Tallinna Ülikool
Informaatika instituut

Tanel Jõeäär

Tarkvara hübriidne arendus- ja ärimudel

Magistritöö

Juhendaja: Kaido Kikkas

Autor: ...……...…….... “…..” …… 2008
Juhendaja: ....…….……..……... “…..” …… 2008
Instituudi direktor: ..…….....……... “…..” …… 2008

Tallinn 2008

Sisukord

Sissejuhatus

Viimastel aastatel on plahvatuslikult kasvanud huvi avatud lähtekoodiga tarkvaralahenduste vastu. [Lerner&Tirole, 2002] Avatud lähtekoodiga tarkvara ei kujuta endast enam vaid "patsiga poiste" vaikset nohistamist, vaid on tõusnud ka suurte IT ettevõtete huviorbiiti.

Kõik tarkvaraettevõtted eksisteerivad selleks, et teenida kasumit. Selleks otsitakse uusi viise tulu kasvatamiseks ning kulude kärpimiseks. Üha enam kasutavad ettevõtted avatud lähtekoodiga tarkvara selleks, et saavutada mõlemad eesmärgid.

Käesoleva magistritöö eesmärgiks on välja selgitada, milliseid võimalusi pakub avatud lähtekoodiga tarkvara arendus ettevõttele ning kuidas avatud lähtekoodil baseeruvat hübriidset arendus- ja ärimudelit on võimalik juurutada.

Lisaks viis autor läbi uuringu, selgitamaks välja, kas Eestis leidub avatud lähtekoodi arendajaid ning milline on nende motivatsioon mõnes avatud lähtekoodi projektis kaasa löömiseks.

Töö 1. ja 2. peatükis tutvustakse avatud lähtekoodi, selle arendajaid, erinevaid aspekte ning litsentse, mille all avatud lähtekood avalikustatakse.

3. peatükis käsitleb autor avatud ja suletud lähtekoodi arendusmudeleid ning esitab hübriidse avatud lähtekoodil baseeruva arendusmudeli, mida ettevõtte saab kasutada avatud lähtekoodiga tarkvaraprojekti loomisel.

4. peatükk seob endas võimalusi pakkuda avatud lähtekoodi ärilistel eesmärkidel. Nendeks on avatud lähtekoodil baseeruv ärimudel, kus avatud lähtekoodile saab pakkuda erinevaid lisa- ning tugiteenuseid, mis võimaldavad teistel ettevõtetel saada piisav lisaväärtus avatud lähtekoodi kasutamiseks. Hübriidne ärimudel kirjeldab tarkvara sidumist erinevate tingimustega litsentsikogumi alla. Hübriidne ärimudel ei pruugi kasutada avatud lähtekoodi selle klassikalises mõistes, küll aga vabatahtlikke arendajaid.

5. peatükis esitab autor avatud lähtekoodi küpsusmudeli, mis võimaldab hinnata avatud lähtekoodi küpsust, olgu tegu tarkvaratootega, mida ettevõte soovib kasutada või enda loodud avatud lähtekoodi kasutava tarkvaraprojektiga.

6. peatükis leiab autor avatud lähtekoodi projektiga seotud ohte ning võimalusi. Samuti on juttu lähtekoodi avalikustamise tasuvusest.

Viimases peatükis esitab autor läbiviidud uuringu tulemused. Uuring käsitles olemasolevate ning potensiaalsete avatud lähtekoodi arendajate huvi ning motivatsioone Eestis avatud lähtekoodiga tarkvaraprojektis osalemiseks.

1 Avatud lähtekood

Avatud lähtekoodist on saanud reaalne fenomen. See moodustab juba praegu suure osa infotehnoloogia valdkonnast ning jätkab tõusvas joones domineerimist selle aspektide üle. [Feller, 2004] Suuremad ja väiksemad ettevõtted on näinud avatud lähtekoodi võimalusena suurendada koostööd, vähendada arenduskulusid, luua platvormi oma toodetele ning müüa teenuseid.

Avatud lähtekoodi olemus seisneb selles, et rakenduse lähtekood avalikustatakse koos tarkvaraga. Avatud lähtekood on kõigile kasutamiseks vaba; igaüks võib kasutada rakendust endale sobival moel; uurida selle tööd ning adapteerida seda oma vajadustele; jagada sellest koopiaid; täiustada seda ning jagada täiustatud koodi kogukonnaga, et teisedki sellest kasu saaks.

Lühidalt võib avatud lähtekoodi tuuma kokku võtta kolme olulise tunnusega [OSI]:

- Lähtekood peab olema avalikustatud tarkvaraga kaasas või muul moel ilma täiendava tasuta avalikustatud.
- Igaüks võib tarkvara tasuta levitada, ilma seda maksustamata või litsentsitasusid nõudmata.
- Igaüks võib tarkvara lähtekoodi modifitseerida, tuletada selle abil uusi tarkvara rakendusi ning levitada seda samade tingimuste all.

Avatud lähtekoodi tarkvaraprojektid Linux ja Apache on tõestanud, et suurte ja komplektsete tarkvarasüsteemide ehitamine, arendamine, ülalpidamine ning laiendamine on võimalik. [Benkler, 2002]

1.1. Avatud lähtekood ettevõttes

Juhtumiuuringud Apache ja Mozilla põhjal [Mocus et al., 2003] on avaldanud, et avatud lähtekoodil baseeruv tarkvara arendus aitab toota kõrge kvaliteediga tarkvara odavalt ja kiiresti.

Kui esmapilgul võiks arvata, et avatud lähtekoodiga tarkvara arendamine ning kasutuselevõtt on vaid suurte IT ettevõtete pärusmaa (milliseid Eestis kahjuks ei leidu), siis Bonacossi jt

poolt läbi viidud uuring [Bonacossi et al, 2006] näitas, et suurus ei mõjuta ettevõtte avatust avatud lähtekoodi kasutuselevõtuks. Seega on võimalik avatud lähtekoodi mudeleid rakendada ka keskmistes ning väikestes IT firmades.

Heal avatud lähtekoodi projektil on rohkem potentsiaali paindlikumaks tehnoloogiaks ning kiiremaks innovatsiooniks kui sarnasel ärilise tarkvara projektil. [Borrell, 2001] Lisaks suudab avatud lähtekood paremini toime tulla nn „tarkvarakriisi" [Feller&Fitzgerald, 2000] probleemidega, milleks on kirjeldavad süsteemide liiga pikk arendusprotsess, liigne kulukus ja vähene töökindlus.

1.2. Avatud lähtekoodi kogukonnad

Järgnevad väited iseloomustamaks avatud lähtekoodi kogukondi on refereeritud Sharma et al artiklist [Sharma et al, 2002].

Avatud lähtekoodi kogukonnad erinevad fundamentaalselt teistest traditsioonilistest organisatsioonidest. See koosneb paljudest vabatahtlikest arendajatest, kes teevad kaastööd individuaalselt või osana ajutisest töörühmast.

Avatud lähtekoodi projekti ümber tekkiv kogukond on voolav, arendajad võõrduvad kogukonnast, samas kui uued tekivad juurde. Seetõttu pole kogukonnal ka kindlaid piire.

Avatud lähtekoodi arendajad on geograafiliselt hajutatud ning kohtuvad näost-näkku väga harva. Samuti ei saa nad pühendada suuri blokke oma ajast avatud lähtekoodi projektiga tegelemiseks. Seega võib avatud lähtekoodi kogukonda iseloomustada detsentraliseeritud töötsooni ning asünkroonse kommunikatsiooniga.

Kuigi avatud lähtekoodi kogukonnal on mitmeid eeliseid ettevõtte tarkvara arendajate grupi ees (paindlikkus, jagatud valitsemine, vaba info voolavus), võib avatud lähtekoodi kogukonna mudeli adapteerimine siiski ettevõtetele raskusi valmistada. Siiski peaks võimalik olema mõningate avatud kogukonna aspektide ettevõttesse sulandamine. Luues ettevõttes hübriid-avatud kogukonna, võib senisest hõlpsamaks muutuda kvaliteetse tarkvara arendamine.

1.2.1. Motivatsioon

Lakhani ja Wolf on oma uurimuses [Lakhani&Wolf, 2005] välja toonud neli aspekti, miks arendajad avatud lähtekoodiga tarkvara arendamise projektides osalevad:

- Peamisteks motivaatoriteks on nauding ja õppimine
- Koodi on vaja mittetööalaste kasutaja vajaduste rahuldamiseks
- Tööalased vajadused ning karjääri puudutavad aspektid
- Kohusetunne kogukonna ees ja arvamus, et tarkvara peaks olema vaba/avatud.

Lakhani ja Wolfi järgi jaguneb arendajate motivatsioon loomuomaseks ning väliseks motivatsiooniks. Loomuomaseks motivatsiooniks nimetatakse pigem rahuolul kui kindlatel eesmärkide saavutamise nimel saadav motivatsioon. Nii võib koodi loomine valmistada selle autorile naudingut. Loomuomaseks motivatsiooniks võib pidada ka kohusetunne kogukonna ees.

Välisteks motivaatoriteks võib otseste teguritena lugeda rahalist tasu (ettevõtted võivad maksta avatud lähtekoodi projektis osalemise eest) ning kasutaja isiklikke vajadusi konkreetse koodi järele. Pikemaajalisteks motivaatoriteks võivad olla karjäärialane edasijõudmine või programeerimisoskuste omandamine.

1.2.2. Valitsemine

Üks silmapaistvaid avatud lähtekoodi kogukonna tunnusjooni on enesedistsipliin (*self-governance*). [Sharma et al, 2002] Iseennast organiseerivas kogukonnas võtavad selle liikmed endale vastutuse oma töökoormuse organiseerimise eest, jagavad omavahel ära tööülesanded vastavalt vajadusele ja oskustele ning osalevad meeskonna otsuste tegemisel.

Projektid (olgu nendeks siis arenduse alamosad) on juhtivate süsteemiarhitektide või –disainerite poolt jaotatud alamosadeks, mida haldavad rühmad või üksikisikud. [Sharma et al, 2002]. Näiteks otsustab avatud lähtekoodiga veebiserveri Apache tuleviku üle nn „tuumikarendajatest“ koosnev Apache Group, mis koosneb valitud, pikka aega projektiga tegelenud arendajatest. [Mockus et al, 2005]

1.2.3. Kogukond ettevõtte arendusprojektis

Kui ettevõte alustab avatud lähtekoodiga tarkvara arendamist, ei tohi ära unustada, et arendajatel peaks olema sõnaõigus ka otsustus- ning planeerimisprotsessis. Seetõttu ei pea avatud ei pea olema üksnes lähtekood, vaid ka tarkvaraarenduse artefaktid (nõuded, disain jne). [Robbins, 2005]

Arvestada tuleb ka, et vabatahtlikel programmeerijatel on erinevad isiklikud eesmärgid (ettevõtte omadest erinevad) projektis osalemiseks. Nende eesmärkide mõistmine on ettevõttele oluline, kuna see hoiab arendajat projekti juurde ning aitab juurde meelitada ka uusi. [Woods&Guliani, 2005]

Avatud lähtekoodi kogukondadest tuleb juttu töös edaspidigi.

2 Litsentseerimine

Avatud lähtekoodiga tarkvara kasutusõigused kirjeldab ära sellele valitud litsents. Kõik avatud lähtekoodi litsentsid jagavad põhimõtet, et igaühel peaks olema võimalus lähtekoodi ning sellest kompileeritud rakendust kasutada, kopeerida ja jagada. Selleks, et siduda tarkvaraprojekti mõne avatud lähtekoodi litsentsiga, tuleb esmalt mõista erinevate litsentside omadusi, nende kattumisi ning ebasobivusi.

Hübriidsete tarkvaraprojektide puhul on oluline, et litsents sätestaks ettevõttele vaieldamatud õigused tarkvaratootele, mida too soovib avaliku litsentsiga siduda. Vaid nii saab ettevõte oma tarkvaratoodet maksustada, muuta vajadusel litsentsitingimusi ning tarkvaratoodet erinevatel litsentsitingimustel levitada. [Välimäki, 2003]

Käesolev peatükk kirjeldab mõningad tuntuimad avatud lähtekoodi litsentsid, toob võrdlused suletud lähtekoodi litsentsiga ning vaatleb hübriidset litsentsikogumit QNX Software Systems näitel.

Avatud lähtekoodi ning vaba tarkvara litsentsidest ning nende õiguslikest küsimustest on O'Reilly kirjastuse kodulehel vabalt kättesaadav selleteemaline e-raamat (aadressil http://www.oreilly.com/catalog/osfreesoft/book/).

2.1. Avatud lähtekoodi litsentsid

Erinevate avatud lähtekoodi litsentside tingimuste kattuvuste ning kokkusobimatuste mõistmine on väga oluline aspekt, mida igal ettevõttel tuleks avatud lähtekoodiga tarkvaraprojekti käivitamisel arvestada. [Woods&Guliani, 2005] Näiteks on GPL (GNU Public License) litsentsis kirjeldatud juhud, mil teise litsentsiga seotud tarkvara ka GPL-ga ühildub.

Mõned GPL-ga ühilduvad vabad litsentsid:

- GNU Lesser General Public License (LGPL)
- MIT/X11 litsents
- BSD litsents

Järgnevalt on kirjeldatud mõned levinuimad avatud lähtekoodi litsentsid, mida sobib aluseks võtta kas eraldiseisvana või avatud lähtekoodi litsentsina topeltlitsentsis.

