujen kehittämistä, jolloin muokattua koodia ei jaeta eteenpäin ja esimerkiksi GPL-lisenssin vastavuoroisuusehto ei tule voimaan. SaaS-ehto kuitenkin on voimassa silloin kun ohjelmisto tarjoaa julkista palvelua. SaaS-ehto liittyy kuitenkin yleensä tunnettuihin vastavuoroisiin lisensseihin, joten se on helppo tunnistaa. Tällöin SaaS-ehdon sisältävän mikropalvelun tai palvelun koodi pitää julkaista yhteensopivalla avoimen lähdekoodin lisenssillä, mitä ei haastattelujen perusteella olla sisäistetty. Tämä on merkittävä kehityskohde julkishallinnossa.

Toimenpide-ehdotukset

- Ensisijainen kehitysehdotus on avoimen lähdekoodin hallinnan koulutuksen lisääminen.
- oman lisenssiyhteensopivuustaulukon tekeminen julkishallinnolle.
- kansallisen FOSSology-palvelun kehitys.

Tarkempi raportti on liitteenä 4.

Projekti alitti budjettinsa

Projekti alitti budjettinsa reilusti (liite 5). Projektin hankinnat tehtiin hankintalain mukaisesti. Useimmiten käytettiin niin sanottua kevennettyä kilpailutusta. Projekti jaettiin osaprojekteihin ja toteutukset palasteltiin sopivan pieniksi kokonaisuuksiksi, jotta pystyttiin tekemään kohtuullisen ketteriä hankintoja ja pitämään kokonaisuus hallinnassa. Hankintamenoihin oli allokoitu 498 750,00€ ja toteutuneet hankintakulut olivat: 271 622,17€. Tämän päälle opetus- ja kulttuuriministeriön virkamiehet käyttivät virka-aikaa projektin hallinnointiin ja johtamiseen.

TUIMA – tunnistus, integraatio ja lisenssien yhteentoimivuus

Projektisuunnitelma

Versio 1.0

13.1.2017

Sisällys

Dokumentin versiohistoria

Versio	*Päiväys*	*Laatija*	*Muutoksen kuvaus*
V0.1	7.1.2015	J.M	Ensimmäinen versio
V0.2	8.1.2015	J.M	Täydennetty runko
V0.3	12.1.2015	J.M	Täydennetty vaiheiden tekstikuvaukset
V0.4	6.3.2015	M.M.	Muokattu tunnistusratkaisun osalta
V0.5	6.3.2015	M.M.	Korjattu Henrin kommenttien perusteella
V0.6	9.4.2015	J.M.	Lisätty KaPA palveluväylä näkemykset
V0.7	15.4.2015	J.M.	Tarkennettu hankkeen organisoitumista ja sisältöjä käy-

			tyjen palaverin perusteella.
V0.8	28.4.2015	J.M.	Tarkennettu MM kommenttien pohjalta
V0.9	8.5.2015	J.M.	Lisätty Ohry ja tarkennettu läpivientiä
V1.0	15.5.2015	J.M.	Viimeinen tarkistus ja kieliasun tarkistus

1. Projektin sisältö

1.1. Lopputulokset

Projektin tuloksia ovat:

- Toteutussuunnitelma
- Helposti useisiin palveluihin integroitavissa oleva **sosiaalisen median tunnuksia hyväksikäyttävä tunnistusratkaisu ja kevyt roolienhallinta** ratkaisu. Kokonaisuus on integroitavissa kansalliseen tunnistuksenohjaukseen ja tukeutuu sopivilta osin rooli-ja valtuutuspalveluun.
- **X-Roadin/palveluväylä** osalta eri osapuolten yhteiskäytössä oleva Github varasto (repository), jossa **integrointimallit ja avoimen lähdekoodin koodiesimerkit**
- Viro-Suomi akselilla tapahtuvan **avoimen lähdekoodin päälle rakentuvan yhteistyön kuvaus ja alustavat toimintamallit.** Kuvaus koskee käytännön läheistä työtä, ei sopimuksellisia seikkoja.
- **Selvitys ohjelmistokomponenttien lisenssien hallinnan nykytarpeista ja toimintaehdotukset jatkotoimia varten** yhteensopivuuden maksimoimiseksi ja riskien minimoimiseksi. Selvitys tehdään yhteistyössä VTT:n kanssa, jotta voidaan tarkastella tämän prosessin sovittamista osaks avoimen tuotteen hallintamallia.
- Loppuraportti

1.2. Projektin kuvaus ja kohde

Projekti koostuu kolmesta eri alaprojektista:

1. Alaikäisten tarpeet huomioon ottava tunnistusratkaisu,
2. Palveluväylän integrointimallien kehittäminen ja Viro-Suomi avoimeen lähdekoodin perustuvan yhteistyön kuvaaminen, ja
3. Ohjelmistokomponenttien yhteensopivuuden hallintamallin edellytysten selvittäminen ja sovittaminen osaksi avoimen tuotteen hallintamallia.

1. Alaikäisten tarpeet huomioon ottava tunnistusratkaisu

OKM:n EduCloud-projektissa täydennetään VRK:n johdolla kehitettävää kansallista tunnistautumisratkaisua siten, että kansalaisen vahvan tunnistamisen rinnalle kehitetään heikomman tunnistamisen komponentit, jotka ovat tarpeellisia ja riittäviä monissa käyttötarkoituksissa mm. perusasteen oppimisalustoissa. Tavoite on, että tuotantotasoista ratkaisua pilotoidaan vuoden 2015 lopussa.

Ratkaisu työstetään mahdollisuuksien mukaan yhteistyössä STM:n kanssa, siten että vetovastuu on OKM:llä. Ratkaisu tulee olemaan itsenäinen palvelu, jonka voi tarvittaessa ottaa pienellä vaivalla käyttöön myös muilla sektoreilla. Oletusarvoisesti ja mahdollisuuksien mukaan käytetään avoimen lähdekoodin komponentteja. Myös avoimet rajapinnat ovat oletusarvoinen lähestymistapa. Mikäli joudutaan käyttämään suljetun lähdekoodin komponentteja tai palveluja osana kokonaisratkaisua, tehdään niiden päälle joka tapauksessa avoimet rajapinnat. Kaikki projektissa rahoitettu kehitystyö tehdään avoimella lähdekoodilla, jonka lisenssi määräytyy vuoden 2015 loppuun mennessä ja tulee olemaan yhteensopiva kansallisen arkkitehtuurin lisenssivalintojen kanssa. Lisäksi dokumentaatiot ja määrittelyt tehdään avoimen datan lisenssin alaiseksi (JHS 189 mukaisesti[1]).

Tunnistautumisen lisäksi palvelut tarvitsevat erilaisia tietoja käyttäjistä (käyttäjäattribuutteja), kuten nimi, sähköpostiosoite, mihin ryhmään tai organisaation osaan käyttäjä kuuluu, jne. EduCloud-projektissa perusasteen koulutukseen liittyvien käyttäjien attribuutteja on tarkoitus hakea Opetushallituksen keskitetystä Opintopolku.fi-järjestelmän rekistereistä[2] samoin kuin kansallisessa tunnistautumisratkaisussa haetaan kansalaisia koskevia attribuutteja Väestötietojärjestelmästä.

Palveluiden käyttövaltuudet (auktorisointi) on tehokkainta nivoa käyttäjäattribuutteihin, aina kuin se vain on mahdollista, sillä näin käyttövaltuushallinta voidaan automatisoida ilman, että jokaisessa palvelussa ylläpidetään erikseen käyttäjäkohtaista tietoa kuka saa tehdä mitäkin. Esimerkiksi on paljon tehokkaampaa tehdä sääntö, että kaikki käyttäjät, joilla on attribuutti ’opettaja’ saavat oppimisalustassa oikeuden antaa ja tarkastaa kotitehtäviä (ja käyttäjät, joilla on attribuutti ’oppilas’ eivät), kuin että sovelluksen pääkäyttäjä käy käyttäjäkohtaisesti raksimassa tämän oikeuden.

On ilmeistä, että kaikilla toimialoilla on tarve käyttäjäattribuuttien ja käyttövaltuuksien hallintaan, vaikka tarvittavat attribuutit ja roolit vaihtelevat luonnollisesti toimialasta ja palvelusta riippuen. KaPA-ohjelmaan kuuluvassa rooli- ja valtuuspalvelu -hankkeessa (Rova) kehitetään lisäksi kansallisella tasolla toimivaa rooli- ja valtuutuspalvelua. Eri toimialojen palveluiden vaatimien valtuutusten suhde Rovaan kaipaa vielä täsmennystä – ei liene tarkoituksenmukaista, että Rova palvelisi mikrotason valtuutusasioissa (”onko Rymättylän ala-asteen 4 C -luokan oppilaalla Ville Virtasella oikeus päästä Otavan matematiikan harjoitusmateriaaliin”), vaan keskittyisi laeissa ja asetuksissa määritelyihin rooleihin ja valtuutuksiin (”onko Matti Virtasella oikeus Ville Virtasen

[1] http://www.jhs-suositukset.fi/suomi/jhs189

[2] http://www.minedu.fi/export/sites/default/OPM/Tapahtumakalenteri/2013/05/Tietohallinto/Nokkanen_Erja__Tietohallinnon_yhteistyxkokous_22052013_OPH.pdf

huoltajana ilmoittaa Ville Virtanen ortodoksiseen uskonnonopetukseen"). Rovan kanssa tehdään tiivistä yhteistyötä kuten alustavissa yhteistyöpalavereissa on sovittu, jotta varmistetaan päällekkäisyyksien välttäminen ja yhteentoimivan ratkaisun rakentuminen.

EduCloud-projektissa konseptoitu ja pilotissa testattu roolienhallintamalli implementoidaan tuotantotasoiseksi ja dokumentoidaan siten, että se voidaan pienellä vaivalla siirtää toisille toimialoille. Näin toimimalla vähennetään eri tyyppisten ratkaisujen keksimistä. Kaikki projektissa rahoitettu kehitystyö (komponentit ja rajapinnat) tehdään avoimella lähdekoodilla, jonka lisenssi määräytyy vuoden 2015 loppuun mennessä ja tulee olemaan yhteensopiva kansallisen arkkitehtuurin lisenssivalintojen kanssa. Lisäksi dokumentaatiot ja määrittelyt tehdään avoimen datan lisenssin alaiseksi (JHS 189 mukaisesti[3]).

2. Palveluväylän integrointimallien kehittäminen

Kyseessä on integraatiomallien kehittämistä ja kommunikointia Suomessa eri sidosryhmille, jotta integraatio olisi joustavaa ja tapahtuisi mahdollisimman pienellä vaivalla. Erityiseksi haasteeksi tulee löytää erilaajuisia adapteriratkaisuja, joilla palveluväylän SOAP rajapinta sovitetaan yhteen markkinoilla yhä dominoivamman roolin ottaneisiin REST/JSON rajapintoihin.

EduCloud palvelun kehittäminen on alusta asti tapahtunut yhteistyössä virolaisten kanssa ja siksi onkin syntynyt paljon linkitystä ja keskustelua EduCloud (OKM), Information System Authority (RIA) ja Cybernetica välille. Tätä yhteyttä on järkevä käyttää etsittäessä järkevät integrointimallit, joilla erilaiset organisaatiot tuovat omat palvelunsa palveluväylään. Kehitettävät mallit hyödyttävät kumpaakin maata ja kokonaisarkkitehtuurin ylläpitoa.

Kyseessä on enemmän organisatorinen yhteistoiminta Viron ja Suomen välillä, jossa määritellään arkkitehtuurista toimintaa integroinnin saralla. Lisäksi tuotetaan yhteistyössä yksityisen sektorin kanssa (EduCloud Alliance jäsenet ja muut halukkaat) ja muiden julkisen sektorin toimijoiden (VRK, STM) koditason kokeilut ja esimerkit julkisesti kaikkien käyttöön avoimena lähdekoodina sikäli mikäli mahdollista. Joissakin integrointimalleissa saatetaan joutua käyttämään kaupallisia komponentteja ja silloin avoin lähdekoodi ei ainakaan kaikilta osin ole mahdollista. Tarkoitus ei ole toteuttaa REST/JSON tukea palveluväylään, vaan operoida adapter serveri osuudessa. Tulemme kuitenkin selvittämään reunaehtoja REST/JSON tuen toteuttamiseksi palveluväylän seuraavaan versioon.

Tuottamalla integrointimallit ja esimerkit helpotetaan organisaatioiden liittymistä palveluväylään käyttäen olemassa olevia rajapintaratkaisuja ilman että rajapintoja pitää tehdä uudelleen SOAP teknologialla. Ratkaisut yhtenäistävät käytäntöjä ja siten luovat vakautta ja kokonaisjärjestelmän turvallisuutta. Lisäksi on oletettavaa että mallien ja esimerkkien avulla syntyy kansallisella tasolla merkittäviä kustannussäästöjä.

[3] http://www.jhs-suositukset.fi/suomi/jhs189

3. Ohjelmistokomponenttien lisenssien yhteensopivuuden sovittaminen avoimen tuotteen hallintamalliin

Sinällään ei ole suurta eroa puhutaanko lisenssien hallinnan kohdalla avoimesta vai suljetusta lähdekoodista. Kummassakin tapauksessa lisenssejä tarvitsee valvoa arkkitehtuurimielessä ja vastaamaan kysymyksiin kuten esimerkiksi syntyykö jonkin komponentin lisenssistä riskejä suhteessa kokonaisarkkitehtuuriin. Suomessa on valtionhallinnossa käytössä kokonaisarkkitehtuuriin perustuva ajattelu- ja hallintamalli. Tästäkin lähtökohdasta ajatellen olisi erittäin tärkeää, että JulkICTLab ympäristönä tarjoaa projekteja vetäville tahoille jotain eväitä toteuttaa annettuja ohjeistuksia siten, että kehitys on jouhevaa, sisältää mahdollisimman vähän riskejä ja tuottaa parhaan mahdollisen tuloksen.

Sen lisäksi, että julkisessa hallinnossa voidaan hyödyntää avoimen lähdekoodin ohjelmistoja, antavat avoimet lisensointimallit myös mahdollisuuden levittää julkisin varoin toteutettuja ohjelmistoja vapaasti käytettäviksi. Suosituksessa (JHS 169[4]) todetaan muun muassa, että ohjelmiston hankinta kannattaa tehdä siten, että hankittava ohjelmisto voidaan tarvittaessa jakaa hallinnon ja kansalaisten keskuudessa. Kun avoimen lähdekoodin ohjelmistoon tehdään tai teetetään muutoksia, niin kehitystulokset tulee julkaista avoimen lähdekoodin lisenssillä, ellei kehitystyön sulkemiseen ole erityistä perustetta. Sama suositus varoittaa myös että markkinoilla on ohjelmistoja, joita markkinoidaan avoimina, vaikka niitä ei saa muokata tai jaella edelleen. Näin ollen tulee olla tarkkana lisenssien kanssa.

Yllä mainituista ja monista muista syistä johtuen JulkICTLabiin tulisi nopealla tahdilla määrittää työkalut, joilla voidaan vähentää riskejä jotka liittyvät eri komponenttien lisenssien yhteentoimivuuteen. Projektissa ei tuoteta lisenssienhallinnan työkaluja, vaan selvitys nykytarpeista ja mahdollisuuksista jatkotoimia varten. Erityisen tärkeäksi kyseinen työkalu muodostuu niissä projekteissa, joissa lopullisen tuotteistuksen tekee kaupallinen osapuoli. Toisin sanoen, referenssitoteutuksessa jota julkishallinto ja yksityinen sektori kehittävät yhdessä, voidaan toki käyttää komponentteja joiden lisenssi ei esimerkiksi salli kaupallista toimintaa. Kuitenkin viimeistään tuotteistusvaiheessa tarvitaan tieto riskeistä ja komponenttien lisenssien yhteensopivuudesta.

Vain pitämällä yllä lisenssikarttaa voidaan nähdä komponentit, jotka joudutaan korvaamaan esimerkiksi juuri tuotteistusvaiheessa. Tuotteistus pitää tehdä, jotta valtio voi hankkia palvelun. Lisäksi voidaan nähdä että tämä liittyy myös avoimen tuotteen hallintamalliin, jota VTT:n johdolla on kehitetty julkishallinnon hallintatyökaluksi avoimen lähdekoodin projekteihin.

Kansallisen palveluarkkitehtuurin (KaPA) hankkeissa tuotettu ohjelmakoodi tullaan julkaisemaan avoimena lähdekoodina ja toteutuksissa hyödynnetään myös valmiita avoimen lähdekoodin komponentteja. Hankkeissa tuotettavan lähdekoodin lisenssien sekä valmiiden komponenttien lisenssien yhteentoimivuuden varmistaminen on tästä johtuen ensi arvoisen tärkeää. Tällä hetkellä lisenssien yhteentoimivuuden varmistaminen täytyy tehdä manuaalisesti komponenttikohtaisesti, joka on aikaa ja lisensseihin liittyvää erityisosaamista vaativaa työtä. Tehtävään soveltuvien valmiiden työkalujen saatavuus JulkICTLa-

[4] http://www.jhs-suositukset.fi/suomi/jhs169

bin kautta vähentäisi merkittävästi lisenssien yhteentoimivuuteen liittyviä riskejä sekä yhteentoimivuuden arvioimiseen liittyvää työmäärää

1.3. Projektin tavoitteet

Projektin keskeisimmät tavoitteet ovat:

- Luoda erityisesti nuoret ikäryhmät ja heidän eri tarpeet huomioon ottavan tunnistautumisen osa-ratkaisu:
    - joka täydentää kansallista tunnistautumisratkaisua
    - heikkojen tunnistusvälineiden komponentit, joita voidaan käyttää osana erilaisia pilvipalveluja eri toimialoilla pienillä integrointikuluilla.
    - Konseptoitu ja pilotissa testattu käyttäjäattribuuttien ja roolienhallintamalli. Ratkaisu voidaan pienellä vaivalla siirtää toisille toimialoille.
- Saada aikaan integrointimallit palveluväylään:
    - Selvitetään järkevät integrointimallit muun muassa REST/JSON tekniikan kohdalla yhdessä virolaisten arkkitehtien kanssa yhteistyössä.
    - Julkisia online seminaareja, joissa käsitellään integrointimalleja ja tuetaan testaamista.
    - Vähintään yksi isompi päivän kansainvälinen työriihi/konferenssi yhdessä virolaisten kanssa.
    - Dokumentoidut integrointimallit ja minimissään 1 koodiesimerkki jokaisesta mallista Githubiin. Tehdään yhteistyössä virolaisten kanssa.
    - Testataan ratkaisut oikeilla julkisen ja yksityisen sektorin palveluilla käyttäen hyväksi EduCloud Alliance yhteisöä.
    - Pilotoitu REST rajapintakytkentä palveluväylään yhteistyössä VRK:n ja PRH:n kanssa.
- Selkiyttää komponenttien lisenssienhallintaa:
    - JulkICTlabissa olevien projektin käyttöön ohjelmistosuositukset tai pilvipalvelut joiden avulla voidaan ylläpitää karttaa arkkitehtuurin sisältämistä lisensseistä ja niiden suhteista toisiinsa. Kaupalliset työkalut ostaisi kukin projekti itse, ei siis JulkICT-Lab. Evaluoidaan vähintään 3 eri ohjelmistovaihtoehtoa.
    - Ohjeistus (lyhyt) miten yhteensopivuustarkistukset voisi suorittaa osana kehitystyötä.
    - Toimintamalli yhteensopivuuden maksimoimiseksi ja riskien minimoimiseksi.

1.4. Projektin omistaja

Projektin omistajana toimii: Jarkko Moilanen, joka vastaa sen läpiviennistä.

1.5. Toteutuksen osapuolet

Omat ja sidosryhmien osapuolet

Organisaatio	Rooli toteutuksessa
VM	Rahoitus, KaPA koordinaatio ja ohjaus
CSC	Vastaa tunnistautumisen ja roolienhallinnan toteuttamisesta
VTT	Kehittää avoimen tuotteen hallintamallia. Sisällytetään kehitettävät ratkaisut osaksi kyseistä mallia.
Sosiaali – ja terveysministeriö, STM	Otetaan mukaan määrittelyyn ja siten huomioidaan yhteiskunnallisia tarpeita laajemmin.
Opetushallitus, OPH	Otetaan mukaan määrittelyyn ja siten huomioidaan yhteiskunnallisia tarpeita laajemmin.
Väestörekisterikeskus, VRK	Vastaa palveluväylän kehittämisestä ja käyttöönotosta, keskeinen toimija integrointimallien kehittämisessä yhteistyössä virolaisten kanssa. Toteuttaa palveluväylän REST pilotin yhdessä PRH:n kanssa käyttäen hyväksi yhdessä määriteltyjä työkaluja ja käytäntöjä soveltaen. Vastaa kansalaisen tunnistuspalvelun ja roolien ja valtuustusten hallinnan palveluiden kehittämisestä kansallisessa palveluarkkitehtuurissa.
Cybernetica (Viro)	Kehittää X-Road ratkaisua. Keskeinen toimija integrointimallien kehittämisessä. Vastaa omista kuluistaan.
Information System Authority (RIA, Viro)	Koordinoi kansallisen arkkitehtuurin kehitystä Virossa. Keskeinen toimija integrointimallien kehittämisessä. Tuottaa integrointimallien kehikot. Vastaa omista kuluistaan.

Kansallinen palveluarkkitehtuuriohjelma

Hanke / projekti	Tarvitaan
Tunnistautuminen	Tulee varmistaa ettei tehdä päällekkäisiä tai yhteentoimimattomia ratkaisuja. Tapahtuu pitämällä säännöllisesti palavereja joissa jaetaan tietoa toteutuksista.
Rooli- ja valtuutuspalvelu	Tulee varmistaa ettei tehdä päällekkäisiä tai yhteentoimimattomia ratkaisuja. Tapahtuu pitämällä säännöllisesti palavereja joissa jaetaan tietoa toteutuksista.

2. Toteutus

2.1. Läpiviennin yleiset periaatteet

Projektin läpiviennistä vastaa Jarkko Moilanen.

Sosiaalisen median tunnuksia hyväksikäyttävän tunnistusratkaisun ja kevyen roolienhallinta -ratkaisun ohjaus kytketään Väestörekisterikeskuksessa käynnissä olevan kansalaisen tunnistamispalvelun ohjaukseen siten, että projektipäällikkö Manne Miettinen kutsutaan ohjausryhmän jäseneksi siksi aikaa, kun tässä projektisuunnitelmassa kuvatan kokonaisuuden toteuttaminen on käynnissä. Miettinen raportoi ohjausryhmälle työn etenemisestä, ja ohjausryhmä varmistaa että tehtävät ratkaisut ovat linjassa kansalaisen tunnistamisen palvelun kanssa.

Palveluväylän integrointimallien kehittäminen ja Viro-Suomi avoimeen lähdekoodin perustuvan yhteistyön kuvaaminen tapahtuu toimimalla tiiviissä yhteistyössä VRK:n palveluväylä-hankkeen kanssa. Yhteistyön ja tiedonkulkemisen varmistamiseksi Eero Konttaniemi kutsutaan TUIMA:n ohjausryhmään. Projektipäällikkö Taija Björklund Sampo Software Oy:stä raportoi edistymisen Palveluväylän ohjausryhmälle. Projektipäällikkö Björklund osallistuu Viro-Suomi yhteistyöpalavereihin, dokumentoi prosessia ja auttaa yhteistyön rakentamisessa toimien tiiviissä yhteistyössä palvelupäällikkö Petteri Kivimäen kanssa.

