Haaga-Helia
ammattikorkeakoulu Oy

Avoimen lähdekoodin merkitys ja hyöty ohjelmistonkehittäjälle

Juhani Turpeinen

Opinnäytetyö
30.4.2020

Tekijä(t)
Juhani Turpeinen

Koulutusohjelma
Tietojenkäsittely Tradenomi

Raportin/Opinnäytetyön nimi	**Sivu- ja liitesivumäärä**
Avoimen lähdekoodin merkitys ja hyöty ohjelmistonkehittäjälle	25 + 5

Opinnäytetyöhön sisältyy tietoperusta avoimen lähdekoodin menetelmästä ja tutkimus liittyen, kuinka iso rooli sillä on ohjelmistoprojekteissa. Tämän opinnäytetyön aiheen tarkoitus on tukea itseäni ohjelmistonkehittäjänä ja käsitellä enemmän avoimien lähdekoodien menetelmän aiheesta.

Opinnäytetyön aiheeseen kuuluu, mikä on avoin lähdekoodi ja mitkä asiat määrittävät ohjelmistoprojektin avoimeksi. Eli aihetta lähdetään avaamaan seuraavilla asioilla: mitä avoimen lähdekoodin kehitys on ja miten se eroaa suljetun lähdekoodin projektin kehityksestä. Mitä lisenssiryhmiä on tarjolla avoimeen lähdekoodin kehitykseen ja minkälainen on avoimen lähdekoodin kehityksen elinkaari ja miten se eroaa muista kehityksen elinkaaresta?

Opinnäytetyön tutkimusosiossa pyritään keräämään vastauksia liittyen aiheeseen ja saamaan vastausta kuinka iso rooli ja hyöty avoimen lähdekoodinkehitys on alan ammattilaiselle. Eli kyselyn kohderyhmänä toimisivat ohjelmistonkehityksen alalla toimivat yritykset, joiden kautta niissä työskentelevät ammattilaiset vastaisivat kyselyyn.

Vastausten pohjalta kirjoitan analysoivan raportin liittyen, kuinka tärkeä avoimen lähdekoodin ekosysteemi on alan ammattilaiselle ja onko mahdollisesti olemassa jonkinlaisia tilanteita, jossa käytetty avoimen lähdekoodin paketti hidastaisi ohjelmistonprojektin edistymistä.

Asiasanat
Avoin lähdekoodi, Avoimen lähdekoodin lisenssit, Avoimen lähdekoodin rooli ohjelmistonkehittäjälle
http://finto.fi/fi/

Sisällys

1 Johdanto 1

2 Avoimen lähdekoodin esittely 2

2.1 Vapaehtoiset avoimen lähdekoodin järjestöt 3

2.2 Avoimen ja suljetun lähdekoodin vertailu 3

3 Avoimen lähdekoodin lisensointi 5

3.1 Käytetyimmät avoimen lähdekoodin lisenssit 7

3.1.1 MIT-lisenssi 8

3.1.2 Apache 2.0 -lisenssi 8

3.1.3 GPL 3.0 -lisenssi 8

4 Avoimen lähdekoodin hyödyllisyys 9

5 Avoimen lähdekoodin haasteita 11

6 Avoimen lähdekoodin elinkaari 12

6.1 Käytettyjä ohjelmistonkehityksen elinkaaria projekteissa 13

6.1.1 Agilen ohjelmistonkehityksen malli 14

6.1.2 Vesiputouksen elinkaari 15

6.2 Yhteenveto 16

7 Avoimen lähdekoodin työkalut lyhyesti 17

7.1 GitHub 17

8 Tutkimus liittyen aiheeseen 18

8.1 Kuinka iso rooli avoimilla lähdekoodin teknologioilla on ohjelmistoprojekteissanne? 18

8.2 Ratkaisun valitseminen projektin tarpeisiin 19

8.3 Mistä etsiä avoimen lähdekoodin ratkaisua tai työkalua projekteihin? 19

8.4 Kriteerit oikean avoimen lähdekoodin valitsemisessa 20

8.5 Avoimen lähdekoodin haittavaikutukset ammattilaisten näkökulmasta 21

8.6 Vastaajien kokemus avoimen lähdekoodin kehityksestä 22

8.7 Projektit ilman avoimen lähdekoodin teknologioita 24

9 Pohdinta 25

Lähteet 26

Liitteet 29

Liite 1. Tutkimuskysymykset 29

1 Johdanto

Opinnäytetyön aiheeksi valitsin avoimen lähdekoodin merkityksen ja sen hyödyn ohjelmistonkehittäjälle. Olen jakanut opinnäytetyön sisällön kahteen osaan, jotka pohjautuvat tietoperustaan ja tutkimukseen. Tietoperustassa käydään läpi asioita, jotka liittyvät kokonaisuudessa avoimen lähdekoodin kehityksen menetelmään. Tutkimusosiossa käsitellään kyselyn kysymyksiä ja kyselylomakkeen avulla saatuja vastauksia. Kyselyyn sisältyy 10 kysymystä, jossa kyselyn kohderyhmänä ovat it-yritykset, joiden toiminta pohjautuu ohjelmistonkehitykseen ja avoimien lähdekoodien käyttöön.

Opinnäytetyön aiheen valitsemisen syynä on se, että olen erikoistumassa ohjelmistonkehittäjäksi ja tulevissa työtehtävissä tulen todennäköisesti käyttämään erilaisia avoimen lähdekoodin pohjalta olevia työkaluja ja muita sovelluksia.

Opinnäytetyön sisältö alkaa tietoperustalla, jossa pyritään kertomaan laajemmin avoimen lähdekoodin menetelmästä. Millä tavoin jokin sovellus tai muunlainen kehitelty työkalu luokitellaan avoimeksi lähdekoodiksi, minkälaisia lisenssityyppejä on olemassa avoimelle lähdekoodille ja yleisesti minkälaisia hyötyjä ja haittoja liittyy avoimen lähdekoodin kehitykseen ja käyttöön?

Tutkimusosiossa käsitellään vastauksia, jotka on saatu alan ammattilaisilta kyselylomakkeen avulla ja pohditaan, kuinka merkittävä avoimen lähdekoodin menetelmän käyttö on ammattilaisessa ympäristössä.

Tulevaksi ohjelmistonkehityksen ammattilaiseksi mielestäni avoimen lähdekoodien kehitys on yksi tärkeä osa-alue, jossa olisi hyvä olla jonkinlainen tietämys liittyen avoimen lähdekoodin menetelmiin ja miten niitä käsitellään. Opinnäytetyön tavoitteena saada mahdollisimman konkreettista tietoa, millaisia asioita sisältyy avoimen lähdekoodin menetelmään, ja saada vastauksia aikaisempiin mainittuihin tutkimuskysymyksiin.

”Milloin on hyödyllistä käyttää avoimen lähdekoodin ratkaisuja?”
”Mistä löytää tarpeellinen ratkaisu projektille, joka on avoimen lähdekoodin muodossa?”
”Milloin mahdollisesti on haitallista käyttää avoimen lähdekoodina olevaa ratkaisua?”
”Onko avoimen lähdekoodin käyttäminen tärkeätä ohjelmistonkehittäjälle?”

2 Avoimen lähdekoodin esittely

Avoimen lähdekoodin menetelmän tarkoituksena on julkaista kehitelty idea kaikille saataviksi ja antaa vapautta toisille saada ladattua jaetun lähdekoodin ja muokata lähdekoodia omien tarpeiden mukaisesti, minkä vuoksi avoimen lähdekoodin menetelmän suosio lähti nousemaan. Mikä tahansa ohjelmistonkehittäjän tarpeen tai idean pohjalta kehitelty sovellus tai ratkaisu voi olla avoimena lähdekoodina, joka on asetettu kaikille vapaasti saatavaksi, joko itse kehitellylle verkkosivulle tai yleiselle versionhallinnan alustalle kuten GitHub, GitLabs tai BitBucket, jossa ohjelmistonkehittäjillä on mahdollisuus jakaa oman rakennetun lähdekoodin toisille. Myös muut ohjelmistonkehittäjät pääsevät käsiksi jaetulle ratkaisun sisällölle ja voivat halutessa ladata ja muokata lähdekoodia omien tarpeiden mukaisesti. (Fitzgerald & Kesan 2011, 8–27.)

Avoimen menetelmän kehittymisen ansiosta projektissa kehiteltävän sovelluksen sisältö voi olla olemassa olevasta jaetusta lähdekoodin paketista tai tyhjästä ideasta luotu sovellus, jonka ohjelmistonkehittäjä on voinut julkaista sen kaikille saatavaksi.

Avoimen lähdekoodin menetelmiin ja ilmaisen sovelluksen määritelmiin voi sisältyä pientä epäselvyyttä liittyen siihen, että molemmat ovat vain ilmaiseksi saatavana, mutta avoimella lähdekoodilla on enemmän merkitystä kuin ilmaiseksi saatavalla sovelluksen määritelmällä. Avoimen lähdekoodin menetelmän ideana on, että jaetuilla lähdekoodeilla on lupa uudelleen käyttää omien tarpeiden mukaisesti ja mahdollisesti uudelleen jakaa muokattua versiota eteenpäin, tosin riippuen asetetusta lisenssistä. Avoimen lähdekoodin menetelmä eroaa ilmaisesta sovelluksen jakelusta siten, että jaetusta ilmaisesta sovellusta voi asentaa vapaasti, mutta sen lähdekoodia ei ole saatavilla erikseen. (Fitzgerald & Kesan 2011, 8–27.)

Suurin osa ohjelmistonkehittäjistä on päätyneet avoimen lähdekoodin kehityksen menetelmään, minkä syynä on ollut aikaisempia lähdekoodiprojekteja, jotka ovat onnistuneet avoimena lähdekoodina, kuten suosittu käyttöjärjestelmä Linux. Linux-käyttöjärjestelmän ensimmäisen version kehitti suomalainen Linus Torvalds, joka oli päättänyt julkaista kehittämänsä käyttöjärjestelmän lähdekoodin muille nähtäväksi ja käytettäväksi. (Maruping & Daniel & Cataldo 2019.)

2.1 Vapaehtoiset avoimen lähdekoodin järjestöt

Julkaistun avoimen lähdekoodin sovelluksen ylläpitävät yleensä vapaehtoiset ohjelmistonkehittäjät, jotka ovat päättäneet auttaa jotain tiettyä heitä kiinnostavaa lähdekoodin sovellusta, jonka tuloksena sen suosio jatkaa nousemista ohjelmistonkehittäjien keskuudessa ja seurauksena siitä kehittyy tiivis yhteisö, jossa jaetaan tuntemusta ja kehitetään yhdessä sovellusta paremmaksi. (Engard 2010, Part 1 Chapter 1).

Järjestö nimeltään Open Source Initiative, jonka visiona on tukea ja ylläpitää avoimen lähdekoodien jakamista sekä huolehtia, että avoimille lähdekoodeille on saatavilla tarpeen mukainen lisenssi. OSI-järjestössä on kehitetty lista, jonka tarkoituksena on tukea avoimen lähdekoodin määritystä ja he mm. ylläpitävät osan tarjolla olevista avoimen lähdekoodin lisenssejä, jotka kattavat avointa lähdekoodin määritystä. (Open Source Initiative 2007.)

Myös Free Software Foundation-järjestö, jonka missiona on ylläpitää vapaata ohjelmistoa saatavilla verkossa ja ylläpitää GNU (General Public License) -nimistä lisenssiä, joka kattaa vapauden käyttää sen lisensoitua ohjelmaa vapaasti. Free Software Foundation järjestö ja GNU-käyttöjärjestelmän ylläpitäjät ovat kehittäneet yhdessä määritelmän, joka kattaa vapauden jaetuissa avoimissa lähdekoodeissa. Vapaus määritelmä koostuu neljästä eri vapauden liittyvästä elementistä, jotka ovat seuraavat:

1. Käyttää jaettua sovellusta vapaasti.
2. Tutkia ja opiskella vapaasti jaettua avointa lähdekoodia.
3. Vapaus jakaa kehiteltyä lähdekoodia auttaakseen muita.
4. Muokata jaetun avoimen lähdekoodin sisältöä ja jakaa takaisin sen yhteisölle.