2.1.1. GNU avalik litsents

GNU avalikku litsentsi (*GNU Public License*, GPL) peetakse kõigi avatud lähtekoodi litsentside eelkäijaks ning see on siiani arendajate seas eelistatuim (mõningatel hinnangutel kuni 85% kõigist projektidest). [Woods&Guliani, 2005]

GNU avalik litsents sisaldab endas selle looja, Richard Stallmani poolt kasutusele võetud, nn *copyleft*-printsiipi. Kui traditsiooniline autoriõigus (*copyright*) sätestab kõigi õiguste automaatse kuulumise autorile, siis *copyleft* annab kasutajale õiguse litsentsiobjekti vabalt kasutada, paljundada ja muuta, seni kuni ta omalt poolt ei hakka saadud õigusi edasisel levikul kitsendama. Nõnda peab iga GPL litsentsiga lähtekoodi derivatsioon olema samuti GPL litsentsi all avalikustatud. Säilitama peab ka viited algse(te)le autori(te)le.

GPL litsentsiga koodi segamine kommertstarkvarasse või katse levitada suletud lähtekoodiga rakendust on seega käsitletav GPL litsentsitingimuste rikkumisena. Lõppkasutaja võib koodi siduda ka kommertsrakendustega, eeldusel, et see püsib „majas sees“. Võtmesõnaks on siinkohal levitamine – mitte ainult müük, vaid ka valminud koodi jagamine äripartneri või kolmanda osapoolega on keelatud.

Kaks täpsemat GPL litsentsi toimimise viisi on oma artiklis kirjeldanud McGowan [McGowan, 2005].

2.1.2. LGPL

Lesser GNU Public License ehk LGPL on veidi leebem versioon GPL litsentsist. See võimaldab LGPL litsentsi kasutavaid programmiteeke ehk alamprogrammide kogumikke siduda kommertstarkvaraga. Muus osas sarnaneb see GPLiga, st. kehtib *copyleft*-i põhimõte.

LGPL sätestab, et koodi, mis kasutab LGPL litsentsi all olevaid programmiteeke (mitte ei ole seotud selle lähtekoodiga), ei saa lugeda selle derivatsiooniks ning langeb seetõttu antud litsentsiskoobist välja. [LGPL]

2.1.3. BSD

BSD litsents on vaba litsents, mis arendati välja 1970ndatel aastatel Berkeley Ülikoolis, litsentseerimaks BSD Unixit. Tegemist on enimkasutatava mitte-*copyleft* litsentsiga. [Kikkas, 2006]

BSD litsentsi peetakse äärmiselt lubavaks, selle sätted võimaldavad piiramatut lähtekoodi ülevõtmist teisteks tarkvaraprojektideks või BSD Unixi kinnisteks kommertsversioonideks, tingimuseta lähtekoodi jagada. Seda võimalust on kasutanud nii Apple oma Mac OS X arenduses [Woods&Guliani, 2005] kui ka Microsoft Windows [Kikkas, 2006].

2.1.4. MIT/X11

MIT/X11 litsents pärineb algselt Unixi graafilise keskkonna *X Windows System* loojatelt. Sarnaselt BSD litsentsiga annab see kasutajale õigused tarkvara kasutada, kopeerida, muuta, levitada ja müüa. MIT/X11 litsents on GPL-ühilduv.

2.1.5. Apache litsents 2.0

Apache veebiserver on üks tuntumaid ja paindlikemaid avatud lähtekoodi projekte. Apache litsents, mille versioon 2.0 kinnitati jaanuaris 2004, kuulub nn teise generatsiooni litsentside hulka. Need loodi eraldiseisvatena seni tuntud litsentsidest ning võivad seega olla mõneti eralduvad ka GPL arusaamadest. [Woods&Guliani, 2005]

Versioon 2.0 sisaldab varasemaga (v1.1) võrreldes lisandusi, andes muuhulgas litsentsi valdajale kõik autori- ja patendiõigused.

2.1.6. Mozilla avalik litsents

Mitmed iteratsioonid läbinud Mozilla avalik litsents (*Mozilla Public License*, MPL) on üks populaarsemaid tänapäevaseid alternatiive GPL-litsentsile [Woods&Guliani, 2005]. Nagu ka GNU avalikus litsentsis, tuleb MPL litsentsiga kaetud lähtekood teha kättesaadavaks

arendajate kogukonnale, erinevuseks on, et suuremamahuline Mozilla (firma) sees toodetud kood pea olema avalikustatud. Seda kasutab Mozilla oma äritegevuses.

Peamiseks MPL litsentsi eeliseks ongi, et see kaitseb ettevõtte ning selle partnerite intellektuaalset omandit. MPL ei ole GPL'ga ühilduv, kuid Mozilla on alustanud pingutusi, et tõsta kogu Mozilla kood kolmiklitsentsi kaudu GPL ja LGPL alla, võimaldades Mozilla koodi kasutada ka eraldiseisvates GPL-projektides. [Woods&Guliani, 2005]

2.2. Kinnised litsentsid

Kinnised litsentsid sätestavad suletud lähtekoodiga tarkvara kasutamisõigused. Näitena on toodud mõned võrdluseks olulised väljavõtted ühest tuntuimast, Microsoft Windows XP Professional'i lõppkasutaja litsentsilepingust (*End User License Agreement*, EULA) [EULA, 2001]:

- **Kasutaja võib paigaldada, kasutada, kuvada ja käivitada koopia tootest ühel tööjaamal (*workstation*).** [EULA, 2001]

  Võrdluseks: Avatud lähtekoodi litsentsid ei määra, mitmel arvutil võib antud tarkvaratoodet kasutada, vaid soosivad tarkvaratoote võimalikult laialdast levikut.

- **Kasutaja ei või toodet pöördprojekteerida (*reverse engineering*), dekompileerida (*decompilation*) ega osadeks lammutada.** [EULA, 2001]

  Võrdluseks: Weber toob samasuguse analoogia Coca-Colaga [Weber, 2004]. Kuigi Coca-Cola toote-etiketilt võib lugeda joogi koostisosi, pole seda võimalik pöördprojekteerida (Weber kasutab sama terminit) algosadeks. Samuti pole viimaseid kombineerides võimalik täpselt samasugust toodet ise luua. Sarnaselt on MS Windows tarkvaratoode, mida saab kasutada, kuid mille reprodutseerimine, (paremaks) muutmine ning oma versiooni levitamine on litsentsi alusel keelatud. Avatud lähtekoodi üks printsiipe seevastu on, et lähtekood oleks kõigile vabalt kättesaadav ning sellele kohustusele tuginevad ka kõik avatud lähtekoodi litsentsid.

- **Nõusolek andmete kasutamiseks / Internetil tuginevate teenuste komponendid / Turvapaigad** [EULA, 2001]
  Need sätted (*EULA – Description of other rights and limitations*) määravad üheselt, et kasutaja lubab Microsoftil ja selle partneritel koguda ja kasutada tootega seotud tehnilist infot tugiteenuste osana.

  Võrdluseks: Richard Stallmani vaba tarkvara idee kohaselt ei tohi tarkvara kasutaja järele nuhkida. [Stallman, 2006]

Hübriidse tarkvara arendusprojekti koha pealt on kinnine litsents oluline juhul, kui ettevõte soovib müüa avatud lähtekoodiga tarkvaraga suhtlevat ning sellele lisaväärtust loovat kinnist tarkvara.

2.3. Hübriidsed litsentsid

2.3.1. Duaallitsenseerimine

Hübriidse tarkvara levitamisel on kõige sobilikum kasutada nn. duaallitsentseerimist (*dual license*). Duaalsus tähendab antud mõistes seda, et vaba tarkvara distributsioonimudel ning traditsionaalne tarkvara äriprotsess on kombineeritud. Tehniliselt tähendab see kahte litsentsi, ühte vaba levitamise ning teist tarkvara ärilistel eesmärkidel kasutamiseks.

Välimäki viis läbi juhtumiuuringud kolme duuallitsentsiga tarkvara tootvate ettevõtte kohta, uurides põhjusi, miks need topeltlitsentsi kasutusele võtsid, kuidas see töötab ning kui efektiivne see on. Juhtumiuuringutest ilmnes, et topeltlitsents on rakendatav lahendus, samas seab see tarkvarale mitmed organisatoorsed, legaalsed ja majanduslikud nõudmised ning piirangud, mis võivad praktikas limiteerida selle mudeli kasutatavust. [Välimäki, 2003]

2.3.2. Hübriidne litsentsikogum

Manussüsteemide tarkvaraga tegelev ettevõte QNX Software Systems on hübriidsele tarkvara arendusmudelile välja töötanud kolm erinevat litsentsi. Niimoodi on moodustunud hübriidne litsentsikogum, mis võimaldab ettevõttel avatud lähtekoodi kasutavaid tarkvaratooteid müüa:

- ***Non-commercial End Users License Agreement*** **(NCEULA)** [QNX[1]]**–** Litsents nn „mitteärilistele lõppkasutajatele“, kes ei kasuta toodet kasu saamiseks (kuigi võivad seda näiteks tulevikus plaanida). Jaguneb omakorda kolmeks litsentsiklassiks:
    a) **Evalvatsiooni litsents** (*Evaluation License)* – Võimaldab tarkvara kasutada 90 päeva, määramaks antud toote sobivust sihtsüsteemiga ning prototüüpide loomiseks. Tüüpiline jaosvara (*shareware*) litsents.
    b) **Mitte-ärilise arendaja litsents** (*Non-Commercial Developer License*) – Tarkvara kasutamine isiklikeks vajadusteks, kuid vaid mitteärilistel eesmärkidel. Sisaldades ka eelnevas litsentsis kirjeldatud õigusi, võimaldab see lisaks arendada uusi rakendusi, osaleda QNXi arendajate kogukonna (Foundry27) projektides ning levitada derivatiivseid arendustöid teistega.
    c) **Akadeemiline litsents** (*Academic Faculity License*) – Tarkvara kasutamine õppe-eesmärkidel klassiruumides ja laborites. Lisanduvad kõik mitte-ärilise arendaja litsentsis kirjeldatud õigused. Kui eelnevad litsentsid määrasid kasutusõiguse üks litsents tööjaama kohta, siis akadeemiline litsents võimaldab ühte litsentsi kasutada terves institutsioonis.

- ***Partner Software License Agreement*** **(PSLA)** [QNX[2]] **–** Litsents nn „kogukonna partneritele” – kasutajatele, kes soovivad võimalust pakkuda oma lisaväärtust loovaid tooteid ja teenuseid QNX klientidele. PSLA on tasuta litsents, mis on loodud üldkasutatava tehnoloogia ja teenuste kahepoolseks (QNX ja selle klientide vahel) arenduseks. Jaguneb kaheks:
    a) **Partneri litsents** (*Partner License*) **–** loovad üldkasutatavaid (mitte lõppkasutajale määratud) tooteid, mis töötavad koos QNXi tuumikrakendusega. Partnerlitsentsiga on võimalik kasutada QNX tarkvara partnertoote arenduseks ning demonstratsiooniks. Partnertoote eest ei tohi küsida tasu ning see peab olema avalikustatud QNX kogukonnale.
    b) **Iseseisva konsultandi litsents** (*Independent Consultant License*) – saavad kasutada tarkvaratoodet eesmärgiga assisteerida teisi QNX kogukonna liikmeid (partnereid, kliente jne). Litsentsiomanik ei või kasutada tarkvara oma rakenduse loomiseks, haldamiseks või toetamiseks.

Veel kohtab litsentsilepingus aga Microsofti EULAst tuttavat klauslit. Nimelt ei tohi partneri või iseseisva konsultandi litsentsi omanik ise ega lasta teistel tarkvara dekompileerida, dekrüptida, tõlkida või muul moel pöördprojekteerida, välja arvatud juhul kui antud osa tarkvarast on avatud lähtekoodi litsentsi all. Litsents täpsustab, et kui pole eraldi märgitud, ei ole lähtekood avatud. Seega on loodud lähtekood „avatud“ vaid kogukonna, mille moodustavad litsentseeritud kasutajad, sees.

- ***Commercial Software License Agreement*** **(CSLA)** [QNX[3]] – Litsents, mis lubab QNX tarkvara kasutada ärilistel eesmärkidel, sisaldab äriklientidele vajalikke garantiisid ning tugiteenuseid. Võimaldab ligipääsu QNX poolt patenteeritud toodetele. Ärilitsentsi omanikul on ka kõik eelnevate litsentsitingimuste õigused.

QNX hübriidne litsentsikogum on hea näide sellest, kuidas ettevõte saab erinevaid litsentsitingimusi rakendades korraga kasutada avatud lähtekoodi poolt pakutavaid eelised kui ka samas kaitsta oma intellektuaalset omandit.

3 Hübriidne tarkvara arendusmudel

Hübriidne tarkvara arendusmudel võimaldab ettevõttel luua avatud lähtekoodil baseeruvat ning avatud lähtekoodi arendajate poolt loodavat tarkvara, pakkudes sellele omalt poolt lisateenuseid kasutajatoe, kommertsteenuste või mõnel muul moel.

Vaatleme suletud ja avatud lähtekoodiga tarkvara arendusmudeleid ning nende põhimõtteid.

3.1. Ülevaade kinnise tarkvara arendusmudelist

Kommertstarkvara valmib sootuks teistel alustel kui avatud lähtekoodiga tarkvara. Selle loomeprotsess toimib põhimõttel, et võimalikult paljud inimesed oleksid nõus loodava tarkvaratoote eest maksma. Tarkvarakompaniid peavad selgeks tegema, mida eeldatavad kliendid soovivad. Sellega kaasneb palju riske, sest nõuete koostamine tähendab palju eelduste tegemist tulevaste klientide suhtes. [Woods&Guliani, 2005]

Tüüpilised tarkvaraarenduse protsessi etapid:

- Projekti planeerimine, eesmärkide seadmine
- Süsteemianalüüs, nõuete kirjeldamine
- Disain
- Arendus
- Integratsioon ja testimine
- Juurutamine
- Hooldus

Vanim ja tuntuim tarkvaraarenduse mudel on kaskaad- e koskmudel. [Kay, 2002] Mudeli eesmärgiks on lõpetada iga tegevus enne, kui liigutakse edasi järgmisesse etappi. Projekti meeskond lõpetab nõutud tegevused loogilises järjekorras, aga puuduseks on see, et projekti liikmed ei saa oma tööd parandada, sest „kosest ei saa üles ronida“. [Kaljula, 2004]

Koskmudeli tänapäevasemaks lahenduseks on iteratiivne arendusmudel, mis on agiilsete arendusraamistike, nagu Rational'i unifitseeritud protsess (Rational Unified Process, RUP) ja ekstreemprogrammeerimine (*extreme programming*, XP), põhialuseks. Iteraatiivsete

arenduste puhul ei pea tegema pikaajalisi tarkvara nõuete ennustusi, selle asemel muudetakse vajadustel lühiajaliste prognooside raames vastu võetud otsuseid. [Woods&Guliani, 2005]

3.2. Ülevaade avatud tarkvara arendusmudelist

Avatud lähtekoodiga tarkvara ei sünni ärilisi aspekte silmas pidades. Rakenduse „klientideks" on selle arendajad ise, seetõttu on vajaduste väljaselgitamine lihtne.