2.2. Aikataulu ja vaiheistus

Projektin toteutus on jaettu seuraaviin vaiheisiin:

INTEGROINTIMALLIT

1. Kehitettävien integrointimallien valinta
2. Bi-lateriaalinen Viron ja Suomen työhön osallistuvien osapuolten työryhmäpäivä. Muut maat jätetään tässä vaiheessa pois (muun muassa Ukrainasta ilmaistu kiinnostusta osallistua)
3. Mallien suunnittelua ja dokumentointia
4. Bilateriaalinen työryhmäpäivä
5. Mallien tarkennusta
6. Bilateriaalinen työryhmäpäivä
7. Mallien tarkennusta
8. Bilateriaalinen seminaari (tuloksien evaluointia)

YHTEENTOIMIVUUS MALLI

1. Selvitystyö nykytilanteesta ja tarpeista
2. Yhteensopivuustarkistus toimintamallin määrittäminen
3. Mallin pilotointi yhteen tapaukseen, EduCloud

4. Mallin sovittaminen osaksi avoimen tuotteen hallintamallia

TUNNISTUSRATKAISU

1. Perusasteen opetuksen palveluissa tarvittavan tunnistusvälineiden komponentit
2. Roolitietorajapinnan määritys perusasteen opetuksen palveluiden tarpeisiin
3. Em. roolitietorajapintojen implementointi Opintopolku.fi:n päälle
4. Tunnistuskehikkojen testausta ja vertailua
5. Valitun tunnistuskehikon konfigurointi
6. Tunnistuskehikon integrointi roolienhallintaan
7. Pilotointi

Projekti viedään läpi aikavälillä **1.5.2015 - 1.3.2016**

Projektin tarkempi vaiheistus ja läpivienti on kuvattu erillisessä GANTT kaavion sisältävässä taulukossa.

2.3. Vaiheet

Alla on kuvattu projektiin sisältyvien osaprojektien/tehtäväkokonaisuuksien päävaiheet ja niiden keskeiset tehtävät. Seuraavassa esitettävät työvaiheet noudattelevat pääsääntöisesti aikataulugraafissa (GANTT liite) esitettyä vaiheistusta ja kuvaavat näiden työvaiheiden sisällön.

2.3.1. Tunnistusratkaisu

Tässä vaiheessa kehitetään sosiaalisen median tunnuksia hyväksikäyttävä tuotantotasoinen tunnistusratkaisu. Kyseessä on siis heikon tunnistuksen mahdollistava ratkaisu, jota voidaan käyttää täydentämään kansallisen tason vahvaan tunnistamiseen perustuvaa ratkaisua.

Ratkaisu on helposti integroitava kokonaisuus, joka sisältää myös tuotantotasoisen, mutta kevyen roolienhallintaratkaisun. On oletettavaa että kansallisen tason roolienhallintaa tarvitsee laajentaa sektorikohtaisia ratkaisuja tehtäessä. Luomalla tässä kuvatun ratkaisun tai kehikon, voidaan sitä käyttäen ratkaista tai ainakin nopeuttaa huomattavasti monia eri sektoreittein roolienhallintaan ja heikkoon tunnistamiseen liittyviä kehitystarpeita, jotka tulevat ilmi kun palveluja kytketään palveluväylään.

Vaiheessa syntyvät tuotokset:

- Erityisesti nuoret ikäryhmät ja heidän eri tarpeet huomioon ottavan tunnistautumisen osa-ratkaisu, joka niveltyy tulevaan kansalaisen tunnistautumisratkaisuun.
- Sosiaalisen median tunnusten käytön sisältävä tunnistusratkaisumoduli,

jota voidaan käyttää osana erilaisia pilvipalveluja eri toimialoilla pienillä integrointikuluilla.

2.3.2. Integraatiomallit

Vaiheessa syntyvät tuotokset:

- Tuotetaan järkevät palveluväylän integrointimallit muun muassa REST/JSON tekniikan kohdalla yhdessä virolaisten arkkitehtien kanssa yhteistyössä ja lisensoidaan tuotos avoimen datan lisenssillä.
- Julkisia online seminaareja, joissa käsitellään integrointimalleja ja tuetaan testaamista.
- Vähintään yksi isompi päivän kansainvälinen työriihi/konferenssi yhdessä virolaisten kanssa.
- Dokumentoidut integrointimallit ja minimissään 1 koodiesimerkki jokaisesta mallista Githubiin lisensoituna kansallisen palveluarkkitehtuurin lisenssin kanssa yhteensopivasti (avoimen lähdekoodin lisenssi). Tehdään yhteistyössä virolaisten kanssa.
- Testataan ratkaisut julkisen ja yksityisen sektorin palveluilla käyttäen hyväksi EduCloud Alliance yhteisöä, PRH:n ja mahdollisten muiden toimijoiden kanssa.
- Tuotetaan kuvaus Viro-Suomi avoimen lähdekoodin käytännöllisestä yhteistyöstä jatkotoimia varten. Selvitys keskittyy kooditason yhteistyön ja yhteistoiminnan kuvaamiseen.

Vaihe on kaksi osainen: Ensimmäinen osa keskittyy palveluväylään integroitumisen helpottamiseen. Jotta palveluväylään pystytään liittämään jatkossa palveluita mahdollisimman nopeasti, helposti ja turvallisesti, tulee olla olemassa käytännölliset integrointimallit ja avoimen lähdekoodin toteutusesimerkit, joiden lisenssi sama kuin palveluväylän tai vähintään yhteensopiva.Tuotosten tallennus- ja hallinta-alusta Github.

Tässä vaiheessa yhdessä virolaisten kanssa luodaan palveluväylän integrointimallit. Työskentely tapahtuu itsenäisesti ja työryhmäpäivien yhteydessä. Työryhmäpäiviä järjestetään vähintään kolme. Vaiheeseen osallistuu järjestelmäarkkitehti (tai tarvittaessa useita) Viron RIA yksiköstä, palveluväylää kehittävästä Cyberneticasta, Väestörekisterikeskuksen palveluväylästä vastaava päällikkö ja muita tapauskohtaisesti tarpeelliseksi koettuja sidosryhmien edustajia kuten Vero ja Sosiaali-ja terveysministeriö sekä Opetushallitus.

Työskentely tapahtuu avoimen lähdekoodin periaatteiden mukaisesti avoimesti Github palvelussa. Vaiheen tulokset tallennetaan Github palveluun, joten kaikilla on mahdollisuus kommentoida ja osallistua työskentelyyn. Lisäksi näin voidaan taata että tuotokset ovat kaikkien saatavilla.

Toisena rinnakkaisena prosessina syntyy lisäksi raportti, jossa kuvataan nykyiset Suomen ja Viron väliset yhteistoiminnan prosessit erityisesti niiltä osin kun ne tapahtuvat REST gateway komponenttia, integrointimalleja ja EduCloud standardia kehitettäessä. Tulokset tallennetaan Github palveluun lisensoituna JHS 189 suosittelemalla avoimen datan lisenssillä. Raportissa keskitytään käytännönläheisen (koodi, määritykset, tapaamiset, organisointi, viestintä, tavat ja arvot toiminnassa) yhteistyön kuvaamiseen. Mahdollisuuksien mukaan tulee

verrata löydöksiä tutkimuskirjallisuudesta löytyviin malleihin. Lisäksi raportti sisältää alustavia korkeamman tason toimintamallien kuvauksia ja suosituksia. Raportti ja siihen liittyvä työ ei sisällä seuraavia aihealueita: Suomen ja Viron väliset sopimukset tai vastuut tai muut lainopilliset seikat.

2.3.3. Yhteentoimivuusmalli

Vaiheessa syntyvät tuotokset:

- Selvitys ohjelmistokomponenttien lisenssien hallintaan liittyvistä tarpeista JulkICTlab ja kansallisen palveluarkkitehtuurin tasolla.
- Alustava lisenssien hallintaprosessin sovitus osaksi VTT:n kehittämää avoimen tuotteen hallintamallia.
- Selvitys olemassa olevista ratkaisuista lisenssien hallinnan tekemiseksi ja suositukset jatkotoimista.

Vaiheen aikana selvitetään erilaisten markkinoilta löytyvien työkalujen soveltuvuus komponenttien lisenssienhallintaan avoimen lähdekoodin projekteissa. Lisäksi selvitetään tämän hetkiset tarpeet lisenssien hallintaan. Selvitys sisältää jatkotoimenpide-ehdotukset, työkalusuositukset ja esimerkkiprosessin siihen, miten eri komponenttien yhteentoimivuutta edistetään JulkICT labissa. Selvitys antaa perusteet jatkotoimille.

2.3.4. Loppuraportointi

Projektin suunnittelun / toteutuksen ja arvioinnin pohjalta dokumentoidaan tulokset, tuotokset sekä toteutuksen tärkeimmät tehtävät ja toteutukseen käytetyt resurssit loppuraportiksi. Loppuraportissa projektin vastuutaho kuvaa myös oman arvion toteutuksen onnistumisesta suhteessa työlle asetettuihin tavoitteisiin.

Vaiheessa syntyvät tuotokset:

- Loppuraportti
    - Arviointi (miten toteutus onnistui suhteessa tavoitteeseen)
    - Toteutus (vaiheet, tuotokset ja tehtävät)
    - Resurssien käyttö (henkilötyö, rahoitus)

3. Projektin organisointi

3.1. Projektin hallinnollinen organisointi

TUIMA kokonaisuuden koordinointi:

- Kehittämispäällikkö Jarkko Moilanen (OKM)

Tunnistautumisen kehittäminen:

- projektipäällikkö: Manne Miettinen (CSC)

Integraatiomallien kehittäminen ja yhteistoiminnan kuvaaminen ja mallintaminen:

- Projektipäällikkö: Taija Björklund (Sampo Software Oy)

Yhteentoimivuusmallin kehittäminen:

- Projektipäällikkö: Gofore Oy

3.2. Ohjausryhmä

Johtuen projektin lyhyestä kestosta ja pienehköstä koosta, ohjausryhmä pyritään pitämään mahdollisimman pienenä. Lisäki ohjausryhmä tulee kokoontumaan pääsääntöisesti online formaatissa. Projektin alku- ja loppupalaverit pyritään pitämään kuitenkin fyysisinä palavereina.

Projektin ohjausryhmään kutsutaan jäsenet seuraavista organisaatioista:

- Jarkko Leskinen, Kansalaisen tunnistamispalvelun ohjaus (VM/VRK)
- Matti Hiltunen, Kansallinen rooli- ja valtuutuspalvelu (VM/VRK)
- Eero Konttaniemi, Kansallinen palveluväylä (VM/VRK)
- Jari Porrasmaa (STM)

Ohjausryhmää johtaa TUIMA:n projektipäällikkö Jarkko Moilanen (OKM).

3.3. Viestintä ja tiedonvaihto

Koska hanke koskettaa monia organisaatioita ja tietojärjestelmien kehittämisprojekteja ja linkittyy osaksi kansallisen arkkitehtuurin kehittämistä, tulee erityistä huomiota kohdistaa sujuvaan tiedonvaihtoon.

Hankkeen projektipäälliköt ovat vastuussa siitä, että sopivat sidosryhmien kanssa tiedonvaihtotavoista ja säännöllisistä tapaamisista. Muissa tapauksissa vastuu on TUIMA:n hankepäälliköllä.Yhteistoimintapalavereista on tehtävä muistio joka julkisilta osin kirjataan Github:iin projektin repositoryyn.

Esimerkiksi heikon tunnistuksen projekti, jota johtaa Manne Miettinen tulee olla tiiviissä yhteistyössä RoVA –hankkeen ja kansallista tunnistuksenohjausta rakentavan hankkeen kanssa. Heikon tunnistuksen Github repository löytyy osoitteesta: https://github.com/educloudalliance/educloud-sso

3.4. Tuotokset

Oletusarvoisesti kaikki mitä tuotetaan ovat avointa tietoa ja avointa lähdekoodia. Tuotokset tulee sijoittaa versionhallinnan piiriin Github palveluun EduCloud Alliance organisaation alle. Mikäli projektilla ei vielä ole kyseisen organisaation alla omaa reposirotyä (varasto), tulee sellainen luoda heti. Lisätietoja voi kysyä hankepäälliköltä.

Tiedossa olevat hankkeeseen liittyvät repositoryt:

- Heikon tunnistuksen päärepository https://github.com/educloudalliance/educloud-sso

- X-Road REST gateway komponentti (sisältää myös integraatiomallit) https://github.com/educloudalliance/xroad-rest-gateway
- Lisenssien / yhteentoimivuuden repository https://github.com/educloudalliance/license-compliance
- Viro-Suomi yhteistyö https://github.com/educloudalliance/est-fin-cooperation-model (raportti nykyisistä käytännöistä)

4. Työmääräarvio

Projektin vaihekohtainen työmääräarvio henkilötyöpäivinä on seuraava:

Vaihe	OKM htp	VTT htp
Tunnistusratkaisu	100	0
Integraatiomallit	50	0
Yhteentoimivuus	50	30
Yhteensä	**200**	**30**

5. Kustannus–arvio

Liitteessä 3 Kustannusarvio.

TUIMA-projektin loppuraportti

Manne Miettinen, 6.6.2016

Tausta ja tavoitteet

CSC kehitti opetus- ja kulttuuriministeriön ohjauksessa toukokuusta 2015 lähtien maaliskuun 2016 alkuun asti alaikäisten tarpeet huomioon ottavaa MPASS-tunnistusratkaisua osana TUIMA tunnistus, integrointi ja maksaminen -projektia, jonka valtiovarainministeriö pyysi opetus- ja kulttuuriministeriötä toteuttamaan (VM/988/02.02.03.09/2015). Osaprojektin tavoitteena oli luoda nuorille ikäryhmille tarkoitettu tunnistamisratkaisun komponentit, joka täydentäisi kansallista tunnistamisratkaisua (suomi.fi tunnistaminen) heikoilla tunnistusvälineillä, ja joka olisi pienellä vaivalla siirrettävissä toisille toimialoille. Lisäksi tavoitteena oli suunnitella ja testata malleja käyttäjäattribuuttien ja käyttäjäroolien hallintaan.

Tavoitteiden saavuttaminen

Kokonaisuutena voidaan arvioida, että projekti oli erittäin tuloksellinen. Erityisesti toteutettua tunnistusratkaisuun integroitujen henkilöhakemistojen ja palveluiden määrä on ylittänyt odotukset. Tiivistettynä projektin konkreettiset tulokset ovat:

- MPASS-tunnistusratkaisu, johon kuuluu kevyen tunnistamisen komponentit
- Tunnistusratkaisun lähdekoodi on julkaistu GitHubissa[1]
- Oppijan avain tunnistusvälineen pilotointi
- MPASS-verkkosivusto, mpass.fi
- Open ID Connect -tekniikan proof-of-concept toteutus kansallisen tunnistusmallin luottamusverkostoa varten.
- Integroinnit Oulun kaupungin AD-hakemistoon, Hyvinkään kaupungin AD Azure palveluun, Kauniaisten Mäntymäen koulun Unelmakoulu-palveluun, Nettimoppi-palveluun, Peda.net-palveluun sekä Opinsys-käyttäjärekisteriin ja tunnistamiseen

Tavoitteiden toteutuminen kuukausitasolla on esitetty liitteen 1 taulukossa.

Talous

Aikavälillä 1.5.2015–29.2.2016 projektin kulut olivat 154 990,00 €. Jakautuminen kuukausittain henkilötyöhön, ostopalveluihin ja matkakuluihin on esitetty alla olevassa taulukossa.

	05-2015	06-2015	07-2015	08-2015	09-2015	10-2015	11-2015	12-2015	01-2016	02-2016	yht
Henk.t.	7,171	6,567	178	17064	16,259	7,775	17,134	13,481	13,374	12,776	111,779
Ostop.	2 508	3,207	395	0	900	1,995	2,2261	18,624	2,055	12,344	61,781
Matkak.	0	630		32	598	896	257	1,091	156	181	3,481
Yht.	**9,989**	**10,404**	**573**	**17,096**	**17,575**	**10,666**	**39,652**	**33,197**	**15,585**	**25,300**	**154,990**

[1] https://github.com/Digipalvelutehdas/MPASS-proxy

Liite 1. Tavoitteiden toteutuminen kuukausitasolla.

	Aikaansaannokset	Seuraavaksi	Ostot
05-2015	Testi-instanssin ylläpito kehityksen tukena	Shibboleth IdP v3 -pohjaisten moduulien kehitys: Tunnistus-ID laskenta -toteutus Auth Data API client -toteutus Proxy -toteutus Sosiaalisen median tunnistuksen toteutuksen aloitus Arkkitehtuurityön jatkaminen	Connector-komponentin toteutus (Haltu Oy) Attribuuttien vaatimusmäärittely (Sampo Software Oy)
06-2015	Testi-instanssin siirtäminen Shibboleth v3 -pohjaiseen toteutukseen Ensimmäiset toimivat versiot Tunnistus-ID laskenta, Auth Data API client ja Connector-komponentin ensimmäisen version toiminnallisuus valmis	Shibboleth IdP v3 -pohjaisten moduulien kehitys: Tunnistus-ID laskenta -testaus Auth Data API client -toteutus ja testaus Proxy -testaus Sosiaalisen median tunnistuksen toteutus	Connector-komponentin toteutus (Haltu Oy) Attribuuttien vaatimusmäärittely (Sampo Software Oy) Juristityö)
07-2015	-	-	-
08-2015	Ensimmäiset versiot valmiina Shibboleth IdP v3 -moduuleista Twitter, Facebook ja Google-kirjautumisille Ensimmäinen raportti potentiaalisten palveluntarjoajien käyttäjäattribuutti-vaatimuksista julkaistu	Shibboleth IdP v3 -pohjaisten moduulien kehitys: Sosiaalisen median tunnistuksen toteutus Valmistelu GitHub-julkaisua varten (koodien Javadoc-kommentointi, unit-testaus) Yle-tunnus integraation valmistelu Starsoft-yhteistyön valmistelu MPASS-testiympäristön rakennus cPouta-palvelimille	- Attribuuttien vaatimusmäärittely (Sampo Software Oy)
09-2015	Proxy-lähdekoodien siirto ja jatkokehitys Digipalvelutehtaan GitHub –hakemistossa Proxy-instanssien pystytys CSC:n cPouta-palveluun (mpass-proxy-test.csc.fi & mpass-social-test.csc.fi) Tunnistusvälineen valinnan toteutus Proxy-palvelulle Virhetilanne-käsittelyn toteutus Proxy-palvelulle OpenID Connect -tuen toteutuksen aloitus (järjestäytymi-	Shibboleth IdP v3 -pohjaisten moduulien kehitys: OpenID Connect -tuen toteutuksen jatkaminen Virhetilanne-käsittelyn toteutus Proxy-palvelulle Integraation aloitus kuntien/koulujen järjestelmiin (Oulu, Valkeakoski)	ECA Auth Data- ja Connector-komponenttien kehitys (Haltu Oy) Palveluntarjoajien vaatimusmäärittely (Sampo Software Oy) Oppijan Avain (Elisa Oyj)

	nen vahvojen tunnistusvälineiden luottamusverkoston teknisen ryhmän kanssa)		
10-2015	MPASS Promosivun vaatimusmäärittely tehty ja tarjouspyynnöt lähetetty Oppijan Avain -tunnistusmetodin vaatimien rajapintojen toteutus Auth Data ja Connector –komponenteille Suoran LDAP/AD -tuen tuki Proxy-komponentille sekä LDAP-testihakemiston pystytys kapasiteetti-testejä varten	Shibboleth IdP v3 -pohjaisten moduulien kehitys: OpenID Connect -tuen toteutuksen jatkaminen (myös vahvojen tunnistajien luottamusverkoston PoC-profiilin mukaisesti) Virhetilanne-käsittelyn toteutus Auth Proxy-palvelulle Integraation aloitus kuntien/koulujen järjestelmiin (Oulu, heidän ympäristöään vastaavan testiympäristön pystytys) Suoran LDAP/AD -tuen toteutus Auth Proxy ja Data –komponenteille Oikeaa käyttäjädataa käsittelevien komponentti-instanssien asennuksesta sovittu (mpass-(proxy/data/connector).csc.fi)	ECA Auth Data- ja Connector-komponenttien kehitys (Haltu Oy) cPouta -virtuaalikoneiden hallinta (Sampo Software Oy) Oppijan Avain (Elisa Oyj)
11-2015	MPASS Promosivun tarjouspyynnöt käsitelty ja projekti aloitettu Mediayhtiö B105:n kanssa MPASS Beta-tuotantopalvelimien pystytys cPouta-palveluun ja käyttöliittymien viimeistelyn.	- Shibboleth IdP v3 -pohjaisten moduulien kehitys (MPASS Proxy): - OpenID Connect -tuen toteutuksen jatkaminen (myös vahvojen tunnistajien luottamusverkoston PoC-profiilin mukaisesti) - Django Framework -pohjaisten moduulien kehitys ja käyttöliittymätyö (MPASS Data ja Connector) - Toiminnallisuudet tuleviin kunta-integraatioihin ja Oppijan Avain -tunnistusvälineelle - Integraatiotyön jatkaminen kuntien/koulujen järjestelmiin (Oulu, Dream School) - MPASS markkinointityö (kustantajat, kunnat (Savonlinna, Jyväskylä, Akaa/Toijala)	- ECA Auth Data- ja Connector-komponenttien kehitys (Haltu Oy) - cPouta -virtuaalikoneiden hallinta (Sampo Software Oy) - Oppijan Avain (Elisa Oyj) - MPASS promosivusto (Mediayhtiö B105 Oy)
12-2015	MPASS Promosivu julkaistu www.mpass.fi MPASS Beta-tuotantopalvelimien ensimmäinen kirjautuminen tapahtunut testipalveluun Oulun eduouka-tunnuksella -oikeat käyttäjätiedot välittäen	Shibboleth IdP v3 -pohjaisten moduulien kehitys (MPASS Proxy): OpenID Connect -tuen toteutuksen jatkaminen (myös vahvojen tunnistajien luottamusverkoston PoC-profiilin mukaisesti) Django Framework -pohjaisten moduulien kehitys ja käyttöliittymätyö (MPASS Data ja Connector)	ECA Auth Data- ja Connector-komponenttien kehitys (Haltu Oy) cPouta -virtuaalikoneiden hallinta (Sampo Software Oy) Oppijan Avain

	Vastaava kirjautuminen on mahdollista nyt myös Unelma-koulu-ympäristön tunnuksilla ja käyttäjätiedoilla OpenID Connect PoC demottu vahvojen tunnistajien luotta-musverkoston tekniselle ryh-mälle	Toiminnallisuudet jo tehtyihin sekä tuleviin kunta-integraatioihin ja Oppijan Avain -tunnistusvälineelle Integraatiotyön jatkaminen kunti-en/koulujen järjestelmiin (Oulu, Dream School, Office365) Markkinointityö kustantajille ja kunnille (Savonlinna, Jyväskylä, Akaa/Toijala, Hyvin-kää)	(Elisa Oyj) MPASS pro-mosivusto (Me-diayhtiö B105 Oy)
01-2016	Omatoiminen tunnusten hallin-ta mahdollista MPASS Connec-tor -palvelussa MPASS Connector -palvelun ulkoasun yhtenäistämi-nen suomi.fi -brändin kanssa	Shibboleth IdP v3 -pohjaisten moduulien kehitys (MPASS Proxy) Testaus-automaation lisääminen ja integ-rointi Travis CI -ympäristöön Django Framework -pohjaisten moduulien kehitys ja käyttöliittymätyö (MPASS Data ja Connector) Erityisesti testaus-automaation lisääminen ja integrointi Travis CI -ympäristöön Kokonaisen MPASS paikallisen kehitysym-päristön pystyttämisen automisointi (Vag-rant. Ansible, Salt -työkalut) Ensimmäinen versio valmiina Proxy-komponentin osalta, Data ja Connector tulossa	MPASS Data- ja Connector-komponenttien kehitys (Haltu Oy) Oppijan Avain (Elisa Oyj)
02-2016	Hyvinkää integroitu MPASS:iin Azure AD:n välityksellä Mikrolinna Oy:n Nettimoppi-sovellus integroitu MPASS:iin Esimerkki MPASS-sovelluksesta valmiilla integraatiolla testi-ympäristöön julkaistu GitHubis-sa	Shibboleth IdP v3 -pohjaisten moduulien kehitys (MPASS Proxy) Django Framework -pohjaisten moduulien kehitys ja käyttöliittymätyö (MPASS Data ja Connector) Kehitysvastuun siirtyminen Haltulta CSC:lle Kokonaisen MPASS paikallisen kehitysym-päristön pystyttämisen automisointi (Vag-rant. Ansible, Salt -työkalut) Ensimmäinen versio valmiina Proxy ja Data-komponenttien osalta, Connector tulossa Integraatioiden tuki Mikrolinna Oy, Nixu Oyj (SanomaPro:n toimittaja), Peda.net, Hyvinkää, Jyväskylä	Oppijan Avain (Elisa Oyj)

Joint X-Road Community

Report on a joint open source community and open development cooperation between Finland and Estonia

Author: Taija Björklund

March 2016

TUIMA project, Ministry of Education and Culture, Finland

CONTENTS

1 BACKGROUND AND SCOPE

1.1 WHAT IS X-ROAD?

X-Road is a technology or architecture for data exchange layer, which is used for ensuring secure Internet-based data exchange between information systems. X-Road has been developed in Estonia and it has been in use in the country since 2003. Republic of Estonia Information System Authority (Riigi Infosüsteemi Amet, RIA) estimates that over 900 organizations use X-Road, locally known as X-tee, in Estonia daily.[1]

In Finland, X-Road is being adopted into use as a part of the National Architecture for Digital Services (Kansallinen palveluarkkitehtuuri). The purpose of the project is to create a national digital infrastructure, aiming at facilitating information transfer between organizations, supporting service development and simplifying the transactions of citizens and companies with the authorities.[2]

There are four subprojects in the national architecture program. One of the subprojects, called National Data Exchange Layer (Kansallinen palveluväylä), is both a data exchange concept and a data exchange layer. As a data exchange layer, it provides a standardized and secure way to publish and use databases and services. Kansallinen palveluväylä will consist of several closed zones and a public communication layer, the development of which is based on the Estonian X-road solution.

