TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MIKKO KARTTUNEN

OHJELMISTOPROJEKTIN ETENEMISEN ENNUSTAMINEN AUTOMAATTISESTI KERÄTYLLÄ DATALLA

Diplomityö

Tarkastaja: Prof. Kari Systä
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunnan
tiedekuntaneuvoston
kokouksessa 09.11.2016

TIIVISTELMÄ

MIKKO KARTTUNEN: Ohjelmistoprojektin etenemisen ennustaminen automaattisesti kerätyllä datalla
Tampereen teknillinen yliopisto
Diplomityö, 77 sivua, 0 liitesivua
Kesäkuu 2017
Tietotekniikan koulutusohjelma
Pääaine: Ohjelmistotuotanto
Tarkastajat: Prof. Kari Systä
Avainsanat: Ennustaminen, projektinhallinta, github, issue, commit, kommentti

Avoimen lähdekoodin projekteja isännöivistä palveluista voidaan kerätä projekteihin liittyvää ohjelmistotuotantodataa automaattisella louhinnalla. Tätä dataa voidaan käyttää ohjelmistoprojektin etenemisen ennustamiseen, mikä auttaa projektipäällikköä ennakoimaan projektin resurssitarpeita. Tämän tutkimuksen tavoitteena oli selvittää voiko avoimen lähdekoodin ohjelmistoprojektin työtehtävien määrän kasvua ennustaa käyttämällä automaattisesti louhittua ohjelmistotuotantodataa. Tutkimuksessa etsittiin ennustuspotentiaalia valituista datapareista viiveellisen yhteyden avulla eli kuinka muutos yhdessä aikasarjassa vaikuttaa viiveellä toiseen tai itseensä.

Tutkimuksessa käytetty kymmenen projektin aineisto kerättiin GitHub-verkkopalvelusta. Seitsemässä projektissa tuotteena on tekstinkäsittelyohjelma ja kolmessa ei. Projekteja tutkittiin visualisoinnin, ristikorrelaation, impulssivastefunktion ja autokorrelaation avulla. Visualisoinnilla tutkittiin kokonaiskuvaa ja sopivaa aikaväliä ennustuspotentiaalin etsimiseen. Matemaattisilla menetelmillä tutkittiin aikasarjojen välistä viiveellistä yhteyttä eli kuinka samanlainen viivästetty versio yhdestä aikasarjasta on toiseen verrattuna.

Suurimmasta osasta projekteja löytyi alle viikon aikajänteellä näkyvä yhteys dataparien välillä. Aikasarjat oli koottu viikkotasolle eli yhteys näkyi ilman viivästystä. Eri dataa sisältävistä datapareista ei löytynyt yleisluontoista viiveellistä yhteyttä yli viikon viiveellä. Työtehtävien sisältä löytyi kuitenkin voimakasta autokorrelaatiota 90% projekteista eli työtehtävien ilmestymisessä on säännönmukaisuutta. Tätä voidaan käyttää ennustamiseen, mutta autokorrelaatiolla ei voida ennustaa yllättäviä muutoksia aikasarjoissa.

Projektipäällikkö voi ennustaa alle viikon aikajänteellä näkyvän yhteyden perusteella työtehtävien ilmestymistä, jos työtahti on tiedossa. Esimerkiksi työtahdin lisääminen nopeuttaa myös työtehtävien ilmestymistä. Työtehtävien autokorrelaation avulla olisi mahdollista antaa projektin etenemisestä suuntaa antava ennuste esimerkiksi GitHubin tehtävienhallintajärjestelmän yhteydessä.

ABSTRACT

MIKKO KARTTUNEN: Forecasting the progress of a software project using automatically collected data
Tampere University of Technology
Diplomityö, 77 pages, 0 Appendix pages
June 2017
Master's Degree Programme in Information Technology
Major: Software Engineering
Examiner: Prof. Kari Systä
Keywords: Forecasting, project management, github, issue, commit, comment

It is possible to automatically collect software engineering data from services that host open source software projects. This data can be used to forecast the progress of a software project, which helps the project manager estimate future resource requirements. The research goal of this master's thesis was to find out whether it is possible to forecast the growth rate of issues in an open source software project using automatically collected software engineering data. The research focused on finding forecasting potential using delayed correlation between specific data pairs. Delayed correlation refers to the change observed in one time series that is later observed in another time series.