Avatud tarkvara arendusprotsess on iteratiivne ja agiilne protsess.

Mocus jt poolt koostatud juhtumiuuring [Mockus et al, 2005] vaatleb lähemalt Mozilla ja Apache arendusprotsessi. Järgnevalt on esitatatud refereering antud juhtumiuuringust.

Mõlema projekti arendusprotsess koosneb viiest etapist:

- **Tehtava töö väljaselgitamine**
  Apache: Taotlused muudatusteks raporteeritakse peamiselt läbi arendajate listi. Kuna edastajaks on reeglina Apache arendajate kogukonna liige, sisaldab raport piisavalt informatsiooni probleemi analüüsimiseks või koguni paika (*patch*) probleemi lahendamiseks.

  Mozilla: Mozilla haldab tulevaste reliiside jaoks ajatelge (*roadmap*), kus on kirjas ka väljalasete graafik. Ajagraafiku määramisel sätestatakse mõistlik aeg, mille jooksul arendajate kommuun on võimeline selles osalema. Vigadest ning täiustusest saavad raporteerida kõik.

- **Arendustöö määramine ja teostamine**
  Apache: Kui vea olemus või täiustus on Apache tuumikarendajate poolt heaks kiidetud, otsitakse vabatahtlikke, kes probleemi kallal tööle asuvad. Tuumikarendajad kalduvad töötama kindlaksmääratud aladel. Tihti ei ole probleemiks mitte olukorra lahendamine vaid otsustamine, millisel viisil on seda kõige otstarbekam teha. Kui ka probleemist raporteerija on oma lahenduse pakkunud, ei pruugi see sobida üldise lahendusena (nt mitte sobida kõikidele platvormidele).

Mozilla: Arendajate kogukond saab sirvida veahaldussüsteemis BugZilla, leidmaks vigu või täiendusi, mille kallal nad soovivad töötada. Erinevad arendajad on enamasti endale valinud kindlaksmääratud alad, kus nad töid teostavad.

- **Testimine**

  Apache: Arendaja testib uut lahendust oma serveril.

  Mozilla: Mozilla teostab igapäevase järgu (*build*) testimise erinevatel platvormidel, kindlustamaks selle stabiilsust, et oleks võimalik arendustööd jätkata. Kui see peaks ebaõnnestuma, ei teostata edasisi arendustöid enne, kui vead on parandatud.

- **Inspekteerimine**

  Apache: Peale testimist lisab tuumikarendaja muutused otse koodi või postitab paiga (*patch*) arendajate listi ülevaatamiseks.

  Mozilla: Koodi inspekteeritakse moodulipõhisest ning kogu koodibaasi vaatenurgast.

- **Väljalaskehaldus**

  Apache: Kui projekt hakkab lähenema uuele väljalaskele, hakkab üks Apache tuumikarendajatest vabatahtlikuks väljalaskejuhiks. Tema ülesandeks on identifitseerida kriitilised probleemid, mis pärsivad väljalaset, selgitama välja, millal on need probleemid lahendatud ning tarkvara on jõudnud stabiilsesse punkti.

  Mozilla: Mozilla kasutab katkematut järguprotsessi (Tinderbox), mis näitab, millistes järkudes ja platvormidel on vead.

Oma olemuselt sarnaneb Mozilla arendusmudel rohkem hübriidse arendusmudeliga, kus ettevõttele kirjutatakse avatud lähtekoodi. Olgu ka mainitud, et Mozilla on Apache kõrval palju suurem arendusprojekt.

3.3. Hübriidne tarkvara arendusmudel

Hübriidne tarkvara arendusmudel kombineerib mõlema arendusmudeli töövõtted. Kuigi ohjad projektijuhtimise osas on justkui ettevõtte käes, tuleb avatud lähtekoodiga arendusprojektis arvestada ka kogukonna arvamusega. Liigne projekti

Käesoleva peatüki edasises osas vaatleme hübriidse tarkvara arendusmudeli elemente.

Tarkvara arendusmudel on tihedalt seotud ka selle ärimudeliga – viimane mõjutab ka tarkvara arendusprotsessi. Kui ettevõtte eesmärgiks on pakkuda avatud lähtekoodiga tarkvaratootele lisateenuseid (avatud lähtekoodi ärimudel), siis suunab ettevõtte oma arendusjõud koos kogukonna arendajatega avatud lähtekoodi loomiseks. Kui aga ettevõtte eesmärgiks on teenida tasu läbi litsentseerimise (täpsemalt kasutades hübriidset litsentsikogumit, vt ptk 2), panustab ettevõtte suletud lähtekoodi kirjutamisse avatud lähtekoodi ümber. Muidugi peab ettevõte selleks kirjutama mõningal määral ka avatud lähtekoodi, et säilitada arendajate kogukonna huvi antud tarkvaratoote vastu.

See, kas ettevõtte põhirõhk on avatud või suletud lähtekoodi kirjutamisel, avaldab mõju ka arendusmudelile. Esimesel juhul sarnaneb see rohkem avatud lähtekoodi arendusmudelile (koos mõningate „tavalise“ tarkvara arendusmudeli elementidega), teisel juhul on ettevõtte arendusmudel konservatiivsem. Siiski peab olema rõhk ka avatud lähtekoodi arendusmudelil, kuna see on see, mille abil ettevõte mõlemal juhul kasumit teenib.

3.4. Hübriidse tarkvaraprojekti algatamine

Üheks põhjuseks, miks avatud lähtekoodiga tarkvaraprojektid läbi kukuvad, on valearvamus, et samad juhtimispraktikad, mis töötavad majasiseste projektide puhul, töötavad ka avatud lähtekoodiga tarkvaraprojektis. [Fogel, 2005]

3.4.1. Osalemine olemasolevas projektis

Enne avatud lähtekoodi tarkvaraprojekti algatamist tuleks uurida, kas soovitud funktsionaalsusega tarkvaraprojekti juba ei eksisteeri. [Fogel, 2005] Seda välja arvatud juhul, kui ettevõtte ei soovi avalikustada mõne oma olemasoleva rakenduse lähtekood. Kui sobiv

projekt juba eksisteerib, saaks ettevõtte suunata oma ressursid juba olemasolevasse projekti, selle asemel, et „leiutada jalgratast“.

Olemasoleva projekti eelised:

- Projektil on juba olemas oma arendajate kogukond.
- Puuduva lisafunktsionaalsuse saab ettevõte ise juurde arendada või esitada selleks soovi.
- Ressursse ei pea kaasama nii palju kui uue projekti puhul.
- Kui ettevõttel pole (veel) vilumust avatud lähtekoodiga tarkvaraprojektide juhtimisel, kaasneb juba käimasoleva projektiga liitumisega vähem riske.

Enne olemasoleva avatud lähtekoodiga projektiga liitumiseks tuleb välja selgitada, kas selle litsentsitingimused sobivad ettevõtte eesmärkidega. [Goldman&Gabriel, 2006] Näiteks ei saa GPL litsentsi all avalikustatud lähtekoodi kasutada suletud projektis.

Avatud lähtekoodiga ja vaba tarkvara otsimiseks sobivad:

- SourceForge.net (http://www.sourceforge.net)
- Freshmeat (http://freshmeat.net)
- Free Software Foundationi (FSF) kataloog (http://directory.fsf.org)

3.4.2. Planeerimine

Kui ükski juba eksisteeriv avatud lähtekoodi tarkvara arendusprojekt ei rahulda firma vajadusi või soovib ettevõte avada enda olemasolevat lähtekoodi, tuleb alustada uut avatud lähtekoodi projekti.

Avatud lähtekoodi algatusfaasis on üks keerulisemaid ülesandeid muuta ettevõtte visioon avalikuks visiooniks. [Fogel, 2005] Ühtlasi tuleb määratleda, mida projektiga soovitakse saavutada ning mida mitte.

Fogel kirjeldab, millised nõuded on vajalikud avatud lähtekoodiga tarkvaraprojekti algatamiseks. Samas tõdeb ta, et kõik allnimetatud nõuded ei pea siiski eelnevalt täielikult valmis olema, vaid nende koostamisel saab abiks olla ka arendajate kogukond.

Vajalikud on:

- Konkreetne missioonikirjeldus
- Tarkvara vajaduste ja omaduste nimekiri
- Arendusstaatuse äramärkimine
- Versioonikontrolli- ja veahaldussüsteemid
- Suhtluskanalid (list, foorum, viki)
- Juhtnöörid arendajatele (pigem sotsiaalse kui tehnilise plaani selgitus)
- Dokumentatsioon
- Projekti koduleht

Mõistagi tuleb ettevõttel selles faasis determineerida ka oma ärihuvid, millisel viisil avatud lähtekoodiga tarkvaraprojekt kasulikuks osutub. Erinevaid avatud lähtekoodi ärimudeleid on vaadeldud järgnevad peatükis.

3.5. Protsess

3.5.1. Otsuste langetamine ja teostus

Otsuste langetamine võtab avatud lähtekoodi projektide puhul rohkem aega, kuid seetõttu võib otsus olla ka kvaliteetsem ning argumenteeritum. Oluline on, et otsuste langetamise protsess oleks endiselt läbipaistev, st et ettevõte ei võtaks otsustamist ainult oma õlule, vaid arvestaks ka kogukonna seisukohtadega. Arendajatega on võimalik saavutada kokkulepe, et ettevõtte poolt hallatav vajaduste ja omaduste nimekiri on primaarne määraja, millal suurem reliis valminud on. Sellisel juhul saab teisi omadusi endiselt lisada. [Goldman&Gabriel, 2005]

Sisemiste arendajate üle saab täielikult otsustada ettevõte. Siiski on halb kui sisemised arendajad töötavad ei tööta vabatahtliku kommuuni huviorbiidis, oluline on, et eesmärke saavutatakse üheskoos. Vastasel juhul võib aset leida kahveldumine (*forking*) – arendajate kogukond jätkab edasist arendust iseseisvalt.

3.5.2. Koodi ülevaatamine

Ettevõte peab otsustama, kellele ta usaldab valminud uue koodi ülevaatamise ning selle lisamise koodibaasi. Mõnede avatud lähtekoodi projektide puhul puudub kontroll sootuks, strateegiliste ning ärihuvidega projekti puhul peab see kindlasti eksisteerima.

Apache veebiserveri koodimuudatuse üle otsustab Apache Group - nn tuumikarendajadest koosnev kogu. [Mocus et al, 2005] Ettevõte ei tohiks liiga karmi kätt mängida uue koodi aktsepteerimisel, kuna see seab ohtu innovaatilisuse, samas peaks ta siiski olema suunavas rollis. Suured projektid, nagu Mozilla, kasutavad formaalselt koodi ülevaatlusprotsessi, mille iga uus koodijupp peab läbima [Goldman&Gabriel, 2005].

Avatud lähtekoodi projektides vaadatakse üldjuhul kood üle terve kogukonna poolt.

3.5.3. Järgud

On väga oluline, et viimane valminud koodijärk (*build*) oleks arendajatele kohe allalaadimiseks saadaval. See võimaldab arendajatel katsetada nende koodi uusimal koodibaasil. Kui seda võimalust pole realiseeritud, on arendajad palju vähem motiveeritud projektis osalema. [Goldman&Gabriel, 2005]

3.5.4. Testimine

Avatud lähtekoodi rakenduse testimises osalevad rangelt väljenduses kõik selle kasutajad. [Goldman&Gabriel, 2005] Mida lihtsamaks on tehtud vigadest teatamise võimalus (nt veahaldussüsteemi abil), seda rohkem kasutajaid seda võimalust kasutab.

Peamine erinevus kommertstarkvara ning avatud lähtekoodiga tarkvara testimise vahel on, et siin võivad veast raporteerivad kasutajad piisava kompetentsi korral ka sellele lahendust pakkuda. See selgus ka Mockuse jt poolt läbi viidud Apache juhtumiuuringust [Mockus et al, 2005].

Kasutajatepoolne testimine võimaldab ka välja selgitada, kas pakutav funktsionaalsus vastab kasutaja tegelikele vajadustele. [Goldman&Gabriel, 2005]

3.5.5. Väljalasked

Avatud lähtekood toimib põhimõttel, et uus lähtekood peaks olema võimalikult varakult ning tihti avalikustatud („*Release early, release often*“). Igakord, kui keegi vormistab muudatuse

lähtekoodis, on tegu selle uue väljalaskega. Viimane on eriti soodne aktiivsetele arendajatele, sest niimoodi töötavad nad alati kõige uuema lähtekoodiga ning ei tööta probleemi kallal, mille keegi on juba lahendanud [Goldman&Gabriel, 2005].

Kasutajatele, kes ei soovi nii sagedasi väljalaskeid koos uusima funktsionaalsuse, vaid eelistavad sellele töökindlust, tuleks väljastada suuremaid väljalaskeid (*major release*) pikema ajavahemiku tagant.