1.2 JOINT X-ROAD COMMUNITY

The 10-year experience of using X-Road and building services around it in Estonia has proven that when different organizations are solving inter-connectivity issues, they tend to think that their issues are unique and cannot be solved in a standard way. It is true that some of the issues related to digitalization of services may be organizational rather than technical. Nevertheless, there seem to be common patterns across organizations regardless of whether we are talking, for example, about tax authorities or health organizations.

To detect and to solve issues that are potentially common for different kind of organizations, the people working for organizations providing X-Road support, RIA or Population Register Centre (Väestörekisterikeskus, VRK) should not be the only ones solving inter-connectivity issues and connections to X-Road. For this purpose, for creating a collaboration platform for anyone developing services on top of the architecture, a community called Joint X-Road has been initiated.

1.3 PURPOSE OF DOCUMENT

This report has been produced by the TUIMA project, which was started in May 2015. The project was coordinated by the Ministry of Education and Culture and financed by the Ministry of Finance in Finland.

The project consisted of three subprojects. One of them, called X-Road integrations, aimed at creating and promoting open source based documentation and code samples for the Finnish Palveluväylä. Prior to the establishment of the TUIMA project there had already been practical community based cooperation with Estonia around the X-Road solution, and the purpose was to continue the cooperation with

[1] https://www.ria.ee/public/x_tee/X-road-factsheet-2014.pdf
[2] http://vm.fi/en/national-architecture-for-digital-services

Estonia. Moreover, the goal of the project was to define and describe the open source based cooperation model between the two countries.

This document provides a theoretical framework on open development and open source communities and describes both the open source based cooperation and the initiation of Joint X-Road community.

2 Open source and Open Development Communities: The Theoretical Framework

2.1 Open Source

The history of free or open source software can be traced back to the 1960's - 1970's and several different phenomena of the era: counterculture, operating system-related development by researchers in scientific networks and software development in the defense research community (mainly ARPANET). The term free software was first coined and defined by Richard Stallman in 1986. According to Stallman, it is software anyone can freely use, study, modify, and share. The term 'free' mainly refers to free as in freedom or liberty rather than free in zero-cost / gratis.[3] The main freedoms of free software are the following:

0. "The freedom to run the program for any purpose."
1. "The freedom to study how the program works, and change it to make it do what you wish."
2. "The freedom to redistribute copies so you can help your neighbor."
3. "The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits."[4]

The software freedoms were institutionalized in the 1980's by the "copyleft" licensing of free software, mainly by GPL, the General Public License. Copyleft is a wordplay on copyright and a copyleft license ensures that the not only the software is free, but any versions of it will remain free. In other words, all derivative works must be distributed on the same licensing term as the original.

Since the term "free software" can be considered ambiguous (free as in freedom or free as in cost free) and a moral stance or ideological crusade started to be attributed to "free software", it made the term problematic for business use. Hence, there was a need for a more pragmatic and less ideological approach.

Open source software is a term promoted by Open Source Initiative (OSI), and it emphasizes open development process as an approach for creating better software. Quality and efficiency are achieved by transparency of development processes and building complex systems out of freely available building blocks. Open source software can even be considered a business model.[5]

According to the OSI, "open source is a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in."[6]

[3] Fogel 2005.
[4] http://www.gnu.org/philosophy/free-sw.html
[5] E.g. Kilamo 2014.
[6] https://opensource.org/about

To avoid taking any sides in the debate for an appropriate term for free / open / libre software, researchers quite regularly refer to the phenomenon as "FLOSS" (= Free/Libre/Open Source Software). In order to avoid using the cumbersome FLOSS acronym, which is generally not used outside academia, this report will refer to free or open source software as open source software.

Freeman[7] describes 4 phases of open source software and open development:

1. Development of UNIX by research-scientists at Berkeley and development of ARPANET in the networks of US armed forces
2. Free software Foundation and GPL
3. Internet-mediated open source software communities, particularly Linux
4. Emergence of hybrid open source communities, comprising of companies and volunteers, examples Open Office, Google

2.2 Governments and Open Source

There seems to be a fifth, more contemporary development for open source where governments and government agencies are adopting open source. For instance, open source has been a strategic choice for Brazil and recommended by Brazil's National Information Technology Institute and presidential cabinet since 2003. For an emerging country like Brazil, the economic motivation can be easily understood, but the strategy seems to be equally motivated by competition, diversification, bridging digital divide, employment and development of knowledge.[8] Other emerging BRICS[9] countries have had similar plans.

Also, the European Commission has defined an open source strategy for 2014-2017[10] and made a strategic choice to "ensure a level playing field to open source software when procuring new software solutions". Software developed by the Commission should have a public license, especially if it is intended to be used by others. The European Union has also created its own open source compliant license, EUPL.

As a recent development, the White House is soliciting public comments on a draft policy for releasing a portion of "federally funded" software to be release as open source. The draft policy published 10 March 2016 proposes a pilot program during which 20 % of either procured software or software developed in-house by federal agencies would be open source software. The draft mentions that releasing open sources software and allowing community participation "makes it easier to conduct software peer review and security testing, to reuse existing solutions, and to share technical knowledge".[11] According to the United States Chief Technology Officer this policy will not only provide cost savings, but also leverage innovation and allow serving the citizens better.[12]

> ***"Today, the Federal Government is already building some of our most important projects using open source, and more are launching all the time. -- For example, if you're struggling to make your mortgage or rent payments, there's an open source tool to find free housing counselors near you, built by the Consumer Financial Protection Bureau, using open data from HUD. Survivors of sexual assault can find resources and data on the open source site, NotAlone.gov, built in the open by 18F. If you're trying to figure out which***

[7] Freeman 2011.
[8] For example http://timreview.ca/article/250
[9] BRICS = Brazil, Russia, India, China, South Africa
[10] http://ec.europa.eu/dgs/informatics/oss_tech/index_en.htm
[11] https://sourcecode.cio.gov/
[12] https://www.whitehouse.gov/blog/2016/03/09/leveraging-american-ingenuity-through-reusable-and-open-source-software

college is right for you, the College Scorecard, the underlying data and API were all built with open source. If you're a journalist and want to find the best open data on an issue facing Americans, you can go to data.gov, which was built using open source, and in the open, using the platforms WordPress and CKAN."[13]

2.3 Open Development Method

The open development method is traditionally characterized (at least) by the following factors:

1. **Volunteerism:** Quite often the open source software projects are perceived to differ from other/proprietary software project by the fact that development team is comprised of volunteers rather than paid employees. This volunteerism or even "hacktivism" is often seen as a representation of a new working ethic or new division of labor. On the other hand, there is actually a large number of company-run open source projects developed by the company employees or hybrid open source projects and communities, where paid employees work alongside the volunteer developers.
2. **Motivation and self-selection of tasks:** One of the key foundations of open source development is self-selection of tasks or self-governance: the famous "scratch your own itch" principle by Eric Raymond.[14] It means that a developer becomes interested in a task and produces good quality software because the task in hand solves the developer's own problem or need.
3. **Transparency of activities** and **peer review**: Open software development exposes both the released software and its source code to decentralized peer review, which improves the software quality. Other artifacts, such as bug reports and documentation, and the development process itself is kept public and transparent as well.
4. Distributed **Internet-mediated teams:** Open development teams use tools that allow the development team members communicate and collaborate regardless of geographic location and time. Moreover, the contribution can take place simultaneously.
5. **Code reuse:** Since the source code is shared, which makes it publicly accessible and modifiable, it is possible to re-use and rewrite existing software modules and solutions. This can result in efficiency and quality improvement allowing others to improve an existing piece of software or to incrementally build a new, more complex software system out of the existing modules.

The most famous comparison between the open source development and the traditional / proprietary software development has been made by Raymond, who compares the traditional software development to building a cathedral - solemn work, careful crafting by talented group of specialists, who work in isolation and reveal nothing before the work is as finalized or as fault-free as possible. Open source development on the other hand is like a "great babbling bazaar of differing agendas and approaches", where the quality of software is not maintained by following standards but rather by numbers: releasing the software early and often and getting feedback potentially from hundreds of developers and users distributed around the globe.[15] [16]

In this report, we will adhere to the following two definitions of open development method or model:

[13] https://www.whitehouse.gov/blog/2016/03/09/leveraging-american-ingenuity-through-reusable-and-open-source-software
[14] Raymond 1998.
[15] Raymond 1998.
[16] Crowston & Howison 2005.

> ***"Internet-mediated geographically dispersed software development activity, in which the source code is publicly accessible, modifiable and redistributable to programmers."***[17]

> ***"A development method with a collection of processes and tools that allow wide variety of users and developers to contribute to the development of the software product."*** [18]

2.4 Open Development Communities

To understand what an open source or open development community is, we must first comprehend what constitutes a community. Kim defines a community as

> ***"a group of people with a shared interest, purpose, or goal, who get to know each other better over time."*** [19]

It is easy to see what would be the shared interest or the common goal for an open development community, if the community exists for the purpose of creating a software product or a system. However, if the open development team is Internet-mediated and might have only limited or no face-to-face contact, what makes it a group, where individuals "get to know each other better"?

Furthermore, it might be fairly difficult to determine the boundaries of such a community. If participation is based on volunteerism and individuals can thus "come and go as they please"[20], integration of volunteers into the community, motivation and building trust become interesting challenges.

Freeman refers to open development community as an "imagined community". [21] The same term has been applied to nations, where the sense of community or belonging to a particular group, the nation, has been largely built through printed press. Similarly, in the case of open development community, the sense of belonging to a particular group is built by "digital press", the written communication that takes place in community mailing lists and other discussions channels. Even software code can be considered to be speech. In a sense, the community becomes alive or exists through writing. Also, the contributions by community members have dependencies to the contributions of other members and that interdependency of the individuals' contributions further unites the community. Freeman also suggests that ties between members are formed because their skills complement each other. [22]

Kilamo[23] has suggested than an open community could be examined through five key aspects or elements:

- **People**: The group(s) of individuals that form the community, for example developers, testers, manager(s), users, followers. Potentially also people from outside the community, such as partners or related open source communities, could be considered.
- **Purpose**: The shared goal or interest that unites the individuals and attracts the people into community.
- **Product**: The piece of software or software system the community is developing.

[17] Freeman 2011.
[18] Kilamo 2014.
[19] Kim 2000.
[20] Freeman 2011.
[21] Freeman 2011.
[22] Freeman 2011.
[23] Kilamo 2014.

- **Policies**: Set of written and unwritten guidelines that direct the working process and the interaction in the community. This includes for example software and documentation licenses, code of conduct, coding conventions, governance structure.
- **Platform**: Project infrastructure, especially the software ecosystem, the community needs to be able to function.

Using the five key elements listed by Kilamo, we could come to a new definition of a open source community: **voluntary association of people, united by a shared purpose or goal, working collaboratively on a software product(s), guided by a set of policies and using a software platform as tool ecosystem for collaboration**.

2.4.1 Open source community structure

When we talk about open source or open development communities, the term "open" implies that the communities are open for anyone interested to join or to contribute.[24] And since one of the core principles is also transparency, they should be open for anyone to follow as well. If the community is both open for any individual to join or to follow, how to determine who belongs to the community? Are only active contributors part of the community? What about potential contributors, passive followers – or users of the software product?

Typically, there are several levels of participation in an open source community or project, and a distinction is made between core members and what Raymond calls the "halo"[25] or periphery, which would consist of more passive members or followers. The level of participation can also change from time: a peripheral member can become an expert or part of the core time, but at the same time a core member might also become a more passive member of the community.[26]

Traditionally, the structure of a healthy open source development community has been compared to a multilayered onion, where each layer represents a participation role coupled with level of engagement.[27] The most active and influential community members are the project leader and the core members, who oversee and coordinate the project and who have participated in the project for a long time. The core members usually contribute substantial amounts of code. [28]

[24] In practice, licenses and terms, even guidelines and conventions may determine, who can contribute and how.
[25] Raymond 1998.
[26] Freeman 2011.
[27] Kilamo 2014.
[28] Freeman 2011.

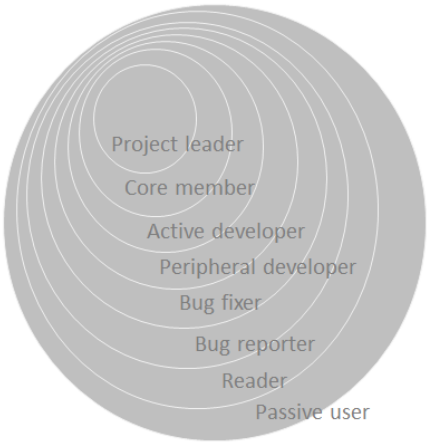


Figure 1: Onion model of an open source community. Based on Kilamo 2012.

With each outer layer, the amount of influence per role is smaller, but the amount of people in that role typically larger. Developers have been divided into active developers, peripheral developers and bug fixers. The more passive roles are active users who report bugs and potentially also provide use cases, and readers, who follow the community discussion forums and mailing list. The outermost layer is formed by the passive users. Even though some of the roles are more passive, they are equally important for community health and sustainability.[29]

2.4.2 From hackers to hybrid communities

Traditionally, open source community members and contributors were "competent user-developers", whose motivation to contribute was based on their own need ("scratching their own itch") rather than working for a salary and who developed software for themselves and for equally competent peers to use.[30] They referred to themselves as hackers – not in the sense of computer criminals, but in the original meaning of computers enthusiasts, expert programmers and bright problem solvers.[31]

The hacker ethics, "engag[ing] passionately and playfully, just for fun, in the creation of useful and socially valuable software" have been contrasted with the protestant work ethic, where the primary incentive is not playful enjoyment but salary and work is considered as an obligation.[32] These two poles (of a simplified view of human motives) need to coexist in the hybrid communities that work on company.-initiated or sponsored open source product.

In the hybrid communities, volunteers and paid employees work side by side to create the software products that are either entirely or partially licensed as open source. There are two hybrid types: 1. **firm-driven**, where the community was built around pre-existing proprietary software code or product, which the company decided to open and 2. **community-driven**, where the software product in question was originally developed by the community, but gained the interest of one or more companies.[33]

[29] E.g. Kilamo 2012, Kilamo 2014.
[30] Freeman 2011.
[31] Raymond 1998.
[32] Freeman 2011.
[33] Freeman 2011.

When working with a voluntary developer community, the company releasing the software should take extra care to make sure that there is equal access to both code and information and that guidelines and conventions are shared between the company and the community. There should be resources that are allocated to support the community. Mutual respect should be established between the two groups of developers.[34]

When the other party of the hybrid community is a public sector organization instead of a community, there can potentially be additional challenges. Public organizations may find true openness and transparency challenging and tend to be rigid in decision making. Fogel[35] has even commented that according to his experience "governments are less naturally suited to participating in free software projects than private-sector". He gives several reasons for this: more hierarchical organization structure, risk aversion, sensitivity to publicity and inflexible procurement.

> ***"Governments have labyrinthine and in certain ways inflexible procurement and employment policies. These policies can make it difficult for a government agency to be nimbly responsive in an open source development community."[36]***
>
> ***"Government agencies tend to be unusually risk-averse -- when development happens in public, the inevitable false starts and wrong turns are also public; if development were internal, no one else would know about it when those things happen."[37]***

2.4.3 Community building

In section 2.2 we introduced the well-known metaphors "the cathedral and the bazaar".[38] While initially viewed as two opposing poles of methods of working and purposes (proprietary software development vs. open source development), it has been suggested that both the cathedral and the bazaar could in fact be necessary evolutionary phases of an open source project. [39] Additionally, there is a third necessary phase, a transition phase, between the two.

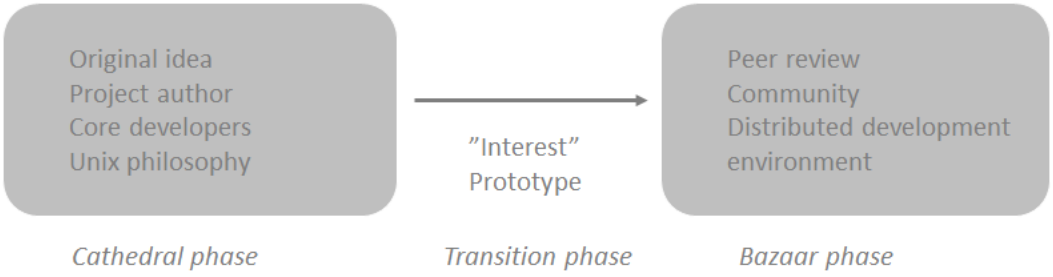


Figure 2: Lifecycle of open source development project. Based on Senyard & Michlmayr 2004.

Senyard and Michlmayr have argued that "[it would be] impossible to originate the project in the bazaar phase." The majority of the open source community members are not part of the core team, but are rather peripheral developers, bug fixers or users making bug reports and feature requests. For them to be

[34] E.g. Freeman 2011, Kilamo 2012.
[35] Fogel 2005.
[36] Fogel 2005.
[37] Fogel 2005.
[38] Raymond 1998.
[39] Senyard & Michlmayr 2004.

able to contribute to the project, the prerequisite is the existence of some kind of initial implementation they can use or test, or in other words, collaborate on. This initial version is usually created by a single individual or a small core team. Furthermore, the transition period between the cathedral phase and the bazaar phase determines whether the open software project will be successful or not. Most open software endeavors actually never go beyond the cathedral base.[40]

> ***"Hence, the collaborative effort starts after someone has written the first version. This is something that often goes unnoticed because the assumption is that open source is collaborative from the start."***[41]

In order to facilitate the collaborative effort by individuals outside the initial, or core, team, the version of the software product needs to be "approachable" enough for others to work on. The approachability and quality of the software product can be enhanced for example by providing accompanying documentation, making the installation or setup as easy as possible, and by improving the source code quality or readability.[42]

In addition to creating an approachable initial version of the software product or platform, it is necessary to prepare for collaboration in general. Open development will for example require a tool ecosystem that enables the Internet-mediated collaboration. Kilamo has listed some of the requirements for setting up community-based development:

> ***"Preparation of the product and the platform is needed long before moving into the open development model. Additionally policies that support collaboration and enforce trust are needed. The community needs to be kept an eye on through some way suitable for the community. Core participation and data support are needed. Addressing and encouraging motivational aspects are also essential."***[43]

Improving the product quality and its approachability, making sure that there are tools and processes available for the collaborators and ensuring that joining the community is as easy as possible can lower a threshold referred to as "hacktivation energy", which could be described the effort community newcomers need to make before they can contribute and potentially also start to get something out of the effort.[44]

Another major and less straightforward task is attracting members into the community: individuals interested in the project or software product and ultimately enthusiastic or willing enough to donate their time and effort to improve the product. Fogel, who has written a cook-book like guide for building open source communities, compares his guidelines about introducing a new open source project to distribution of medication:

> ***"[M]any of its recommendations would sound familiar to a health organization distributing medicine. The goals are very similar: you want to make it clear what the medicine does, get it into the hands of the right people, and make sure that those who receive it***

[40] Senyard & Michlmayr 2004.
[41] Freeman 2011.
[42] Kilamo 2014.
[43] Kilamo 2014.
[44] Fogel 2005.

know how to use it. But with software, you also want to entice some of the recipients into joining the ongoing research effort to improve the medicine."[45]

This task is often referred to as community building. Kilamo challenges the usage of term "building" because a community is not an inanimate object that could be built. Instead, she prefares to compare the establishment of communities to growing: "Growing is an activity where growth happens when certain prerequisites are met."[46]

"Coercive techniques don't work. If people are unhappy in one project, they will just wander off to another one. Free software is remarkable even among volunteer communities for its lightness of investment. Most of the people involved have never actually met the other participants face-to-face, and simply donate bits of time whenever they feel like it. The normal conduits by which humans bond with each other and form lasting groups are narrowed down to a tiny channel: the written word, carried over electronic wires. Because of this, it can take a long time for a cohesive and dedicated group to form. Conversely, it's quite easy for a project to lose a potential volunteer in the first five minutes of acquaintanceship. If a project doesn't make a good first impression, newcomers may wait a long time before giving it a second chance."[47]

3 Cross-Border Open Source Cooperation

3.1 X-Road in Estonia and in Finland

X-Road is a data exchange standard or data exchange layer, used for ensuring secure Internet-based data exchange between information systems. In Estonia, it has been the de facto data exchange standard in public sector since 2002. It has been one of the key building bricks of Estonian e-society. It has been estimated that in 2014 over 900 organizations used X-Road, locally known as X-tee, daily in Estonia.[48]

The development and maintenance of X-Road is the responsibility of the Infomation System Authority (Riigi Infosüsteemi Amet, RIA). RIA is a subdivision of the Estonian Ministry of Economic Affairs and Communications and it "coordinates the development and administration of the state's information system". The actual development of X-Road core software is carried out by a company called Cybernetica.

RIA took the initiative to establish an X-tee community in 2013. The purpose of the local community is to allow discussion and sharing advance information of X-Road development. The community has a mailing list of around 115 developers from approximately 60 organizations. The communication in the mailing list is predominantly one-way: from RIA to developers. Additionally, the community meets twice a year in semiofficial community meetings. Based on the meeting feedback, people feel more connected because of the meetings. Also, regular community meetings allow RIA to stay informed about the developers' concerns and ideas. [49]

[45] Fogel 2005.
[46] Kilamo 2014.
[47] Fogel 2005.
[48] https://www.ria.ee/public/x_tee/X-road-factsheet-2014.pdf
[49] Email and oral report from Heiko Vainsalu, RIA.

In 2013, prime ministers Jyrki Katainen and Andrus Ansip signed a memorandum of understanding[50]about cooperation in the in the field of information technology. As a part of memorandum, it was agreed that Finland would implement its national data exchange layer (Palveluväylä) on top of X-Road source code. The development of the national data exchange layer is the responsibility of Population Register Center (Väestörekisterikeskus, VRK).

3.2 Open Source Based Cooperation: Case REST Gateway

The decision to build the national digital architecture around X-Road faced certain criticism in Finland, because the data transactions in X-Road are based on the SOAP[51] protocol, which was considered legacy technology. The data consumers favored a more light-weight data transfer model called REST[52]. Both of the technologies allow API calls, but SOAP provides inbuilt support for formal contracts on data exchange format.

In 2013, the Finnish Ministry of Education and Culture initiated a project called EduCloud, the purpose of which was to support the development and distribution of e-learning material on national level. Since it would be mandatory to use Palveluväylä also in the educational sector, the project aimed to take the national digital architecture into account right from the beginning. The EduCloud project worked in cooperation with both public educational organizations and EduICT companies. Since the APIs provided by the EduICT companies were all REST/JSON APIs, the EduCloud project leader Jarkko Moilanen wanted to find a solution for connecting REST APIs to Palveluväylä.

Based on the initial investigation, Jarkko Moilanen wrote two blog posts[53] to the API Suomi community website in July 2014. The posts gained the attention of Andres Kütt from RIA, Estonia. Andres had been concerned with supporting organizations in joining the X-Road and decided to contact the author of the blog. This led to a few email and phone calls and eventually to an informal meeting in Helsinki. This meeting was a beginning for what later became open-source based cooperation between Finland and Estonia on REST support for X-Road and documentation for integration patterns.