The studied data, which consists of ten software projects, was collected from the web service GitHub. Seven of the projects produce text editor software and three do not. The projects were studied using visualization methods, cross correlation, impulse-response function and autocorrelation. The visualizations offered an overview of the projects and helped with the selection of the mathematically studied time periods. The mathematical methods were used to search for a delayed correlation between the time series, or in other words to analyze how similar the delayed version of one time series is compared to another time series.

A correlation with a delay of less than a week was found in most of the projects and data pairs studied. The time series were aggregated to weekly level which means the correlation showed up without any delay. No consistent delayed correlations were found in data pairs consisting of heterogenous data with longer delays. However, there was strong autocorrelation in pure issue data in 90% of the projects studied, which means that the rate at which new issues open is not random. This can be used for forecasting, though sudden changes in time series can't be forecast with autocorrelation.

Based on the correlation with a delay of less than a week, a project manager can forecast the rate at which new issues open if the planned rate of development is known, since an increase in the rate of work accelerates the rate at which new issues open. It is possible to estimate the progress of a project using the autocorrelation in issue data. This method could be integrated to GitHub's issue management system.

ALKUSANAT

Kun alunperin mietin diplomityölleni aihetta, minulla oli toiveena yhdistää ohjelmistotuotantoon näkökulmia joko projektijohtamisesta tai web-sovelluskehityksestä. Etsin näihin alueisiin liittyvää aihetta tietotekniikan laitokselta ja kuulin professori Kari Systältä aiemmin laitoksella tehdystä tutkimuksesta, joka liittyi ohjelmistotuotannon datan analysointiin. Aihe herätti nopeasti mielenkiintoni. Projektijohtamisen näkökulma on helppo ottaa osaksi projektien datan analysointia, ja tutkimuksissa käytetty Visu-työkalu oli web-teknologioilla rakennettu. Kaikki toiveeni yhdistyivät samassa aihepiirissä. Vaikka valitsin tarkan tutkimusaiheen itse, tietotekniikan laitoksella tehty aiempi tutkimus inspiroi sen.

Kari Systä sekä ohjasi että tarkasti diplomityöni. Lisäksi hän auttoi antamalla aiempien data-analyysitutkimusten papereita minulle luettavaksi, mikä nopeutti työn alkuun saamista. Tahdon antaa erityiskiitoksen hänelle avusta koko diplomityöprosessin aikana.

Sain työn alkuvaiheessa apua myös Alexey Patrusheviltä ja Anna-Liisa Mattilalta Visu-työkalun käyttöön ja jatkokehitykseen liittyen. Alexey opasti minua Visun muokkaamisessa ja antoi valmiin tietokannan täynnä itse keräämäänsä dataa, jonka perusteella pystyin nopeammin suunnittelemaan omat datatarpeeni. Anna-Liisa puolestaan vastasi kysymyksiini liittyen Visun toimintaan sekä antoi pääsyn Visun lähdekoodiin. Haluan kiittää heitä molempia aikansa käyttämisestä diplomityöni hyväksi.

Lisäksi haluan kiittää puolisoani Marjaana Karttusta valmiin LaTeX-pohjan antamisesta, oikolukemisesta ja diplomityön tekemiseen liittyvistä vinkeistä, joita hän osasi antaa oman diplomityöprosessinsa pohjalta. Tekstin omituisista viittaussuhteista piikittely on selvästi tehokkain tapa varmistaa niiden huomioiminen.