Väljalasete haldamiseks on tihtilugu kasulik määrata eraldi väljalaskehaldur, kes aitab otsustada, mida uude väljalaskesse panna ning mis pole selleks veel valmis. [Goldman&Gabriel, 2005]

3.6. Avatud lähtekoodi töövahendid

Enne avatud lähtekoodi projekti väljakuulutamist peaks ettevõttel olema üles seatud selle töövahendid, mis võimaldavad (esialgu potentsiaalsetel) arendajatel saada informatsiooni ning omavahel suhelda.

Hea infohaldus hoiab ära avatud lähtekoodi projekti varisemise Brooks'i seaduse [Brooks, 1982] all, mis ütleb, et tööjõu lisamine hilinenud tarkvaraprojektile muudab selle veelgi hilisemaks. [Fogel, 2005]

3.6.1. Versioonihaldussüsteem

Versioonihaldussüsteem (*version control system*) on kombinatsioon tehnoloogiatest ja praktikatest, mis võimaldab jälgida ja kontrollida muutusid projekti failides, peaasjalikult lähtekoodis ja dokumentatsioonis, mõningal juhul ka veebilehtedes. [Fogel, 2005]

Nagu mainitud, saab läbi versioonihaldussüsteemi saab avalikustada ka projekti puudutavat dokumentatsiooni, sellegi poolest on tänapäeval selleks olemas mugavamad vahendid, nagu sisuhaldussüsteemid (*content management system*, CMS) ja vikid. Viimastest tuleb juttu edaspidi.

Kõige enamkasutatavaks versioonihaldussüsteemiks on Concurrent Versions System (CVS). [Robbins, 2003] Viimasel ajal on kasutust leidnud ka selle parandatud variant, Subversion.

3.6.2. Veahaldus ja tehniline tugi

Veahaldussüsteeme (*bug tracking system, issue tracking system, request tracker*) ei kasutata vigadest teada andmiseks, vaid ka intsidentidest ning arendussoovidest teavitamiseks. Üldjoontes aitab veahaldussüsteemi kasutamine parandada suhtlemist, tõsta tarkvaratoote kvaliteeti ning kliendi rahulolu.

Üks enimkasutavaid veahaldussüsteeme avatud lähtekoodi projektides on Mozilla projekti jaoks loodud Bugzilla (http://www.bugzilla.org). [Robbins, 2005]

3.6.3. Suhtluskanalid

Järgnevalt on esitatud erinevad suhtluskanalid, mida ettevõtte saab kasutada avatud lähtekoodi projekti kohta informatsiooni levitamiseks. Kahtlematea ei ole nende kõigi kasutamine projekti juures kohustuslik, vaid see sõltub konkreetsetest vajadustest. Erinevaid funktsionaalsusi saab siduda ka ühte suhtluskanalisse (projekti koduleht vikina, foorumil baseeruv veahaldussüsteem), kuid sel juhul tuleb jälgida, et ligipääs nende ja vajadusel kaasa löömine oleks võimalikult mugav ning pakuks vajalikke võimalusi.

- **Listid**

  Listid on vanim ja levinuim avatud lähtekoodi arendajate suhtlemise moodus, mida kasutatakse siiani väga paljudes avatud lähtekoodi projektides. Tänapäeval on tõenäoliselt otstarbekamaks lahenduseks foorumid.

  Ühe tarkvaraprojekti juurde kuuluvad listid võiksid olla:

  - Teadaanded väljalasete ja turvaintsidentide kohta
  - Üldine diskussioon
  - Sügavam tehniline diskussioon

  Mõistagi sõltub listide teemade valik ning arv konkreetse projekti vajadustest.

- **Projekti veebileht**

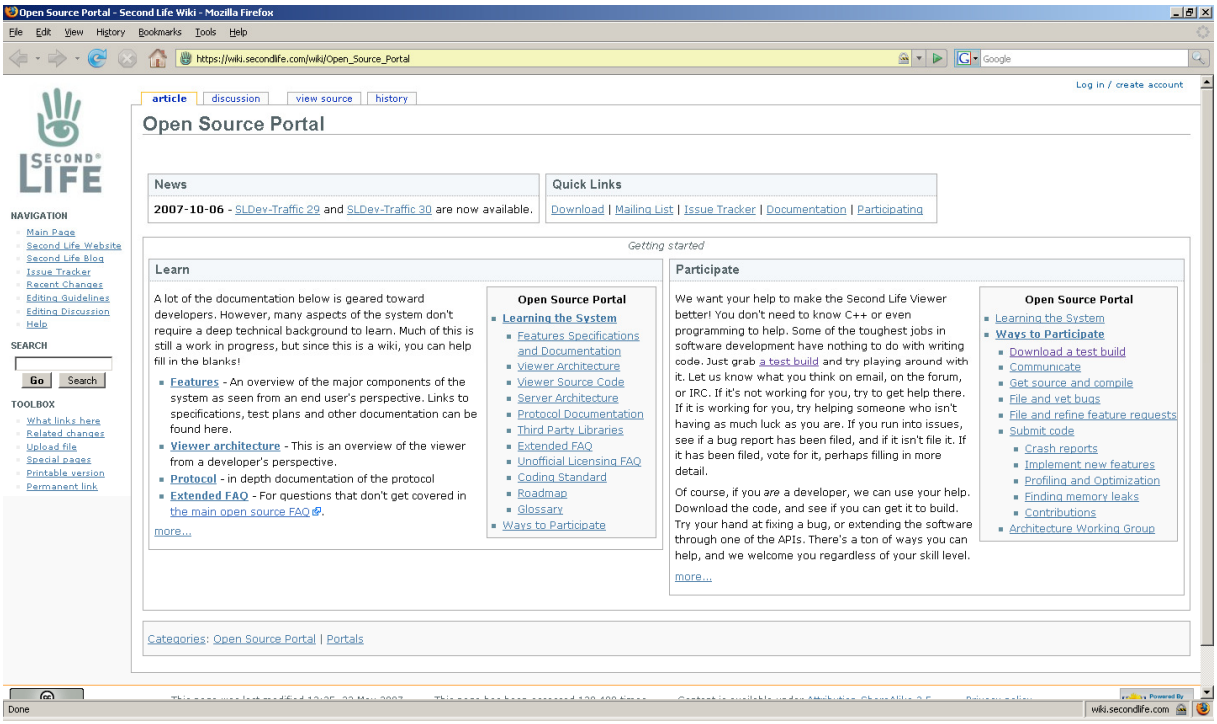
  Avatud lähtekoodi projekti veebileht on peamiseks allikaks, kust asjast huvitatud ammutavad infot projekti kohta. Selleks peavad veebilehel olema viited teistele projekti töövahenditele (dokumentatsioonile, allalaadimisetele, listidele, vikidele, versiooni- ja veahaldussüsteemile jne).

  Veebilehe loomiseks võib kasutada mõnd sisuhaldussüsteemi (*Content Management System*, CMS) või seada koduleht üles vikina.

- **Viki**

  Viki (*wiki*) on veebilehestik, mis võimaldab külastajatel selle sisu muuta ja laiendada. Hetkel ei ole vikid veel kujunenud avatud lähtekoodi projekti suhtluskanalina standardiks, kuid arvatavasti saavad selleks peagi. [Fogel, 2005]

  Virtuaalmaailma Second Life haldav ja arendav Linden Labs kasutab avatud lähtekoodi projektis vikit (Joonis 1). Portaalis on toodud viimased uudised, kiirlingid erinevate avatud lähtekoodi projekti töövahendite juurde ning info õppimis- ja kaasalöömisvõimaluste kohta.



Joonis 1 – Virtuaalmaailma Second Life haldav Linden Labs kasutab avatud lähtekoodi projektis vikit.

- **Foorum**

Foorum sobib tänapäevaseks alternatiiviks listidele. Kuigi foorumid on peamiselt mõeldud diskussioonide läbiviimiseks, saab seda vajadusel kasutada ka kogu projekti kohta vajamineva info esitamiseks.

Heaks näiteks töötavast avatud lähtekoodi projekti foorumist on transpordisimulaatori SimuTrans'i foorum aadressil http://forum.simutrans.com. Foorum sisaldab kogu vajalikku informatsiooni projekti kohta.

3.6.4. Multifunktsionaalsed arenduskeskkonnad

Lisaks erinevaid funktsionaalsusi pakkuvatele rakendustele on olemas ka koostööl põhinevaid arenduskeskkondi (*Collaborative Development Environement*, CDE), mis pakuvad kõiki avatud lähtekoodi projekti läbiviimiseks vajalikke lahendusi ühes kohas.

Üheks selliseks keskkonnaks on SourceForge (http://sourceforge.net), mis võimaldab ligipääsu standardsele töökeskkonnale, mis koosneb veebiserverist projekti sisu jaoks, listidest, intsidendihaldurist ja versioonikontrollsüsteemist.

Koostööl põhinevad arenduskeskkonnad on avatud lähtekoodi projektides laialt kasutust leidnud, pakkudes vajaminevat infrastruktuuri veebipõhiselt. Sellised keskkonnad aitavad vältida debatte töövahendite valiku üle ning keskenduda pigem väärtuse loomisele. [Robbins, 2005]

4 Avatud lähtekoodi ärimudelid

Avatud lähtekoodiga tarkvaratoode ning selle lähtekood on kättesaadav igale sellest huvitatud kasutajale. Vahel osutuvad avatud lähtekoodiga tarkvaratooted on nii nõutuks, et äriklientidel on suur vajadus erinevate tugiteenuste järele. [Krishnamurthy] Need hõlmavad endas installatsiooni, välja õpetamise/sertifitseerimise ning tehnilise toe teenuse.

Arendajate huvides on, et nende poolt teostatav projekt koguks võimalikult palju populaarsust, et seeläbi kasvatada arendajate kogukonda ning muuta projekt mastaapsemaks ning luua uusi kasutusvõimalusi. Seega on nad avatud ettevõtetele, kes võiksid antud tarkvaratoote kasutajaskonda kasvatada ning toetada projekti omapoolsete arendajatega.

4.1. Avatud lähtekoodi ärimudel

Kuna avatud lähtekoodi kontseptsioon ei hõlma endas mingisugust ärimudelit, võib kõiki avatud lähtekoodil baseeruvad ärimudeleid nimetada ühel või teisel viisil hübriidseteks. [Mickos] Avatud lähtekoodil baseeruv ärimudel annab avatud lähtekoodile lisaväärtuse, mis muudab selle atraktiivsemaks.

MySQL AB tegevjuht, Mårten Mickos, identifitseeris 2007. aasta Open Source Business Conference'l San Franciscos kolmteist avatud lähtekoodi ärimudelit. [Mickos] Nende seas on ka mudelid, mis kindlasti ei sobi või pole ettevõttele piisavalt tulutoovad loomaks hübriidse ärimudeliga tarkvaraprojekti. Olgu siin toodud mõned Mickose poolt esitatud ärimudelid, mis pälvivad antud magistritöö kontekstis lähemat vaatlust:

1. Tarkvara on vaba, kuid vajame annetusi ja toetusi (Apache Software Foundation, Eclipse, ObjectWeb)
2. Tarkvara on vaba, kuid müüme reklaami ja paigutust (Mozilla)
3. Tarkvara on vaba, kuid selle sidumisel suletud lähtekoodi küsime tasu (TrollTech, DB4Objects, Funambol, MySQL)
4. Tarkvara on vaba, kuid teenused mitte (Covalent)
5. Tarkvara on vaba, kuid hooldus, monitooring ja provisioneerimine mitte (RedHat)
6. Tarkvara on vaba, kuid mõned kommertsfunktsioonid mitte (SugarCRM, Zimbra, JasperSoft)

7. Tarkvara on vaba, kuid me ehitasime selle ümber suletud lähtekoodiga toote (EnterpriseDB, GreenPlum)
8. Tarkvara on vaba, raudvara mitte (Sun)

Kui Krishnamurthy avaldas oma ideed, tuginedes olemasolevale avatud lähtekoodiga tarkvara lisabõimaluse loomisele, siis Mickose ideid saab rakendada ka juhul, kui ettevõttel on soov seni suletud lähtekoodiga tarkvaratoote kood avalikustada.

Järgnevalt vaatleme mõningaid toimivaid avatud lähtekoodil baseeruvate ärimudelite näiteid laiast maailmast.

4.1.1. Red Hat

Red Hat Software loodi 1995. aastal ideega pakkuda süsteemiadministraatoritele võimalust Linuxit paremini paigaldada, kasutada ja hallata. Linuxi distributsioon, pakituna koos mõningate lisarakenduste ning hea dokumentatsiooniga, pakkudes lisaks kasutajatuge, paisati müügile 50 USA dollariga. Tänu heale ajastusele (Linuxi kasutatavus oli just plahvatuslikult tõusmas) ning paketi kasutusmugavusele sai ettevõttest kiiresti juhtiv Linuxil baseeruva operatsioonisüsteemi turustaja. [Weber, 2004]

Tarkvara, mida Red Hat turustab, on kirjutatud GPL litsentsi all, seega ei ole Red Hat traditsionaalsel viisil selle omanik. Selle heaks illustratsiooniks on kloondistributsioonide (CentOS, WhiteBox Linux, Scientific Linux jt) olemasolu.

Linuxi lähtekood on endiselt avalik ning tasuta, Red Hat'i poolt kirjutatud lähtekood kuulub kõigi reeglite kohaselt samuti GPL litsentsi alla ning on vabalt kättesaadav kõigile. Red Hat saab kasumit, müües lähtekoodile juurde eelnevalt mainitud lisaväärtusi, mis on kommertskasutajale olulised.

Red Hat rahastab samas ise avatud lähtekoodiga tarkvara loomist, makstes palka mitmetele Linuxi arendajatele. [Weber, 2004]

4.1.2. Apache ja IBM

Apache sai alguse 1995. aastal kaheksa inimese (Apache Group) projektist, kes „paikasid“ tollast NCS HTTP veebiserveri rakendust. 1999. aastaks oli tollane lähtekood juba täielikult üle kirjutatud ning rakenduse nimeks sai Apache. Palju vajalikke funktsioone, väga hea töökindlus ja nullhind on teinud Apache'st maailma populaarseima veebiserveri.