(GNU Operating System 2019; Free Software Foundation 2019).

2.2 Avoimen ja suljetun lähdekoodin vertailu

Paulsonin, Succin ja Eberleinin (2004, 2.) kirjoituksen mukaan avoimen lähdekoodin kehityksessä käytettyjen komponenttien toiminnallisuudet ovat rakennettu yksinkertaisemmalla ajattelutavalla kuin suljetun lähdekoodin kehityksessä. Sovelluksen lähdekoodin ylläpito on todennäköisemmin enemmän reagoiva avoimessa kehityksessä. Avoimen sovelluksen lähdekoodit ovat julkisesti näkyvillä netissä, josta kuka vaan siitä kiinnostunut voi vaikuttaa ja mahdollisesti ehdottaa oman näkemyksen tai jakaa oman versionsa muille näkyväksi. Mitä enemmän yhteistyötä ja kommunikaatiota avoimen lähdekoodin kehityksessä ilmenee, sitä enemmän avoimen lähdekoodin laatu kasvaa ja sovelluksen sisältö monipuolistuu.

Kun taas suljetussa lähdekoodin kehityksessä sama ohjelmistonkehityksen tiimi joutuu ratkomaan sovellukseen ilmenevät ongelmat ja päivittää version tarpeen mukaan. Seurauksena sovelluksen kehitys voi olla hitaampaa ja haasteellisempaa kehitystiimille, tietenkin ohjelmistonkehityksen tahti riippuu kehitystiimistä ja sen yhteistyötaidoista. Kun taas avoimessa lähdekoodin kehityksessä ei ole rajaa, kuinka monta henkilöä on mukana ylläpitämässä lähdekoodia.

Taulukon 1 pohjalta nähdään vertailu avoimen- ja suljetun lähdekoodin eroavaisuuksia eli nähdään, missä kohdissa nämä kaksi eroavat toisistaan.

Taulukko 1 Avoimen- ja Suljetun lähdekoodien menetelmien vertailu (Potdar & Chang 2004, 106-110.) pohjalta

	Avoin Lähdekoodi	Suljettu Lähdekoodi
Lähdekoodin kehityksen ajoitus	Ohjelmistonkehittäjä voi periaatteessa ylläpitää sovelluksen kehittämistä niin kauan, kun hänellä tarvetta sille tai sen yhteisö auttaa kehitystä tarpeen mukaisesti, jotta pitää koodin ajan tasalla. Avoimen lähdekoodin kehityksen aika rajaton.	Suljetun lähdekoodin kehityksellä on yleensä määriteltynä jonkinlainen tavoite, milloin sovellus on valmis luovutettavaksi asiakkaalle. Eli aikarajat ovat asetettu.
Lähdekoodin vaatimusten lähtöpohja	Yleensä ohjelmistonkehittäjällä on jokin tarve, josta lähtee kehittämän ratkaisu sille, joka on syynää yleensä avoimen lähdekoodin projektin aloittamiseen.	Suljetulla lähdekoodilla on yleensä kaupallinen tarve, jossa on rajatut vaatimukset sovellukselle, jonka pohjalta rakennettaan sovellus.
Lähdekoodien dokumentaatio	Avoimen lähdekoodin dokumentointi riippuu sen kehittäjästä, onko hän tehnyt dokumentaatiot sovellukselle.	Suljetulla lähdekoodilla on yleensä tarkasti dokumentoitu raportti, joka auttaa asiakasta sovelluksen käyttöönotossa ja sovelluksen ylläpidossa.
Rakennetun lähdekoodin näkyvyys	Lähdekoodi on kaikille näkyvillä, riippuen sen jakajasta.	Lähdekoodi on yleensä näkyville sovelluksen omistajalle ja kehitystiimille riippuen sopimuksesta.
Ylläpito	Avoimen lähdekoodin ylläpito ja päivitys on riippuvainen sen kehittäjästä ja sovelluksen suosioista.	Suljetun lähdekoodin ylläpito on tarkasti suunniteltu ja versioiden julkaisu tiettyinä ajankohtina.

Kuten näemme, avoimen lähdekoodin kehityksen tapaukset menevät enemmän sen alkuperäisen julkaisijan ja yhteisön puolelle, kun taas suljetussa lähdekoodissa kehityksen kohdat menevät enemmän markkinoivan yrityksen puolelle, jossa on yleensä asiakas.

3 Avoimen lähdekoodin lisensointi

Avoimen lähdekoodin lisensointi on mielestäni erittäin tärkeää liittyen avoimen lähdekoodin menetelmän suosion nousemisen kannalta, koska jotkut ohjelmistonkehittäjät eivät välttämättä aseta avoimen lähdekoodin lisenssiä omaan sovellukseen, joka sattuu olemaan erittäin hyvä sisällöltään, mutta sitä ei voi ottaa käyttöön, koska ei ole määritelty lisenssiä kyseisessä lähdekoodissa. Singh ja Phelpsin (2013.) mukaan, jotta itse rakennettu sovellus pidettäisiin avoimen lähdekoodin määritelmän mukaisena, niin olisi hyvä määrittää lisenssi omalle projektille.

Avoimen lähdekoodin menetelmän pohjalta on olemassa oma lisenssiryhmä, joka määrittää enemmän vapautta jaetun lähdekoodin käyttämiselle ja jakelulle. Laajasti kehitetyn avoimen lähdekoodin lisenssivalikoiman ansiosta ohjelmistonkehittäjällä on valinnan varaa valita sopiva lisenssi omaan projektiinsa. Kuitenkin erilaisilla tarjolla olevilla lisensseillä on eroavaisuuksia, jotka ovat muun muassa, mitä asioita jaetulla avoimen lähdekoodin paketilla voi tehdä ja mitä ei voi tehdä. Lisenssin valitseminen riippuu täysin ohjelmistonkehittäjästä eli siitä mihin tarkoitukseen hän haluaa, että hänen julkaisemaansa lähdekoodia käytetään.

Kapitsaki, Tselikas ja Foukarakis (2014.) mukaan on kehitetty määritelmä kolmesta erilaisesta tasosta, jotka pohjautuvat vapaiden levitysoikeuden lisensseihin. Levitysoikeuden tasot jakautuvat tiukkaan, heikkoon ja avoimeen levitysoikeustasoon. Näiden kolmen tason pohjalta on kehitelty erilaisia käytännöllisiä lisenssejä, joita voidaan käyttää erilaisissa avoimissa lähdekoodeissa. Seuraavista mainituista kohdista nähdään, mihin avoimen lähdekoodin lisenssi tulee pohjautua.

> **Tiukka levitysoikeus (”Strong Copyleft”):** Jos ohjelmistonprojektissa on käytetty tämän tason pohjalta olevaa lisenssiä, niin se tarkoittaa sitä, että seuraava ohjelmistonkehittäjä, joka haluaa käyttää jaettua lähdekoodi pakettia omassa projektissa niin, joutuu julkaisemaan oman versionsa samalla lisenssillä, joka oli määritelty alkuperäisessä jaetussa versiossa. (Kapitsaki & Tselikas & Foukarakis 2014.)
>
> **Heikko levitysoikeus (”Weak Copyleft”):** Tämän tason lisenssit antavat vapauden julkaista toisella lisenssillä. Mutta riippuen tilanteesta, jos henkilö on muokannut alkuperäistä koodia toisentyyppiseksi, niin hän joutuu käyttämään samaa lisenssiä, joka on käytetty alkuperäisessä julkaistussa lähdekoodissa. (Kapitsaki & Tselikas & Foukarakis 2014.)

Avoin levitysoikeus (”Permissive”): Tämän tason lisenssit antavat kokonaisuudessa eniten vapautta jaetulle koodille ja niille, jotka haluavat käyttää omassa projektissaan ja myös tämän tason lisenssit antavat vapauden julkaista toisella avoimen lähdekoodin pohjaisella lisenssillä. (Kapitsaki & Tselikas & Foukarakis 2014.)

Permissive	Weak copyleft	Strong copyleft
Apache License v. 2.0	Adaptive Public License v. 1.0	Frameworx License (Frameworx-1.0)
Boost Software License (BSL-1.0)	Apple Public Source License (APSL-2.0)	GNU Affero General Public License v. 3 (AGPL-3.0)
BSD 3-Clause "New" or "Revised"	GNU Library or "Lesser" General Public License v. 2.1 (LGPL-2.1)	GNU General Public License v. 2.0 (GPL-2.0)
BSD 3-Clause "Simplified" or "FreeBSD"	GNU Library or "Lesser" General Public License v. 3.0 (LGPL-3.0)	GNU General Public License v. 3.0 (GPL-3.0)
Academic Free License ("AFL") v. 3.0	Microsoft Reciprocal License (Ms-RL)	Non-Profit Open Software License v. 3.0 (NPOSL-3.0)
Attribution Assurance Licenses (AAL)	Motosoto License (Motosoto)	Open Software License v. 3.0 (OSL-3.0)
EU DataGrid Software License (EUDatagrid)	Mozilla Public License v. 2.0 (MPL-2.0)	Reciprocal Public License v. 1.5 (RPL-1.5)
Educational Community License, v. 2.0 (ECL-2.0)	Nokia Open Source License (Nokia)	Sleepycat License (Sleepycat)
Eiffel Forum License v. 2.0 (EFL-2.0)	Ricoh Source Code Public License (RSCPL)	
Entessa Public License v. 1.0	Sun Public License v. 1.0 (SPL-1.0)	
Fair License	wxWindows Library License (WXwindows)	
ISC License (ISC)		
MirOS Licence		
MIT License (MIT)		
PHP License v. 3.0 (PHP-3.0)		
PostgreSQL License (PostgreSQL)		
Python License (Python-2.0) CNRI		
Python License (CNRI-Python) University of Illinois/NCSA Open Source License (NCSA)		
Vovida Software License v. 1.0 (VSL-1.0)		
W3C License (W3C)X.Net License (Xnet)		
Zope Public License v. 2.0 (ZPL-2.0)		
zlib/libpng License (Zlib)		

Kuva 1. Avoimen lähdekoodin lisenssejä jaettuna kolmeen tasoon (Kapitsaki & Tselikas & Foukarakis 2014, 74.)

Kuten näemme kuvassa 1 on jaoteltu erilaisia avoimen lähdekoodin lisenssejä kolmeen mainittuun tasoihin. Samoissa ryhmissä olevien lisenssien päätarkoituksena on määritellä samalla periaatteella avoimen lähdekoodin käyttöä, mutta niihin on määritelty erilaisia tapauksia, joita lähdekoodilla voi tehdä ja mitä ei voi tehdä.

On olemassa tapauksia, joissa ohjelmistonkehittäjä julkaisee itse rakentamansa lähdekoodin julkiselle alustalle, josta muut ihmiset voivat päästä siihen käsiksi, mutta hän ei ole asettanut omalle lähdekoodille minkäänlaista lisenssiä. Open Source Initiative (2007.) -järjestön mukaan, jos avoimella lähdekoodilla ei ole määriteltyä siihen tarkoitettua lisenssiä, niin jaettua lähdekoodia ei voida pitää vapaana.