Tampereen Hervannassa 3.5.2017

Mikko Karttunen

SISÄLLYS

1. Johdanto 1

2. Ohjelmistotuotantodata 3

2.1 Data ja datalähteet 3

2.2 Datan tutkiminen 4

2.3 GitHub ja avoimen lähdekoodin projektit 6

2.4 Ohjelmistotuotannon visualisointi 7

3. Ohjelmistotuotannon ennustaminen 9

3.1 Ennustaminen 9

3.2 Ohjelmistotuotanto ja ennustaminen 10

3.3 Virheiden ennustaminen 12

4. Ennustusongelma 15

4.1 Ongelman asettelu 15

4.2 Hypoteesit 16

4.3 Kriteerit onnistumiselle 18

5. Menetelmä ja työkalut 20

5.1 Esikartoitus 20

5.2 Päätutkimus 21

5.2.1 Yleiskatsaus 21

5.2.2 Matemaattinen menetelmä 22

5.3 Visu-Forecaster työkalu 24

5.3.1 Esittely ja tausta 24

5.3.2 Datan esikäsittely ja visualisointi 25

6. Tutkitut projektit 27

6.1 Yleispiirteet ja valintaperusteet 27

6.2 Projektit 28

6.2.1 Esitystapa 28

6.2.2 Spacemacs 29

6.2.3 Neovim . . . 31
6.2.4 TextMate . . . 33
6.2.5 Lime . . . 34
6.2.6 LightTable . . . 36
6.2.7 Caret . . . 37
6.2.8 Zed . . . 39
6.2.9 Video.js . . . 40
6.2.10 Fabric . . . 42
6.2.11 Vagrant . . . 43
6.3 Yhteenveto projekteista . . . 45
7. Datan tutkiminen . . . 47
7.1 Yleisiä huomioita . . . 47
7.2 Kirjaukset ja auenneet tehtävät . . . 49
7.3 Pienet kirjaukset ja sulkeutuneet tehtävät . . . 52
7.4 Kommentit ja sulkeutuneet tehtävät . . . 55
7.5 Auenneet tehtävät . . . 57
8. Tulosten arviointi . . . 61
8.1 Yhteenveto tuloksista . . . 61
8.2 Johtopäätökset . . . 65
9. Yhteenveto . . . 70
Lähteet . . . 73