Praeguseks on Apache Group kasvanud 25 liikmeni ning selle arendajate kogukonda hinnatakse peaaegu 400-liikmeliseks [Mocus et al, 2005]

Samal ajal kui Apache oli populaarsust kogumas, püüdis IBM müüa oma kommerts veebiserverit GO, Apache pakkus aga paremat tehnoloogiat ning oli sealjuures tasuta. IBM

IBM oleks võinud võtta Apache lähtekoodi, seda modifitseerida ning välja anda enda kaubamärgi all (kuniks oleks sätestatud, et see tuleneb Apachelt). See oleks aga IBM-i eemale jätnud Apache arendajate kogukonnast ning jätnud selle toetuma nn „majasisestele“ ressurssidele, mis oleks ka olnud halb tehing.

Apache'i poolne nõue oli, et IBMi parimad insenerid osaleks sarnaselt teiste programmeerijatega arendusprotsessis ning teeks kaastööd avatud lähtekoodi näol. Apache arendajad polnud huvitatud rahalisest hüvitisest, vaid soovisid pigem kaastööd. [Friedman, 2006 Samal ajal pakub IBM tasulist ärikasutaja tuge, mis omakorda aitaks saavutada senisest veelgi suuremat turuosa. [Weber, 2004]

1998. aasta juunis teatas IBM Apache koodi ühendamisest oma veebiserveri WebSphere alla, lisades selle patenti ka Apache autoriõiguse [Friedman, 2006].

4.1.3. MySQL

1995. aastal alustas MySQL AB relatsioonilise andmebaasihaldussüsteemi arendamist, mis oli algselt suunatud veebiserveritele, kuid on nüüdseks saanud üldkasutatavaks andmebaasisüsteemiks.

Alguses levitati MySQL andmebaasi oma litsentsitingimustel. See võimaldas tasuta, kuid limiteeritud toote kasutamist ja jaotamist Unixil baseeruvatel süsteemidel. Windows

süsteemidel oli kasutusel jaosvara litsents, mis ei lubanud toote tasuta kasutamist. [Välimäki, 2005]

Alates 2000. aastast kasutab MySQL klassikalist kaksiklitsenseerimise (*dual licensing*) poliitikat. Avatud lähtekoodi projekti raames MySQL andmebaasi kasutajatele kehtib kas GPL või mõni muu Open Source Initiative (OSI) litsents, olenevalt millise litsentsi all on projekti lähtekood avalikustatud. GPL-litsentsiga mitteühilduvate avatud lähtekoodiga litsentsi all on MySQL kasutamine samuti võimaldatud. [MySQL, 2006]

4.2. Hübriidne ärimudel

Hübriidne ärimudel kasutab 2. peatükis äramärgitud hübriidseid litsentsikogumeid ning nende eesmärgiks on pakkuda avatud lähtekoodil baseeruvale tarkvaratootele suletud lähtekoodiga lisafunktsionaalsust.

Üks näiteid hübriidse ärimudeli rakendamisest on QNX Software Systems (QNX). Ettevõte alustas tegevust 1980. aastal. Esimeseks tooteks oli kernel Intel 8088 protsessorile. Hiljem keskendus ettevõte POSIX-laadsete operatsioonisüsteemide tootmine. 2007. aasta septembris teatas ettevõtte oma lähtekoodi avalikustamisest.

QNX töötas välja ning rakendas arendusmudeli, mis lahendaks manussüsteemide (*embedded system*) tarkvara traditsionaalse arenduse ja levitamisega seonduvad fundamentaalsed probleemid. [Rosen, 2007]

Selline samm võeti ette kolmel peamisel põhjusel:

- Probleemid manussüsteemide patenttarkvara arenduses, kus muutused toimuvad nii ruttu, et nendega on nii kliendil kui tarkvara tootjal raske sammu pidada.
- Kasutajad ning kliendid sooviksid teha ise modifikatsioone tarkvaras, kui see oleks võimalik.
- Paljud neist tervitaksid ka võimalust koostööd teha ning jagada kollektiivse arenduse tulemusi.

QNX leidis, et avatud lähtekoodi lähenemine, mis tähendanuks täielikku loobumist intellektuaalse omandi kontrolli üle ning viimase tasuta jagamist, ei teeni klientide hüve.

Avatud lähtekoodi kasutaks konkurendid ja ettevõttel ei ole mõtet sellesse investeerida. Liiga tihke litsentsitingimused hirmutaks jällegi eemale kliendid.

Intellektuaalse omandi äristrateegiate määratlemine on üsnagi keeruline protsess. Kui ettevõtte piirab litsentsitingimustega liialt toote kasutamist või on kasutustingimused liialt keerulised, halvab see klientide soovi ja võimaluse toodet kasutada. Kui ettevõtte autoriõigused või patendid ei ole piiratud, kaob ettevõttel mõte antud toote arendamisse investeerida.

QNX hübriidse tarkvara arenduse mudeli omadused:

- Põhikomponendid on avatud lähtekoodiga Apache 2.0 litsentsi all.
- Võtmekomponendid on litsentseeritud (loe: kaitstud).
- Kõik saavad kaasa lüüa ja saavad QNX litsentsi, kuid nad ei tohi kasutada seda kolmandate isikute hüvanguks, kellel litsents puudub.

Seega, QNX poolt hallatav tarkvara ei ole avatud lähtekoodis (st vabalt kättesaadav ja modifitseeritav kõigi poolt), kuid on selle reaalne modifikatsioon, mis on valminud avatud lähtekoodi arendajate kogukonna abil.

5 Avatud lähtekoodi küpsusmudel

Käesolevas magistritöös on toodud Woods ja Guliani poolt [Woods&Guilani, 2005] esitatud avatud lähtekoodi küpsusmudel. Sarnane mudeli on Akadeemilise Vaba Litsentsi (Academic Free Licence 2.1) all publitseeritud Navica konsultatsioonifirma [Golden, 2005].

Enne kui ettevõte saab rakendada loodud avatud lähtekoodi, tuleb hinnata selle küpsust. Hindamine annab vastuse küsimusele, kui palju nõuab valitav rakendus edasiarendust, enne kui seda saab ettevõttes kasutusele võtta. Kui mõned küpsuse võtme-elemendid on puudu – informatsiooni on raske leida, kood on halvasti struktureeritud, või dokumentatsioon puudub – tähendab see rohkem tööd, enne kui rakendust saab kasutada.

Küpsed avatud lähtekoodi projektid pakuvad kõike, mida patenteeritud tootedki, lisaks tulu jõudsalt kasvavast kogukonnast, võrgus asuvatest ressurssidest ning täielikult arenenud ökosüsteemist. Kõige vähema küpsusega projektid on võivad olla vaid veidi idee ning veidi koodi, mis ideed realiseerib. Sellise lähtekoodi kasutuselevõtt tähendab sisuliselt ühinemist projekti arendustiimiga.

Avatud lähtekoodiga tarkvaratoote küpsuse hindamata jätmine võib hiljem kaasa tuua järgnevad probleemid:

- Juurutamine nõuab rohkem tööd kui planeeritud.
- Toode ei tööta eksisteerivatel (planeeritud) süsteemidel
- Toodet on keerulisem laiendada kui esialgu arvatud
- Arendustiimilt abi saamine on raskendatud

Ilma formaalse metodoloogiata, mis implementeerib standardiseeritud analüütilist raamistikku, ei saa hinnata toote küpsust hinnata ega ka tuvastada, millised tooteelemendid vajavad täiustamist. [Golden, 2005]

Goldeni ning Woods ja Guliani poolt esitatud küpsusmudelid erinevad hinnatavate kriteeriumite kui ka ülesehituse poolest.

Goldeni poolt esitatud mudelis soovitatakse avatud lähtekoodil põhineva rakenduse küpsust hinnata järgnevate sammudena:

- Defineeri nõudmised
- Lokaliseeri ressursid
- Hinda küpsust
- Määra skoor

Avatud lähtekoodi küpsusmudel sobib ka ettevõtte enda poolt arendatava hübriidse tarkvaraprojekti küpsuse hindamiseks. See on oluline mõistmaks, kui raske on ettevõttel valminud rakendust juurutada kui ka seda, kas kui valmis on toode selleks, et pakkuda seda teistele.

5.1. Küpsuskriteeriumid

Järgnevad küpsuskriteeriumid on refereeritud Woods ja Guliani [Woods&Guliani, 2005] raamatust.

5.1.1. Tootekriteeriumid

- **Iga**
  Liialt noored avatud lähtekoodi projektide rakendamisega ettevõttes kaasneb suurem risk.

- **Mitmed toetavad platvormid**
  Nõutumad on tooted, mis töötavad mitmel platvormil (nt Windows ja Unix).

- **Väljalasete sagedus**
  Väljalasete sagedus võimaldab hinnata selle kasutamise efektiivsust. Kui uuendusi ei väljastata piisavalt sagedasti, peavad arendajad liialt kaua ootama, et testida viimast koodi ning ei saa programmi efektiivselt kasutada. Samuti ei ole sel juhul võimalik saada piisavalt ruttu veaparadusi.

- **Populaarsus**
  Populaarsed rakendused on rohkem (erinevates süsteemides, olukordades) testitud ning seega küpsemad.

- **Disaini ja koodi kvaliteet**

  Disaini ja koodi kvaliteet on tähtsaim pea kõigist teistest faktoritest, samas aga ka üks keerulisemaid, mida hinnata. Koodi ülevaatamine eeldaks rohkesti programmeerimisoskusi ning aega. Vihjeteks on siinkohal aga koodi struktuur, modualiseeritus ning see kui loetavaks ja arusaadavaks see on tehtud.

5.1.2. Kasutuskriteeriumid

- **Lõppkasutaja tugi**

  Lõppkasutaja tugi on projekti küpsuse võtme-elemendiks ning üks kriitilisemaid aspekte avatud lähtekoodi tarkvara kasutuselevõtul. Kehvalt koostatud dokumentatsioon võib viidata ka kehvadele arenduspraktikatele ning hoolimatusele koodi lõppkasutajate suhtes.

- **Paigalduskulud**

  Mõned avatud lähtekoodi tarkvaratooted on saadaval vaid lähtekoodis, mille lõppkasutaja peab ise kompileerima, teised tulevad koos mugava paigalduspaketiga. Samuti tasuks uurida, kas eksisteerib paigaldusjuhis.

- **Kasutuskulud**

  Kasutuskuluna võiks defineerida aega, mis kulub rakenduse kohta vajalike juhiste või nõuannete saamiseks. Kui lihtne on saada infot probleemide korral, kas eksisteerivad käsiraamatud või on info saadaval vaid otseselt arendajatega kontakteerudes. Mõningad firmad pakuvad ka kolmanda osapoole koolitusteenuseid (see on üks avatud lähtekoodi ärimudelitest).

5.1.3. Integratsioonikriteeriumid

- **Modulaarsus**

  Modulaarsuse hindamisel tuleb leida vastus küsimustele, kas koodi on võimalik laiendada ning kui hästi on see moduleeritud ning kas eksisteerib rakendusliides (*Application Programming* Interface, API).

- **Koostöö teiste tarkvaratoodetega**

  Üks olulisi küsimusi tarkvaratoote juurutamisel oma süsteemi on selle sobivus teiste tarkvaratoodetega, ka suletud lähtekoodil põhinevatega. Samuti on oluline, kas rakendus töötab ka juhul, kui olemasolev süsteem on vahetatud teiste rakendustega süsteemi vastu.

- **Standardite soosivus**

  Avatud lähtekoodi tarkvara vastavus kehtivatele standarditele on üks olulisi küpsuse näitajaid.

- **Arendajate tugi**

  Kas projektil eksisteerib hea lahendus arendajate tekkinud küsimustele vastamiseks? Raskesti kättesaadavad vastused teevad antud tarkvara arendamise ettevõttes keeruliseks.

5.2. Hindamine

Hinne = 1 (ebaküps)

Toode on puudulik. Ärikriitilistes funktsioonides kasutamine on riskantne. Sellises faasis on projektid, mis pole jõudnud esimese suurema redaktsioonini.

Hinne = 2 (küps)

Tootel on arvestatavalt pikk eduka ning stabiilse juurutamise ajalugu.

Hinne = 3 (väga küps)

Tootel on pikk ja stabiilne ajalugu, lai kasutajate kogukond jne.

Alltoodud avatud lähtekoodi küpsusmudel on üldine. Kahtlemata tasub hinnata vaid neid kriteeriume, mis on antud hetkel ettevõtte jaoks olulised ning mis aitavad määrata lähtekoodi poolt ettevõtte IT valdkonnale määratavad nõudmised.

Mõistagi loeb ka see, kuidas valminud projekti ettevõttes kasutatakse.

5.3. Mudel

Tabelis 1 on kujutatud avatud lähtekoodi küpsusmudel, mille on välja toonud Woods ja Guliani [Woods&Guliani, 2005].