At the same time, the need for REST support had also not gone unnoticed at VRK. The planning for component providing the REST support, called initially REST Proxy, was started in September 2014. Petteri Kivimäki, the technical lead of Palveluväylä, had access to X-Road source code and was able to study alternatives for creating the REST support.

In December 2014 these two separate paths were merged and there was a Skype meeting between Jarkko Moilanen, Andres Kütt, Petteri Kivimäki and representatives from Cybernetica. In the meeting, it was decided to initiate the implementation of the first version of an extension that would enable REST support. The component was named X-Road REST Gateway.

The first practical step was to create a shared Github[54] repository[55]. It was created under EduCloud organization, since the REST Gateway did not have an "official" home and EduCloud had been one of the

[50] https://www.riigikantselei.ee/valitsus/valitsus/et/uudised/Failid/2013/ICT_MoU_FI-EE_10dec2013.pdf

[51] SOAP = Simple Object Access Protocol. XML-based data exchange protocol that allows transferring API calls.

[52] REST = Representational State Transfer. An architectural model for implementing APIs.

[53] http://apisuomi.fi/2014/07/how-to-build-rest-support-to-x-road-laughing-version/, http://apisuomi.fi/2014/07/soap-rest-json-and-x-road-x-tee/.

[54] GitHub is a web-based service for hosting software projects. It provides features for code management, revision control and software team collaboration.

[55] https://github.com/educloudalliance/xroad-rest-gateway

initiators for the REST requirement. The establishment of a new repository did not create an obligation for anyone to participate in the development and no official agreements were made to formalize the Finnish-Estonian collaboration. Instead, the idea was to start a voluntary development project that would work according to open source principles.

The voluntary developers contributed code both in isolation and during developer days organized during spring 2015. The developer days were organized both in Tallinn and in Helsinki. The activities in the developer days consisted both of development and testing. Development was performed as team work, "pair programming" as a group around a common screen. The results of the development effort were verified in test environments for both Finland and Estonia. The way of working proved to be very fruitful and effective.

With just a handful of contributors, there has been significant development on the X-Road REST Gateway from its first version 0.1 published in January 2015 to the current 0.7 version. The component is listed as an additional component to X-Road on the RIA pages[56], but it is neither part of official X-Road source code nor part of the security server repository opened by VRK.

3.3 JOINT X-ROAD COMMUNITY

3.3.1 Birth of the community

The open source cooperation between Finland and Estonia around X-Road had been initiated by the REST Gateway development and creation of integration patterns. Tuima project was aiming to take the cooperation further by joint creation of openly available developer material, like code samples.

In June 2015, Petteri Kivimäki from VRK, Jarkko Moilanen from Ministry of Education and Culture and Taija Björklund from the Tuima project, had a meeting concerning the next steps and hopes for open source cooperation between Finland and Estonia in X-Road area. The purpose was to create an action plan for creating integration patterns, code samples and other developer material, and organizing developer workshops and other community gatherings. Based on the meeting, it seemed necessary to contact RIA for discussing items like licensing integration packages, cooperating on developer package creation and organizing community events.

The meeting concerning the creation of developer package was organized as an online meeting in September 2015. The meeting participants represented RIA, VRK and Tuima project: Heiko Vainsalu (RIA, X-Road Area Manager), Andres Kütt (RIA, Architect), Petteri Kivimäki (Systems Manager, VRK), Jarkko Moilanen (Senior Advisor, Ministry of Education and Culture), Taija Björklund (Project coordinator, Sampo Software Oy). In the meeting, it became clear that it is not enough just to create publicly available documentation and sample code, but rather it would make sense to initiate a developer community around X-Road. Furthermore, it should be a joint community for both Finnish and Estonian developers. Potentially, it could even become a global community.

> ***"During the process we have come to situation where creating joint X-Road developer community between the countries makes sense and it was considered as a natural step to take. This is a historical day."[57]***

[56] https://www.ria.ee/en/additional-components-of-x-road.html
[57] https://tarinoitadigitalisaatiosta.wordpress.com/2015/09/04/birth-of-joint-x-road-community/

One of the key elements when establishing a community are the platforms that allow collaboration and communication. Github had already been used as the collaboration platform during the REST Gateway development, but it was decided to open a new Joint X-Road organization in Github: https://github.com/jointxroad. For day-to-day communication, a discussion forum was established at https://jointxroad.slack.com[58]. Since this is a shared community, the language used in the forum is English.

The name that was selected for the community, Joint X-Road, refers to the name of the architecture in international use. Both Estonia and Finland have a local name for the data exchange layer, X-tee in Estonia and Palveluväylä in Finland. The selected name was considered to be understandable and relevant for both Estonian and Finnish developers, since the common denominator is the underlying technology.

The word "joint" was included in the name to distinguish the community from any (existing or potential) local communities, such as the Estonian X-tee community. The decision to name the community Joint X-Road later faced some criticism in Slack, because "it gives the impression that this is yet another useless umbrella community."

3.3.2 Community Manager

The newly created community needed someone to coordinate the activities and promote the community, in other words a community manager. Finland took the initial initiative of finding and hiring the community manager through the Tuima project. After a public bidding process, Karri Niemelä from BeAn Solutions was selected as the community manager.

The community manager's tasks have so far entailed promoting the community in events, activating the community, setting up community portal[59] and participating in the creation of developer material, such as the code samples. Community manager's role also entails following Slack discussion and questions tagged as xroad in Stack Overflow. The most relevant – or reoccurring - technical questions and discussion threads can be picked up and consolidated into technical patterns or training material.

3.3.3 Determining focus

After the initial meeting and the establishment of the community in September 2015, two other meetings were organized with the purpose of defining the purpose and focus of the community. While it was obvious that the community should not replace any official support or information channel, it was also clear that the community can allow a two-way discussion between the organizations developing X-Road and the organizations using it, whether public or private. This would enable RIA and VRK to stay aware of problems the consumers are facing and potentially get development ideas from the community.

More importantly, consumer organizations should be able to share their own know-how to others in the community and to learn from the experience of others, even reuse their existing solutions. Ultimately, the official organizations could hand-over some of the issues related X-Road for the community to solve. However, there seem to be no plans to open the source code for the X-Road core.

[58] The Joint X-Road Slack channel can be joined through https://joinxroadcommunity.herokuapp.com/
[59] http://jointxroad.github.io/

3.3.4 5 P's of Joint X-Road community

At the moment, the Joint X-Road community is still an embryo of a community. If we use the five elements or 5 P's proposed by Kilamo[60] to examine the current status of the Joint X-Road community, it becomes clear that some of the initial elements - the ones that are most easy to set up, like collaborating platform - are in place. At the same time, the more challenging elements are missing, like attracting and activating community members.

Element	FLOSS
Platform	Github, Slack, community portal. *The basic collaboration platforms and tools are in place.*
Product	Material that helps organizations connect to X-Road. Solution library of reusable software components, like REST Gateway. Developer package or training material: code samples, pattern descriptions, API guidelines. *Partially in place, but this requires more work so that the community portal could become a true knowledge base. Useful material is one of the key elements in attracting people to the community, particularly followers and readers.*
Purpose	To become a knowledge base and experience sharing platform for developers working in organizations using X-Road. *Community mission and purpose have been defined at least officially, but it remains to be seen whether this is a purpose that will attract community members.*
People	RIA, VRK, Cybernetica (company that develops X-Road). Developers and potentially also more business oriented people that work for public or private organizations that need or want to connect to X-Road. *Most of the current contributors are paid employees, working for either RIA or VRK or their subcontractors and/or have had a role in the initiation of the community.*
Policies	All software and documentation licenses should be public. For example, Integration patterns are licensed as cc-by-nc-sa 4.0. REST Gateway software license is the EUPL V.1.1. Community should be free for anyone to join. *Apart from that, code of conduct, governance structure, coding conventions or any other community policies are yet undefined.*

The Joint X-Road community can also be said to be in pre-bazaar phase and preparations are being made for opening up the community. The next steps should focus on:

- Creating truly useful material, whether it is software components or documentation
- Promoting the community: participating in events, writing blog posts
- Arranging community events
- Defining the community policies.

3.3.5 Community Slack channel

The Joint X-Road Slack channel was established in September 2015. It currently has 48 members:

- 6 RIA employees
- 3 VRK employees
- 1 employee of an Estonian ministry
- 3 employees of Finnish ministries or governmental agencies
- 1 employee of a ministry or a governmental agency outside Finland or Estonia

[60] Kilamo 2014.

- 4 employees of Estonian companies
- 13 employees of Finnish companies (2 people primarily working with community building)
- 3 employees of companies outside Estonia or Finland
- 3 persons from other organizations, such as educational institutions, interest groups
- 11 persons with unknown affiliation

3.4 QUESTIONNAIRE ON JOINT X-ROAD COMMUNITY

In February – March 2016 a questionnaire was sent to key people involved in setting up the Joint X-Road community. All the people selected as respondents had been involved either in initiating the community or in the open source based X-Road related cooperation between Finland and Estonia preceding the initiation of Joint X-Road community. The purpose was to find out their goals, motives and hopes for building or nurturing the community.

The questionnaire has been modified from a questionnaire created by Freeman[61]. Please see appendix A for questionnaire details. Altogether six questionnaires were sent out by email. Five people answered the questionnaire in writing, while one person was interviewed by Skype.

3.4.1 Community purpose

Community built around X-Road was first and foremost seen by respondents as a collaboration and know-how sharing platform. The purpose is to give support to organizations that need or want to join the local X-Road environments so that they would have a starting point other than relying on documentation only.

As a minimum, organizations responsible for maintaining the X-Road can use the community as a channel to easily reach X-Road consumers and developers, to hear about the problems they are facing, and to get development ideas from them. Ideally, the community members would share their experiences and code samples to help other organizations for example to set up security servers. Organizations developing and maintaining X-Road, such as RIA and VRK, cannot alone collect and document all possible real-world challenges and solutions as samples, patterns and instructions. Instead, providing guidelines and samples should be augmented by providing an opportunity and framework for sharing information and experiences. Being able to learn from others and to share the experiences can lower the barrier of adopting new technology.

Ultimately, as one of the respondents wrote, this could lead to “efficient and citizen friendly e-government”, since each organization does not have to re-invent the wheel when integrating to X-Road and either building or utilizing services around X-Road. Community that is so tightly related to national digital services can also provide a way for citizen participation.

3.4.2 Local vs. joint community

Even if the data exchange layer architecture is built around a common core technology, X-Road, it is possible that there are some national flavors in the implementation, i.e. in Palveluväylä and in X-tee. Why should there be a joint community for Finland and Estonia – or ultimately a global community – instead of local communities?

[61] Freeman 2011.

For a true open source community to work in long-term, it would need a significant number of members. Neither Finland nor Estonia alone is large enough to sustain such a community around one area of technology. Also, both Finnish and Estonian organizations and developers should be able to learn from one another. Estonians have over 10 years of experience of building an e-society and developing digital services around X-Road, while Finns have mature organizations and experience on administration and integration of large and complicated information systems.

On the other hand, the respondents were concerned, whether the need to use English instead of the national languages could constitute a barrier for joining or participating in the community. The experience from Estonia has proven that it is "difficult to get a community going as it is". One respondent suggested that maybe the language barrier should not be seen as a real obstacle. It is possible or even likely for national communities to exist within the joint community. This sounds like a more viable solution than having two separate national communities, which poses a danger of community needs diverging too greatly. That could potentially lead to one solution not being able to fulfil the needs of both communities.

3.4.3 Community members and growth

As potential members or contributors of the community, the respondents mentioned both organizations and individuals. Organizations can be either public or private, the key factor being the interest towards X-Road or rather the need to develop services that need to connect with X-Road. By individuals the respondents referred mainly to the employees working in the organizations. The individuals can work in different roles, for example "developers, DevOPS, architects, technically skilled process operators and owners".

The key motivating factor in joining the community is the amount of information and know-how available in the community, because information sharing and learning from the technical solutions made by others can reduce the need for "reinventing the wheel" for data exchange through X-Road layer. Other potential motivating factors are getting peer support, networking and influencing future development of the core technology.

Some of the respondents thought there could be a recruitment strategy for the community, because without a sufficient number of both active members and more passive followers the community does not take off. Others commented that rather than recruit the community should attract new members. The community can be attractive only, if it can provide solutions and information for the real-world problems the potential members are facing.

Even if there would be no recruitment strategy, active promotion of the community is required nonetheless: "You should be able to tell the potential new members, why it's worthwhile to join and what is the benefit both for the individual and the organization." This kind of communication or promotion should not only take place in official X-Road related events, but also in other developer or target group events.

One of the respondents mentioned that one of the key factors of promoting Joint X-Road community would be doing instead of talking. The community should accomplish and develop something together, something that is "relevant for the big picture", but not critical for the core technology. This would require courage from the governmental agencies to give room and have trust in the ability of the community to solve problems. To build trust in the community the agencies responsible for the X-Road development can start by giving small enough problems or "low-hanging" fruits for the community to solve and then allow the community to work on the problem. Participation by the community should be made visible.

The experience from the Estonian X-tee community has also shown the importance of arranging community events regularly. These events have contained both information of core development, but also possibilities to participate in workgroup activities. X-tee community has also been advertised as “unofficial communication channel”, that provides “insights and early information on X-Road core development”. The community events that have taken place in rural areas have been a success, but in many other ways the information flow is from RIA towards the developers. The respondents also mentioned that it takes a long time for the members to actually “buy-in”.

3.4.4 Valuable contributions

So far the open source based cooperation and the Joint X-Road community has produced architectural patterns, code samples and ready-made components. The respondents were asked what can be considered a valuable contribution in this community and the answers varied from code to “courage to ask questions”. Basically anything that has value for the other community members could be considered a valuable contribution.

On the other hand, the community is not a true community yet, but rather an idea or an embryo of a community. Therefore, one of the respondents pointed out that a valuable contribution at this point could also be anything related to building the community or promoting it. Such a contribution could for instance be a blog post that informs potential interested parties about the community and about its place and function in the big picture. There should be more than one person writing about the Joint X-Road community or else the communication does not have credibility.

3.4.5 Next steps

The respondents were also asked to evaluate the success of the open source cooperation so far. There has been some very positive feedback on the usefulness of the open source components, which were implemented very fast and by a handful of people. However, it would be good to attract more people who could bring a larger scale of understanding of different technologies.

Most of the open source development effort has been made by Finnish implementers and it would be good to see more contributors from Estonia. There is also room for improvement in know-how sharing and coming up with a larger number of code samples. On the other hand, it is already an achievement that there is a common intention between Finland and Estonia to try establishing a common, cross-border open source community around X-Road.

The next steps for the community would be creating more base material for the community: both code samples and documentation, like integration patterns and catalogue of reusable components. Also, there should be more regularity in the community activities. This could for instance mean regular Joint X-Road events or gathering. One of the respondents also commented that “opening the whole X-Road source code could have a positive impact on community growth”.

As the outcome of community building activities, there should be an active community that consists of people with varying backgrounds: developers and managers and paid employees and volunteers side by side. This would make the community a true asset for all actors involved. Community would provide the opportunity for contribution for people interested in X-Road or people interested in improving governmental digital services, or governmental sector in general.

4 CONCLUSIONS AND RECOMMENDATIONS

The Joint X-Road community is only in pre-bazaar phase. To make a successful transition to a true open community, it needs to offer interesting and relevant material about integration of services in X-Road. This material serves two purposes: attracts members into the community and serves as a starting point for collaboration so that the new members can potentially start from a more passive follower or reader role in the beginning.

The history of the local Estonian X-tee community has proven the value of community events, which allows the community members to feel "more connected to the community". Also, the experience from the Estonian community has shown that there is a long buy-in for the community. The Estonian local C-tee community has been established in 2013, but X-tee has been used as the data exchange architecture since 2002. It is mandatory to use X-tee for data exchange between (public) digital services and public information systems, but the community is still mainly a one-way information channel between RIA and community members. One can also speculate whether the community members think that they will need to accept whatever will be developed and do not expect to be able to influence the development or to be able to contribute to any area.

In Finland, it seems to be challenging to attract people to the community when basic functionality or key services are not in place in yet. Developers may not feel the urgency to learn about the X-Road integrations or solve potential integration challenges, if the services of the organizations they are working for or interested in are somewhere on the roadmap.

For a vibrant and open Joint X-Road community to emerge, some adjustment in the mindset of governmental organizations may also be required concerning openness and true transparency. Not only will it be necessary to ensure that the information flows between RIA, VRK and the community, but public organizations also may need to learn to justify why certain decisions are made instead of just stating that this is the decision and this is what you will need to live with.

Furthermore, open community members typically would like to be able to contribute to something that is either useful or interesting for them or that will provide some value. This means that a community would not like to end up being treated as a rectangle in diagram describing an organization or a development setup diagram, particularly if the shape representing the community seems to have been added as an afterthought. Hence, the public organizations would somehow find a way to give room to the community and let it contribute in some meaningful way. Ideally, the contributions should somehow be acknowledged, maybe even contributed upstream.

From the practical point of view, the Joint X-Road community will need to have a community manager to keep things rolling also in the future. This may require sharing the responsibility between the two states for hiring the community manager and also being available for the selected community manager in keeping the community going.

5 BIBLIOGRAPHY

Crowston, K., & Howison, J. (2005). *The social structure of open source software development teams*. First Monday, 10 (2). Available at http://firstmonday.org/ojs/index.php/fm/article/view/1478/1393. Accessed 2015-08-05.

Fogel, K. 2005: *Producing Open Source Software.* O'Reilly Media, Sebastopol, CA, USA. Available at http://producingoss.com/ Accessed 2016-03-29.

Freeman, S. 2011: *Constructing a Community: Myths and Realities of the Open Development Model.* PhD thesis, University of Helsinki. Available at https://helda.helsinki.fi/handle/10138/28432. Accessed 2016-03-29.

Kilamo , T,. Hammouda, I., Mikkonen, T. & Aaltonen, T. 2012: *From Proprietary to Open Source – Growing an Open Source Ecosystem.* In Journal of Systems and Software vol. 85, issue 7. Elseviere Science Inc.

Kilamo, T. 2014: *Essential Properties of Open Development Communities. Supporting Growth, Collaboration and Learning.* Tampere University of Technology. Publication 1194. Available at http://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/22140/kilamo.pdf?sequence=3&isAllowed=y. Accessed 2016-03-29.

Kim, A. J. 2000: . *Community Building on the Web: Secret Strategies for Successful Online Communities.* Boston, Addison-Wesley Longman Publishing.

Raymond, E. S. 1998: *The Cathedral and the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary.* O'Reilly.

Senyard, A. & Michlmayr, M. 2004: *How to Have a Successful Free Software Project.* In Proceedings of the 11th Asia-Pacific Software Engineering Conference. IEEE Computer Society. Available at https://pdfs.semanticscholar.org/a5e4/f4892359b16a0625a2e5a8c2fc56395c6ed1.pdf. Accessed 2016-03-29.

Appendix A.

QUESTIONNAIRE ON JOINT X-ROAD COMMUNITY

> **Note**: This questionnaire refers to **open-source based X-Road cooperation and community** (later referred to as **Joint X-Road**). It is not related to other X-Road related cooperation between Estonia and Finland.

1. Background and community involvement

1.1. What is your education and working history?

1.2. What do you hope to get out Joint X-Road, professionally?

1.3. How did you get involved with Joint X-Road? Have been involved with X-tee community and how did you get involved with that?

1.4. What does Joint X-Road mean to you?

1.5. With whom do you collaborate in Joint X-Road and/or in local community?

1.6. Is this your professional activity (partially or totally compensated by employer), spare time activity or both?

1.7. Are you a member of any other open-source based communities, whether local, national or global?

1.8. Could you describe how you got involved with the other communities?

1.9. What about people – with whom do you collaborate in the other communities?

2. Recruiting and guiding community members

2.1. From your point of view, who could be potential Joint X-Road members or contributors?

2.2. Why do you think someone would want to join Joint X-Road and/or local community?

2.3. Should there be a "recruiting strategy" for Joint X-Road?

Questions for Finnish interviewees:

2.4.1 From your point of view, what would be the best way or best activities to attract new community members? Why would members be interested in joining the Joint X-Road community?

2.5.1 What kind of problems can emerge when trying to recruit members?

2.6.1 In what ways do you think newcomers need to be helped out?

Questions for Estonian interviewees:

2.4.2 From your point of view, why were members interested in joining the X-tee community? What activities were used in attracting new members and what worked best in recruiting newcomers?

2.5.2 In your experience, what kind of problems emerged when trying to recruit new members to X-tee community?

2.6.2 From your point of view, what would be the best way or best activities to attract new community members? Why would members be interested in joining the Joint X-Road community?

2.7.2 In what ways do you think newcomers needed to be helped out?

2.8.2 Do you know any people who have left X-tee community? If so, why have they left?

3. How and by what means Joint X-Road works

3.1. How would you describe your contribution to Joint X-Road or local community (e.g. X-tee community in Estonia)? In what ways have you participated?

3.2. What can be considered a valuable contribution in this community?

3.3. What are the most important tools used in the project?

3.4. What are the next steps and how would you like to participate?

3.5. What does the 'open development model' mean in Joint X-Road?

4. The significance of the community meetings and events

4.1. Have you participated in any X-Road and/or local community meetings?

4.2. If you have, why?

4.3. What did you learn or hoped to learn?

4.4. Who did you meet there?

4.5. If not, why?

5. Cooperation between Finland and Estonia

5.1. What benefit can a joint community (between Finland and Estonia) bring as opposed to having two separate local communities?

5.2. What would be the benefit of having two separate communities? Do you think there could be conflicts of interest between community members from Finland and Estonia? Why?

5.3. How would you evaluate the success of the open source based cooperation so far (e.g. REST Gateway, integration patterns, code samples, API guidelines) so far? What have been the biggest achievements? Which parts of cooperation have not worked?

5.4 What would you like to see as a result of the joint community?

Any other comments or suggestions:

GOFORE

OHJELMISTOKOMPONENTTIEN LISENSSIEN HALLINNAN NYKYTARPEET JA TOIMINTAEHDOTUKSET

Opetus- ja kulttuuriministeriö

Hämeenkatu 16
FI-33200 Tampere

Urho Kekkosen katu 7 B
FI-00100 Helsinki

Y-tunnus 1710128-9
Puh. 010 439 7777

gofore.com

31.5.2016

GOFORE

GOFORE

GOFORE

1. Johdanto

Tämä selvitys kuvaa julkishallinnon avoimen lähdekoodin lisenssien hallinnan nykytilan, nykytilasta aiheutuvat riskit ja potentiaaliset esteet julkishallinnon tuottaman avoimen lähdekoodin jatkokäytölle.

Lisäksi selvitys kuvaa lisenssien hallintaan käytettäviä menetelmiä, työkaluja ja palveluita, sekä suosittelee millaisia käytäntöjä julkishallinnon avoimen lähdekoodin projekteissa tulisi käyttää jatkossa.

Selvitys pohjautuu tilaajien ja toimittajien haastatteluihin seuraavien julkishallinnon organisaatioiden hankkeissa:

- Helsingin Yliopisto
- Opetushallitus
- Trafi
- Työterveyslaitos
- Vantaan kaupunki
- Väestörekisterikeskus

Lisätietoja selvityksestä voi pyytää selvityksen tilaajalta, Jarkko Moilaselta (jarkko.moilanen@minedu.fi), tai tekijöiltä, Salum Abdul-Rahmanilta (salum.abdul-rahman@gofore.com) ja Iiris Saarenpäältä (iiris.saarenpaa@gofore.com).

2. Taustaa

Avoimen lähdekoodin lisenssi määrittää millä ehdoilla lähdekoodia voi käyttää. Lähdekoodin lisenssillä on suuri merkitys käytettävien komponenttien kannalta, mutta myös itse tuotetun lähdekoodin jatkokehityksen kannalta.

Tässä selvityksessä lisenssien hallinnan nykytilaa ja riskejä arvioidaan OSSLI-kehyksen avulla (Open Source Software License Investigation). OSSLI-kehys soveltuu monipuolisesti lisenssien vaikutuksen hallinnan kyvyn arviointiin, koska se huomioi lisenssien hallinnan monella eri tasolla ja useasta eri näkökulmasta. OSSLI-kehyksessä lisenssien hallintaa tarkastellaan normatiivisella tasolla, lain soveltamisessa, liiketoiminnassa, ohjelmistokehitysprosessissa, ohjelmistoarkkitehtuurissa sekä sosiaalisella tasolla. (http://dspace.cc.tut.fi/dpub/handle/123456789/22333).