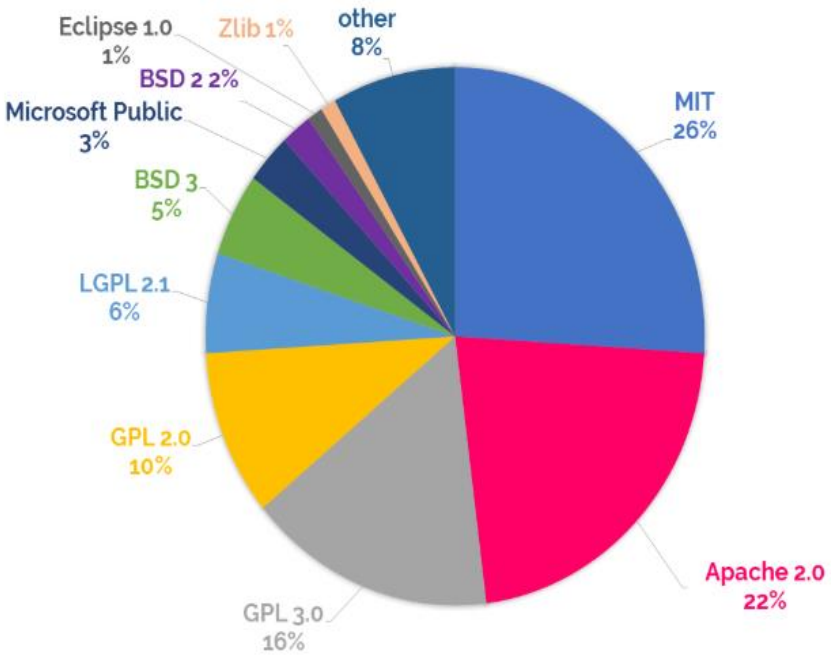
Lähteen mukaan jaettua koodia pidetään tekijänoikeudellisena eli toisilla ei ole oikeuksia käyttää jaettua projektia markkinointikäyttöön tai uudelleen jakaa omalla lisenssillä eteenpäin. Mutta järjestö suosittelee, että kannattaa ottaa yhteyttä alkuperäisen lähdekoodin jakajaan ja pyytää mahdollisesti erillinen lupa tai pyytää tekijää lisäämään avoimen lähdekoodin lisenssin näkyvälle olevalle projektille. Lisenssin laittaminen omaan

rakennettuun koodiin ei ole pakollista, mutta se saattaa vaikuttaa koodin käyttöön, koska jaettua lähdekoodia pidetään tekijänoikeudellisena ja se rajoittaa jaetun lähdekoodin uudelleenkäyttöä. (Choosealicense 2019; Open Source Guides. 2019.)

Toisena vaihtoehtona lähdekoodin jakajalla on mahdollisuus julkaista oman projektin tekijäoikeusvapaus lisenssinä, jos hän ei halua määritellä omaan projektiinsa minkäänlaista lisenssiä. Toisin sanoen tekijä antaa yleisesti kaikki vapaudet yhteisölle tai koodista kiinnostuneille lähdekoodin uudelleenkäytettäväksi ilman rajoituksia.

3.1 Käytetyimmät avoimen lähdekoodin lisenssit

Blogissa (WhiteSource 2018.) mainitussa tutkimuksessa, jossa oli kerätty dataa liittyen siihen, millaisia lisenssejä on käytetty eniten ohjelmistonprojektien yhteydessä. Tutkimukseen sisältyi monia erilaisia julkaistuja projekteja, joissa niihin oli määritelty erilaisia avoimia lähdekoodin pohjalta olevia lisenssejä. Tutkimuksen tekijät olivat päätyneet kuvassa 2 esitettyyn tulokseen, että vuonna 2018 eniten käytettyä avoimen lähdekoodin lisenssejä olivat MIT ja Apache 2.0, jotka molemmat kuuluvat avoimeen levitysoikeuden lisenssiryhmään ja kolmanneksi suosituin lisenssi oli GPL 3.0, joka taas kuului tiukempiin avoimen levitysoikeuden lisenssiryhmään.



Kuva 2. Tulokset piirakkakuviossa, joka on lainattu vuoden 2018 pohjalta olevasta tutkimuksen tuloksesta (WhiteSource 2018.)

Miksi sitten MIT-lisenssi on suosituin, ja miksi sen suosio jatkaa kasvamistaan ohjelmistonkehittäjien kesken?
(Goldstein 2020.) julkaistussa blogissa on käsitelty asiaa siten, että se on niin yksinkertainen ja vaivaton, että se tukee avoimen lähdekoodin menetelmää sisältyvää vapaudellisuutta eniten. Se myös takaa, että jaetusta lähdekoodista kiinnostuneet, jotka haluavat kehittää koodia eteenpäin, voivat tehdä sen huoletta. Toki vaatimuksena on, että jatkokehittäjät pitävät alkuperäisen lähdekoodin lisenssin mukana tai mainitsevat alkuperäisen julkaisijan.

3.1.1 MIT-lisenssi

MIT-lisenssi kuuluu avoimpien lähdekoodien lisenssiryhmään. MIT-lisenssin avoimuuden ja sisällön yksinkertaisuuden ansiosta MIT-lisenssin määrittänyt antaa vapaudet tehdä jaetulle lähdekoodille mitä vain. Muun muassa muokata lähdekoodia omaksi versioksi ja jopa julkaista toisen lisenssin alla. Vaadittuna asiana MIT-lisenssissä on, että jos haluaa uudelleen julkaista muokatun koodin toisella lisenssillä, niin täytyy mainita sen sovelluksen alkuperäisen jakajan nimen. (Lawerence 2004.)

3.1.2 Apache 2.0 -lisenssi

Apache 2.0 -lisenssi kuuluu samaan avoimpien lisenssi ryhmään, kuin edellä mainittu MIT-lisenssi. Apache 2.0 -lisenssi antaa myös vapauksia muokata ja julkaista toisen lisenssin alla. Jos sovelluksessa olevat lähdekooditiedostot pidetään alkuperäisenä, niin henkilö, joka haluaa julkaista sen omana versiona, joutuu käyttämään samaa Apache 2.0 -lisenssiä niissä tiedostoissa, joihin hän ei ole tehnyt muutosta. (Kaufman 2018.)

3.1.3 GPL 3.0 -lisenssi

GPL 3.0 -lisenssi kuluu tiukempiin avoimien lähdekoodien lisensseihin eli toisin sanoen, jos henkilö haluaa muokata tai lisätä muita toiminnallisuksia ja uudelleen julkaista omana versiona, niin hän joutuu asettaman saman lisenssin muokattuun versioon. (GNU Operating System 2019 B.) GPL lisenssin julkaisijan mukaan, että GPL 3.0 lisenssi on suositeltu isoimmille avoimille ohjelmistoprojekteille, joiden suosio tulee kasvamaan ajan myötä.

4 Avoimen lähdekoodin hyödyllisyys

Avoimen lähdekoodin suosion kasvun ansiosta on kehittynyt erilaisia hyötyjä eri alan toimiville osapuolille, jossa Johnsonin ja Tannerin (2013) mukaan siitä on kehittynyt myös vahva vaihtoehto kaupalliseen käyttöön. Syynä on ollut avoimeen lähdekoodiin lisenssien pohjalta sisältyvä vapaus niiden käytöstä.

Yritysmaailmassa todennäköisesti pieni- ja keskikokoiset yritykset voivat hyötyä eniten avoimen lähdekoodin käytöstä, koska se olisi heille suotuisa vaihtoehto ja kustannukset olisivat alhaisia. Suosituimpien avoimien lähdekoodien ansiosta on mahdollista, että tarjolla olevien lähdekoodien ja sovelluksien laatu voi olla laadultaan yhtä hyvä tai jopa parempi kuin tarjolla olevissa maksullisissa vaihtoehdoissa. Jaetun avoimen lähdekoodin paketin kehittäjät ylläpitävät sen toiminnallisuutta ja korjaavat mahdollisesti tulevista sovelluksen ongelmista, jotta avoimen lähdekoodin käyttäminen olisi suotuisaa ja vaivatonta. (Zeimer & Stenz 2012.)

Uusien ohjelmoinnista kiinnostuneiden tulokkaiden näkökulmasta avoimien lähdekoodien käyttö ja kehitys sekä niiden pohjalta rakennetuilla versioilla on suuri hyöty henkilökohtaisessa oppimisessa ja kehityksessä. Avoimen lähdekoodin vapauden ansiosta on mahdollisuutena ottaa kiinnostava projekti omaan käyttöön ja analysoida sen sisältöä ja ohjelmointityyliä, josta voi oppia enemmän avoimen lähdekoodin menetelmän käyttöä. Myös suurimmasta osasta julkaistujen avoimien lähdekoodien projekteista on tehty erilaisia dokumentaatioita ja ohjeistuksia erilaisiin käyttötarkoituksiin, jotka mahdollisesti auttavat ohjelmoinnin alan opiskelijoita ja myös ammattilaisia ymmärtämään, mitä toiminnallisuuksia jaettu lähdekoodi tarjoaa. Myös uudet ohjelmoinnista kiinnostuneet henkilöt pääsevät helposti mukaan avoimen lähdekoodin kehitysympäristöön sekä näkemään avoimen lähdekoodien kehityksen innovaatiota, jonka suosio on kasvanut ajan myötä. (Gunjan & Pawan 2011.)

Myös alan ammattilaisille on hyötyä avoimen lähdekoodien menetelmän käytöstä, jossa avoimen lähdekoodien pakettien kehittäminen tai käyttö auttaa parantamaan syventävästi heidän taitojaan ja ongelmanratkaisun kykyä. Eli avoimen lähdekoodin menetelmien käytön ansiosta ammattilainen voi nähdä ongelman eri näkökulman pohjalta, josta hän pyrkii ratkoman ongelman eri tavalla tai kehittää nykyistä sovelluksen versiota paremmaksi. (Roberts & Hann & Slaughter 2006.)

Myös aloittelijan osallistuminen avoimien lähdekoodien yhteisöihin, joissa myös on mahdollisuutena saada palautetta kehitysideoista kokeneemmilta henkilöiltä, auttaa tukemaan henkilön ohjelmointitaitojen kehittymistä ja parantaa hänen itsevarmuuttansa. Tällä tavoin yhteisössä työskenteleminen tukee aloittelijan taitojen kehitystä sekä hiomista paremmaksi ajan myötä. (Tackaberry. 2018; Lee. 2015.)

Tietyt alan yritykset, etenkin pienet yritykset, jotka käyttävät avoimen lähdekoodin pohjalta olevia sovelluksia tai muita teknologioita, voivat hyötyä monessa eri asiassa, kuten esimerkiksi resurssien käytössä, koska suurimmaksi osaksi työkalut ovat vapaasti saatavilla verkossa. Sovelluksien avoimuus myös helpottaa niiden muokattavuutta tietyn yrityksen tarpeisiin, kun verrataan esimerkiksi tuotannossa oleviin maksullisiin sovelluksiin tai muihin työkaluihin ja teknologioihin. Kuten aikaisemmassa (2.2 Avoimen ja suljetun lähdekoodin vertailu) kappaleessa vertailtiin avoimen ja suljetun lähdekoodin menetelmiä, niin avoimen lähdekoodin menetelmän ansiosta yrityksillä on valinnanvaraa sovelluksen muokattavuudessa ja saatavuudessa, koska sen saa heti käyttöön.
(Congdon 2015.)

Miten yritykset hyötyvät, kun ne päättävät julkaista oman lähdekoodin kaikille saataviksi? (Kraus 2016.) artikkelissa kerrotaan, että julkaisemalla projektin lähdekoodin voi saada enemmän tukea kasvavalta yhteisöltä. Jos julkaistun lähdekoodiin tulee ongelmia vastaan, niin apua on saatavissa yhteisöltä nopeasti ja ongelmatkin huomataan nopeammin. Myös on mahdollisuutena nostaa yrityksen näkyvyyttä it-alalla, kun julkinen yleisö käyttää enemmän jaettua ohjelmistoa ja mahdollisesti auttaa sovelluksen ylläpidon kehittävyyttä tulevaisuudessa. Se myös helpottaisi ohjelmistonkehittäjien rekrytoinnissa, koska ohjelmistonkehittäjät ovat päässeet vapaasti tutustumaan koodiin ja tutkimaan sen sisältöä, mikä auttaisi pääsemään helposti mukaan yrityksen toimintaan tarpeen vaatiessa. (Kraus 2016.)