Tabel 1 - Avatud lähtekoodi küpsusmudel

Küpsuskriteerium	**Hinne = 1**	**Hinne = 2**	**Hinne = 3**
Tootekriteeriumid			
Iga	Alla 6 kuu	6 kuud – 2 aastat	Üle 2 aasta
Toetatavad platvormid	Üks platvorm	Palju sarnaseid platvorme	Palju heterogeenseid platvorme
Väljalasete hulk	Mitte ühtegi väljalaset poole aasta jooksul	Alla kahe väljalaske viimase aasta jooksul	Regulaarsed väljalasked
Populaarsus	Tundmatu projekt	Rakendatav alternatiiv	Valdkonna liider
Disaini kvaliteet	Monoliitne rakendus	Mitmed komponendid	Hästi defineeritud API
Kasutuskriteeriumid			
Paigalduskulud	Kasinalt dokumenteeritud paigaldusprotsess; kehv dokumentatsioon; arendajapoolne tugi	Hästi dokumenteeritud paigaldusprotsess; korralik dokumentatsioon; arendajapoolne tugi; foorumid	Hästi dokumenteeritud paigaldusprotsess; paigaldusskriptid ja/või viisardid; korralik dokumentatsioon; arendajapoolne tugi; foorumid; kolmanda osapoole paigaldusteenus
Kasutuskulud	Kasin või olematu dokumentatsioon; tugi vaid läbi otse arendajatega	Kasutusjuhendid saadaval; tugifoorumid	Kolmanda osapoole koolitusteenused saadaval

	kontakteerudes		
Lõppkasutaja tugi	Foorumid ja/või listid puuduvad	Mõned foorumid ja/või listid	Hästi hallatud foorumid ja listid koos arhiivi ja otsinguga; kolmanda osapoole tugiteenused
Integreerimiskriteeriumid			
Modulaarsus	Monoliitne struktuur; võimalik, kuid raske laiendada	Mitu moodulit, võimalik laiendada	Mitu moodulit, hästi defineeritud API, võimalik ja kerge laiendada
Koostöö teiste tarkvaratoodetega	Teadmata	Integratsioonijuhud teada	Palju integratsiooni-võimalusi dokumenteeritud
Standardite soosivus	Teadmata või patenteeritud standard	Aegunud standard	Hetkel kehtiv standard
Arendaja tugi	Puuduvad foorumid ja meililistid	Mõned foorumid ja maililistid	Kolmanda osapoole toe võimalus

5.4. Küpsuse mõju oskustele ja ressurssidele

Selleks, et teha avatud lähtekoodil baseeruv tarkvara ettevõttes võtmekomponendiks, vajab ettevõtte IT valdkond teatavat oskustekomplekti. [Woods&Guliani, 2005]

Mida vähemküps on avatud lähtekoodiga tarkvaratoode, seda rohkem teadmusbaasi on vaja, et seda juurutada. Nagu lähtub küpsusmudelist, pakuvad küpsed avatud lähtekoodi tarkvaratooted kõike, mida kommertstarkvaragi. Kõige vähem küpsete avatud lähtekoodi projektid eksisteerivad vaid ideena ning koodijupina, mis osaliselt seda ideed rakendab. Sellise tarkvaratoodete kasutamine eeldab aga sisuliselt astumist selle arendajate ridadesse. Seega võib väita, et see kui küpse või väheküpse avatud lähtekoodi tarkvara ettevõte saab rakendusse võtta, sõltub suuresti ettevõtte IT-kompetentsist.

Nii Woods ja Guilani kui ka Golden on kirjeldanud ettevõtte oskustasemeid Gaussi kõverana, kus eesotsas asuvad innovaatorid ning varased rakendajad, millele järgneb nii-öelda „kuristik“ – tühi maa, mille ületamisel erinevad teisel poolel asujate IT-kompetents oluliselt.

Innovaatoritel ning varastel rakendajatel on kõrge riskitaluvus ning kõrge kompetents, neid tõmbab uus funktsionaalsus. Varasem enamus ja hilisem enamus on kirjeldatud madala riskitaluvusega ning neid ei huvita „toores“ funktsionaalsus, vaid teatud ärilisi probleeme lahendava tehnoloogia omandamine.

Ettevõte peaks arendama piisavalt kompetentsi tunnustamaks ning rakendamaks küpset avatud lähtekoodi tarkvara, mis ületab eelpoolnimetatud kuristikku.

Woods ja Guilani on kirjeldanud erinevad oskuste komplektid IT-kompententside tasemete hindamiseks. Neid antud magistritöö piiratud mahu raames lähemalt ei vaadelda.

6 Avatud lähtekoodiga projekt ettevõttes

6.1. Lähtekoodi avalikustamise tasuvus

Otsus, kas ettevõttel tasub mõne oma rakenduse lähtekood avalikustada, vajab kahtlemata eelnevat analüüsi.

Selleks on üks Apache tuumikarendajaid, Brian Behlendorf, loonud näite IT-ettevõttest, mille tegevusalaks on andmebaasitarkvara müük. [Behlendorf, 1999]

Behlendorfi näites jagunevad ettevõtte tulud järgmiselt:

- 40% - andmebaasi tarkvara müügist
- 15% - kasutajatoe pakkumise eest
- 10% - konsultatsioonid
- 10% - kiirarenduse (*rapid application development*, RAD) vahendite müük
- 10% - graafiliste tööriistade müük
- 10% - andmebaasi protseduuride ja programmiteekide müük
- 5% - raamatud ja koolitused

On ebareaalne, et ettevõte võiks loobuda 40 protsendist tulubaasist, muutes tarkvara lähtekoodi kõigile avatuks. Seda muidugi juhul, kui on eeldatud, et ülejäänud tulubaas jääb samaks.

Pidevat arendamist vajava tarkvara lähtekood sobib avalikustamiseks. Kõik teised tuluallikad säilivad ning Behlendorf soovitab teiste teenuste hinda tõsta, nüüd kus tarkvara on kliendile vabalt kättesaadav.

Võib järeldada, et enne mõne oma rakenduse koodi avalikustamist peab ettevõtte analüüsima, kas antud rakendus omab küllalt võimalusi edasiarendamiseks ning millist tulu selle avalikustamisest on oodata. Kui tegemist on liialt spetsiifilise rakendusega, mille lähtekoodi avalikustamisel ei saa luua piisavalt uut funktsionaalsust, ei tasu lähtekoodi avalikustamine ära.

Samuti võib probleemiks olla see, kui rakenduse kasutusala piirneb oma spetsiifilisuse tõttu üksnes Eesti Vabariigi territooriumiga. Sellisel juhul on raske kaasata vabatahtlikke arendajaid väljastpoolt Eestit.

Kui aga ettevõtte suudab tarkvaratoote populaarsust tänu lähtekoodi avalikustamisele tõsta, kasvab seetõttu ka tasuliste teenuste kasutajate arv.

Tänu avatud lähtekoodile ja tasuta kättesaadavusele kasvab kindlasti antud tarkvaratoote kasutajate arv. Küsimuseks on, kui suur on nii-öelda „tasuta sõitjate" („*free riders*") ja kui suur tasuliste lisateenuste kasutajate vahekord. [Raymond, 2000] Kui tarkvaratoote kliendiks on teised ettevõtted, siis soovivad need saada antud tootega ka kindlustunnet tugiteenuste näol ning võimalust rakendus ettevõttele sobivamaks muuta.

Pöördudes tagasi Behlendorfi juurde – kui andmebaasitarkvara hakatakse selle tasuta saadavusele kasutama senisest kaks korda rohkem ning kasutajad oleksid sama motiveeritud ostma tugiteenuseid, arendustööriistu ning teeke kui varem, kasvaks ettevõtte tulud nendel aladel. Need on kompensatsiooniks kaotatud tootemüügi tuludele.

Üheks võimalikuks ajendiks, miks ettevõte lähtekoodi avalikustab on juhtum, kus ettevõte ei soovi enam tarkvaratoodet edasi arendada.

Võimalusi, mida veel pakub lähtekoodi avalikustamine, vaatleme antud peatükis edaspidi.

6.2. Võimalused

- **Klientide arvu kasv**
  Kuigi paljud saavad toodet kasutada tasuta, on senisest suuremale hulgale vajalikud ka ettevõtte poolt pakutavad lisateenused.

- **Konkurentsieelis**
  Kui ettevõtte poolt avalikustatud lähtekoodiga tarkvaratoode või algatatud projekt saavutab piisavalt suure populaarsuse, võib see antud segmendis saavutada konkurentsieelise. See viib konkurendid olukorda, kui neil tuleb hakata pakkuma ettevõtte tarkvaratootega ühilduvust, kaotamaks turgu.

- **Rohkem arendajaid**

  Projekti kallal töötab senisest rohkem arendajaid. Seda muidugi juhul, kui tarkvaratoode on piisavalt atraktiivne.

- **Kulude kokkuhoid**

  Edukas avatud lähtekoodi tarkvaraprojekt aitab kokku hoida arenduskulusid.

- **Innovatsioon**

  Tarkvaratoote kasutajad saavad pakkuda või ise realiseerida just sellise funktsionaalsuse nagu neil vaja. Sellest võib hiljem saada osa tarkvaratoote baasfunktsionaalsusest.

6.3. Ohud

Järgnevalt on autor välja selgitanud mõningad ohud, mis võivad avatud lähtekoodiga tarkvaraprojektile saatuslikuks saada.

- **Konkurendid kasutavad avatud lähtekoodi uute võimaluste realiseerimiseks.**

  Avalikustatud lähtekood on kättesaadav mõistagi ka ettevõtte konkurentidele, kes võivad seda kurjasti ära kasutada. Sellisel juhul ei ole mõttekas ettevõtte intellektuaalset vara avalikustada. Lahenduseks on sellisel juhul tarkvaratoote põhi- ning võtmekomponentide eristamine, millest esimene on avalikustatud lähtekoodis, rakendusele lisaväärtuse andvad võtmekomponendid aga suletud lähtekoodiga ning ettevõtte sees arendatavad.

- **Arendajate vähene huvi projekti vastu**

  Arendajate vähene huvi tõttu võib avatud lähtekoodi projekt hääbuda. Seda saab ära hoida ettevõttesiseste arendajate kaasamisega, kuid ainuüksi sellest võib väheks jääda. Ettevõtte peaks üle vaatama tarkvaraprojekti missiooni ning ka infoallikad (veebileht, foorum), püüdmaks leida vähese huvi põhjuseid.

- **Probleemid, millega keegi tegeleda ei oska**

  Kui kommertstarkvaraga käib kaasas kasutajatugi või vähemalt võimalus tarkvaratootjalt probleemide korral nõus küsida, siis avatud lähtekoodiga tarkvara

korral on selle kasutaja rohkem omapäi. Eksisteerivad küll listid ja foorumid, kuid nendest ei pruugi mingi spetsiifilise probleemi lahendamisel abi olla. Seda enam, kui ettevõtte satub enda arendatud avatud lähtekoodi tarkvaraga probleemi ette, millele ei tea vastust keegi majasiseselt ega ka kogukonnas. Selline olukord on küll pigem erandlik, kuid siiski arvestatav.

- **Terviklikkus**

  Kindlalt võib väita, et kommertstarkvara on terviklikum. Paigaldusprogramm, administreerimisliides ning dokumentatsioon on enamasti paremad kommertstarkvara puhul. [Woods&Guliani, 2005] Samas on ettevõtte eelduseks avatud lähtekoodi projekti alustades omada piisavalt kompetentsi ka vähem terviklikuma tarkvaratootega toimetulekuks. Avatud lähtekoodil baseeruva tarkvara küpsust on käesolevas magistritöös eraldi peatükk.

- **Kvaliteet**

  Ettevõtte poolt arendatav avatud lähtekoodi tarkvara ei pruugi saavutada juurutamiseks vajalikku kvaliteeditaset. Selline olukord on lahendatav majasiseste arendajate kaasamisel. Woods ja Guliani [Woods&Guliani, 2005] adresseerivad kvaliteediaspekti hoopis teise tahu alt – ostes ettevõttesse sisse kommertstarkvara, ei saa olla kindel selle komponentide kvaliteedis. Tegu võib olla rakenduse või funktsionaalsusega, mis ei toimi nii nagu ettevõttele vajalik. Samuti on kommertstarkvara lähtekood suletud, mistõttu ettevõtte ei saa ka ise (kasvõi vaid firmasiseselt) probleemiga midagi ette võtta.

- **Turvalisus**

  Puudub üks-ühene arvamus, kas avatud lähtekood kujutab endast suuremat turvariski kui suletud või mitte. Laurie väidab, et turvalisuse probleemid on avatud lähtekoodiga tarkvaras täpselt sarnased. Väga palju vastutust lasub lõppkasutajal, kes ei uuenda oma tarkvara. [Laurie, 2005]

  Palju on vaieldud teemal, kas lähtekoodi kättesaadavus muudab tarkvaratoote ebaturvalisemaks või mitte. Raymond väidab, et kui rakendusel on piisav beetatestijate (kasutajate) ning arendajate baas, leitakse ja parandatakse vead kiirelt. [Raymond,

2000] Selline „paljude silmade“ ideoloogia ei ole aga avatud lähtekoodi turvalisusprobleemide puhul hetkel täielikult rakenduv. [Viega, 2004; Laurie, 2005]

Avatud lähtekoodiga tarkvara arenduses puuduvad aga üldtunnustatud ja standardiseeritud protsessid turvaauditi läbiviimiseks. [Viega, 2004]

7 Uuring

Magistritöö analüüsiosas uurib autor, millistel tingimustel on Eestis võimalik läbi viia avatud lähtekoodiga tarkvaraprojekti.

Selleks viis autor läbi uuringu, leidmaks vastuse küsimustele, kui suur on vabatahtlike programmeerijate huvi avatud lähtekoodi vastu ning kaardistas avatud lähtekoodiga tarkvaraprojektiga kaasnevaid riske.

Uuringu sihtgrupiks olid Hinnavaatlus.ee foorumi ning Eesti Linux User Group (LUG) liikmed.

Hinnavaatlus.ee on suurima kasutajate ning teemade arvuga arvuti- ja tehnikaalane foorum Eestis. Uurimuse läbiviimise hetkel oli foorumis registreeritud kasutajaid 27 990.