Vaihtoehtoisia malleja lisenssien hallinnan kypsyyden arviointiin ei ole juuri kehitetty. Ainoa vaihtoehtoinen malli on Linux-säätiön Open Compliance Program, mutta se keskittyy kypsyyden arviointiin vain GPL-perheen lisenssien näkökulmasta (http://www.linuxfoundation.org/programs/legal/compliance).

2.1 Avoimen lähdekoodin lisenssien aiheuttamat riskit

2.1.1 Potentiaaliset tekijänoikeusloukkaukset

Jokainen avoimen lähdekoodin lisenssi myöntää kolme oikeutta:

- oikeus kopioida ja levittää lähdekoodiin perustuvaa ohjelmaa
- oikeus tutustua ohjelmiston lähdekoodiin
- oikeus tehdä muutoksia ohjelmistoon.

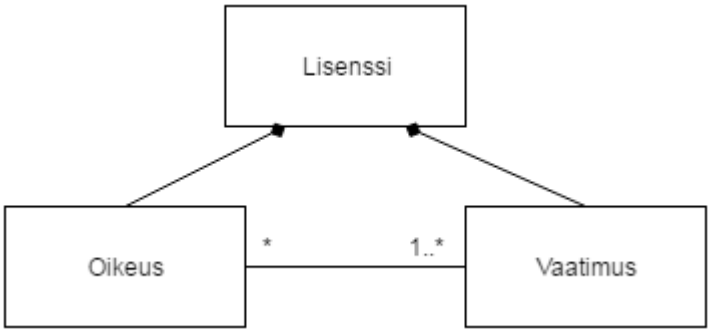
Lisenssistä riippuen on olemassa lisäksi tiettyjä vaatimuksia, jotka tulee täyttää ennen kuin nämä oikeudet saa. Eri lisensseissä on erilaisia vaatimuksia liittyen oikeuksiin, jotka se myöntää.

Jos kyseessä on ohjelmisto, joka muodostuu eri lisensseillä lisensoiduista komponenteista, on tilanne monimutkaisempi. Tällöin tulee huolehtia, että jokaisen lisenssin jokainen käytettyyn oikeuteen liittyvä vaatimus täyttyy. Mikäli nämä vaatimukset ovat keskenään

ristiriidassa, ei ohjelmistoa silloin voi käyttää kyseisellä tavalla syyllistymättä tekijänoikeusrikkomukseen.

Esimerkiksi GPL-lisenssin vastavuoroisuusehto rajoittaa komponentin sisältävän ohjelmiston jakelemista lisenssillä, jossa on GPL-lisenssiin kuulumattomia rajoituksia.

JSON-lisenssissä on ehto, jonka mukaan sillä lisensoitua koodia ei saa käyttää pahaan, kun taas GPL-lisenssissä tämänlaista ehtoa ei ole. Näin ollen sellaisen ohjelmiston levittäminen, joka sisältää sekä GPL-lisenssin että JSON-lisenssin rikkoo GPL-lisenssin ehtoja, ja syyllistyy tekijänoikeusloukkaukseen.



Kuva 1 Avoimen lähdekoodin lisenssin myöntämään oikeuteen voi liittyä vaatimuksia

Melkein kaikissa lisensseissä on vastuuvapauslauseke, jossa tekijänoikeuden haltija sanoutuu irti komponentin sopivuudesta yhtään mihinkään tarkoitukseen. Koska lähdekoodin on avoimesti saatavilla, voi jokainen periaatteessa itse päätellä, onko komponentti sopiva, eikä tätä vastuuvapausehtoa ole tiettävästi toistaiseksi haastettu. Vastaava ehto voisi kaupallisessa sopimuksessa olla Suomen kuluttajanturvan alla ongelmallinen.

Vastuuvapauslausekkeen perusteella ja lähdekoodin saatavuudesta johtuen on kuitenkin täysin mahdollista, että avoimen lähdekoodin komponentti sisältää tekijänoikeusloukkauksen, jolloin sen hyödyntäminen rikkomatta lakia on mahdotonta.

Tekijänoikeusloukkaus voi johtua esimerkiksi avoimen lähdekoodin komponentin sisältämästä koodista, joka on kopioitu sinne ilman tekijänoikeudenhaltian lupaa. Komponentti voi myös sisältää

lisensoimatonta koodia tai koodia, jonka avoin lisenssi ei ole yhteensopiva komponentin muiden lisenssien kanssa.

Avoimen lähdekoodin yhteisöt itse kyllä valvovat ja tutkivat lähdekoodeja ja löytäessään tällaisia virheitä yhteisö yleensä korjaa ne. Suurin riski on komponenteissa, joilla on vähän käyttäjiä ja ylläpitäjiä, koska näiden koodia tarkastellaan harvemmin kuin aktiivisessa kehityksessä olevien komponenttien koodia.

On kolme pääasiallista tapaa jolla tekijänoikeusloukkaukset voivat päätyä ohjelmistoon:

1. Kehittäjä lisää itse tekijänoikeusrikkomuksen aiheuttavan koodinpätkän, komponentin tai konfiguraation.
2. Kehittäjä lisää komponentin, jonka mukana tullut riippuvuus aiheuttaa tekijänoikeusloukkauksen
3. Kehittäjä hyväksyy kontribuution, joka sisältää tai aiheuttaa tekijänoikeusloukkauksen

2.1.2 Tekijänoikeudenhaltijat

Suurimmalla osalla kypsiä avoimen lähdekoodin komponentteja tai kehyksiä on useita kehittäjiä, jotka voivat olla joko yrityksiä tai yksityishenkilöitä. Tällöin jokaisella riittävän suuren kokonaisuuden lisänneellä tekijällä on tekijänoikeus ohjelmistoon. Tämän ansiosta jokaisella tekijänoikeudenhaltijalla on oikeus vaatia toimia asianomaisena, jos tekijänoikeutta on rikottu.

Toisaalta mikäli tämänkaltaisessa tilanteessa haluaisi neuvotella poikkeuksen lisenssiehtoihin, pitäisi jokaiselta tekijänoikeudenhaltijalta saada suostumus muutokseen, mikä voi olla käytännössä mahdotonta. Mikäli tämän kaltaisen yhteisön tekijänoikeuksia loukkaa, on epätodennäköistä, että he olisivat kovinkaan aggressiivisia korvausten tai oikeustoimien suhteen, vaan ennemminkin yrittäisivät neuvotella lisenssiehtoja kunnioittavan ratkaisun. Liian aggressiivinen käytös voitaisiin nähdä yhteisön, ohjelmiston tai avoimen lähdekoodin mainetta vahingoittavana.

Myös yritykset ja yhdistykset tuottavat kokonaisia avoimen lähdekoodin komponentteja, ja tällöin yhdellä toimijalla saattaa hyvinkin olla mahdollisuus neuvotella poikkeuksista lisenssiin. Poikkeukset voidaan mahdollistaa kontribuutiosopimuksella. Sen

lisäksi että kontribuutiosopimuksen ehdoissa voidaan antaa ohjelmiston ylläpitäjällä mahdollisuus neuvotella poikkeuksista lisenssiehtoihin, niissä saatetaan sitoutua siihen, että allekirjoittajan kontribuutiot eivät sisällä tekijänoikeusloukkauksia. Kontribuutiosopimusta käytetään yleensä vain hyvin kypsissä tai kaupallisesti merkittävissä projekteissa.

Jos yritys tarjoaa avoimen lähdekoodin komponentista tai kehyksestä myös kaupallista lisenssiä, voi se olla huomattavasti aggressiivisempi neuvotteluissa, korvausvaatimuksissa tai oikeudenkäynnissä kuin epämuodollisempi avoimen lähdekoodin yhteisö.

2.1.3 Tekijänoikeusloukkauksen löytyminen

Perinteisesti avoimeen lähdekoodiin liittyviä tekijänoikeusloukkauksia on etsitty vertailemalla jonkin ohjelmiston käännettyä koodia tunnettujen komponenttien käännettyyn lähdekoodiin. Mikäli samankaltaisuuksia on löytynyt riittävästi, on asiaa ruvettu selvittämään olemalla yhteydessä mahdolliseen rikkojaan. Lähdekoodiin päästään tutustumaan todennäköisesti viimeistään oikeuskäsittelyssä oikeuden määräyksestä.

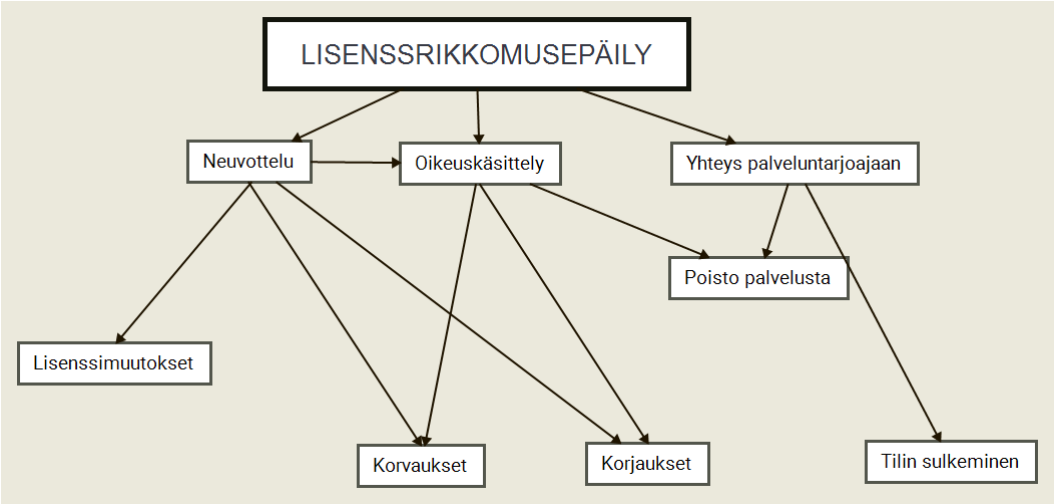
Julkishallinnon palveluissa käännettyä koodia harvemmin jaetaan eteenpäin. Teoriassa virtuaalipalvelimien tai konttien kuvat voidaan laskea jakeluiksi, mutta niihin sisältyy sama ongelma kuin perinteisiin palvelimiin. Jotta käännettyä koodia pääsee tutkimaan palvelimilla, on ensin syyllistyttävä tietomurtoon. Tietomurron kautta saatu epäily tekijänoikeusloukkauksesta on erittäin ongelmallinen tekijänoikeudenhaltijan kannalta, koska silloin haltija oletettavasti kuuluisi rikoksesta epäiltyjen joukkoon. Lisäksi todisteet olisivat tällöin vähintäänkin kyseenalaisia.

Tekijänoikeusloukkaus voi paljastua myös sitten, kun avoimella lisenssillä lisensoitu koodi on julkaistu ja tekijänoikeudenhaltija tai kolmas osapuoli on päässyt tutustumaan siihen.

2.1.4 Tekijänoikeusloukkauksen seuraamukset

Jos tekijänoikeusloukkausta epäillään, tekijänoikeudenhaltijalla on mahdollisuus yrittää ottaa yhteyttä tekijänoikeuden loukkaajaan, aloittaa oikeuskäsittely tai yrittää poista tekijäoikeusloukkaus sen mahdollistaneen kolmannen osapuolet palveluntarjoajan kautta.

Näiden toimien mahdollisiin seurauksiin kuuluvat lisenssimuutokset, korvaukset, korjaustoimenpiteet, tekijänoikeusloukkauksen poisto palvelusta tai tekijänoikeudenloukkaajan tilin poisto.



Kuva 2 Lisenssirikkomusepäilyn eteneminen

Kun tekijänoikeudenhaltija saa tiedon tekijänoikeusloukkauksesta on todennäköistä, että hän lähestyy tekijän oikeusrikkomuksen tekijää ja tiedottaen havaitusta virheestä yrittää neuvotella ratkaisusta tekijänoikeusrikkomukseen ja mahdollisista korvauksista. Ratkaisuna voi olla lisensointi eri lisenssillä, kuten sen maksullisella versiolla.

Tekijänoikeudenhaltijalla on myös mahdollisuus vaihtaa lisenssiä tai tehdä muutoksia ohjelmistoon, ohjelmiston käyttöön tai liiketoimintaan niin, että lisenssiehdot täyttyvät. Kyseisen lähdekoodin käyttö voidaan esimerkiksi lopettaa tai lisensoida oma tuotos avoimella lisenssillä ja julkaisemalla se.

Mikäli neuvotteluissa tekijänoikeudenhaltijan kanssa päästään yhteisymmärrykseen korjaustoimenpiteistä, kannattaa ne tehdä välittömästi. Myös epäselvässä tilanteessa voi olla järkevää tehdä korjauksia jotka selkeyttävät tilannetta ja voivat siten heikentää epäilyä tekijäoikeusrikkomuksesta.

Korjaamisen voi tehdä usealla tavalla, riippuen tietysti rikotusta lisenssistä ja ohjelmiston rakenteesta. Yksi mahdollinen korjaus voi olla lähdekoodin julkaiseminen sopivalla lisenssillä. Myös loukatun komponentin poistaminen järjestelmästä ja sen toiminnon

korvaaminen omalla koodilla tai lisenssien kannalta sopivammalla komponentilla voi olla vaihtoehto. Kolmas vaihtoehto on yrittää eristää loukattu tai loukkaava komponentti muusta järjestelmästä siten, että lisenssiehdot täyttyvät. Joka tapauksessa korjaamistoimenpiteet vaativat työtä ja voivat muodostua kalliiksi, mutta todennäköisesti vähemmän työlääksi ja kalliiksi kuin oikeusprosessi.

Mikäli tekijänoikeudenhaltijan ei ole tyytyväinen neuvottelujen lopputulokseen tai neuvotteluyhteyttä ei saada aikaiseksi, voi tämä ilmoittaa viranomaisille tekijänoikeusrikkomuksesta ja asian käsittely voi siirtyä oikeuteen. Tekijänoikeusloukkausta käsittelevä oikeudenkäynti vaatii asianajajien, ohjelmistonkehittäjien ja tekijänoikeus- tai lisenssiasiantuntijan aikaa. Sen lisäksi oikeus voi asettaa tekijänoikeusloukkauksen sisältävän järjestelmän käyttökieltoon joko käsittelyn päätteeksi tai sen ajaksi. Käsittelyn päätteeksi hävinnyt osapuoli saattaa joutua maksamaan voittajan oikeudenkäyntikulut sekä korvauksia.

Mikäli tekijänoikeudenhaltija ei saa lainkaan yhteyttä tekijänoikeudenloukkaajaan neuvotteluja varten, on hänellä myös tällöin mahdollisuus oikeuskäsittelyyn. Oletuksena on, että tilanteessa, jossa suomalaisen julkisen hallinnon tuottamassa ohjelmistossa epäiltäisiin tekijänoikeusrikkomusta, kyseinen tekijänoikeusrikkomus olisi tapahtunut Suomessa.

Suomessa tehdyt tekijänoikeusrikkomukset käsitellään Suomen oikeusjärjestelmässä. Toisin toimitaan vain, jos lisenssin, jonka ehtoja on rikottu, ehdoissa on maininta, että kyseisen lisenssin rikkomukset käsitellään jossakin toisessa oikeudessa.

Voidaan kuitenkin teoreettisesti argumentoida, että jos koodia jaellaan jonkin kansainvälisen palvelun kautta, kuten versionhallintatietovarannon tai mobiilisovelluskaupan kautta tai ajetaan ulkomailla sijaitsevalla palvelimella, että tekijänoikeusrikkomus tapahtuisi silloin jossain muussa maassa. Näissä tapauksissa palveluja kuitenkin hallinnoidaan Suomesta käsin, joten oikeushenkilö, joka tekijänoikeusrikkomuksen on tehnyt, on silloin todennäköisesti ollut Suomessa ja oikeuskäsittely on tällöin Suomalaisen oikeusistuimen alainen. Tämänkaltainen palvelu ei ole oikeushenkilö, eikä se sellaisenaan voi syyllistyä tekijänoikeusrikkomukseen.

Palvelun tarjoaja on kuitenkin oikeushenkilö ja se saattaa tekijänoikeudenhaltijan kanssa käytyjen neuvottelujen tai ulkomaisen oikeusistuimen päätöksen johdosta poistaa epäillyn tekijänoikeutta loukkaavaan ohjelmiston tai palvelun omista järjestelmistään. Tällöin on mahdollista, että myös palvelun tarjoaja katsoo, että heidän käyttöehtojaan on loukattu ja sulkee kyseisen epäillyn tilin palvelussa. Tämä vaikuttaa silloin myös muihin palveluihin tai kehitysprojekteihin, jotka riippuivat kyseistä tilistä.

Lisenssirikkomusepäily, oikeudenkäynti tai tuomio luovat vastuulliselle organisaatiolle tai henkilölle huonoa mainetta. Tämä voi näkyä asiakkaiden haluttomuutena tehdä yhteistyötä tai haasteena rekrytoida päteviä ohjelmistokehittäjiä. Huono maine myös nostaa riskiä, että tekijänoikeudenhaltijat alkavat selvittää löytyykö huonomaineisen organisaation järjestelmistä muitakin tekijänoikeusloukkauksia.

2.2 Avoimen lähdekoodin lisenssien vaikutukset jatkokäyttöön

Avoimen lähdekoodin lisenssien perusoikeudet mahdollistavat niillä lisensoidun lähdekoodin jatkokäytön. Tutkimuksissa on ilmennyt, että vastavuoroiset lisenssit käytettynä julkishallinnon yleishyödyllisisä projekteissa saattavat olla keskimääräistä kiinnostavimpia kohteita kontribuutioille. Vastavuoroisuusehto velvoittaa julkaisemaan muutokset jatkokehitettyyn koodin tehdyt muutokset ja korjaukset, mikä on hyödyllistä myös alkuperäiselle projektille.

Jotkin toimijat kuitenkin välttelevät vastavuoroisia lisenssejä ja suosivat sallivia lisenssejä. Varsinkin yritykset, joiden liiketoimintamalliin ei sovellu koodin julkaiseminen, voivat vältellä vastavuoroisilla lisensseillä lisensoitua koodia. Vaikka heidän liiketoimintamalliinsa ei sovi oman koodin avaaminen, voisivat he silti julkaista korjauksia käytettyyn komponenttiin.

Kaupalliset toimijat julkaisevat koodinsa vastavuoroisella lisenssillä, koska silloin kilpailija ei voi kopioida suoraan uuden toiminallisuude toteuttavaa koodia julkaisematta oma lähdekoodiaan tai jatkokehittämään siihen tehtyjä parannuksia julkaisematta niitä. Julkishallinnon palvelut ovat usein uniikkeja ja niiden toisintaminen ei luo palveluntarjoajalle merkittävää kilpailijaa. Siksi

vastavuoroisten lisenssien käyttö kilpailijoiden epäreilun hyötymisen estämiseksi voi olla turha.

Lisenssivalinta voi myös perustua järjestelmän pohjana toimivan palvelun tai ohjelmistokehyksen lisenssiin. Näin vältytään mahdollisilta lisenssikonflikteilta. Toinen mahdollisuus on, että jokainen komponentti julkaistaan juuri sille sopivalla lisenssillä, ja komponentit kootaan yhteen toimivaksi palveluksi. Tämänlainen modulaarinen toimintopohjainen julkaiseminen lienee paras tapa edesauttaa komponentin jatkokäyttöä, kun sillä on yksi selkeä toiminto, jonka jatkokäyttö on selkeää. Tällöin motivaatio uudelleenkäyttöön perustuu komponentin toiminnallisuuteen ja lisenssi toimii vain mahdollistajana. Modulaarinen arkkitehtuuri, jonka jokainen komponentti on julkaistu itsenäisesti, vaatii kehitystiimiltä jonkin verran enemmän työtä ja osaamista. On harkittava projektikohtaisesti, vaaditaanko julkaisuilta tämän kaltaista laatua, vai onko tärkeämpää saada toimiva ratkaisu käyttöön nopeasti.

2.3 Avoimen tuotteen hallintamalli

Avoimen tuotteen hallintamalli on julkisen sektorin toimintamalli, jonka avulla omistajat hallitsevat yhteisesti kehitettyä ja rahoitettua ohjelmistoa. Avoimen tuotteen hallintamallissa tilaaja vastaa ohjelmiston immateriaalioikeuksista (IPR) ja ohjelmiston elinkaaresta. Omistaminen tuo valtaa, mutta myös vastuuta. Omistamisen myötä tilaajalle tulee vastuu tuotteenhallinnasta tai sen järjestämisestä. Ohjelmistotuotteen hallinnalla tarkoitetaan käytännössä toimia, jotka mahdollistavat ohjelmiston hallitun kehityksen ja kehityksen seurannan sen elinkaaren aikana.

Hallintamallissa keskeisiä määriteltäviä asioita ovat omistajuus ja muut roolit, sekä niihin liittyvät käytänteet, tuotteen elinkaarenhallinta ja tuotteen kehittämisen rahoitus. Päätavoite on, että tuote on käytettävissä myös muilla tarvitsijoilla, jolloin ominaisuudet täytyy kehittää vain kerran, ja niistä maksetaan vain kerran.

Avoimen tuotteen hallintamalli määrittelee formaalit säännöt, joiden avulla julkisen hallinnon organisaatioille voidaan perustaa käyttäjäyhteisö ohjelmistotuotteen ohjausta ja hallintaa varten.

Ohjelmiston lisenssi on osa tuotteen hallintaa, vaikka tuotteen hallintaan kuuluu paljon muutakin.

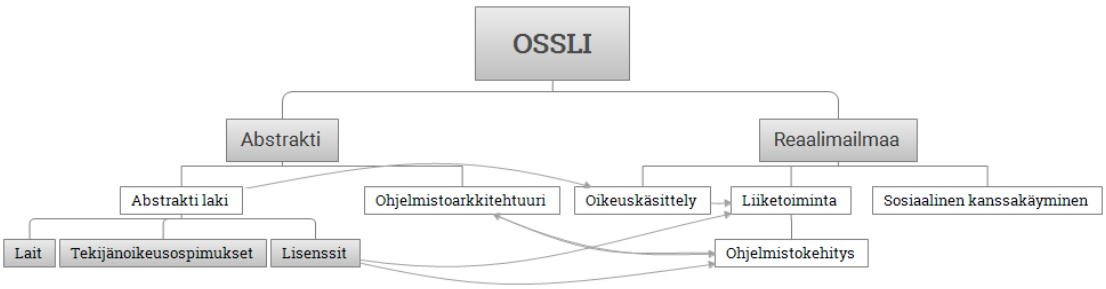
Avoimen tuotteen hallintamalli ei kuitenkaan suoranaisesti ota kantaa lisenssien hallintaan. Hallintamallissa määritellään erilaisia rooleja. Eräs niistä on avoimuuden edunvalvoja, jonka tehtävänä on huolehtia tilattavan ohjelmiston lisenssin avoimuudesta, tehdä tuotteelle vastaanottotarkastus, jolla varmistetaan tarvittava dokumentaatio, metadata, versiotieto, yhteystiedot ja lisenssin avoimuus. Tehtävää voi suorittaa myös tuotepäällikkö. Tuotepäällikön tehtävänä on yleisesti huolehtia tuotteesta ja sen kehittymisen suuntaviivoista. Tuotepäällikkö kerää tarpeita kehitystä varten, seuraa teknologian muutosta, seuraa muita tarjolla olevia tuotteita ja lakimuutoksia. Tuotepäällikön rooli on hyvin yleinen yrityksissä, mutta julkisella sektorilla käytäntö on uusi.

Tuotteen omistajan, ja siitä tarkemmin huolehtivan roolin nimeäminen on tärkeää, jotta on tuotteesta huolehtiva taho. Huolehtiva taho varmistaa, että tuote ei pääse rapautumaan ja hallitsee muutoksia, jotta päällekkäistä työtä ei tehdä, eikä erilaisia versioita pääse syntymään. Näin saadaan tehdyt muutokset kaikkien hyödynnettäväksi.

Avoimen tuotteen hallintamallissa ei oteta kantaa lähdekoodin julkaisuun avoimen lähdekoodin lisensseillä, vaan sitä voidaan käyttää myös suljetun tuotteen yhteiskehityksessä. Avoimen tuotteen tuoteomistajan tai tuotepäällikön täytyy tällöin ottaa vastuu lisenssien hallinnan tasosta, kuten tilaajan missä tahansa avointa lähdekoodia hyödyntävässä kehitysprojektissa.