5 Avoimen lähdekoodin haasteita

Avoimeen lähdekoodin käyttämiseen ja kehittämiseen sisältyy myös haasteitakin. Ajatellaan, että ohjelmistonkehittäjä rakentaa sovelluksen ja julkaisee projektin näkyville, mutta julkaistu projekti ei saa tarpeeksi huomiota ja palautetta. Tämän seurauksena aloitettu projekti jää keskeneräiseksi. Myös toisena sivuvaikutuksena on, että julkaistun avoimen lähdekoodin ylläpitäjä lopettaa rakennetun koodin ylläpitämisen, jonka seurauksena se ei pysy ajan tasalla, sekä vanhentunut koodi altistuu tietoturvaongelmille. (Durkovic & Vukovic & Rakovic 2008.)

Joskus tapahtuu näin, että ohjelmistonkehittäjä on julkaissut oman projektinsa ilman käytännöllistä dokumentaatiota tai koodiin sisältyvä tieto on vaikeasti saatavilla. Tämä voi rajoittaa projektin jatkokehitystä sekä hankaloittaa sen käyttämistä. Näin toiset eivät edes välttämättä vaivautu käyttämään jaettua koodia tai sovellusta, mikä vaikuttaa sen suosioon ja sen jakaneen sovelluksen käytettävyyteen.

Yrityksille avoimen lähdekoodin käytöstä voi tulla ylimääräisiä kustannuksia, jotka voivat liittyä sen ylläpitoon eli esimerkiksi yritys tarpeen vaatiessa joutuu palkkaamaan ja kouluttamaan ammattilaisen, jotta se voisi ylläpitää käytössä olevaa sovellusta, jotta myös käytössä olevan sovelluksen sisältö ja siihen liittyvät riippuvaiset paketit pysyisivät ajan tasalla ja mahdollisesti tulevia ongelmia pystyttäisiin ratkomaan ajoissa.
(Investintech 2018.)

(Golden 2004) kirjassa mainitaan, että riippuen käytetyn lähdekoodin lisensoinnista ja siitä, kuinka useita eri avoimen lähdekoodin paketteja käytetään yrityksen omassa projektissa, joten avoimen lähdekoodin paketin käytöstä voi olla epäsymmetrinen riski yritykselle. Eli kun yritys rakentaa oman sovelluksen avoimilla lähdekoodin paketeilla, niin heidän oma sovelluksensa voi olla tavallaan myös avoin. Seurauksena projekti, joka käytetään markkinoinnillisessa tarkoituksessa voi tulla niin sanotusti kaapatuksi, eli jos joku huomaa, että projekti on tehty avoimien lähdekoodin pakettien pohjalta, niin hän voi pyytää lähdekoodin jaettavaksi, jonka seurauksena henkilö voi kaapata projektin itselleen tai uudelleen jakaa ja se vaikuttaisi yrityksen toimintaan. Myös aina voi olla riskinä, että yritysten projekteissa käytetään useita erilaisia avoimia lähdekoodeja ja voi olla tilanteita, että joitakin käytettyjä avoimia lähdekoodeja ei ylläpidetä tarpeeksi. Silloin yrityksen projektin tietoturva voi olla riskialttiina ja se vaikuttaisi myös yrityksen toimintaan ja luotettavuuteen.

Voi olla tilanteita, että joku avoimen lähdekoodin ylläpitäjistä asettaa haitallisen koodin jaettuun projektiin ja sen asetetun lähdekoodin haitallinen vaikutus voi olla vaikeasti huomattavissa. Tällä tavoin henkilö voi saada arkaluonteista tietoa näkyville tai aiheuttaa muuta haittoja jaetun lähdekoodin kautta käyttäjille. (Golden 2004.)

6 Avoimen lähdekoodin elinkaari

Avoimen lähdekoodin kehitykselle ei ole määritelty virallista kehityksen elinkaaripohjaa. Yleensä lähtökohtaisesti avoimen lähdekoodin kehityksessä on mukana yksi tai useampi ohjelmistokehittäjä, jotka päättävät kehittää ratkaisun omille tarpeilleen tai yleiseen ongelmaan ja julkaista sen avoimena lähdekoodina, josta ei ole periaatteessa olemassa lopullista versiota, vaan jokainen halukas voi ottaa lähdekoodiin ja kehittää oman mieluisen version.

Lähdekoodin kehityksen aloitus lähtee yhdestä tai mahdollisesti useamman tiimin ohjelmistonkehittäjästä ja lähdekoodin elinkaaren kehitys voi päättyä melkein heti alussa, jos ohjelmistonkehittäjä päättää, että ei ole tarvetta lähteä kehittämään uutta projektia, jos sille ei ole minkälaista tarkoitusta. (Sourav & Shyamalendu & Palash 2011.)

Avoimen lähdekoodin lisenssin johdosta saadun vapauden ansiosta kehityksen elinkaari ei välttämättä päädy valmiiksi tuotteeksi ollenkaan. Kun ajatellaan, että avoimen lähdekoodin projektin aloitus johtuu ohjelmistonkehittäjän tarpeesta ja jossain vaiheessa hän päättää jakaa sen avoimen lisenssin pohjalta julkiselle alustalle ja lopettaa sen ylläpidon. Sen julkaisun ansiosta joku muu kiinnostunut ohjelmistonkehittäjä voi mahdollisesti jatkaa sen projektin kehittämistä, jonka ansiosta ohjelmistonprojektin ”elämä” jatkuu. (Guliani & Woods 2005.)

Lähteen (Sourav & Shyamalendu & Palash 2011.) kirjoitetussa teoksen pohjalta on edustettu avoimen lähdekoodin elinkaari visualisoituna ja vaiheisiin purettuna:

Tarve/idea: Lähdekoodin kehittämisen projekti lähtee mahdollisesti tietystä ohjelmistonkehittäjän tai yleisesti esille tulleesta tarpeesta tai ideasta, jota halutaan toteuttaa.

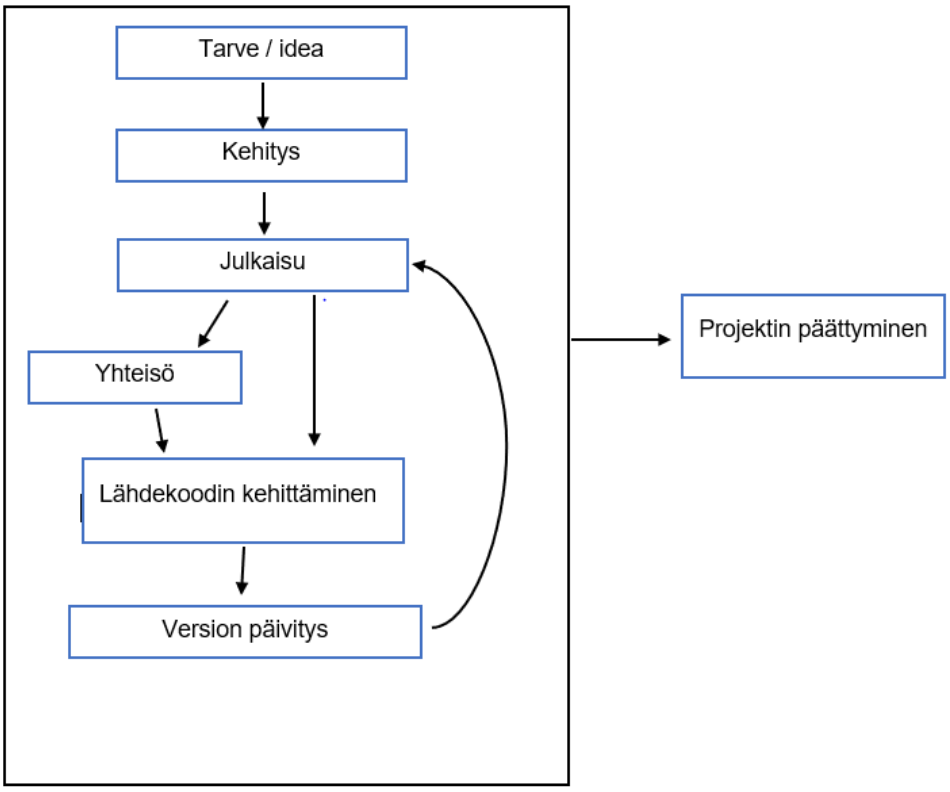
Kehitys: Tässä vaiheessa ohjelmistonkehittäjä rakentaa omaan tahtia ratkaisun kehitetylle idealle.

Julkaisu: Tässä vaiheessa, kun ohjelmistonkehittäjä on rakentanut tarpeeksi stabiili version omasta ohjelmastaan, niin se on mahdollista julkaista avoimen lähdekoodin lisenssin alla julkiseksi.

Yhteisö: Avoimen lähdekoodin julkaisun jälkeen projektin suosio voi alkaa nousta ja projektista kiinnostuneet eri osapuolet päättävät tulla mukaan auttamaan projektin kehitystä.

Ylläpito: Tässä vaiheessa, kun yhteisö ja lähdekoodin ylläpitäjät kehittävät ja parantavat lähdekoodin versiota ajan myötä, projektin kehitys jatkuu riippuen yhteisön aktiivisuudesta.

Projektin päättyminen: Kuten aikaisemmin mainittu avoimen lähdekoodin kehitys voi pysähtyä tai päättyä, kun sen jakaja tai yhteisö lopettavat sen ylläpidon ja kehityksen hetkellisesti tai lopullisesti kunnes joku päättää jatkaa sen projektin kehittymistä tai sitten alkaa kehittää omansa versiota alkuperäisestä.



Kuva 3. Mahdollinen avoimen lähdekoodin elinkaari (Sourav & Shyamalendu & Palash 2011.)

6.1 Käytettyjä ohjelmistonkehityksen elinkaaria projekteissa

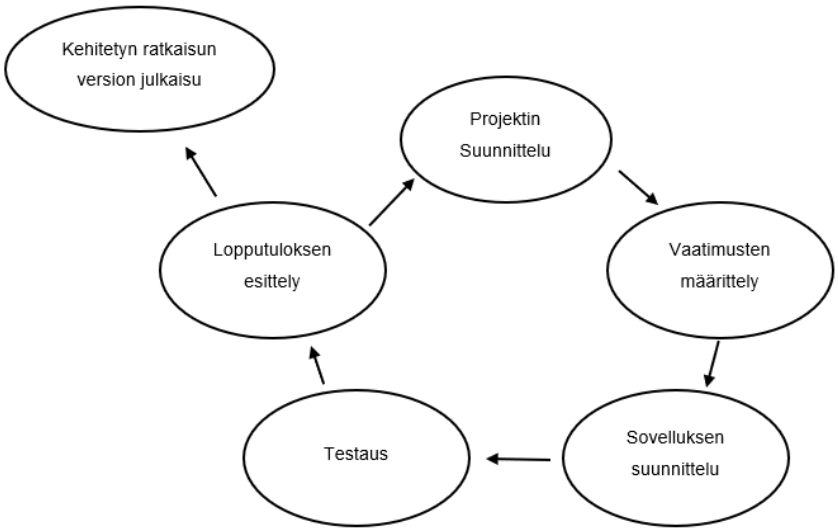
Ohjelmistonkehityksen yritystoiminnassa käytetään suurimmaksi osaksi erilaisia käytännöllisempiä ohjelmistonkehityksen elinkaarien malleja, kuten esimerkiksi Agile ja vesiputousmalli. Näiden käytössä olevien ohjelmistonkehitysten mallien tarkoituksena on tukea yritysten sovelluskehityksen projekteja, jotta valmis ratkaisu olisi käytännöllinen ja projektin liittyvät kulut olisivat pieniä. Suurimman osan sovelluskehityksen elinkaarien malleissa on jaettu vaiheisiin, jonka pohjalta mahdollisesti projektitiimit etenevät ja rakentavat sovelluksia käytössä olevan ohjelmistonkehityksen elinkaarin mallin mukaisesti.