Uuring ehitati üles, kasutades Lakhani ja Wolf'i [Lakhani&Wolf, 2005] ning Ghosh'i [Ghosh, 2005] poolt läbi viidud uuringutes esitatud küsimusi. Sellisel viisil on võimalik hiljem tulemusi võrrelda ning seeläbi analüüsida, kui palju erinevad Eesti arendajad teistes uuringutes osalenud arendajatest. Eelpool nimetatud uuringud käsitlesid siiski avatud lähtekoodi arendajaid

Uuring viidi läbi veebipõhiselt, avatud lähtekoodil baseeruva uuringutarkvaraga LimeSurvey. Ankeet postitati Hinnavaatluse foorumisse ning saadeti Linux User Group listi, personaalselt ankeeti kellelegi ei saadetud.

Uuring kestis 14. aprillist kuni 27. aprillini 2008. a. (k.a.). Sellel perioodil täitis ankeedi 50 vastanut.

7.1. Tulemuste analüüs

7.1.1. Üldandmed

Vanus

Nagu tabelist 2 lähtub, jäi vastanute vanus 14 ja 34 eluaasta vahele, keskmiseks kujunes 22 eluaastat. Valdava enamuse vastajate vanus jäi 22…24 eluaasta vahele.

Kogemus

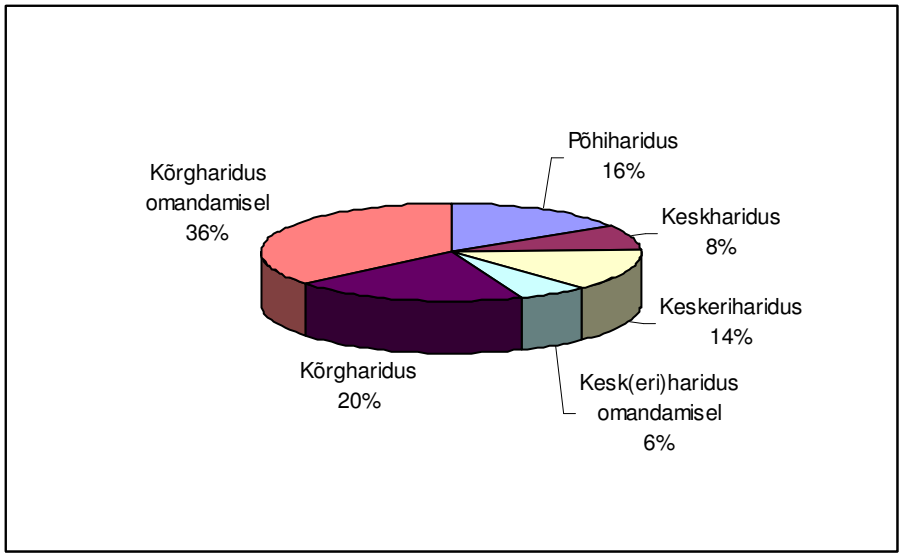
Vastanutel paluti sisestada ka nende programeerimiskogemus aastates. Kuigi see ei näita ühegi programmeerimiskeele valdamise taset, annab see siiski mõningase ülevaate vastanute kompetentsist.

Tabel 2 – Küsimustikule vastajate üldandmed

Muutuja	Keskmine	Standardhälve	Min	Max
Vanus	22,08	3,80	14	34
Aastad kogemust	3,34	3,43	0	16

Haridustase

Vastanute haridustase avaldus nagu näidatud tabelis joonisel 2.



Joonis 2 Vastajate haridustase

Enamus vastanutest oli kõrgharidusega (20%) või seda omandamas (36%). Pea võrdselt oli põhi- ning keskeriharidusega vastanuid (vastavalt 16 ja 14 protsenti). Vähem oli keskharidusega ja seda omandavaid vastanuid (8 ja 6 protsenti).

Vastajate haridustasemes on variandid "kesk(eri)haridus omandamisel" ning "kõrgharidus omandamisel", eristamaks vastavalt gümnasiste ning ülikoolitudengeid. Viimaste osakaal oli kõige suurem, moodustades üle kolmandiku kõigist vastanutest (36%). Üle poole vastanutest olid kõrgharidusega või seda omandamas.

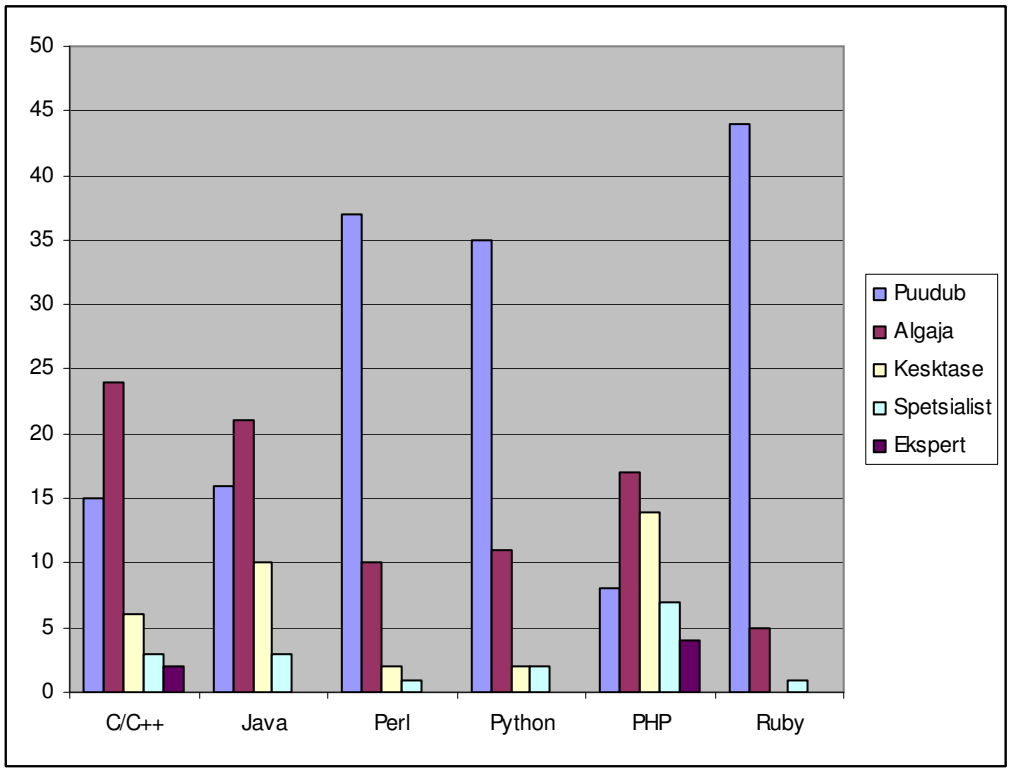
Töökoht

Vastanutest omab töökohta kaks kolmandikku (66%). Käesoleva magistritöö alguses kirjeldati avatud lähtekoodi kogukondade organisatsioonilisi karakteristikuid. Sealsest lähtuvalt ning ka uuringu tulemuste põhjal võib väita, et võimalikel tarkvaraprojekti arendajatel pole vabatahtlikuks tööks eraldada ajaliselt suuri blokke, kuna programmeerimine käib vabast ajast põhitöö kõrvalt. Töö aspekti juurde pöördume käesolevas uuringus veel tagasi.

7.1.2. Programmeerimisoskus

Programmeerimisoskuste väljaselgitamiseks paluti vastajatel Likerti skaalal hinnata oma oskusi erinevates programmeerimiskeeltes.

Küsimustikku valiti need programmeerimiskeeled, mis on toodud Woods'i ja Gulian'i raamatus [Woods&Guliani, 2005] populaarsemate avatud lähtekoodi arenduskeelte nimistus (C/C++, Perl, Python, Java, PHP, Ruby).



Joonis 3 Programmeerimisoskused

Tulemustest (vt joonis 4) selgus, et kõige enam vallatavad programmeerimiskeeled küsitletute seas on PHP, Java ja C/C++.

Kõige kompetentsem arendajate baas tundub olevat programmeerimiskeelel PHP. Seega oleks Eestis perspektiivi veebipõhistel avatud lähtekoodiga tarkvaralahendustel. See ei ole aga ainus järeldus, peaaegu sama häid tulemusi andsid ka C/C++ ning Java, kuigi programmeerimisoskuse tase oli mõlemal juhul madalam. Perl'i ja Python'i oskust esines vähem, Eestis alles viimastel aastatel populaarsust koguma hakanud Ruby pole veel kuigi palju oskajaid koguda jõudnud.

Ankeedis oli vastajal võimalik ka oma programmeerimisoskusi täpsustada. Mainiti programmeerimiskeeli C#, Haskell ja Visual Basic (3 korral), Pascal ja JavaScript (2 korral). Mitmel korral mainiti ka HTML, CSS ning erinevaid SQL-keeli, kuid neid ei saa nimetada

programmeerimiskeelteks nende spetsiifilisuse tõttu (HTML on märgendikeel, CSS laadilehtede keel ning SQL andmebaasi programmeerimiskeeled).

7.1.3. Kokkupuude avatud lähtekoodiga

Tabel 4 illustreerib vastanute varasemat kokkupuudet avatud lähtekoodil baseeruvate projektidega.

Tabel 3 – Varasem osalus avatud lähtekoodi projektis

Kas oled avatud lähtekoodi ise kirjutanud või täiendanud?	*Vastanuid*
Jah, olen aktiivselt osalenud avatud lähtekoodi projektis.	5
Jah, korra olen kaasa löönud.	3
Jah, kuid vaid isiklikuks või firma tarbeks.	9
Ei, kuid tahaksin küll.	18
Ei.	15

Varasem kokkupuude avatud lähtekoodiga osutus vastajate seas mitmekesisemaks kui esialgu arvatud. Kokku kirjeldasid 17 vastajat oma kokkupuudet avatud lähtekoodiga. Nende seas oli vigadest raporteerijaid (*bug reporting*), paikade (*patch*) kirjutajaid ning erinevate rakenduste eesti keelde tõlkijaid. Enamus vastajaid piirdus vaid projekti nimetamisega, milles nad on osalenud.

7.1.4. Avatud lähtekood ja töö

Kuna avatud lähtekoodi projektis kaasa löömine ei ole enamasti indiviidi põhiametiks, uuriti vastanute suhteid töö ja avatud lähtekoodi vahel. Tulemused on toodud tabelis 5.

Tabel 4 – Avatud lähtekood ja töö

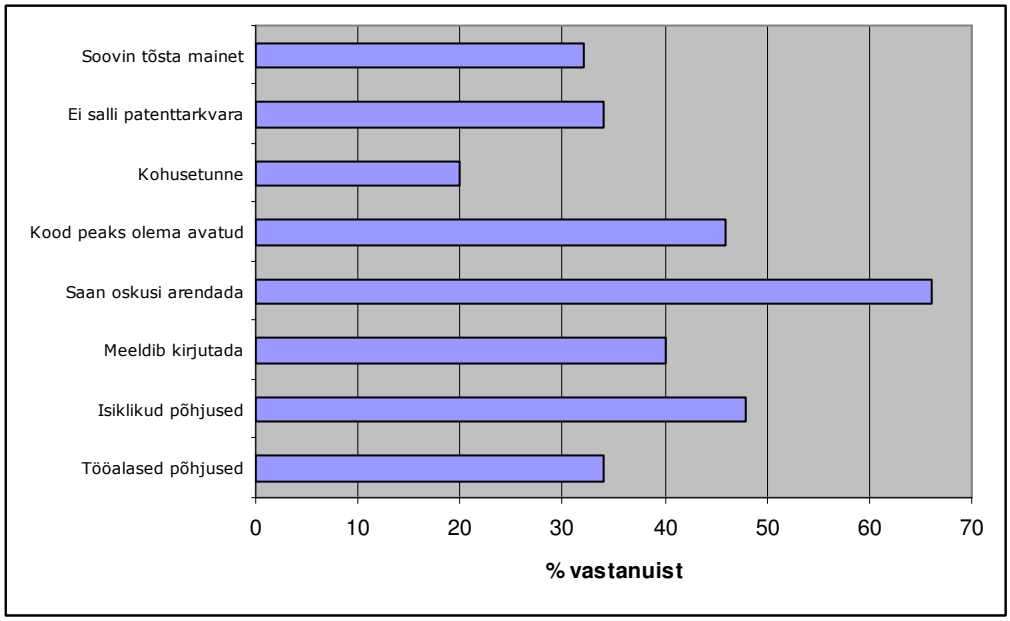
Kas tegeled avatud lähtekoodi kirjutamisega tööajast?	*Protsent*
Jah, tööalal on see vajalik.	4
Jah, tööandja on sellest teadlik.	5
Jah, kuid tööandja pole sellest teadlik.	3
Ei.	80

Kuna eelmisest küsimusest selgus, et avatud lähtekoodiga on tegelenud 17 vastajat, on seletatav ka vaieldamatult suur protsent neid, kes ei tegele avatud lähtekoodiga tööajast. Lisaks ka veel need, kes on küll avatud lähtekoodiga tegelenud, kuid mitte tööajast.

7.1.5. Motivatsioon

Avatud lähtekoodi projektis osalemise motivaatorid valiti tuginedes Lakhani ja Wolfi [Lakhani&Wolf, 2005] ning Ghosh'i [Ghosh, 2005] samateemaliste uuringute baasil läbi viidud uuringutele.

Vastuste jagunemine on kujutatud joonisel 5. Korraga uuriti nii avatud lähtekoodiga kokku puutunud kui mitte kokkupuutunud vastajate arvamust. Selline lähenemine tagas rohkem vastuseid ning andis võimalikest motivaatoritest parema pildi. Ettevõttele, kes soovib algatada avatud lähtekoodiga tarkvara projekti, on oluline ka nende arvamus, kes seni pole sellises projektis osalenud kuid õige(te) motivaatori(te) ajel seda valmis tegema oleks.



Joonis 5 Motivatsioon avatud lähtekoodi projektis kaasalöömisel

Tulemuste järgi (joonis 5) on küllaltki populaarsed kõik motivaatorid (üle kolmandiku vastajatest), vaid „kohusetunne kogukonna ees“ on pälvinud vähem vastanud. Seda arvatavasti põhjusel, et avatud lähtekoodi kogukonnad pole projektides mitteosalenutele tuttavad.