2.4 OSSLI-kehys

OSSLI-kehys perustuu ontologioihin ja sen käsitteet ovat perinteisten ontologiamallien tapaan jaettu abstrakteihin ja realisiin tekijöihin. Abstrakteihin käsitteisiin kuuluvat normatiivinen taso eli abstrakti laki ja ohjelmistoarkkitehtuuri. Reaalimaailmaan kuuluvat liiketoiminta, ohjelmistokehitys, sovellettu laki eli oikeuskäsittelyt ja sosiaalinen taso.



Kuva 3 OSSLI-kehyksen analyysitasot ja niiden suhteita toisiinsa

2.4.1 Abstrakti laki

Normatiivisella tasolla käsitellään lakeja, tekijänoikeussopimuksia ja lisenssejä. Käytännössä tällä tasolla on käsitteitä, jotka ilmaisevat onko jokin asia sallittu, vaadittu, tai kielletty jossain yhteydessä sekä näiden suhteita toisiinsa.

Jokaisessa maassa tekijänoikeuslainsäädäntö määrittää tekijänoikeudet ja tarjoaa mahdollisuuden tekijänoikeudenhaltijalle ylipäätään lisensoida tuotoksensa. Maakohtainen tekijänoikeuslaki myös määrittää teoskynnyksen eli sen rajan, jossa vaiheessa jokin tuotos on niin omaperäinen tai kompleksinen, että se muodostaa itsenäisen teoksen ja on näin tekijänoikeuden alainen. Ohjelmistokehityksessä käytetään usein esimerkkikoodeja ohjeista, blogeista, keskustelu- ja kysy-ja-vastaa -palstoilta, mutta usein ne ovat niin lyhyitä, että teoskynnys ei ylity.

Ohjelmistokehityksessä on ollut käytäntönä, että rajapinnan määrittelevät dokumentit kuten niin sanotut otsikkotiedostot eivät ole tekijänoikeuden alaisia. Kuitenkin juuri tällä hetkellä käynnissä oleva Oraclen ja Googlen välinen Android java-api -oikeudenkäynti voi muuttaa tulkintaa, vaikka Yhdysvalloissa tehty päätös ei suoranaisesti vaikuta Suomen lainsäädäntöön nykyisten tekijänoikeussopimusten puitteissa. Normatiivinen taso luo kehyksen jonka puitteissa lakia sovelletaan eli oikeuskäsittelyn tason.

2.4.2 Ohjelmistoarkkitehtuuri

Ohjelmistoarkkitehtuurilla tarkoitetaan käsitystä siitä, miten ohjelmiston toiminnot on jaettu eri komponenttien kesken ja miten nämä komponentit suhtautuvat toisiinsa. Osa tulkitsee

ohjelmistoarkkitehtuurin olevan olemassa vain silloin kun tämä on muodollisesti kuvattu.

Ohjelmistoarkkitehtuurikuvauksia on monia erityyppisiä ja eri tarkkuusasteilla. Yhdestä ohjelmistosta onkin usein monia eri arkkitehtuurikuvauksia, jotka yhdessä auttavat ymmärtämään ohjelmiston arkkitehtuuria. Ohjelmistoarkkitehtuuri sekoitetaan välillä tietojärjestelmäarkkitehtuuriin, mutta on matalamman tason kuvaus. Ohjelmistoarkkitehtuuri kuvaa yhden ohjelmiston, joka sitten saattaa näkyä yhtenä komponenttina tietojärjestelmäarkkitehtuurissa.

Toisaalta pilvipalveluiden kehittymisen myötä myös tietojärjestelmätason komponentteja kuten palvelimia hallinnoidaan tietokoneohjelmien avulla, jolloin perinteisesti tietojärjestelmäarkkitehtuurin sisältä löytyviä komponentteja saattaa siirtyä ohjelmistoarkkitehtuurin puolelle. Ohjelmistoarkkitehtuuri sekä heijastaa ohjelmistokehitystyössä tehtyjä ratkaisuja, että ohjaa niitä.

2.4.3 Oikeuskäsittely

Mikäli tekijänoikeusrikkomusta ei saada neuvoteltua tai toinen osapuoli ei ole halukas neuvotteluihin, on tekijänoikeudenhaltijalla mahdollisuus tekijänoikeusrikkomuksen oikeuskäsittelyyn. Tuomarit ja asianajat yrittävät tulkita lainsäädäntöä arvioidessaan potentiaalista tekijänoikeusloukkausta, mutta eivät ole ohjelmistokehityksen asiantuntijoita. Siksi oikeuskäsittelyn osapuolten on tuotava käsittelyyn teknisiä asiantuntijoita, joiden todistuksia käytetään tukena päätöksenteossa.

Oikeuskäsittelyn päätöksillä on vaikutus myös liiketoimintaan koska oikeuskäsittelyyn osallistuminen vaatii resursseja, joita voisi käyttää liiketoimintaan ja oikeuden päätökset voivat vaikuttaa suoraan siihen mahdollisiin liiketoimintaratkaisuihin. Joissain maissa aiempien oikeuskäsittelyn päätökset vaikuttavat lain tulkintaan, jolloin voidaan katsoa niiden vaikuttavan myös normatiiviseen tasoon.

2.4.4 Liiketoiminta

Liiketoiminta on se taso, joka tuottaa ohjelmistokehityksessä käytettävät resurssit. Liiketoiminta määrittää myös ohjelmiston

toiminnalliset tavoitteet ja tällä tavalla vaikutta siihen, mitä oikeuksia tarvitaan niiltä komponenteilta, joita ohjelmistossa käytetään. Liiketoiminnan taso tuottaa lisäksi resurssit, joita käytetään mahdollisessa oikeuskäsittelyssä.

Liiketoiminnan ei kuitenkaan tarvitse määritelmän mukaan välttämättä tuottaa rahaa. Esimerkiksi julkisen sektorin palvelut voivat tuottaa arvoa niiden käyttäjille, vaikka niiden resursointiin käytetyt varat eivät tule suoraan käyttäjiltä.

2.4.5 Ohjelmistokehitys

Ohjelmistokehityksen tasolla tuotetaan ohjelmistoja ja asennetaan ne käytettäväksi liiketoimintatavoitteiden saavuttamiseksi. Ohjelmistokehitys tasolla tapahtuvat toimenpiteet komponenttien integroimiseksi järjestelmään, jolloin ohjelmistokehitys on se taso, jossa potentiaalisesti voidaan tehdä tekijänoikeusrikkomuksia.

2.4.6 Sosiaalinen taso

Sosiaalisella tasolla kuvataan ihmisten ja organisaatioiden kanssakäymistä ja kommunikaatiota. Organisaatioiden ja henkilöiden maine ja suhteet vaikuttavat siihen, miten liiketoiminnassa, ohjelmistokehityksessä ja oikeuskäsittelyssä viestitään, jolloin se vaikuttaa suoraan näissä tehtäviin päätöksiin.

3. Julkishallinnon lisenssien hallinta

Tämän selvityksen tavoitteena oli selvittää sekä tilaajien että toimittajien edustajien kykyä hallita avoimen lähdekoodin lisenssejä. Selvitykseen haastateltiin julkisen hallinnon projekteista vastaavia henkilöitä. Selvityksen suppeuden ja aikataulullisen lyhyyden vuoksi haastattelut ovat luonteeltaan kvantitaavisia ja eksploratiivisia, eikä niiden tuloksia voi käsitellä täysin kattavina.

3.1 Haastattelut

Haastatteluissa hyödynnettiin OSSLI-kehystä, jonka avulla haastattelurunko muodostettiin. Haastattelu jaettiin seitsemään eri teemaan, joita olivat perustiedot, liiketoiminta, sosiaaliset tekijät, lakitekniset kysymykset, oikeuskäsittely, ohjelmistoarkkitehtuuri ja ohjelmistokehitys. Haastattelurunko ja -kysymykset ovat esiteltynä seuraavaksi luvussa 3.1.1.

3.1.1 Haastattelukysymykset

Peruskysymykset

1. Projektin/Hankkeen nimi
2. Oma rooli projektissa
3. Kuinka laaja projekti on (htp) ja kuinka pitkällä ollaan?
4. Montako eri ohjelmistokomponenttia järjestelmässä on?
5. Monellako eri lisenssillä lisensoitua koodia järjestelmässä käytetään?
6. Millä lisenssillä järjestelmä julkaistaan, jos julkaistaan?
7. Käytetäänkö projektissa avoimen tuotteen hallintamallia?

Liiketoimintaa koskevat kysymykset

1. Onko projektin/hankkeen liiketoimintamalli yhteensopiva käytettyjen avoimen lähdekoodin lisenssien vaatimusten kanssa?
2. Miten avoimen lähdekoodin käyttö vaikuttaa tuottajan tai toimittajan aineettomaan pääomaan? (henkilöstön osaaminen, asiakassuhteet, organisaation rakennepääoma jne.)
3. Miten avoimen lähdekoodin käyttö vaikuttaa tuottajan tai toimittajan suhteisiin muiden organisaatioiden kanssa (esim. asiakkaat, alihankkijat)?
4. Mitä resursseja tarvitaan avoimen lähdekoodin lisenssien hallintaan?

5. Mitä resursseja tarvitaan avoimen lähdekoodin tekijänoikeusloukkauksesta johtuvassa oikeuskäsittelyssä?

Sosiaaliset

1. Miten lisenssien ehdot vaikuttavat projektin/hankkeen asiakkaisiin?
2. Miten lisenssien ehdot vaikuttavat potentiaalisiin työntekijöihinne?
3. Miten lisenssien ehdot vaikuttavat suhteeseen kehittäjäyhteisön kanssa?
4. Miten lisenssien ehdot vaikuttavat ohjelmistokehitystiimin työhön?
5. Miten lisenssirikkomussyytös tai oikeudenkäynti vaikuttaisi asiakassuhteisiinne?
6. Miten lisenssirikkomussyytös tai oikeudenkäynti vaikuttaisi työntekijöihinne tai potentiaalisiin työntekijöihinne?
7. Miten lisenssirikkomussyytös tai oikeudenkäynti vaikuttaisi suhteeseenne kehittäjäyhteisön kanssa?
8. Miten lisenssirikkomussyytös tai oikeudenkäynti vaikuttaisi ohjelmistokehitystiimiin?

Lakitekniset kysymykset

1. Miten tunnistatte lisenssin?
2. Miten selvitätte, onko lisenssiehdoissa vaatimuksia oikeusistuimesta, jonka alaisuudessa lisenssiloukkaukset käsitellään?
3. Miten selvitätte lisenssin lisenssiehdot?
4. Miten selvitätte, mitä oikeuksia lisenssi myöntää?
5. Miten selvitätte, mitkä oikeudet riippuvat mistäkin lisenssiehdosta?
6. Miten tunnistatte lisensoijat (tekijänoikeudenhaltijat)?
7. Miten tunnistatte eri lisenssien ehtojen muodostamat ristiriidat?
8. Miten voitte ratkaista lisenssiehtojen muodostamia ristiriitoja?

Oikeuskäsittely

1. Missä oikeusistuimessa lisenssirikkomukset käsitellään?
2. Miten kyseinen oikeusistuin näkee avoimet lisenssit?
3. Miten oikeusistuin näkee lisenssien ehdot?
4. Mitä resursseja tarvitsette hallitsemaan lisenssirikkomuksiin liittyvää lainsäädäntöä?
5. Ovatko tekijänoikeudenhaltijat eri oikeusistuimen alaisia kuin te?

6. Miten oikeusistuin ottaa huomioon kulttuurillisia normeja?

Ohjelmistoarkkitehtuuri

1. Onko ohjelmiston arkkitehtuuri muodollisesti kuvattu?
2. Miten eri komponenttien lisenssit tunnistetaan arkkitehtuurissa?
3. Voiko arkkitehtuurin perusteella arvioida, mitkä komponenttien lisenssien ehdot ovat yhteen sopimattomia keskenään?
4. Voiko arkkitehtuurin perusteella arvioida, mitkä komponenttien lisenssien ehdot ovat yhteen sopimattomia eri käyttötarkoituksissa?
5. Voiko ohjelmistoarkkitehtuuria käyttää lisenssiehtojen rikkomusten korjaamiseen?
6. Voiko ohjelmistoarkkitehtuuria käyttää lisenssiehtojen rikkomusten ehkäisemiseen?

Ohjelmistokehitys

1. Vastaako implementaatio arkkitehtuuria?
2. Miten joka komponentin jokainen lisenssi on tunnistettu?
3. Miten ohjelmistokehittäjät voivat tunnistaa lisenssiristiriitoja?
4. Miten ohjelmistokehittäjät voivat ratkaista lisenssiristiriitoja?

Lopuksi haastatteluille annettiin vielä mahdollisuus vapaasti kertoa ajatuksiaan.

3.2 Nykyiset lisenssien hallinnan käytännöt

Haastattelujen perusteella avointa lähdekoodia käytetään julkishallinnossa valmiina tuotteina sekä tietojärjestelmien tai infrastruktuurin osana erityisesti pilvipalveluissa. Avointa lähdekoodia hyödynnetään suurissa hankkeissa ja pienissä projekteissa. JHS-suosituksen ansiosta avoin lähdekoodi on mainittu myös ohjelmistokehityksen palvelukehityksessä.

Liiketoiminnan tavoitteiden kannalta oman lähdekoodin julkaisu avoimen lähdekoodin lisensseillä koetaan julkishallinnossa toimivaksi. Haastattelujen perusteella lisenssien hallinnan kypsyys ei kuitenkaan korreloi sen perusteella, onko tehty päätös julkaista tuotos vai ei tai millä lisenssillä julkaistaan.

Vaikka tuotoksia julkaistaan, ei julkishallinnossa ole kuitenkaan kyvykkyyttä edistää kehittäjäyhteisön syntymistä tai tukemista.

Selvitystä varten haastateltiin sekä tilaajan, että toimittajan edustajia yhdestätoista projektista. Pienimmän projektit olivat suuruudeltaan 1000-2000 henkilötyöpäivää ja suurimmat olivat yli 100 000 henkilötyöpäivän hankkeita. Melkein kaikissa projekteissa osa palveluista oli jo loppukäyttäjien käytössä.

3.2.1 Liiketoiminta

Julkishallinnossa on sekä asiakkaiden, että toimittajien kesken yhteinen näkemys siitä, että avoimet lisenssit sopivat hyvin julkishallinnon ohjelmistojen liiketoimintamalleihin. Avointen lisenssien hyödyntäminen vaatii kuitenkin asiakkaan ja toimittajan ymmärtämyksen lisenssien vaatimuksista. Kaikki haastateltavat olivat sitä mieltä, että avoimen lähdekoodin lisenssit sopivat julkihallinnon ohjelmistojen liiketoimintamalleihin, mutta yleensä vain toimittaja tai asiakas osasi ilmaista, miten lisenssiehdot käytännössä vaikuttavat liiketoimintaan. Vain yhdessä haastatellussa projektissa oli epävarmuutta sekä asiakkaan, että toimittajan mielestä liiketoiminnan ja lisenssivaatimusten yhteensopivuudesta.

Avoimen lähdekoodin hyödyntämisessä tärkeimmiksi hyödyiksi koetaan osaavien tekijöiden hyvä saatavuus, mahdollisuus ulkoiseen jatkokehitykseen ja avoimempi yhteistyö toimittajan ja asiakkaan välillä. Kehittäjille tärkeää on myös oman työn auditoitavuus.

Moni haastateltu tunnisti, että lisenssien hallintaan tarvitaan erityisasiantuntija, eikä lakimies riitä. Ensisijaisesti lisenssien hallintaa pyritään tekemään kehittäjien ohjeistuksen kautta. On suuri riski, että useat kuitenkin kokivat lisenssien hallinnan tarpeettomaksi. Yllättävän harva myöskään tiedosti, että lisenssien loukkaukseen liittyvässä oikeudenkäynnissä tarvitaan juristin lisäksi ohjelmisto- ja lisenssiasiantuntijoita.

3.2.2 Sosiaalinen

Sosiaalisesta näkökulmasta tiedostetaan, että osa kehittäjistä työskentelee mielellään avoimen lähdekoodin kanssa, eivätkä kehittäjät ainakaan vastusta sitä. Toisaalta kaikki haastatellut ohjelmistokehittäjät olivat valikoituneet tehtäviin, joissa käytetään ja mahdollisesti julkaistaan avointa lähdekoodia. Myös muut haastatellut olivat ensisijaisesti tekemisissä tämän kaltaisten kehittäjien kanssa, joten otantaa ei voi pitää edustavana.

Riskinä sosiaalisesta näkökulmasta on, että yhteyksiä avoimen lähdekoodin kehittäjäyhteisöihin ei juuri ole. On kuitenkin tiedostettu, että lisenssirikkomus voisi heikentää niitä ennestään. Lisenssirikkomuksen seurauksena erityisesti maineen pelätään huonontuvan, jos rikkomus saa julkisuutta. Osa kehittäjistä ottaa myös huomioon, että lisenssirikkomuksen seuraukset häiritsisivät kehitystiimin työtä.

3.2.3 Lakitekniset

Neljäsosa haastatelluista tunnistaa lisenssit tarkkuudella, ovatko ne avoimia vai suljettuja. Neljäsosa erottelee sallivat ja vastavuoroiset lisenssit. Neljäsosassa tapauksia on vastuu lisenssien hallinnasta ulkoistettu kehitystiimiltä ja asiakkaan yhteyshenkilöiltä. Vain yksi henkilö osasi ottaa huomioon SaaS-ehdon, joka on merkityksellinen varsinkin palvelukehityksessä. Riski on myös, että kukaan haastatelluista ei sanonut tarkastavansa, onko käytetyissä lisensseissä vaatimuksia oikeusistuimesta.

Lisenssiehtojen tulkinta jakautuu kolmeen osaan. Kolmasosa on ulkoistanut vastuun lisenssiehtojen ymmärtämisestä. Toinen kolmasosa ei lue lisenssiehtoja lainkaan ja viimeinen kolmannes lukee ne itse. Lisenssien myöntämien oikeuksien suhteen tilanne on melko sama.

Yksikään haastateltava ei maininnut nimeämisvaatimuksia, jotka määräävät pitääkö valmiissa palvelussa julkaista tietoa tekijänoikeudenhaltijoista. Nimeämisvaatimukset ovat siinä mielessä haastavia, että osassa lisensseistä nimeämisehto vaati tekijänoikeudenhaltijoiden esittämisen julkiasun osassa ja osassa lisensseistä julkiasu erityisesti kielletään.

Yksi haastatteluissa esiin noussut lisenssien hallintamalli on, että käytetään vain yhdellä lisenssillä julkaistua lähdekoodia. Koska kyseessä on hyvin tunnettu vastavuoroinen GPL-lisenssi ja myös oma tuotos julkaistaan GPL-lisenssillä, on tämä toimiva ratkaisu. Toinen löydetty malli on, että käytetään vain sallivia lisenssejä ja omaa koodia ei julkaista. Tämä malli on periaatteessa toimiva, koska on toimiva käytäntö, että toimittaja hoitaa komponentin arvioinnin ohjeistuksen perusteella ja epäselvissä tapauksissa selvittää asiakkaan kanssa, onko lisenssi sopiva. Valittavasti myöskään tässä

ei käynyt ilmi, että tähän prosessiin olisi kuulunut sallivissa lisensseissä yleisen nimeämisehdon käsittely.

Haastattelujen perusteella on huomattava, että lisenssien yhteensopivuus ei suurella osalla ole hoidettu järjestelmällisesti. Neljäsosassa tapauksia vastuu yhteensopivuudesta on ulkoistettu, joista kuitenkin vain osassa vastaava on pätevä tulkitsemaan yhteensopivuuksia. Vain kolme toimittajaa oli tietoisia yhteensopivuustaulukoista. Ainoastaan yksi lähtisi selvittämään mahdollista ongelmaa tekijänoikeudenhaltijan kautta.

3.2.4 Oikeuskäsittely

Haastatelluilla ei ole juurikaan ymmärrystä lisenssirikkomusten oikeudenkäynteihin liittyvistä asioista. Kolme neljäsosaa haastatelluista olettaa, että lakimies riittää ymmärtämään oikeudenkäynnin vaatimuksia. Neljäsosa tiedostaa, että tarvitaan myös lisenssiasiantuntijaa.

Nykytilanteessa ei osata huomioida, että lisenssien hallinnan dokumentaatio olisi hyvä olla oikeuskäsittelyä varten olemassa. Kukaan haastatelluista ei käsitellyt dokumentaatiota tältä kannalta.

3.2.5 Ohjelmistoarkkitehtuuri

Arkkitehtuuridokumentaatio on projekteissa yleisesti ottaen hyvällä tasolla. Arkkitehtuuritasolla ei kuitenkaan ole kuvauksia immateriaalioikeuksia, vain kolmasosalla on listattuna komponenttikohtaiset lisenssit.

Arkkitehtuuri nähdään ohjaavana apuvälineenä, ei työkaluna. Haastateltujen mukaan modulaarisuus helpottaa komponenttien vaihtoa, komponenttien riippuvuuksien muuttamista ja voi ohjata komponentin lisenssinvalintaa. Vain muutama osaisi kuitenkaan hyödyntää ohjelmistoarkkitehtuurin kuvausta lisenssirikkomuksen korjaamisessa. Kolmasosa kuitenkin sanoo, että ohjelmistoarkkitehtuuria voi käyttää apuna lisenssien hallinnassa rikkomusten korjauksessa ja ehkäisyssä, mutta ei osaa tarkemmin määritellä miten.

3.2.6 Ohjelmistokehitys

Noin puolet projekteista tarkistaa lisenssin käyttöönoton yhteydessä. Suurin osa ei kuitenkaan tarkasta riippuvuuksien lisenssejä. Neljäsosa tarkastaa lisenssejä työkalun avulla.

Projekteissa ei yleisesti hyödynnetä auditointeja yhtä lukuun ottamatta. Projektissa auditointi on kuitenkin satunnainen erikoispalvelu, joten tieto mahdollisesta ongelmasta saattaa tulla kuukausia rikkomuksen jälkeen, mikä on todella hidas takaisinkytkentä ketterään kehitykseen.

Kolme neljästä kehittäjästä uskoo, että kehittäjä voi ehkäistä lisenssiyhteensopivuusongelmia, mutta ei välttämättä tiedä miten. Neljäsosa ei osaa sanoa tai ei usko, että kehittäjä voi ehkäistä lisenssien yhteensopivuusongelmia. Jos ongelma lisenssien yhteensopivuudessa tulisi ilmi, puolet vaihtaisi ja korvaisi lisenssiä loukkaavan komponentin pois. Muista korjaustoimenpiteistä ei ole tietämystä.

3.2.7 Vapaa sana

Muodollisten haastattelujen lisäksi haastateltavat nostivat esiin muutamia näkökulmia. Avoimen lähdekoodin lisenssien ehtojen tarkka selvittäminen ja noudattaminen nähtiin joidenkin haastateltavien mielestä turhana. He totesivat, että lisensseistä aiheutuva tekijänoikeusloukkaus on vaikea havaita. Henkilöt arvioivat myös olevan epätodennäköistä, että tekijänoikeudenhaltija olisi valmis lähtemään oikeuskäsittelyyn. Osa haastateltavista taas koki että aiheesta ei yksinkertaisesti ole saatavilla riittävästi tietoa varsinkin julkishallinnon palvelu- ja ohjelmistokehityksen näkökulmasta ja toivoivat yhteisiä koulutuksia tai käytäntöjä.

Julkishallinnon avoimen lähdekoodin käyttö ja kehitys koettiin kuitenkin sekä toimittajien että tilaajien mielestä erittäin positiiviseksi asiaksi. Ilman nopeasti kehittyviä avoimen lähdekoodin komponentteja ei esimerkiksi moderneihin pilvipalveluihin perustuvia ratkaisuja yksinkertaisesti olisi mahdollista ottaa käyttöön kustannustehokkaasti.