Käytännöllisemmät ohjelmoinnin kehityksen elinkaarien mallit eroavat avoimen lähdekoodin elinkaaresta siten, että avoimen lähdekoodiin mallilla ei ole kiireellistä aikataulutusta, että jokin versio pitäisi olla tietyssä ajassa valmis. Kun taas näissä käytännöllisemmissä ohjelmistonkehityksen elinkaarissa on määritelty aikataulu.

Avoimissa lähdekoodin elinkaarissa keskitytään enemmän sen lähdekoodin yhteisöön ja lähdekoodin kehitykseen ja ylläpitoon, jossa yhteisöllä on erilaisia rooleja, jotka keskittyvät johonkin tiettyyn aiheen alueeseen, josta sitten koodin alkuperäiset ylläpitäjät päättävät onko se tarpeeksi hyvä, jotta voidaan liittää alkuperäiseen versioon. Kun taas asiakasprojekteissa käytettävissä kehityksen elinkaarissa tulevat ohjelmiston muutokset vaativat lisää yhteistyötä, aikaa ja sovittelua sidosryhmien kanssa.

6.1.1 Agilen ohjelmistonkehityksen malli

Agilen ohjelmistonkehityksen elinkaaren mallin tarkoituksena on tukea määriteltyyn projektinryhmän ohjelmistonkehityksen tuotantoa, jotta rakennettu ratkaisu olisi mahdollisimman nopeasti saatavilla käyttöön. Agilen menetelmän käyttäminen ohjelmistonkehityksessä tukee projektissa olevien henkilöiden päätöksentekoa ja parantaa kommunikaatiota sidosryhmien kanssa, jotta ohjelmistonkehityksen projekti olisi tuottoisa yritykselle. (Greene & Stellman 2014.)



Kuva 4. Agilen kehityksen elinkaaren iteraatiomalli (Tutorialspoint 2019.)

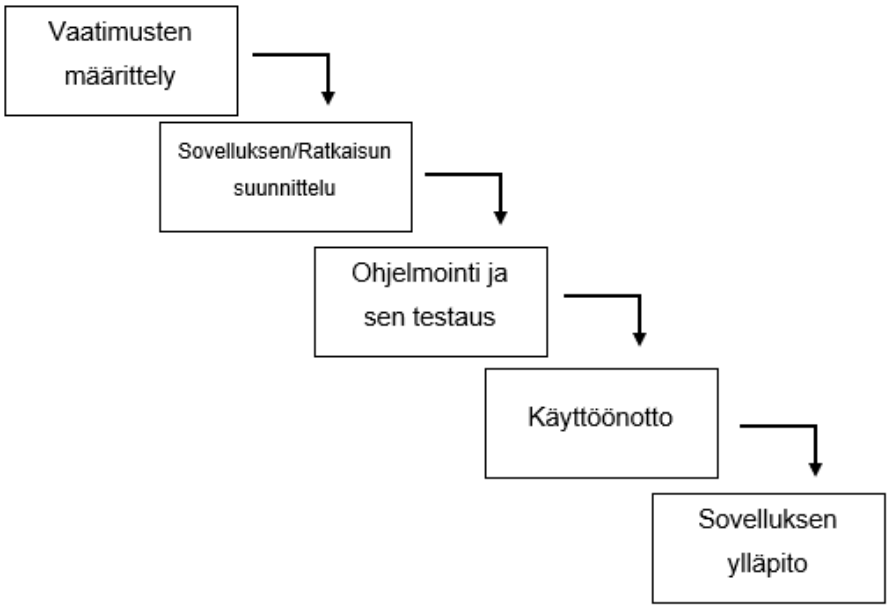
Agilen ohjelmistonkehitykseen mallin tarkoituksena on käyttää iteroinnin tyylistä ohjelmistonkehitystä eli projektin kehitys jaetaan ajastettuihin jaksoihin, joissa jokaisen iteroinnin tavoitteena on rakentaa saatujen vaatimusten pohjalta sovellus, josta aina jokaisen uuden iteraation tavoitteena on kehittää sovellusta käytännöllisempään päin saatujen palauteiden ja ideoiden pohjalta, jotta saadaan jokaisella kerralla uudempi ja kehittyneempi version tuotantoon. (Tutorialspoint 2019.)

Kuten kuvassa 4 näkyy, jokaisen iteraation lopussa rakennetun version tuotos esitellään asiakkaalle ja käsitellään, että onko ratkaisu tarpeeksi hyvä ja toimiva. Jos iteraation aikana kehitelty versio hyväksytään, niin hyväksytty versio julkaistaan tai jos esittelyn aikana on tullut erimielisyyksiä tai muutosehdotuksia ratkaisulle, niin käynnistetään iteraatio uusiksi ja kehitellään ratkaisua uusien ideoiden ja vaatimusten pohjalta.

6.1.2 Vesiputouksen elinkaari

Vesiputouksen ohjelmistonkehityksen elinkaari on yksi vanhempia käytössä olevista ohjelmistonkehityksen elinkaarien käyttötavoista. Vesiputouksen elinkaaren ideana sovelluskehityksessä on, että vaiheet käsitellään peräkkäisesti yhteen suuntaan päämäärää kohti eli tehdään alusta loppuun asti yhtenä iteraationa. Seuraavaan vaiheeseen siirrytään vasta silloin, kun edellinen vaihe on suoritettu lopullisesti. (javaTpoint 2019)

Lähteen (Lucidchart Content Team 2017.) blogissa kirjoitetun tekstin mukaan vesiputousmalli on tehokkain malli projekteissa, joissa asiakkaan kanssa tehdyt vaatimusmäärittelyt ovat tarpeeksi laajat ja yksityiskohtaiset, jotta ratkaisun rakentaminen olisi menestyksekäs. Vesiputousmallin käyttäminen ei ole suositeltavaa, jos iteraation aikana tulee useita erilaisia muutoksia, koska tämä malli vaatii, että projekti alkaisi siinä tapauksessa alusta.



Kuva 5. Vesiputouksen ohjelmistonkehityksen elinkaari kuvitettuna (JavaTpoint 2019.)

6.2 Yhteenveto

Avoimen lähdekoodin kehityksen elinkaari eroaa siten, että sen julkaistun version jälkeen tulee koko ajan palautetta yhteisöltä ja myös sen yhteisön kautta saadaan nopeammin korjausehdotuksia ja tietoa mahdollisilta koodin haavoittuvuuksilta. Kun taas tietyssä suljetussa projektissa yksityiselle asiakkaalle tai ryhmälle projektin ongelmien löytäminen ei ole välttämättä nopeata. Omasta mielestäni ohjelmistonkehityksen elinkaarien mallit pohjautuvat samoille ideoille, mutta avoimen lähdekoodin kehityksen elinkaaressa otetaan mukaan enemmän eri osapuolia sovelluksen kehitykseen.

7 Avoimen lähdekoodin työkalut lyhyesti

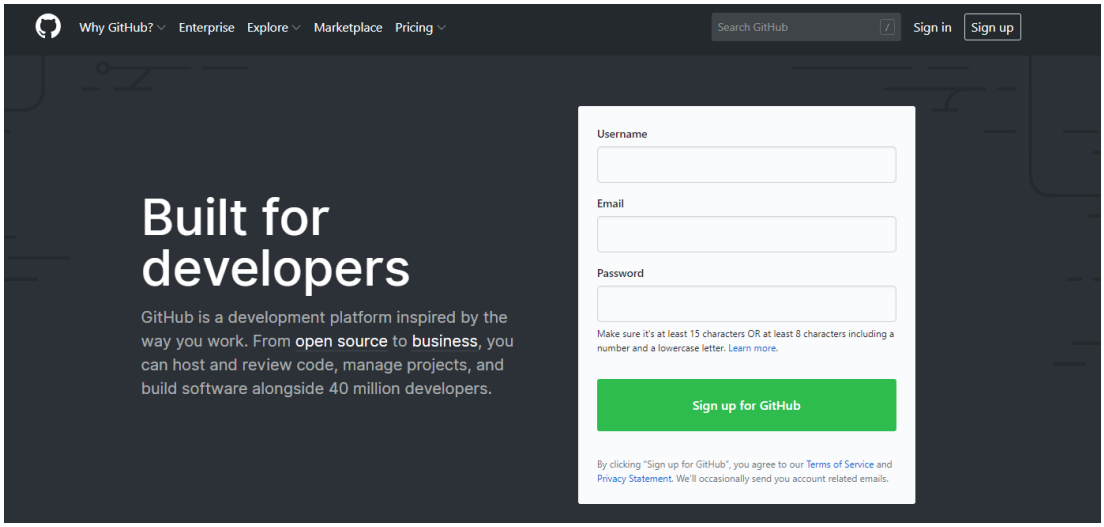
Avoimelta lähdekoodin ja erilaisten ohjelmointikielien pohjalta on kehitelty erilaisia ohjelmoinnin liittyviä ekosysteemejä, työkaluja ja muita hyödyllisiä sovelluksia, jotka mahdollisesti tukevat ja helpottavat ohjelmistonprojektien etenemistä ja niiden ylläpitoa.

Ohjelmistonkehitykseen kuuluu erilaisia suuntautumisen rooleja kuten muun-muassa Front-end, Back-end ja DevOps, joihin jokaiselle mainitulle suuntautumisen roolille on tarjolla laaja valikoima työkaluja, jotka tukevat heidän toiminnassansa ja työskentelyssä.

Avoimen lähdekoodin suosion nousemisen pohjalta on luotu erilaisia teknologioita eri tarpeisiin, jotka myös tukevat alan it-ammattilaisia jopa työskentelyssä erilaisissa projekteissa ja niiden ylläpidossa. Ajan myötä alalla toimivat ammattilaiset ovat käyttäneet erilaisia suosittuja avoimien lähdekoodin teknologioita ja myös niiden pohjalta on rakennettu erilaisia suosittuja sovelluksia ja verkkosivuja erilaisiin käyttöön. Sivustoja, jotka mahdollistavat erilaisten avoimen lähdekoodin projektien tai sovellusten jakamista ovat mm. GitHub tai GitLabs.

7.1 GitHub

GitHub lähdekoodin versionhallinta-alusta julkaistiin vuonna 2007 ja on yksi mahdollisista ohjelmistonkehittäjien versionhallinta-alustoista, joka mahdollistaa avoimen lähdekoodin tallentamista ja jakamista ympäri maailmaa. GitHub -alusta käyttää Git -nimistä versionhallintatyökalua, jota käytetään käyttöjärjestelmien komentorivin käyttöliittymissä, mikä helpottaa projektien ja sovellusten lähdekoodin tallentamista ja niiden versioiden päivittämistä. (Finley 2012.)



Kuva 6. GitHubin Etusivu (GitHub 2019.)

8 Tutkimus liittyen aiheeseen

Opinnäytetyön tutkimuksen osiossa tein kyselyn liittyen avoimen lähdekoodin tarpeellisuuteen ja kyselyn kohderyhmäksi asetin yrityksiä, jotka toimivat it-alan ohjelmoinnin tai muun digitaalisen aiheen kehityksessä. Tarkoituksena oli saada kerättyä alalla toimivien ammattilaisten vastauksia aihealueeseen. Kyselyistä vastausten ajankohdaksi valitsin 13.1 – 27.1, jossa tällä ajalla pyrin etsimään alalla toimivia yrityksiä, jotka voisivat vastata kyselyyn, jonka jälkeen voin analysoida tuloksen.