Kõige olulisemaks motivaatoriks peetakse võimalust ennast arendada (66% vastajatest), sarnane tulemus ilmnes ka Ghoshi uuringust [Ghosh, 2005].

Kokkuvõte

Käesolev magistritöö annab ülevaate avatud lähtekoodil põhinevast hübriidsest arendus- ning ärimudelist. Samuti annab magistritöö vastuse küsimusele, kuidas on võimalik läbi viia avatud lähtekoodiga tarkvaraprojekti ettevõttes.

Hübriidne arendusmudel kombineerib endas nii ettevõtetes kasutatavat kui avatud lähtekoodi arendusmudeli elemente. Projekti planeerimine sarnaneb tarkvaraprojekti planeerimisega ettevõttes, arendus- jt iteratiivsed protsessid avatud lähtekoodi arendusmudelile.

Avatud lähtekoodi kasutavad ärimudelid keskenduvad avatud lähtekoodile lisa- ja tugiteenuste pakkumisele, mis annavad äriettevõtetele lisagarantii avatud lähtekoodiga tarkvara kasutamiseks. Tuntuim avatud lähtekoodi ärimudelit rakendav ettevõte on Red Hat Linux. Hübriidsed ärimudelid kasutavad litsentsikogumeid, mis määravad ära tarkvara kasutamise õigused eri aspektidel. Sellist ärimudelit kasutav tarkvara ei pruugi olla avatud lähtekood selle klassikalises mõistes.

Uuringu tulemustest selgus, et huvi avatud lähtekoodi tarkvara arendamise vastu Eestis on olemas. Mitmed vastanud on avatud lähtekoodi projektides juba ka ühel või teisel moel osalenud.

Kõige enam kompetentsi oli vastanute seas programmeerimiskeeles PHP. Ka Java ning C++ keeltega kokkupuutunuid leidus küllaltki palju.

Peamise motivaatorina avatud lähtekoodi arendamisel toodi välja võimalust ennast arendada. Vastanute seas oli keskkooliõpilasi ning tudengeid, kellele jaoks oleks avatud lähtekoodi projektis osalemine õppimisvõimalusi andev ning kasulik hiljem tööturul konkureerimisel.

Summary

Current Master thesis gives an overview of hybrid development and business models based on open source software (OSS).

Hybrid development model combines both elements of software development at firms and development practices of open source software. Planning to start an open source software project is similar to planning "regular" software projects at firm while the development and other iterative processes are more common to open source software development practices.

Hybrid business models use open source software in combination with additional and supportive services. These give an extra guarantee to enterprises wanting to deploy OSS. Hybrid business models also cover software projects with hybrid licenses, which give different rights to different types of users. This kind of software might not be using open source software under its classic term but is still benefiting from the volunteer community.

Author also conducted a survey to determine interest and motivations of participating on open source development project in Estonia. The results showed that there exists an interest, most common motivation to participate was the opportunity to increase one's knowledge.

Allikate loend

[Behlendorf] Behlendorf, B. (1999); Open Source as a Business Strategy; In: Open Sources: Voices from the Open Source Revolution; [edited by C. DiBona et al]; 149-170; O'Reilly Media Inc; ISBN 978-1565925823

[Benkler] Benkler, Y. (2002); Coase's Penguin, or Linux and the Nature of the Firm; URL: http://www.benkler.org/CoasesPenguin.PDF; 2008-04-12

[Bonaccorsi et al] Bonaccorsi, A., Giannangeli, S., Rossi, C. (2006); Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry; Management Science; Vol. 52; No. 7; July 2006; 1085-1098

[Borrell] Borell, J. (2001); Changing mankind; Information Systems Journal; Volume 12; Issue 1; January 2002; 7-25

[Brooks] Brooks, F. P. (1982); The Mythical Man-month: Essays on Software Engineering; Addison-Wesley; ISBN:0201006502

[EULA] Microsoft, (2001); Microsoft Windows XP Professional End-User License Agreement; URL: http://download.microsoft.com/documents/useterms/Windows%20XP_Professional_English_9e8a2f82-c320-4301-869f-839a853868a1.pdf

[Feller&Fitzgerald] Feller, J., Fitzgerald, B. (2000); A framework analysis of the open source software development paradigm; International Conference of Information Systems; Sydney; Australia; 14.-16. detsember 2000

[Friedman] Friedman, Thomas L. (2006); The World is Flat; Penguin Books; ISBN: 0141022728

[Ghosh] Ghosh, R. A. (2005); Understanding Free Software Developers: Findings from the FLOSS Study; In: Perspectives on Free and Open Source Software; [edited by J. Feller et al.]; 23-46; MIT Press; Cambridge, MA, USA; ISBN 0262062461

[Golden] Golden, B. (2005); Making Open Source Ready for The Enterprise: The Open Source Maturity Model; Navica Inc.; San Carlos, CA, USA

[Goldman&Gabriel] Goldman, R., Gabriel, R. P. (2005); Innovation Happens Elsewhere: Open Source as Business Strategy; Morgan Kaufmann; ISBN: 9781558608894 URL: http://dreamsongs.com/IHE/IHE.html; 2008-

04-12

[Kaljula] Kaljula, K. (2004); Tarkvara testimine nõuete formuleerimise, analüüsi ja disaini etapis - magistritöö; Tartu

[Kay] Kay, R. (2002); QuickStudy: System Development Life Cycle; URL: http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=71151; 2008-04-12

[Kikkas] Kikkas, K. (2006); Vabad litsentsid – mis see on?; URL: http://www.kakupesa.net/kakk/docs/vabadlitsentsid.pdf; 2008-04-12

[Krishnamurthy] Krishnamurthy, S. (2003); An Analysis of Open Source Business Models; URL: http://faculty.washington.edu/sandeep

[Lakhani&Wolf] Lakhani, K. R., Wolf, R. G. (2005); Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects; In: Perspectives on Free and Open Source Software; [edited by J. Feller et al.]; 3-22; MIT Press; Cambridge, MA, USA; ISBN 0262062461

[Laurie] Laurie, B. (2005); Open Source Security; In: Open Sources 2.0: The Continuing Revolution; [edited by C. DiBona et al]; 57-70; O'Reilly Media Inc.; Sebastopol, CA, USA; ISBN: 9780596008024

[Lerner&Tirole] Lerner, J., Tirole, J.; (2002); Some Simple Economics of Open Source; The Journal of Industrial Economics; No.2; June 2002; Volume L

[LGPL] Free Software Foundation (FSF), (2008); GNU Lesser General Public License; URL:http://www.gnu.org/licenses/lgpl.html

[McGowan] McGowan, D. (2005); Legal Aspects of Free and Open Source Software; In: Perspectives on Free and Open Source Software (2005); [edited by J. Feller et al.]; 361-392; MIT Press; Cambridge, MA, USA; ISBN 0262062461

[Mickos] Mickos, M. (2007); Open Source: Why Freedom Makes a Better Business Model; Open Source Business Conference (OSBC); 23.05.2007; San Francisco, CA, USA; URL: http://akamai.infoworld.com/weblog/openresource/archives/OSBC2007%20-%20Marten%20Mickos%20Keynote.pdf; 2008-03-23

[Mockus et al] Mockus, A., Fielding, R. T., Herbsleb, J. D. (2005); Two Case Studies

of Open Source Software Development: Apache and Mozilla; In: Perspectives on Free and Open Source Software (2005); [edited by J. Feller et al.]; 163-210; MIT Press; Cambridge, MA, USA; ISBN 0262062461

[MySQL] MySQL AB (2006); MySQL Licensing Policy; URL: http://www.mysql.com/about/legal/licensing/; 2008-04-27

[OSI] Open Source Licenses; Open Source Initiative; URL: http://www.opensource.org/licenses; 2008-03-15

[QNX[1]] QNX Software Systems (2007); QNX Non-Commercial End User License Agreement; URL: http://www.qnx.com/legal/licensing/non_commercial.html; 2008-04-23

[QNX[2]] QNX Software Systems (2007); QNX Partner Software License Agreement; URL: http://www.qnx.com/legal/licensing/partners.html

[QNX[3]] QNX Software Systems (2007); QNX Commercial Software License Agreement; URL: http://www.qnx.com/legal/licensing/commercial.html; 2008-04-23

[Raymond] Raimond, E. S. (2000); The Catherdal and the Bazaar; O'Reilly Media Inc.; Sebastopol, CA, USA; ISBN: 9781565927247; URL: http://catb.org/~esr/writings/cathedral-bazaar/; 2008-05-04

[Rosen] Rosen, L. (2007); The New QNX Hybrid Software Model: Combining open source and proprietary benefits for embedded systems; URL: http://www.qnx.com/download/download/16868/qnx_whitepaper_hybrid_software_model.pdf; 2008-03-28

[Robbins] Robbins, J. (2003); Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools; In: Perspectives on Free and Open Source Software; [edited by J. Feller et al.]; 245-264; MIT Press; Cambridge, MA, USA; ISBN 0262062461

[Sharma et al] Sharma, S., Sugumaran, V., Rajagopalan, B. (2002); A framework for creating hybrid-open source software communities; Information Systems Journal; Vol. 12; Issue 1; 7-25

[Stallman] Stallmann, R. (2006); The Free Software Movement and The Future of Freedom; 09.03.2006; URL:

http://www.fsfeurope.org/documents/rms-fs-2006-03-09.en.html; 2008-05-01

[Viega] Viega, J. (2004) Open Source Security: Still a Myth; O'Reilly Media Inc.; URL: http://www.onlamp.com/pub/a/security/2004/09/16/open_source_security_myths.html; 2008-04-04

[Välimäki] Välimäki, M. (2003); Dual Licensing in Open Source Software Industry; Systèmes d'Information et Management (SIM); No.8; 2003; Number 1

[Weber] Weber, S. (2004); The Success of Open Source; Harvard University Press; ISBN 0674012925

[Woods&Guliani] Woods, D., Guliani, G. (2005); Open Source for the Enterprise; O'Reilly Media Inc.; ISBN 0596101198

Lisa: Uuringuankeet

Avatud lähtekoodi arendamine Eestis

Tere!

Olen Tanel Jõeäär ning kirjutan Tallinna Ülikoolis IT juhtimise alal magistritööd "Hübriidsed tarkvara arendus- ja ärimudelid" ning uurin **avatud lähtekoodiga tarkvara arendamise võimaluste kohta Eestis**.

Selleks püüan välja selgitada, kas ja kui palju leidub Eestis avatud lähtekoodiga tarkvara arendajaid (ka potentsiaalseid). Palun leia võimalusel kohvipausi või vaba hetke kõrvale aega, et see küsimustik täita.

Küsimuste ja ettepanekutega võib pöörududa mailiaadressile tiiger[ät]tlu.ee

Ette tänades,
Tanel Jõeäär

Selles küsitluses on 11 küsimust.

Üldine

***Sinu vanus?**

Sellele väljale võib sisestada vaid numbreid

***Haridustase**
Choose one of the following answers

- põhiharidus
- keskharidus
- keskeriharidus
- kesk(eri)haridus omandamisel
- kõrgharidus
- kõrgharidus omandamisel

***Kas käid hetkel tööl?**

- Jah
- Ei

***Mitu aastat oled programmeerimisega tegelenud?**

Sellele väljale võib sisestada vaid numbreid

Milliseid programmeerimiskeeli valdad?

	Pole kokku puutunud	Algaja	Kesktase	Spetsialist	Ekspert	Vastus puudub
C/C++	☐	☐	☐	☐	☐	◉
Java	☐	☐	☐	☐	☐	◉
Perl	☐	☐	☐	☐	☐	◉
Python	☐	☐	☐	☐	☐	◉
PHP	☐	☐	☐	☐	☐	◉
Ruby	☐	☐	☐	☐	☐	◉

? Valikus on avatud lähtekoodi arendamiseks kasutatavad levinuimad vabalt levivad keeled. *(Woods & Guliani "Open Source for the Enterprise", O'Reilly Media 2005)*

Avatud lähtekood

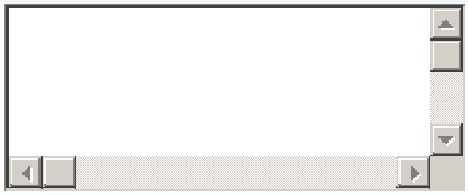
***Kas oled avatud lähtekoodi rakendust ise kirjutanud või täiendanud?**

Choose one of the following answers

- ☐ Jah, olen aktiivselt osalenud avatud lähtekoodi projektis.
- ☐ Jah, korra olen kaasa löönud.
- ☐ Jah, kuid vaid enda või firma tarbeks.
- ☐ Ei, kuid tahaksin küll.
- ☐ Ei.

? Ka näiteks tõlkinud rakenduse eesti keelde, ei pruugigi tingimata koodi kallale minna.

Millises avatud lähtekoodi projektis oled osalenud? Võimalusel kirjelda täpsemalt.



***Kas tegeled avatud lähtekoodi kirjutamisega tööajast?**
Choose one of the following answers

- ○ Jah, tööalal on see vajalik
- ○ Jah, tööandja on sellest teadlik
- ○ Jah, kuid tööandja pole sellest teadlik
- ○ Ei

Osaled/osaleksid avatud lähtekoodi projektis, sest:
Märkige kõik mis sobivad

- ☐ Koodi on vaja tööalastel põhjustel
- ☐ Koodi on vaja isiklikel põhjustel
- ☐ Meeldib koodi kirjutada
- ☐ Saan oma oskusi arendada
- ☐ Arvan, et lähtekood peaks olema avatud
- ☐ Kohusetunne avatud lähtekoodi kogukonna ees
- ☐ Ei salli patenttarkvara
- ☐ Soovin tõsta oma mainet avatud lähtekoodi kogukonnas
- ☐ Muu: _____