Haastatteluissa kävi myös ilmi, että kaikki haastateltavat eivät ymmärrä, että avoimen lähdekoodin lisenssiehdot eivät juuri vaikuta tietojärjestelmätasolla. Open Source Initiativen avoimen lähdekodin

lisenssin määritelmän mukaan avoimen lähdekoodin lisenssi ei voi rajoittaa lisensoidun koodin käyttöä. Lisenssiehdot saavat rajoittaa muokkaamista ja levittämistä. Tämän vuoksi avoimen lähdekoodin lisensseillä ei ole vaikutusta tietojärjestelmätasolla kuin ohjelmistokehitykseen liittyen.

4. Lisenssien hallinnan ratkaisut ja työkalut

4.1 Menetelmät

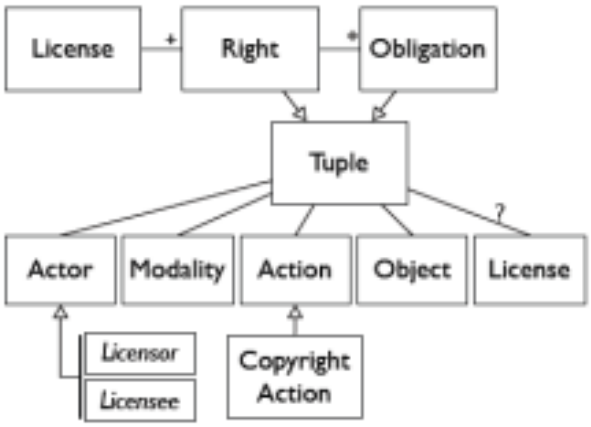
Seuraavaksi esitellään kartoitus avoimen lähdekoodin lisenssien hallinnan menetelmistä.

4.1.1 Koulutus

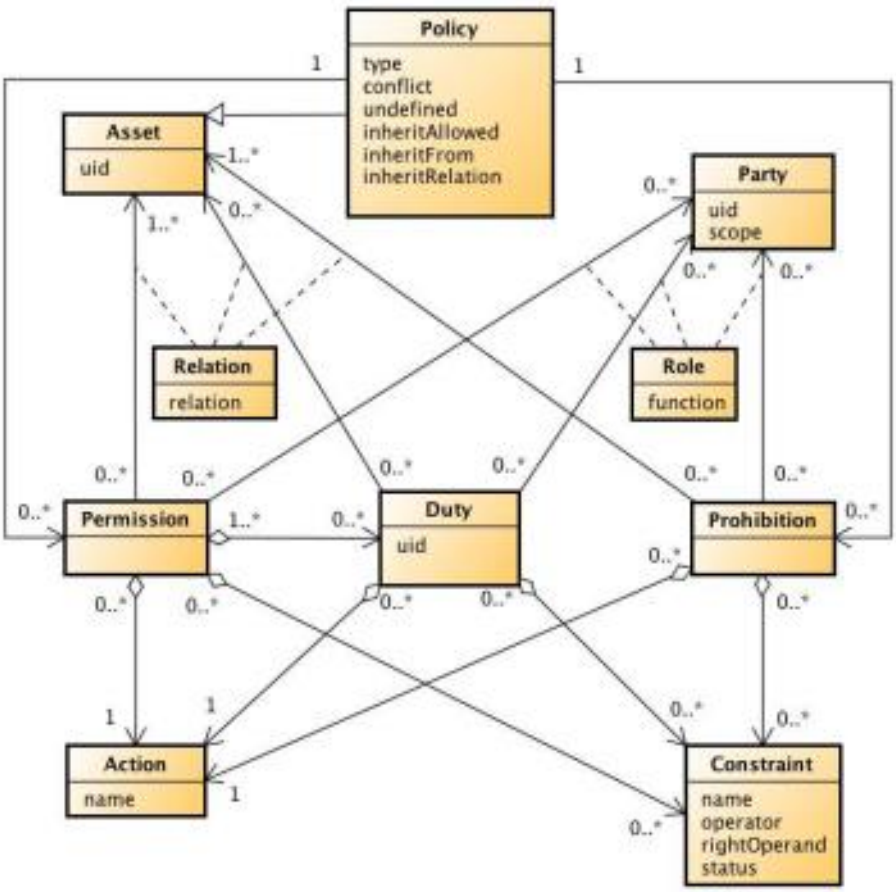
Tietoisuus avoimen lähdekoodin tekijänoikeusriskeistä ja niihin vaikuttavaista tekijöistä mahdollistaa sen, että kehittäjät ovat osaavat varoa itse tekemästä lisenssiyhteensopivuusvirhettä. Myös tietoisuus heikosti ylläpidettyjen tai vähän käytettyjen ohjelmistokomponenttien potentiaalisesta tekijänoikeusriskeistä mahdollistaa muiden tekijänoikeusloukkauksien välttelyn tai ainakin monipuolisemman arvioinnin käyttöönoton yhteydessä.

4.1.2 Lisenssien mallintaminen

Lisenssien mallintaminen on tarkoittaa lisenssien analyysia ja purkua helpommin ymmärrettävään muotoon. Formaaleja malleja on sekä esitelty tieteellisissä julkaisuissa, että XML-pohjaisina tiedonkuvaus ja -siirtoformaatteina. Mallintamisessa lisenssitekstistä kaivetaan esiin sen myöntämät oikeudet ja sen vaatimukset. Tämän jälkeen voidaan tämän yksinkertaisemman mallin perusteella helpommin arvioida lisenssin soveltuvuutta käyttöön. On myös mahdollista viedä mallinnus vielä tarkemmalle tasolle ja esimerkiksi piirtää yhteydet vaatimuksien välille, mitkä näyttävät vaatimukset, jotka liittyvät mihinkin oikeuteen. Monimutkaisemman mallin perustella voi siten arvioida tarkemmin lisenssin sopivuutta käyttöön. Akateemisten mallien lisäksi esim. ccREL (https://wiki.creativecommons.org/wiki/CC_REL) ja ODRL (https://www.w3.org/community/odrl/) kuvauskieliä voi käyttää hyväksi. XML-pohjaiset kuvaukset myös sopivat automaattisen lisenssien hallinnoinnin pohjaksi. Esimerkit lisenssimalleista on esiteltynä kuvissa 4 ja 5.



Kuva 4 Lisenssimalli (Alspaugh et al. http://dx.doi.org/10.1109/RE.2009.22)



Kuva 5 ODRL 2.0 lisenssimalli (http://www.w3.org/community/odrl/two/model/)

Lisenssien mallintaminen on hyödyllisintä silloin, kun haluaa ymmärtää uutta lisenssiä, jota ei ole tullut ennen vastaan. Myös tuttujen lisenssien mallintaminen on hyvä työkalu lisenssien hallinnan oppimiseen.

4.1.3 Auditointi

Komponenttien auditointi tarkoittaa yksinkertaisesti koodin tutkimista. Komponentin ja sen riippuvuuksien lähdekoodeista etsitään merkintöjä lisensseistä ja tekijänoikeuksista. Myös kopioitua koodia voidaan etsiä. Vaikka auditointi voidaan tehdä manuaalisesti, on siihen kehitetty tueksi työkaluja. Työkalut auttavat etsinnässä, sillä varsinkin kopioidun koodin etsiminen manuaalisesti voi olla varsin tehotonta. Periaatteessa auditoinnin voi tehdä milloin vain, mutta se on hyödyllisintä tehdä ennen uuden komponentin käyttöönottoa. Kaikkia komponentteja ei ehkä tarvitse auditoida, mutta jos vähemmän suosittu komponentti sisältää palvelun kannalta oleellisen toiminnallisuuden, voi olla hyödyllisempää auditoida se, kuin jokin suosittu tai vähemmän kriittinen komponentti. Myös olemassa olevan järjestelmän lähdekoodi voidaan auditoida esimerkiksi ennen julkaisua tai käyttöönottoa. Komponentin tekijänoikeus auditoinnin tulokset voi kirjat esimerkiksi SPDX-kielellä. (https://spdx.org/)

Jos tekijänoikeusauditoinnin yhteydessä käydään läpi myös riippuvuuksien versiotiedot, on luonnollista, että samalla tarkistetaan myös, onko näissä komponenteissa tunnettuja tietoturvahaavoittuvaisuuksia.

4.1.4 Ohjeistus

Ohjeistus lisenssien hallinnan menetelmänä tarkoittaa sitä, että kehitettävän ohjelmistoon liitettäville komponenteille asetetaan lisenssivaatimus. Lisenssivaatimukset voivat olla joko yksikäsitteisiä tai tulkinnanvaraisia. Yksikäsitteisyys tarkoittaa, että käytetään vain tiettyjä lisenssejä. Tulkinnanvaraisuus tarkoittaa sitä, että käytetään vain sallivia lisenssejä tai käytetään vain valitun oman lisenssin kanssa yhteensopivia lisenssejä. Monitulkintainen ohjeistus vaatii ohjelmistokehittäjiltä enemmän osaamista, jotta he osaavat soveltaa sitä, mutta tällöin komponenttivalinta voi olla vapaampaa. Oleellista on, että asiakkaan kehittäjille antama ohjeistus tai kehitystiimin itse

määrittelemä lisenssinvalinnan ohjeistus on yhteensopiva liiketoiminnan tarpeiden kanssa.

Lisenssiohjeistuksen perusteena on yleensä yhteensopivuus oman koodin julkaisuun käytettävän lisenssin kanssa. Näin voidaan muodostaa lista tunnetuista sallituista lisensseistä. Kypsempi malli perustuu siihen, että tunnistetaan liiketoiminnan tavoitteet ja analysoidaan, mitkä tunnetut lisenssit ovat yhteensopivia liiketoimintatavoitteiden kanssa ja yhteensopivia oman koodin julkaisulisenssin kanssa.

Ohjeistuksen ei tarvitse rajoittua vain käytettyyn lisenssiin. Ohjeistus voi myös määrittää, että käytetty komponentin haetaan vain tietyistä avoimen lähdekoodin jakelukanavista, tai että komponentin pitää olla riittävän suosittu tai riittävän kypsä. Jälkimmäisen avulla minimoidaan riskiä, että lähdekoodissa olisi piilossa muita lisenssejä tai tekijänoikeusrikkomuksia, sillä pitkään käytössä olleen tai suositun komponentin koodia on useampi käyttäjä tutkinut. Mikäli suositusta komponentista löytyy ongelma, on myös suurella määrällä kehittäjiä tarve korjata se, jolloin ratkaisu ongelmaan muodostuu nopeammin.

4.1.5 Dokumentointi

Lisenssien hallintamenetelmien dokumentointi ja lisenssien hallintaan liittyvien päätösten dokumentointi on hyödyllistä, mikäli haluaa osoittaa, että mahdollinen tekijänoikeusloukkaus ei ole ollut tahallinen tai johtunut välinpitämättömyydestä. Dokumentaatio mahdollistaa myös lisenssien hallinnan käytäntöjen kommunikoimisen projektiin liittyville uusille kehittäjille. Dokumentaatiota voidaan käyttää myös lisenssinhallintamenetelmien kehittämiseen.

4.1.6 Toimintojen erottelu

Modulaarinen ohjelmistoarkkitehtuuri helpottaa komponenttien vaihtamista. Moni ohjelmistokehittäjä osaakin poistaa tekijänoikeusrikkomuksen aiheuttavan komponentin. Tapauksesta riippuen komponentin korvaaminen toisella voi olla pieni työ ja joskus taas korvaava toiminto pitää implementoida itse.

Jossain tapauksissa on mahdollista muokata tietojärjestelmää siten, että toiminnallisuus siirretään ohjelmiston ulkopuolelle ja sitä

käytetään jonkin yleisen rajapinnan kautta. Esimerkiksi joidenkin vastavuoroisten lisenssien yleinen tulkinta mahdollistaa sen, että niiden toiminnallisuuksia kutsutaan käyttöjärjestelmän järjestelmäkutsun kautta. Tällöin on mahdollista, että ohjelmisto jaellaan kahtena eri pakettina, jolloin vain se osa, joka sisältää vastavuoroisen lisenssin, on vastavuoroisuusehtovaatimuksen alainen.

Toiminnallisuuden siirtäminen toiselle palvelimelle tai mikropalveluun voi olla eräs ratkaisu lisenssikonflikteihin, mikäli lisenssin SaaS-ehto aiheuttaa yhteensopivuusongelmia. Verkkopalvelurajapinnan kautta tehdyt kutsut eivät ole suoria riippuvuuksia, jotka kantaisivat vastavuoroisuusehtoa ja vain se ohjelma, joka sisältää vastavuoroisen lisenssin, pitää julkaista vasavuoroisen lisenssin kanssa yhteensopivalla lisenssillä.

Taulukossa 1 on vielä esiteltynä kootusti ratkaisumalleja lisensseiltään yhteensopimattomien komponenttien integraatioon.

Malli	**Ongelma**	**Kuvaus**
Käyttäjä integroi	Kompositio	Jaellaan ohjelmisto kahtena osana eri lisensseillä ja käyttäjä yhdistää toiminnallisuudet lopulliseen ratkaisuun
Interaktion muuttaminen	Interaktio	Muutetaan komponenttien välistä linkityksen tasoa löyhemmäksi
Tarjoaminen palveluna	Komponentti Interaktio	Tarjotaan toisen komponentin toiminnallisuus erillisessä palvelussa yleiskäyttöisen rajapinnan läpi
Välityskomponentti	Interaktio	Komponentit eivät keskustele suoraan toistensa kanssa, vaan kolmas komponentti välittää niiden tarvitsemat ja tuottamat tiedot toisilleen

Taulukko 1: Ratkaisumalleja lisensseiltään yhteensopimattomien komponenttien integraatioon.

4.1.7 Useita lisenssejä

Mikäli ohjelmisto on modulaarinen, on mahdollista, että sen eri osat julkaistaan eri lisensseillä. Tämä luo lisenssien hallintaan kompleksisuutta, mutta voi edistää komponenttien jatkokäyttöä, jos kukin komponentti on julkaistu sille sopivimmalla lisenssillä. Tällä tavalla voidaan välttyä järjestelmän lähdekoodin julkaisemiselta. Mikäli se palvelee liiketoiminnan tavoitteita, voidaan julkaista vain sopivia osia.

Sen lisäksi että eri osat tietojärjestelmästä voidaan julkaista eri lisensseillä, voidaan itse tehty komponentti tai kirjasto julkaista useammalla kuin yhdellä lisenssillä. Tämä voi auttaa ratkaisemaan joitain lisenssien ristiriitatilanteita. Tällöin komponentin tai kirjaston jatkokäyttäjä saa itse päättä kumpaa lisenssiä hän käyttää.

4.1.8 Kontribuutiosopimus

Mikäli tuotettu ohjelmisto ottaa vastaan koodikontribuutioita, voidaan käyttää erillistä kontribuutiosopimusta. Kontribuutiosopimuksessa voidaan vaatia, että kontribuution tekijä ei lisää tekijänoikeusloukkauksia sisältävää koodia tai riippuvuuksia kontribuutionsa kautta, tai että hänen lisäämänsä koodi tai riippuvuudet eivät sisällä epäyhteensopivia lisenssejä. Mikäli sopimuksessa on lisäksi ehto, että ohjelmiston hallinnoija voi neuvotella kontribuution tekijän puolesta lisenssimuutoksista tai poikkeuksista, voi ohjelmiston jatkokäyttö ollut houkuttelevampaa.

On kuitenkin otettava huomioon, että kontribuutiosopimukset luovat ylimääräisen kerroksen hallintoa, mikä itsessään saattaa vähentää ulkopuolisten kehittäjien intoa osallistua. Kontribuutiosopimukset eivät ole kovin yleisiä, ja niitä käytetään usein suuremmissa hankkeissa, joissa on selkeät kaupalliset intressit. Projektin kehittäjien kuuluu joka tapauksessa katselmoida kaikki kontribuutiot ennen kuin ne liitetään palveluun, joten pätevällä kehittäjällä on hyvä mahdollisuus tunnistaa lisenssikonfliktiriskejä. Osa toimijoista valmiita hyväksymään sen, että avoimia kontribuutioita jää saamatta, sillä kontribuutiosopimus lisää yhden tason turvaa avoimien kontribuutioiden riskejä vasten.

4.1.9 Neuvottelu

Mikäli jonkin komponentin kanssa on lisenssiristiriita ja komponentin tekijänoikeudenhaltija on tavoiteltavissa, on

mahdollista yrittää neuvotella tekijänoikeudenhaltijan kanssa komponentin lisensoinnista toisella lisenssillä tai mahdollisesta poikkeuksesta lisenssin ehdoista.

4.1.10 Yhteensopivuustaulukot

Lisenssien yhteensopivuuksia analysoivia taulukoita on saatavilla Internetissä useita. Ne ovat helppokäyttöisä ja selkeitä, mutta niitä ei välttämä ole tutkittu Suomen lain mukaan tai julkishallinnon ohjelmistojen liiketoimintatarpeiden näkökulmasta. On myös huomioitava, että osa yhteensopivuustauluista on avoimen lähdekoodin harrastajien tekemiä. Harrastajat eivät välttämät ole lainoppineita, joten käytettäessä taulukoita on niiden lähdettä arvioita huolellisesti.

4.2 Työkalut

Seuraavaksi esitellään avoimen lähdekoodin lisenssien hallinnan työkalujen kartoituksen tulokset.

4.2.1 ASLA

ASLA (Automated Software License Analyzer) on Realtorin kehittämä työkalu, joka hyödyntää staattista analyysiä tiedostojen lisenssi- ja copyright-tiedon havaitsemiseen ASLA takaisinmallintaa ohjelma-arkkitehtuurin perustuen kääntäjän kykyyn havaita dynaamisia ja staattisia linkkejä lisenssien välillä ja analysoida mahdollinen lisenssikonflikti ennalta määritellyn sääntöjoukon avulla.
(http://dx.doi.org/10.1109/CSMR.2006.10)

4.2.2 DCT Dependency checker tool

DCT (Dependency checker tool) on Linux Foundationin työkalu, joka analysoi linkkejä binaarien välillä. Se käyttää pakettienhallintaohjelmasta saatavaa lisenssitietoa potentiaalisesti ristiriitaisten linkitysten havaitsemiseen lisenssien välillä. Säännöt lisenssien ja linkkien konflikteille ovat Linux Foundationin kehittämät.
(http://www.linuxfoundation.org/programs/legal/compliance/tools)

4.2.3 Debian's licensecheck (in devscripts)

Debian linuxin kehittäjätyökaluihin kuuluva skripti, jonka tavoitteena on tunnistaa ohjelmistojen lisenssitiedot lähdekoodista tai käännettyjen pakettien sisältä.

(https://packages.debian.org/unstable/devscripts)

4.2.4 FOSSology

FOSSology hakee ohjelmiston lähdekoodista tiedostoja ja paketteja, jotka sisältävät copyright- ja lisenssitietoa. Työkalua voidaan käyttää tietokannan rakentamiseen tiedostojen ja pakettien copyright-tiedoista. Tietokannassa käytetään notaationa SPDX:ää tai jotain muuta ennalta määriteltyä skeemaa. FOSSology on avoimen lähdekoodin projekti, jolla on monia suuria teollisia käyttäjiä. FOSSologyyn voi myös integroida Ninka-työkalun.
(http://www.fossology.org/projects/fossology)

4.2.5 HUT OSLC

HUT OSLC on lähdekoodin analysointityökalu. Tästä Helsingin yliopistossa kehitetystä Open Source Licence Checkeristä on tarjolla muutama erilainen versio ja ominaisuudet näissä versioissa eroavat jonkun verran. Työkalu tunnistaa staattisen analyysin avulla lähdetiedostoista lisenssi- ja copyrighttietoa. Työkalu hyödyntää ennalta määritettyä yhteensopivuustaulua lisenssikonfliktien havaitsemiseen. (https://sourceforge.net/projects/oslc/)

4.2.6 Markos

Markos on avoimeen lähdekoodiin perustuva työkalu, jolla voidaan muodostaa versionhallintatietovarannoista RDF-pohjaiset kuvaukset arkkitehtuurista ja tekijänoikeustiedoista. Markos voi tehdä RDF-mallista lisenssikonfliktianalyysin, mutta siinä on lisäksi myös rajapinta, jonka kautta voidaan tehdä SPARQL-kielellä kyselyitä ja käyttää niitä omina lisenssikonfliktin tunnistamissääntöinä. Markos on prototyyppiasteella.
(http://markosproject.sourceforge.net/final-prototype/)

4.2.7 Ninka

Ninka havaitsee lisenssejä ja kopioitua koodia lähdekooditiedostoista. Ninka tarvitsee pääsyn versionhallintatietovarantoon, johon verrata koodia. Työkalu raportoi lähdekoodin kaksoiskappaleista, jos koodi muistuttaa julkaistua avointa lähdekoodia. (http://ninka.turingmachine.org/)

4.2.8 Npm license checker

Npm-license checker hakee npm-pakettien lisenssitiedot pakettien metatiedoista ja lähdekoodista. Se hakee metatiedoista myös

riippuvat paketit, sekä näiden metatietojen ja lähdekoodin pohjalta lisenssitiedot. Näin se muodostaa rekursiivisesti kuvauksen kaikista käytetyistä lisensseistä ja riippuvuuspuusta. (https://www.npmjs.org/package/license-checker)

4.2.9 Qualipso Carneades

Qualipso Carneades on työkalu, jota käytetään lisenssikonfliktien päättelyyn ohjelmistoarkkitehtuurissa olevien lisenssimallien avulla. Lisenssimallit on kuvailtu käyttäen LKIF-OWL:ia ja arkkitehtuuri kuvaukset on muutettu OWL:iin UML:stä. Näitä OWL malleja prosessoidaan Carneadesin logiikkatoiminnolla hyödyntäen OWL-mallinnettua sääntöjoukkoa ristiriitojen havaitsemiseksi. (https://github.com/carneades/carneades-3/tree/master/examples/LKIF/Argument%20Graphs/open_source_licensing)

4.2.10 Qualipso OSLC Open Source License Checker

Qualipso OSLC havaitsee lisenssejä lähdekooditiedostoista ja lähettää ne web-palveluun konfliktien tarkistamista varten. Web-palvelun käyttämää konfliktimekanismia ei ole kuvailtu. (http://dx.doi.org/10.1109/CISE.2010.5676875)

4.2.11 Scancode

ScanCode on koodianalyysityökalu, joka tunnistaa lisenssi- ja tekijänoikeustietoja sekä lähdekoodista, että käännetyistä paketeista. ScanCode julkaisee löytämänsä tiedot JSON tai HTML muodossa. (https://github.com/nexB/scancode-toolkit/wiki)

4.3 Palvelut

Seuraavaksi esitellään kartoitus avoimen lähdekoodin lisenssien hallinnan palveluntarjoajista ja niiden tarjoamista palveluista.