Lähetettyjä kyselyitä eri yrityksille oli 22 kappaletta, joista saatujen vastaajien määrä oli 13 kappaletta. Tietenkin tavoitemäärästä se oli odotettua alhaisempi ymmärtäen, että kaikilla ei ole sattumalta vapaata aikaa kyselyn vastaamiseen. Myös ottaen huomioon sen, että kysely oli suomeksi, mikä voi vaikuttaa vastaajien määrään. Syynä voi olla, että monien it-alan yrityksien työkielenä voi olla joku muu kuin suomi.

Kyselyn pääaiheena oli avoin lähdekoodi ja sen merkitys projekteissa ja ammattiympäristössä. Sekä millaisia ongelmia voi ilmetä avoimien lähdekoodien käytössä ja mistä mahdollisesti ammattilaiset etsivät ratkaisun projektin haasteisiin tai ilmeneviin ongelmiin. Tarkemmin kyselyn kysymyksiin saa tutustua työn lopussa (liite 1).

8.1 Kuinka iso rooli avoimilla lähdekoodin teknologioilla on ohjelmistoprojekteissanne?

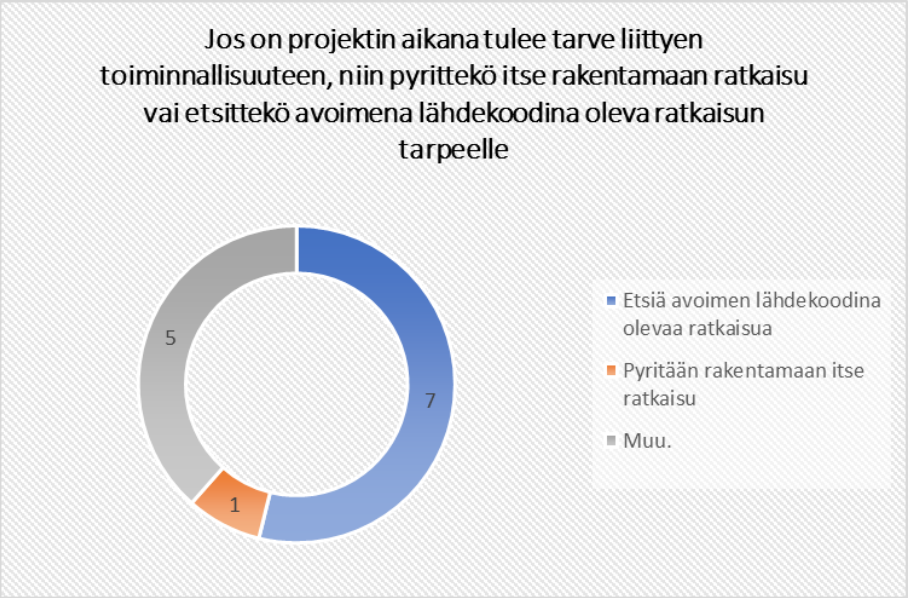
Tämän kysymyksen ajatuksena oli saada vastauksia siihen, kuinka iso rooli on avoimella lähdekoodilla ja siihen liittyvillä teknologioilla projekteissa ja myös työskentely ympäristössä, jotta voidaan saada arvion avoimen lähdekoodin suosiosta.

Kysymykseen saatuun vastausten perusteella melkein kaikki vastaajat totesivat, että avoimen lähdekoodin rooli on erittäin suuri ja yksi vastaajista kirjoitti, että heidän toiminnassansa avoin lähdekoodi on ainoa vaihtoehto heidän projekteihinsa. Jotkut kyselyn vastanneista mainitsivat sen, että tarjolla olevien avoimien lähdekoodin kirjastot, jotka ovat julkaistu kaupallisen käytön lisenssin pohjalta on iso tekijä heidän projekteissaan. Lisäksi tuli esille, että avoimien lähdekoodien ansiosta yrityksen työstävien projektien resurssien kulut pysyvät matalina.

Voi olla, että joissakin yritysten toiminnassa avoimen lähdekoodin rooli ei ole kovin merkittävä, mutta todennäköisesti se on tarpeeksi merkityksellinen projekteissa, kuten suurin osa vastaajista totesivat.

8.2 Ratkaisun valitseminen projektin tarpeisiin

Ohjelmistonprojekteissa saattaa tulla eteen, että tarvitaan uusi toiminnallisuus, jonka täytyy lisätä projektiin. Työtiimi tai yksittäinen ohjelmistonkehittäjä pohtii vaihtoehtoja ongelman ratkaisemiseksi. Vaihtoehtona voi olla ratkaisu, että rakennetaan itse, käytetään avoimen lähdekoodin pohjalta rakennettua ratkaisua tai kahden mainitun ratkaisun yhdistelmä.



Kuva 7. Jos projektin aikana tulee jonkinlainen tarve liittyen toiminnallisuuteen, niin pyrittekö itse rakentamaan ratkaisu sille vai pyrittekö etsiä avoimen lähdekoodina olevan ratkaisun tarpeelle?

Kuten kuvassa 7 näkyy, saatujen vastausten määrästä noin puolet olivat valinneet, että etsivät suoraan tarvittavan toiminnallisuuden avoimena lähdekoodina. Yksi vastanneista kirjoitti, että rakentaa tarvittavan ratkaisun itse. Toiseksi eniten vastanneista vastasivat, että ratkaisun valinta on tilannesidonnainen. Yksi vastaajista totesi, että ”valitaan avoimen lähdekoodin ratkaisua, mikäli se ei vaadi liian suuria kompromisseja”. Kuten huomataan, että avoimen lähdekoodin rooli työkaluna on suuri.

8.3 Mistä etsiä avoimen lähdekoodin ratkaisua tai työkalua projekteihin?

Mistä ammattilaiset etsivät tarpeellista avoimen lähdekoodin ratkaisua, joka mahdollisesti helpoittasi tärkeiden projektien tai omien projektien edistymisessä. On olemassa jo jonkin verran avoimen lähdekoodin projektien palveluita, jotka jakavat julkaistuja projekteja eteenpäin muille saataviksi erilaisiin tarpeisiin, kuten esimerkiksi GitHub.

Jos on tarvetta saada valmis ominaisuus käyttöön, niin mistä sitä lähdetään heti etsimään. Kun ajatellaan, että kaikki tarjolla olevat jakelupalvelut ei ole välttämättä niin turvallisia kuin toiset.

Vastaajien keskuudessa on tullut esille suurimmaksi osaksi, että he etsivät netistä tietyillä avainhakusanoilla, jonka tuloksena on pyritty etsimään: esimerkiksi linkki GitHubin, GitLab tai ohjelmistonkehittäjien suosima kyselypalsta StackOverflow, jotka tarjoavat mahdollisen tarpeellisen ratkaisun projektin etenemiseen tai aloittamiseen. Tietenkin myös mitä suositumpi on jonkin avoimen lähdekoodin tarjoama ekosysteemi tai kirjasto, niin sitä isommalla todennäköisyydellä sen käyttöönottamista harkitaan. Isoimmassa yrityksessä kuitenkin pyritään hyödyntämään siellä työskentelevien ammattilaisten kokemusta ja kerätään tarpeeksi laajasti mielipideitä, joiden pohjalta saadaan tarpeeksi hyvä idean, millä avoimen lähdekoodin teknologioilla lähtevät rakentamaan ratkaisua.

8.4 Kriteerit oikean avoimen lähdekoodin valitsemisessa

Kun ohjelmistonkehittäjä on valitsemassa tietynlaista avoimen lähdekoodin pakettia tai työkalua mukaan projektiinsa, niin tietenkin joitakin kriteereitä on oltava sen valitsemisessa. Minkälaisia asioita avoimessa lähdekoodin paketissa täytyy sisältyä? Tälle kysymykselle oli tullut ihan mielenkiintoisia vastauksia siihen, minkälaiset kriteereitä ammattilaiset asettavat, kun valitsevat jonkin avoimen lähdekoodin teknologian tai paketin oman projektiinsa mukaan.

Kuva 8. Avoimen lähdekoodin paketin valinta kriteerit saatujen vastausten pohjalta.

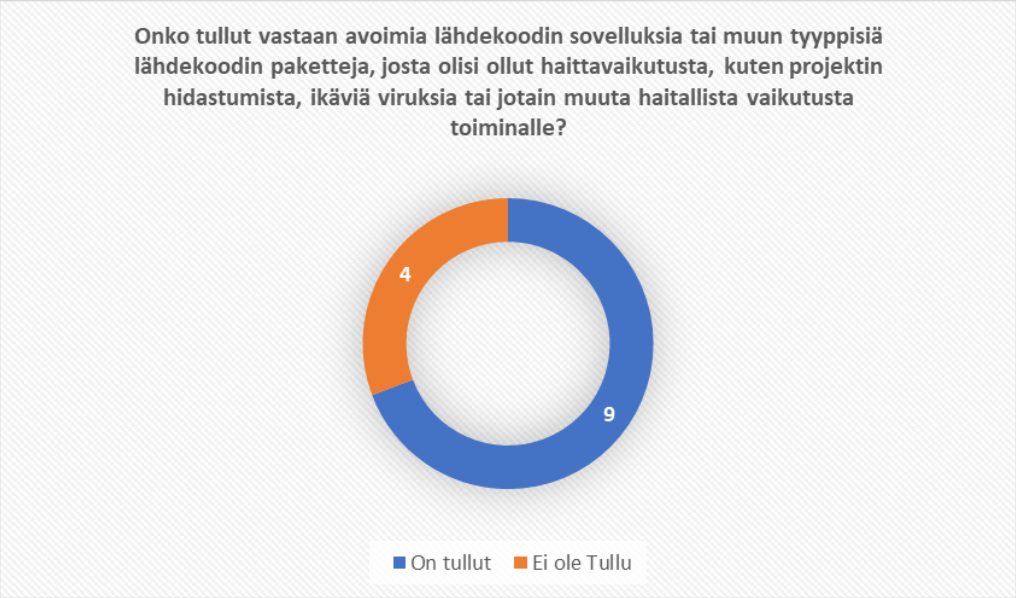
Aika usein mainittu kriteeri on ollut se, miten usein lähdekoodia päivitetään, mikä kertoo sen toimivuudesta, turvallisuudesta ja sen suosiosta. Mitä isompi suosio, sitä todennäköisemmin siitä pidetään huolta ja kaikki haavoittuvaisuudet todennäköisesti korjataan tehokkaammin. Vastanneille on myös tärkeä, millainen kokemus muilla on ollut sen käyttämisestä esimerkiksi, onko kukaan muu toteuttanut aikaisemmin esille tullutta ominaisuutta jossain toisessa projektissa ja kuinka helppokäyttöinen se on hänen mielestänsä. Mielenkiintoista on se, että vastanneista vain kaksi henkilöä mainitsi

avoimen lähdekoodin turvallisuudesta erikseen, koska tietoturvallisuus on yksi tärkeistä asioista ohjelmistonkehityksessä ja se vaikuttaa erittäin paljon sovelluksen luotettavuuteen.

8.5 Avoimen lähdekoodin haittavaikutukset ammattilaisten näkökulmasta

Kun ajatellaan avoimen lähdekoodin tarjontaa, se on laaja ja jossain määrin helposti saatavilla. Mutta, jos ei tutkita sen sisältöä kunnolla, niin seuraksensa voi tulla haitallinen vaikutus projektiin ja mahdollisesti yrityksen toimintaan. Esimerkiksi jakaja on voinut asettaa viruksen, haitallisen scriptin tai ylläpitänyt lähdekoodia huonosti. Asian korjaamiseksi menee aikaa ja resurssia, kun ammattilainen joutuu uudelleen rakentamaan sovelluksen.

Edellä kuvatuista syistä olen jakanut tämän kysymyksen kahteen osaan, eli ”Onko tullut jonkinlaista haittavaikutusta avoimen lähdekoodin käytöstä?”. Jos vastaus on kyllä, niin jatkokysymyksenä oli ”Miten ongelma oli korjattu?”. Jos minkäänlaisia ongelma ei ole tullut vastaan, niin miten on pyritty välttämään tulevilta ongelmilta.



Kuva 9. Vastaukset jaettunaan kahteen ryhmään liittyen, onko tullut haittavaikutuksia avoimen lähdekoodin käytöstä.

Kuten kuvassa 9 näkyy, suurimalle osalle vastaajista on ollut tullut vastaan jonkinlainen ongelma avoimen lähdekoodin käyttämisen yhteydessä.

Saatujen vastausten mukaan ongelmat on korjattu seuravilla tavoin kuten mm. päivittämällä käytettyjen avoimien lähdekoodien pakettien versioita, korjaamalla käytettyä pakettia itse tai paikkaamalla haitallisen paketin omatekoisella ratkaisulla.

Vastaajilta tuli myös hyviä neuvoja, miten välttää ongelmatilanteita, kun käytetään avoimia lähdekoodin paketteja projekteissa. Mainittuja asioita olivat mm. käyttämällä suosittuja

avoimen lähdekoodin kirjastoja, monitoroida projektissa käytettyjä avoimia lähdekoodin paketteja, tutkimalla haluttua lähdekoodin sisältöä tarkemmin (esim. lisenssi tai koodi), rajoittamalla avoimien lähdekoodien käyttö projektissa ja huomioimalla aikaisempi tuntemus aiheesta.

Eli kuten huomaamme, niin pitämällä riman korkealla avoimen lähdekoodin paketin valitsemisessa pärjää hyvin, mutta silti on hyvä olla tarkkaavainen versioiden päivityksessä ja tutkia avoimen lähdekoodin paketin sisältöä tarkemmin, jotta välttyy tulevilta ongelmilta.

8.6 Vastaajien kokemus avoimen lähdekoodin kehityksestä

Aiheeseen liittyen tietenkin kiinnostaa, kuinka moni kyselyn vastaajista on ollut mukana jossain avoimen lähdekoodin kehityksessä tai jopa mahdollisesti itse julkaissut avoimen lähdekoodin pakettia. Kun aikaisemmin käsitellyssä kappaleessa (4. Avoimen lähdekoodin hyödyllisyys) on todettu, että ohjelmistonkehittäjä voi hyödyntää itseään kehittämällä erilaisia asioita käyttämällä avoimen lähdekoodin teknologioita, joten luonnollista oli myös kysyä vastaajilta heidän omasta kokemuksestaan.

Kuva 10. Vastaukset liittyen onko vastaajista itse ollut mukana avoimen lähdekoodin kehityksessä.

Suurin osa vastaajista on ollut jossain muodossa mukana kehittämässä tai julkaissut avoimen lähdekoodin pakettia. Tästä huomaa, että oleminen mukana yhteisöpohjaisessa ohjelmistonkehityksessä auttaa kehittämään taitoja ja tuo enemmän näkemystä avoimen lähdekoodien käytöstä.

Projekteissa käytettävien avoimen lähdekoodien saatavuus voi helpottaa projektien etenemistä huomattavasti. Mutta voiko olla mahdollista, että jossain menee raja eli onko tullut käytettyä projekteissa liian monta erilaista avoimen lähdekoodin pakettia, jotka loppujen lopuksi hidastavat projektin kulkua.

Kuva 11. Vastaajien mielipide liiallisesta avoimen lähdekoodin pakettien käyttämisestä.

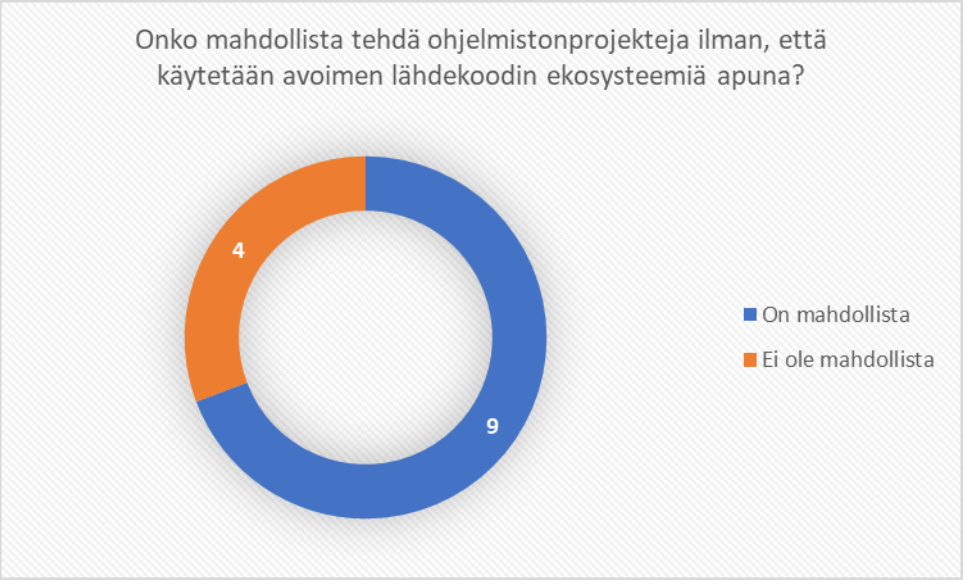
Osa vastaajista totesi, että liiallisesta avoimen lähdekoodin pakettien käytöstä saattaa tulla jonkinlaista haittavaikutusta. Kuten vastauksissa oli mainittu mm. projektin sovellus voi viedä liikaa kokonaismuistia ja mahdollisesti myös muita projektissa olevia resursseja ja projektin ylläpidon tarve nousee. Myös vastauksissa oli mainittuna, että avoimien lähdekoodien pakettien toistensa riippuvuudet nousevat ja sen kautta koodissa voi ilmetä erilaisia virheitä. Lopuksi oli mainittu, että yritysasiakkailla voi olla jonkinlainen epäluottamus eli he eivät välttämättä luota tuntemattomaan koodiin.

Toinen osa vastaajista sitten totesi, että avoimella lähdekoodin käytöllä ei ole rajaa. Kuten yksi vastaajista mainitsi, että projektin voi tehdä kokonaan avoimista lähdekoodeista, kun ohjelmistonkehittäjällä on tiedossa minkälaisia paketteja hän käyttää projektissa. Toisin sanoen, pitää olla tarkkaavainen ja pohtia kahdesti, että tarvitaanko projektissa esille tullutta avoimen lähdekoodin pakettia vai ei.

8.7 Projektit ilman avoimen lähdekoodin teknologioita

Tietenkin yleisesti avoimen lähdekoodien pohjalta olevien teknologioiden tai työkalujen suosio nousee koko ajan. Nykyaikana käytetään enemmän avoimia teknologioita omissa projekteissa, jonka pohjalta kehitetyt sovellukset ovat riippuvaisia avoimista teknologioista, joten tulee mietittyä, onko mahdollista kehittää sovelluksia ilman avoimien lähdekoodien teknologioiden apua.

Kyselyssä oli kaksiosainen kysymys, jossa ensimmäisessä osassa kysyin, onko mahdollista tehdä projekteja ilman avoimien lähdekoodin teknologioiden apua ja toisessa osassa pyysin perustellut vastaukselle.



Kuva 12. Onko mahdollista tehdä ohjelmistonprojekteja ilman, että käytetään avoimen lähdekoodin ekosysteemiä apuna?

Suurin osa vastaajista valitsivat kahdesta vaihtoehdosta, että on mahdollista tehdä projekteja ilman avoimen lähdekoodin ekosysteemin apua, mutta jatkokysymyksen vastauksissa he olivat sitä mieltä, että se ei ole järkevää. Syinä on ollut, että projektien kustannukset nousisivat liian korkeaksi ja ajallisesti se vaatisi enemmän aikaa saada tuotantoon. Siis teoriassa on mahdollista tehdä projekteja ilman avoimen lähdekoodin teknologioita, mutta käytännössä ei ole. Kuten yksi vastaajista totesi, että ei voi välttää avoimen lähdekoodin teknologioita nykyajan ohjelmistonkehityksessä.

Eli aiheeseen tiivistettynä avoimien lähdekoodin teknologioiden käyttö ja niiden menetelmät ovat yksi iso osa nykypäivää, joka vaikuttaa huomattavasti ohjelmistonkehityksen alaan.

9 Pohdinta

Opinnäytetyössä käsiteltyjä aiheita liittyen avoimen lähdekoodin menetelmään ja myös tutkimus sen pohjalta oli kokonaisuudessa erittäin kiinnostavaa ja jossain määrin opettavaista. Ajatellen, kun itse olen erikoistumassa ohjelmistonkehittäjäksi ja työkaluina tulee olemaan todennäköisesti avoimen lähdekoodin pohjalta olevia teknologioita, joita olen aikaisemmin käyttänyt, mutta tieto teknologioiden ”avoimuuden” termistä ei ollut kovin iso.

Joten opinnäytetyön tavoitteena oli kokonaisuudessa saada jonkinlainen pohjanäkemys avoimen lähdekoodin menetelmästä ja kuinka iso rooli sillä on ammattilaisessa ympäristössä, josta saada käsitys avoimen lähdekoodin käytöstä.

Kokonaisuudessa opinnäytetyön paketin sisällöstä olen omasta mielestä ihan tyytyväinen tulokseen verraten alussa asettamaniin tutkimuskysymyksiin. Toki saatuun kyselyn vastauksien määrä olisi voinut olla suurempi, jotta olisi voinut tarkemmin avata käsiteltävän aiheen. Mutta olen ihan tyytyväinen saaduista vastausten laadusta, josta sitten pystyin kirjoittaman tutkimuspohjan. Toisaalta saaduista vastausten määrän perusteella ei voi tehdä isoja johtopäätöksiä, mutta voi saada pohjaidean avoimen lähdekoodin roolista ja avoimien teknologien käytöstä.

Lyhyenä yhteenvetona on hyvä pohtia, kuinka tärkeä avoimen lähdekoodin kehitys on. Omasta näkökulmasta ajattelen, että avoimen lähdekoodin kehitysmenetelmät ovat aika tärkeätä nykyajassa ja kuinka paljon resurssia alan yritykset mahdollisesti voivat säästyä käyttämällä tiettyjä julkaistuja avoimen lähdekoodin teknologioita ja ekosysteemiä. Myös teknologian kehittymisen myötä enemmän teknologioita tullaan käyttämään avoimena lähdekoodina, mitä tuli esille opinnäytetyön tekemisen aikana. Ison ohjelmistonkehityksen yhteisön ansioista on tapauksia, jossa halutaan antaa takaisin yhteisölle jollain tapaa, joka auttaa avoimen lähdekoodin menetelmän pitäminen elossa.

Lähteet

Choosealicense. 2019. No License. Luettavissa: https://choosealicense.com/no-permission/. Luettu: 14.10.2019.

Congdon, L. 03.02.2015. 8 advantages of using open source in the enterprise. Luettavissa: https://enterprisersproject.com/article/2015/1/top-advantages-open-source-offers-over-proprietary-solutions. Luettu: 22.10.2019.

Durkovic, J. & Vukovic, V. & Rakovic, L. 2008. Open Source Approach in Software Development – Advantages and Disadvantages. Luettavissa: https://pdfs.semanticscholar.org/dfa1/68ff667c50bf21e7dc3521b65a26b9537fde.pdf. Luettu: 26.10.2019.

Engard, N. C. 2010. Practical open source software for libraries. Chandos Publishing 2010. Englanti. Part1. Chapter 1.

Finley, K. 14.8.2012. What Exactly is GiHub Anyway? Luettavissa: https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/. Luettu: 10.11.2019.

Fitzgerald, B. & Kesan, J. P. 2011. Adopting Open Source Software: A Practical Guide. MIT Press 2011. Englanti.s. 8-27.

Free Software Foundation 2019. Luettavissa: https://www.fsf.org/about/. Luettu: 5.10.2019.

GitHub. 2019. Saatavilla: https://github.com/. Käyty: 10.11.2019.