4.3.1 Antepedia

Antepedia on web-palvelu, jota voidaan käyttää avoimen lähdekoodin komponenttien lisenssidokumentaation, kuten lisenssi- ja tekijänoikeustiedon etsimiseen. Julkisten projektien lisäksi palvelun avulla voidaan etsiä tietoa myös sisäisten komponenttien lisensseistä. (http://www.antepedia.com/)

4.3.2 Black Duck Protex

Black Duck tarjoaa palveluita, joilla voidaan turvata ja hallinnoida avoimen lähdekoodin ohjelmistoa. Heidän avullaan voidaan tutkia ohjelmistojen koostumusta, etsiä uudelleenhyödynnettävää lähdekoodia, hallinnoida oikeuksia ja löytää niihin liittyviä turvallisuuspuutteita. Black Duck Protex on ratkaisu, jolla voidaan hallinnoida avoimen lähdekoodin lisenssien noudattamista. Se integroituu kehitystyökaluihin ja automaattisesti skannaa, tunnistaa ja varastoi avoimen lähdekoodin ohjelmistoa. Lisäksi se analysoi lisenssien vaatimukset, konfliktit ja riskit. Riskejä voidaan hallita palvelulla, sillä löydettyjen tietojen avulla saadaan selville lisenssienmukaisuus ja yrityksen käytäntövaatimukset. (https://www.blackducksoftware.com/products/protex)

4.3.3 Dejacode

Dejacode tarjoaa palvelua, jonka avulla voidaan hallinnoida ohjelmistokomponentteja ja lisenssitietoa sekä sisäisistä, että ulkoisista lähteistä, toimittajilta ja sopimuskehittäjiltä. Palvelussa on kolme pääasiallista ominaisuutta: tuoteportfolion, komponenttien ja lisenssien hallinta. Tuoteportfolion avulla voidaan hallinnoida ohjelmistokäytänteitä, lisenssitietoa ja komponentteja yhdellä alustalla. Näin voidaan tukea komponenttien uudelleenkäyttöä ja lisenssivaatimusten täyttymistä. Komponenttien hallinnan avulla voidaan varastoida ja hyödyntää komponenttien metadata, alkuperä, lisenssi, teknologia ja toiminnallisuus. Lisenssien hallinnan työkalu on listaus avoimen lähdekoodin ja luvanvaraisista lisensseistä, sekä niiden käyttöehdoista. Työkalun avulla voidaan hallinnoida lisenssejä yhdessä paikassa. (http://www.dejacode.com/index.html)

4.3.4 OpenLogic

OpenLogic on Rogue Wave Softwaren tuotemerkki, joka tarjoaa organisaatioille työkaluja avoimen lähdekoodin lisenssien hallintaan. OpenLogic tarjoaa myös paljon muita palveluita, kuten analytiikkaa ja tietoturvallisuutta. Avoimen lähdekoodin hallinnan palvelun avulla voidaan ymmärtää miten ja missä avointa lähdekoodia hyödynnetään organisaatiossa. Sen avulla voi myös ohjata lisenssinvalintaa lajittelemalla lisenssit hyväksyttyihin, kiellettyihin ja hyväksynnän vaativiin ja seurata näiden toteutumista. Palvelun avulla voidaan koostaa raportti organisaatiossa käytetyistä avoimen lähdekoodin kokonaisuuksista ja komponenteista, sekä niissä hyödynnetyistä

lisensseistä. Palvelussa voidaan myös luoda lisenssivaatimuksiin perustuva lista tunnistamalla lisenssit, niiden vaatimukset, käyttötapaukset ja tuomalla esiin mahdolliset yhteensopimattomuudet. (http://www.openlogic.com)

4.3.5 Palamida

Palamida tarjoaa palveluita kolmansien osapuolien ja avoimen lähdekoodin lisenssien hallintaan ja turvallisuuteen. Sen avulla voidaan analysoida lähde- ja binaarikoodia ja löytää niistä lisenssi- ja tekijänoikeustietoa sekä muualla raportoituja tietoturvaongelmia. Löydetyt lisenssit näytetään tiedostopuussa, jonka avulla voidaan tarkastella tiedostoja, jotka sisältävät lisenssin. Palvelun avulla voidaan myös hallita lisenssien hyväksyntää organisaatiossa joko automaattisesti tai manuaalisesti. Palvelun avulla voidaan lisäksi koostaa auditointiraportti kolmansille osapuolille ohjelmiston luovuttamista varten. (http://www.palamida.com)

4.3.6 TripleCheck

TripleCheck tarjoaa palvelua, joka etsii annetusta lähdekoodista metatietojen avulla lisenssejä ja toisaalta taas vertaa lähdekoodia suureen avoimen lähdekoodin komponenttien tietokantaan. Tuloksena on raportti, joka kertoo, miten hyvä lisenssien hallinnan laatu on. Jos lisensoinnissa on puutteita tai ristiriitoja, raportti antaa ehdotuksen sen korjaamiseen. TripleCheck tarjoaa lisäksi auditointipalvelua kertaluontoiseen selvitykseen. (http://triplecheck.net)

4.3.7 Protecode Enterprise System

Protecode Enterprise System on ohjelmistopalvelu, joka tarjoaa useita työkaluja lisenssien hallintaan organisaatioissa. Protecode hyödyntää omaa tietokantaansa tunnetuista avoimen lähdekoodin projekteista ja tunnistaa sen avulla annetun avoimen lähdekoodin komponentit, niiden lisenssi- ja tekijänoikeustiedot, sekä lisenssien vaatimukset erilaisissa käyttötapauksissa. Protecode tarjoaa laajan valikoiman erilaisia auditointiraportteja. Niihin voidaan koostaa tunnistetut ohjelmiston komponentit ja lisenssivaatimukset tai toimenpiteet lisenssien yhteensopivuuden varmistamiseksi tunnistaen lisenssien yhteensopimattomuudet. Palvelun avulla voidaan etsiä myös tunnettuja tietoturvariskejä ja salata lähtevät tiedot. (http://www.protecode.com)

4.3.8 Validos

Validos on yhdistys yrityksille ja yhteisöille, jotka hyödyntävät avointa lähdekoodia toiminnassaan. Yhdistyksen jäsenet voivat pyytää ohjelmistokomponenteille oikeudellista validointia, joka lisätään yhdistyksen tietopankkiin ja on tämän jälkeen kaikkien jäsenten käytettävissä. Validointiin sisältyvät tarkastukset voidaan jakaa neljään luokkaan, jotka ovat paketin kytkös päälisenssiin, ohjelmistokomponentin sisältämät alakomponentit, ohjelmistopaketin lähdekoodin läpikäynti työkalun avulla ja tekijöiden ja tekijänoikeudenhaltijoiden dokumentointi. Tuotoksena muodostetaan raportti, sekä ohjeet jäsenille komponentin hyödyntämiseksi yrityksissä. Ohjeissa keskitytään erityisesti komponentin jakamiseen eteenpäin, joko sellaisenaan tai muokattuna, osana suurempaa kokonaisuutta tai sellaisenaan. Ohjeissa kerrotaan myös, mitä lisenssiehtoja tai muita tekstiä tulee sisällyttää dokumentaatioon ja edelleen jaettaessa. (http://www.validos.org)

4.3.9 VersionEye

VersionEye on pakettienhallintaohjelmistoihin perustuva palvelu. Pakettienhallintaohjelmiston tiedostoja voidaan analysoida GitHubissa tai Bitbucketissa. VersionEye tutkii riippuvuuksia ja tunnistaa vanhentuneet riippuvuudet. Näin tunnistetaan lisenssit ja voidaan ilmoittaa saatavilla olevista uusista versioista. Lisäksi tunnistetaan lisenssirikkomukset ja tietoturvauhkat. (https://www.versioneye.com)

4.3.10 WhiteSource

WhiteSource on palvelu, joka muodostaa jatkuvasti listaa käytetyistä avoimen lähdekoodin komponenteista tuotteittain huomioiden riippuvuudet. Tätä listaa täydennetään WhiteSourcen tietokannalla, joka sisältää ohjelmistokomponenttien lisenssitiedot. Palvelu tarjoaa näiden tietojen perusteella analyysin riskeistä ja lisenssien rajoitteista, kuten myös korjausehdotukset mahdollisiin puutteisiin lisenssien vaatimusten noudattamisessa. Palvelu tarjoaa informaatiota myös versioinneista ja tietoturvariskeistä. (http://www.whitesourcesoftware.com)

4.3.11 Windriver

Windriver on julkinen pilvipalvelu, jonka avulla voidaan tutkia avoimen lähdekoodin paketteja. Työkalun avulla voidaan luoda SPDX-tiedostoja paketeista. Palvelu hyödyntää algoritmeja ja heuristiikkoja lisenssi-informaation havaitsemiseen kaikista tiedostostoista perustuen yksinomaan vain tiedoston sisältämään informaatioon. (http://spdx.windriver.com/)

5. Toimenpide-ehdotukset

5.1 Riskianalyysi

Tässä osiossa esitellään nykyisten käytäntöjen riskit ja seuraukset sekä niistä aiheutuvat mahdolliset julkishallinnon tuottamien ohjelmistojen jatkokäytön estävät riskit.

Haastattelujen perusteella sekä asiakkaan, että toimittajien puolella on epäselvää, mitä vaikutuksia ohjelmistosta löytyvästä tekijänoikeusloukkauksesta olisi. Pahimmat seuraukset olisivat palvelun tai ohjelmiston poistuminen käytöstä oikeuden tai palveluntarjoajan toimesta. Toiseksi suurimmat vaikutukset olisivat oikeuskäsittelyn tai korjauksen vaatimat työmäärät, sekä näiden vaikutukset asiakkaan ja toimittajan toimintakykyyn. Suomalaisessa oikeuskäsittelyssä määrätyt korvaukset tekijänoikeusloukkauksesta ovat pienet, ellei kyseessä ei ole komponentin levittäminen, mitä julkishallinnon hallinnon palveluissa ei yleensä tehdä. Poikkeuksena on tietysti selaimessa ajettava koodi, mutta selaimessa ajettava koodi on yleisesti lisensoitu sallivilla lisensseillä, eikä komponenteilla tai kehyksillä ei ole erillisiä kaupallisia versioita, joiden perusteella komponentin hinnan voisi määrittää.

Haastatteluissa kävi ilmi, että monet haastatelluista pelkäävät mahdollisen rikkomuksen vaikuttavan negatiivisesti maineeseen. Mainevaikutukset eivät kuitenkaan ole julkishallinnon palvelukehityksessä vakava seuraamus, sillä suurimmalla osalla palveluista ei ole kilpailijoita. Tällöin asiakkailla ei ole mahdollisuutta vaihtaa kilpailevaan tuotteeseen. Lisäksi myöskään avoimen lähdekoodin yhteisöihin ei ole yleensä ole suhteita, joita huono maine voisi pilata. Vakavimmat seuraukset loukkauksesta on koottu vielä alla olevaan taulukkoon 2.

Tekijänoikeusloukkauksen vakavimmat seuraukset
Palvelun tarjoaja sulkee käyttäjätilin, joka liittyy epäiltyyn tekijänoikeusloukkaukseen ja tiliin liittyvät palvelut eivät ole käytössä.

Palvelun tarjoaja sulkee palvelun, joka liittyy epäiltyyn tekijänoikeusloukkaukseen.
Epäillyn **tekijänoikeusloukkauksen korjaamiseen** vaadittava työmäärä.
Epäillyn **tekijänoikeusloukkauksen oikeuskäsittelyyn** vaadittava työmäärä.

Taulukko 2: Tekijänoikeusloukkausien vakavimmat seuraukset.

Korkeammalla tasolla ongelmana julkishallinnossa on, että asiakas ei aina osaa ottaa omistajuutta lisenssien hallinnasta. Lisenssien hallinnassa ensimmäinen kulmakivi on se, millä lisenssillä oman palvelun lähdekoodi julkaistaan vai julkaistaanko olleenkaan. Asiakas on aina liiketoiminnan asiantuntija ja liiketoiminnan asiantuntijaa tarvitaan, koska oman koodin julkaisuun käytettävän lisenssin pitää sopia liiketoiminnan tavoitteisiin. Mikäli julkaisuun käytettävää lisenssiä valittaessa on tiedossa, että jokin komponentti tai kehys on olennainen osa tuotettua palvelua, on myös näiden lisenssien yhteensopivuus otettava huomioon.

Mikäli asiakas ei osaa määrittää projektille lisenssien hallinnan tarpeita on täysin kehittäjien oman aktiivisuuden varassa, miten asiat hoidetaan. Vaikka toimittaja haluaisikin hoitaa lisenssien hallinnan kunnolla ja sitä sopimuksessa vaaditaan, ei jokaiselle kehittäjällä ole kompetenssia ottaa avoimen lähdekoodin lisenssien ehtoja huomioon työssään ilman ohjausta. Haastatteluissa kävi myös ilmi, että asiakkaat epäilevät, että tekijänoikeusloukkauksesta kärsisi sopimuksen ansiosta vain toimittaja, joka joutuisi tekemään korjaukset, maksamaan sopimussakkoja ja saattaisi menettää sopimuksen. Tosiasiassa nämä vaikutukset tulisivat voimaan vasta sen jälkeen kun seuraamukset olisivat tulleet asiakkaalle.

Lisensseihin liittyvien tekijänoikeusloukkauksen suurimmat riskit liittyvät oikeuskäsittelyehtoon, nimeämisehtoihin ja SaaS-ehtoon. Mikäli lisenssiloukkauksen oikeuskäsittely tapahtuu jossain muualla kuin Suomessa, oikeuskäsittelyn kustannuksia sekä mahdollisen korvauksen määrää on melkein mahdotonta arvioida. Osa nimeämisehdoista kieltää tekijänoikeudenhaltijan tietojen

julkaisemisen palvelussa, kun taas osaa vaatii sitä. Esimerkiksi BSD-lisenssin eri versioissa ovat nämä vastakkaiset ehdot. Tämän vuoksi nimeämisehdot on käytävä tarkasti läpi ja tekijänoikeudenhaltijoiden nimet on julkaistava sopivassa paikassa niin vaadittaessa. Suurin osa julkishallinnon ohjelmistokehityksestä on verkkopalvelujen kehittämistä, jolloin muokattua koodia ei jaeta eteenpäin ja esimerkiksi GPL-lisenssin vastavuoroisuusehto ei tule voimaan. SaaS-ehto kuitenkin on voimassa silloin kun ohjelmisto tarjoaa julkista palvelua. SaaS-ehto liittyy kuitenkin yleensä tunnettuihin vastavuoroisiin lisensseihin, joten se on helppo tunnistaa. Tällöin SaaS-ehdon sisältävän mikropalvelun tai palvelun koodi pitää julkaista yhteensopivalla avoimen lähdekoodin lisenssillä, mitä ei haastattelujen perusteella olla sisäistetty. Tämä on merkittävä kehityskohde julkishallinnossa.

Riskialttiimmat lisenssiehdot on vielä koottu alla olevaan taulukkoon 3.

Lisenssiehdot	**Riski**
SaaS-ehto	Jos lisenssissä on SaaS-ehto, on lisenssin vastavuoroisuusehto voimassa kun palvelu tarjotaan julkisesti käyttöön, vaikka koodia ei jaeta.
Oikeuskäsittelyehto	Ulkomailla käyty oikeuskäsittely vaatii paljon enemmän resursseja ja korvaukset voivat olla huomattavasti suuremmat kuin suomessa.
Nimeämisehto	Nimeämisehdot ovat ristiriitaisia, joten niiden noudattamisessa vaaditaan tarkkuutta.

Taulukko 3: Riskialtteimmat avoimen lähdekoodin lisenssiehdot.

5.2 Suositeltavat toimenpiteet

Tässä osioissa esitellään tutkimuksen perusteella asiakkaan tarpeisiin suositeltavat menetelmät ja työhön käytettävät työkalut.

Myös suositusten kustannusarviot tuodaan esiin sekä esitellään etenemisehdotus suositeltujen menetelmien integroimiseksi avoimen tuotteen hankintamalliin. Projektikohtaiset kehitysehdotukset on listattu liitteissä.

Ensisijainen kehitysehdotus on avoimen lähdekoodin hallinnan koulutuksen lisääminen. Toimittajien tarjoamat ohjelmistokehittäjät pystyvät luomaan toimivat lisenssien hallinnan käytännöt, mikäli asiakas osaa asettaa heille sopivat vaatimukset.

Suuri osa lisenssien hallintaan liittyvästä käytännön työstä tapahtuu ohjelmistokehittäjien toimesta, joten ei ole mielekästä, että asiakas itsenäisesti suunnittele lisenssien hallinnan käytännöt. Tässä tapauksessa käytännöt saattavat olla teknisesti riittämättömät tai liian työläät ottaen huomioon projektin merkittävyyden. Kun ohjelmistokehitystiimi on mukana määrittelemässä sopivia käytäntöjä, tulee niistä toteuttamiskelpoisia ja tiimin on helpompi pitää niistä kiinni. Tällöin suurin osa lisenssien hallintaan käytettyjen työkalujen valinnoista kuuluu myös tiimille.

Sovittujen käytäntöjen tulee kattaa vähintään edellä mainitut suurimmat riskit eli nimeämisehtojen, oikeuskäsittelyn sijaintivaatimuksen ja SaaS-ehtojen tunnistamisen ja niistä aiheutuvien riskien hallinnan. Nimeämisehtojen julkaisun lisäksi on tärkeää, että palvelusta tai julkaistusta lähdekoodista löytyy tieto kanavasta, jonne voi ilmoittaa tiedon epäillyistä tekijänoikeusrikkomuksista. Tällöin tulee olla olemassa myös prosessi tai käytännöt epäilyjen käsittelylle, jotta epäilijälle voidaan vastata ripeästi tai ainakin antaa signaali, että asiaa käsitellään. Avoimen koodin arviointi ja ongelmista ilmoittaminen on avoimen lähdekoodin yhteisöjen luonnollinen tapaa hoitaa mahdolliset ongelmat.

Lisenssien hallinnan käytännöt ja lähdekoodista löytyvät lisenssit tulee dokumentoida, jotta voidaan näyttää, että näiden ehtoja on yritetty noudattaa. Avoin tieto lisenssien hallinnan käytännöistä ja lisenssidokumentaatio lisää myös jatkokäyttäjien varmuutta siitä, että tuotettu koodi ei sisällä tekijänoikeusluokkauksia.

Toinen kehitysehdotus on oman lisenssiyhteensopivuustaulukon tekeminen julkishallinnolle. Lisenssiyhteensopivuustaulukon tulisi ottaa huomioon julkishallinnon yleisimmät käyttötapaukset eli

lähinnä verkkopalvelukehitys sekä lisenssiyhteensopivuuden tulkinnat Suomen lain mukaan. Lisäksi dokumentti, jossa olisi suomenkieliset tulkinnat eri lisensseistä suomen lain puitteissa helpottaisi lisenssien hallintaa.

Kolmas ehdotus on kansallisen FOSSology-palvelun kehitys. Jos kaikki julkishallinnon käyttämien komponenttien tekijänoikeus- ja lisenssitiedot kerättäisiin yhteen paikaan, voitaisiin niiden auditointi tehdä vain kerran ja juurikin julkishallinnon tarpeiden ja suomen lain näkökulmasta. Jos palvelu olisi avoin, myös muut suomalaiset yritykset, yhteisöt ja ohjelmistokehittäjät voisivat käyttää sitä. Tätä ehdotusta voidaan kritisoida siitä, että monet listatut palveluntarjoajat tarjoavat vastaavaa palvelua ja ovat kehittäneet osaamista tähän, jolloin ne pystyvät todennäköisesti ylläpitämään palvelua tehokkaammin kuin julkishallinnon toimijat. Näiden palveluntarjoajien liiketoimintamalli ei kuitenkaan välttämättä taivu tämänlaiseen avoimeen tekijänoikeustiedon jakamiseen. Tällöin niiden palvelun hyödyt rajoittuisivat julkishallinnon sisälle, eivätkä välttämättä edistäisi lähdekoodin jatkokäyttöä.

Palveluntarjoajia on kuitenkin paljon ja jossain tapauksessa koko lisenssien hallinnan ulkoistus palveluntarjoajalle voisi olla kustannustehokas vaihtoehto, varsinkin jos omaa lähdekoodia ei aio julkaista jatkokäytettäväksi. On myös mahdollista neuvotella palveluntarjoajien kanssa siitä, miten koodin jatkokäyttäjät voisivat hyötyä näiden tarjoamasta tekijänoikeustietojen hallinasta.

Mikäli haluaa edistää julkaistun koodin jatkokäyttöä, on ensisijaista, että koodi on modulaarista, helposti löydettävää ja selkeää käyttää. Vasta kun perusasiat ovat kunnossa, voi joku innostua käyttämään koodia. Vasta tässä vaiheessa mahdolliset hyödyntäjät alkavat tarkastella lisenssi- ja tekijänoikeusasioita, kun jatkokäyttäjä alkaa harkita komponentin käytön tekijänoikeusriskejä. Vaikka sallivien ja vastavuoroisten lisenssien vaikutukset jatkokäytön motivaatioon saatavat olla hieman erilaisia, kannatta alkuperäinen lisenssivalinta tehdä projektin tarpeiden, eikä potentiaalisten jatkokäyttäjien perusteella. On myöskin huomioitava, että jatkokäytön näkökulmasta riittävän laadukasta koodia ei pääse syntymään, mikäli ohjelmistokehitysprojekti on toteutettu pienimmällä mahdollisella budjetilla. Lisenssinvalinnassa kannattaa myös muistaa, että niin kauan kuin tekijänoikeudet ohjelmistoon ovat kokonaan itsellä, voi oman koodin julkaista useammalla kuin yhdellä lisenssillä.

GOFORE

Avoimen tuotteen hallintamallissa ei oteta kantaa lähdekoodin julkaisuun avoimen lähdekoodin lisensseillä, vaan sitä voidaan käyttää myös suljetun tuotteen yhteiskehityksessä. Mikäli avoimen tuotteen kehityksessä käytetään avointa lähdekoodia, tuoteomistajan tai tuotepäällikön täytyy tällöin ottaa vastuu lisenssien hallinnan tasosta, aivan kuten tilaajan missä tahansa avointa lähdekoodia hyödyntävässä kehitysprojektissa. Avoimen tuotteen hallintamallia voi kuitenkin hyödyntää eri organisaatioiden yhteisen tuotekehityksen mallina kehitettäessä yhteiskäyttöistä avoimen lähdekoodin tuotetta.

GOFORE

Sanasto

Termi	Selitys
Avoin lisenssi	Lisenssi, joka ei rajoita käyttäjiä, vaan jokainen, joka hyväksyy lisenssin ehdot ja noudattaa niitä, saa hyödyntää lisenssin myöntämiä oikeuksia
Avoin lähdekoodi	Lähdekoodi, jonka lisenssi sallii sen muokkaamisen ja muokatun version levittämisen, eikä rajoita koodin käyttöä
Kontribuutiosopimus	Sopimus, jolla avoimen lähdekodin ohjelmistoon kontribuution tehnyt kehittäjä antaa oikeuksia tai sitoumuksia ohjelmiston ylläpitäjälle
Kontti	Riisuttu versio virtuaalikoneesta, jonka käyttöjärjestelmä sisältää vain tietyn sovelluksen ajamiseen tarvittavat palvelut ja sovellukset
Pilvipalvelu	Palvelu, joka perustuu virtuaalikoneisiin tai vastaaviin palveluihin, joiden avulla voidaan dynaamisesti skaalata palvelun tarvitsemia tehoja
SaaS-ehto	Vastavuoroisuusehto, joka on voimassa jos palvelu tarjotaan julkisesti käyttöön, vaikka koodia ei jaella
Salliva lisenssi	Avoimen lähdekoodin lisenssi, jossa ei ole vastavuoroisuusehtoa
Vastavuoroisuusehto	Lisenssiehto, joka vaatii, että komponentista riippuvat tai komponentin kanssa ohjelman muodostavien muiden komponenttien lähdekoodi on myös julkaistava avoimena
Virtuaalikone	Virtuaalinen käyttöjärjestelmä, joka käyttää isäntänä toimivan tietokoneen resursseja

Seuranta- ja vertailuraportti suppea

TUIMA

Vuosi 2015

Projekti		Toimittaja		LKP -tili		Toteuma 001-016.2015 EUR
6000A-OKM0023	TUIMA	10001260	CSC-Tieteen tietotekniikan keskus O	Asiantuntija- ja tutkimuspalvelut	43920000	100 000,00
6000A-OKM0023	TUIMA	10001260	CSC-Tieteen tietotekniikan keskus O	Muut asiantuntija- ja tutkimuspalvelut	43920900	155 000,00
6000A-OKM0023	TUIMA	10037069	BeAn Solutions Oy	Asiantuntija- ja tutkimuspalvelut	43920000	3 100,00
6000A-OKM0023	TUIMA	10043603	Osuuskunta Valomo	Asiantuntija- ja tutkimuspalvelut	43920000	1 708,14
6000A-OKM0023	TUIMA	10043603	Osuuskunta Valomo	Muut asiantuntija- ja tutkimuspalvelut	43920900	1 179,44
6000A-OKM0023	TUIMA	10043603	Osuuskunta Valomo	Muut asiantuntija- ja tutkimuspalvelut	43920900	2 440,20
6000A-OKM0023	TUIMA	10043603	Osuuskunta Valomo	Muut ulkopuoliset palvelut	43990000	2 694,39
6000A-OKM0023	TUIMA	#		Yhteistoiminnan kustannusten korvaukset valtion vi	39670000	-266 122,17
Kokonaistulos						0,00

Vuosi 2016

Projekti		Toimittaja		LKP -tili		Toteuma 001-012.2016 EUR
6000A-OKM0023	TUIMA	10037069	BeAn Solutions Oy	Asiantuntija- ja tutkimuspalvelut	43920000	5 500,00
6000A-OKM0023	TUIMA	#		Yhteisrahoitteisen toiminnan tuotot valtion virast	39780000	-5 500,00
Kokonaistulos						0,00

VM:ltä laskutettu yhteensä	-271 622,17