GNU Operating System 2019. What is free software? Luettavissa: https://www.gnu.org/philosophy/free-sw.html. Luettu: 5.10.2019.

GNU Operating System 2019. Frequently Asked Questions about GNU Licenses. Luettavissa: https://www.gnu.org/licenses/gpl-faq.html#WhyUseGPL. Luettu: 11.11.2019.

Goldstein A, Open Source Licenses in 2020: Trends and Predictions. Luettavissa: https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions. Luettu 6.2.2020.

Golden, B. 2004. Succeeding with Open Source. Addison-Wesley Professional 2004. Luettu 26.10.2019.

Greene, J. & Stellman, A. 2014. Learning Agile. O'Reilly Media 2017. Englati. Luettu 3.11.2019.

Guliani, G. & Woods, D. 2005. Open Source for the Enterprise. O'Reilly 2005. Englanti. Luettu: 29.10.2019.

Gunjan, K. & Pawan, K. 2011. Open Source Software (oss): Realistic Implementation of OSS in School Education. Vol. 7. Issue 2. Luettu: 17.10.2019.

Investintech. 26.3.2018. Pros & Cons of Open Source in Business. Luettavissa: https://www.investintech.com/resources/blog/archives/7975-pros-cons-open-source-business.html. Luettu: 26.10.2019.

JavaTpoint. 2019. Waterfall model. Luettavissa: https://www.javatpoint.com/software-engineering-waterfall-model. Luettu: 5.11.2019.

Johnston, K. & Begg, S. & Tanner, M. 2013. Exploring the Factors Influencing the Adoption of Open Source Software in Western Cape Schools. Vol. 9. Iss. 2. Luettavissa: https://eric.ed.gov/?id=EJ1071356. Luettu: 17.10.2019.

Kapitsaki, M. G. & Tselikas, D. N. & Foukarakis, E. F. 2014. The Journal of Systems and Software: An insight into license tools for open source software systems. Cyprus. s. 2 Luettavissa: https://www-sciencedirect-com.ezproxy.haaga-helia.fi/science/article/pii/S0164121214002945. Luettu 9.10.2019.

Kaufman, R. J. 16.2.2018. How to make sense of the Apache 2 patent license. Luettavissa: https://opensource.com/article/18/2/how-make-sense-apache-2-patent-license. Luettu: 11.11.2019.

Kraus, J. 22.11.2016. How your company can benefit from contributing to open source. Luettavissa: https://www.sitepoint.com/how-your-company-can-benefit-from-contributing-to-open-source/. Luettu: 23.10.2019.

Lawerence, E. R. 2004. Open Source Licensing: Software Freedom and Intellectual Property Law. Prentice Hall 2004. Englanti. Luettu 10.11.2019.

Lee, J. 20.5.2015. Why do people contribute to open source projects. Luettavissa: https://www.makeuseof.com/tag/people-contribute-open-source-projects/. Luettu: 18.10.2019.

Lucidchart Content Team. 2.10.2017. The Pros and Cons of Waterfall Methodology. Lähde: https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology. Luettu: 5.11.2019.

Maruping, L. M. & Daniel, S. L. & Cataldo, M. 2019. Developer centrality and the impact of value congruence and incongruence on commitent and code contribution activity in open source software communities.

Open Source Guides. 2019. The Legal Side of Open Source Luettavissa: https://opensource.guide/legal/. Luettu: 14.10.2019.

Open Source Initiative 2007. The Open Source Definition. Luettavissa: https://opensource.org/osd. Luettu: 5.10.2019.

Paulson, J. W. & Succi, G. & Eberlein, A. 2004. An Empirical Study of Open-Source and Closed-Source Software Products. New York. S. 2. Luettavissa: https://search-proquest-com.ezproxy.haaga-helia.fi/abitrade/docview/195578711/fulltextPDF/A9312F323CC74660PQ/1?accountid=27436. Luettu: 6.10.2019.

Potdar, V. & Chang, E. 2004. Open Source and Closed Source Software Development Methodologies. Australia. s. 106 – 110 Luettavissa: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.8413&rep=rep1&type=pdf#page=106. Luettu: 6.10.2019.

Roberts, A. J. & Hann, I-H. & Slaughter, A. S. July 2006. Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects, Vol. 52. No. 7. Luettu 18.10.2019.

Singh, P. V. & Phelps, C. 2013. Networks, Social Influence and the Choice Among Competing Innovations: Insights from Open Source Software Licenses. Vol. 25 Issue. 3 Luettu: 7.10.2019.

Sourav, M. & Shyamalendu, K. & Palash, R. 2011. Open Incremental Model – A open Source Software Development Life Cycle Model (OSDLC). Volume 21. No. 1. Luettavissa: https://www.researchgate.net/profile/Sourav_Mandal13/publication/241412148_Open_Incremental_Model_A_Open_Source_Software_Development_Life_Cycle_Model_'OSDLC'/links/5718d9d208aed43f63232abc/Open-Incremental-Model-A-Open-Source-Software-Development-Life-Cycle-Model-OSDLC.pdf. Luettu: 29.10.2019.

Tackaberry, A. 10.3.2018. Why you should contribute to open source software right now. Luettavissa: https://medium.com/@austintackaberry/why-you-should-contribute-to-open-source-software-right-now-bec8bd83cfc0. Luettu: 18.10.2019.

Todorovic, A. 10.07.2015. Open source licensing at GitHub. Luettavissa: https://opensource.com/life/15/7/interview-ben-balter-github. Luettu: 14.10.2019.

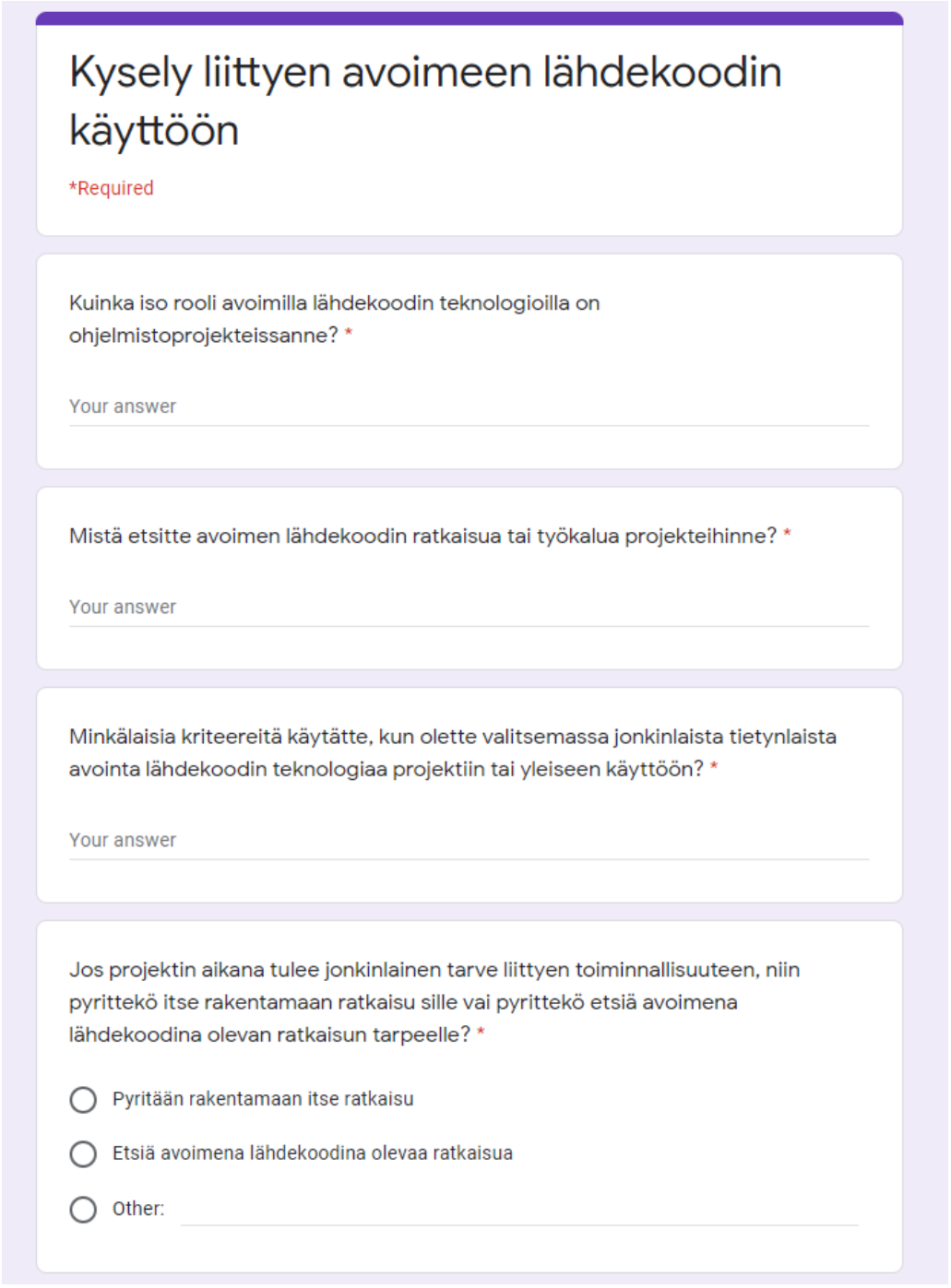
Tutorialspoint. 2019. SDLC – Agile Model. Luettavissa: https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm. Luettu: 3.11.2019.

WhiteSource 2018. Top 10 Open Source Licenses in 2018: Trends and Predictions. Luettavissa: https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions. Luettu: 14.10.2019.

Zeimer, S. & Stenz, G. 2012. The case for open source software in aeronautics. Vol. 84. Iss 3. Luettu 17.10.2019.

Liitteet

Liite 1. Tutkimuskysymykset



Kysely liittyen avoimeen lähdekoodin käyttöön

*Required

Kuinka iso rooli avoimilla lähdekoodin teknologioilla on ohjelmistoprojekteissanne? *

Your answer

Mistä etsitte avoimen lähdekoodin ratkaisua tai työkalua projekteihinne? *

Your answer

Minkälaisia kriteereitä käytätte, kun olette valitsemassa jonkinlaista tietynlaista avointa lähdekoodin teknologiaa projektiin tai yleiseen käyttöön? *

Your answer

Jos projektin aikana tulee jonkinlainen tarve liittyen toiminnallisuuteen, niin pyrittekö itse rakentamaan ratkaisu sille vai pyrittekö etsiä avoimena lähdekoodina olevan ratkaisun tarpeelle? *

- Pyritään rakentamaan itse ratkaisu
- Etsiä avoimena lähdekoodina olevaa ratkaisua
- Other:

Onko tullut vastaan avoimia lähdekoodin sovelluksia tai muun tyyppisiä lähdekoodin paketteja, josta olisi ollut haittavaikutusta, kuten projektin hidastumista, ikäviä viruksia tai jotain muuta haitallista vaikutusta toiminalle? *

- ◯ On
- ◯ Ei ole

(Jatkoa edelliseen kysymykseen) Miten olette pyrkineet välttämään, jos ei ole tullut vastaan tai jos on, niin miten olette ratkaissut ongelman? *

Your answer

Oletteko aikaisemmin itse julkaissut avoimena lähdekoodina olevaa projektia tai olleet mukana jossain olemassa olevassa avoimen lähdekoodin kehityksessä? *

- ◯ Kyllä
- ◯ Ei

Onko liiallisella avoimen lähdekoodien käytöllä haittavaikutuksia projekteissa tai muussa käytössä? *

Your answer

Onko mahdollista tehdä ohjelmistonprojekteja ilman, että käytetään avoimen lähdekoodin ekosysteemiä apuna? *

- ◯ On
- ◯ Ei ole

(Jatkoa edelliseen kysymykseen) Jos on tai jos ei ole, niin miksi? *

Your answer