KUVALUETTELO

6.1 Projektin Spacemacs kirjaukset ja auenneet tehtävät ajan funktiona. . 31

6.2 Projektin Neovim kirjaukset ja auenneet tehtävät ajan funktiona. . . 32

6.3 Projektin TextMate kirjaukset ja auenneet tehtävät ajan funktiona. . 34

6.4 Projektin Lime kirjaukset ja auenneet tehtävät ajan funktiona. 35

6.5 Projektin LightTable kirjaukset ja auenneet tehtävät ajan funktiona. 37

6.6 Projektin Caret kirjaukset ja auenneet tehtävät ajan funktiona. . . . 38

6.7 Projektin Zed kirjaukset ja auenneet tehtävät ajan funktiona. 40

6.8 Projektin Video.js kirjaukset ja auenneet tehtävät ajan funktiona. . . 41

6.9 Projektin Fabric kirjaukset ja auenneet tehtävät ajan funktiona. . . . 43

6.10 Projektin Vagrant kirjaukset ja auenneet tehtävät ajan funktiona. . . 44

TAULUKKOLUETTELO

6.1 Projektista Spacemacs louhitun datan määrä. 30

6.2 Projektin Spacemacs virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät. 30

6.3 Projektista Neovim louhitun datan määrä. 32

6.4 Projektin Neovim virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät. 33

6.5 Projektista TextMate louhitun datan määrä. 33

6.6 Projektin TextMate virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät. 33

6.7 Projektista Lime louhitun datan määrä. 35

6.8 Projektin Lime virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät. 36

6.9 Projektista LightTable louhitun datan määrä. 36

6.10 Projektin LightTable virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät. 36

6.11 Projektista Caret louhitun datan määrä. 38

6.12 Projektin Caret virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät. 39

6.13 Projektista Zed louhitun datan määrä. 39

6.14 Projektin Zed virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät. 39

6.15 Projektista Video.js louhitun datan määrä. 41

6.16 Projektin Video.js virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät. 42

6.17 Projektista Fabric louhitun datan määrä. 42

6.18 Projektin Fabric virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät. 42

6.19 Projektista Vagrant louhitun datan määrä. 44

6.20 Projektin Vagrant virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät. 45

6.21 Yhteenveto projektien datamääristä ja perusluonteista. 45

7.1 Kirjausten ja auenneiden tehtävien ristikorrelaation tulokset 50

7.2 Kirjausten ja auenneiden tehtävien impulssivastefunktion tulokset . . 51

7.3 Pienten kirjausten ja sulkeutuneiden tehtävien ristikorrelaation tulokset 53

7.4 Pienten kirjausten ja sulkeutuneiden tehtävien impulssivastefunktion tulokset 54

7.5 Kommenttien ja sulkeutuneiden tehtävien ristikorrelaation tulokset . 55

7.6 Kommenttien ja sulkeutuneiden tehtävien impulssivastefunktion tulokset 56

7.7 Auenneiden tehtävien autokorrelaation ja ristikorrelaation tulokset . . 58

7.8 Auenneiden tehtävien impulssivastefunktion tulokset 59

8.1 Yhteenveto ristikorrelaation merkitsevistä tuloksista 62

8.2 Yhteenveto impulssivastefunktion merkitsevistä tuloksista 63

8.3 Merkittäviä tuloksia sisältävien projektien osuus tutkituista projekteista 66

TERMIT

Dataluokka	samanlainen joukko dataa, datatyyppi, esimerkiksi kommentti tai tehtävä
Datalähde	tietojärjestelmä, josta haetaan dataa tutkimista varten
Kirjaus	eng. commit, pysyvä muutos versiohallintajärjestelmän ylläpitämään tietovarastoon
Liitospyyntö	eng. pull request, ohjelmistokehityksen työnkulkuun liittyvä tapahtuma, jossa kehittäjä pyytää ylläpitäjää lisäämään tehdyt muutokset osaksi projektia
Merkintä	eng. label, tehtävään liitetty sana, joka kuvaa tehtävän tyyppiä
Tehtävä	eng. issue, työkokonaisuus, joka voi olla esim. virheraportti tai ominaisuustoive
Tehtävienhallintajärjestelmä	eng. issue tracker, sisältää projektin tehtävät ja hallinnoi niitä
Tietovarasto	eng. repository, sisältää versiohallintajärjestelmän ylläpitämän datan
Ominaisuustoive	eng. feature request, käyttäjän toivoma lisäys ohjelmistoon
Versiohallintajärjestelmä	eng. version control system, varastoi ja versioi mm. lähdekoodidataa
Virhe	eng. bug tai error, lähdekoodissa oleva ongelma
Virheraportti	eng. bug report, tehtävä, joka sisältää ilmoituksen ohjelmistossa olevasta viasta

1. JOHDANTO

Yksi oleellisimmista keinoista tehdä säästöjä ohjelmistoprojektissa on reagoida ongelmiin mahdollisimman aikaisessa vaiheessa. Tätä periaatetta sovelletaan paljon ohjelmistojen teknisessä suunnittelussa. Sama pätee kuitenkin projektinhallintaan: projektin resurssitarpeiden ennustamisella voidaan ennaltaehkäistä virheraporttien kasautumista ja varmistaa tasainen kehitystahti [35]. Yksi tapa ennustaa projektin resurssitarpeita on tarkastella projektin työtehtäviä ja niihin liitoksissa olevia välituotteita. Tällaista ohjelmistotuotantodataa voidaan kerätä automaattisesti projektien käyttämistä versio- ja tehtävienhallintajärjestelmistä [28].

Tämän tutkimuksen tavoitteena on selvittää voiko avoimen lähdekoodin ohjelmistoprojektin työtehtävien määrän kasvua ennustaa käyttämällä automaattisesti louhittua ohjelmistotuotantodataa. Tutkimusongelma on jaettu neljään hypoteettiseen kysymykseen: 1. Aiheuttaako työn tekeminen viiveellä uusia työtehtäviä? 2. Aiheuttaako virheenkorjaukseksi oletettu työnteko viiveellä työtehtävien valmistumista? 3. Aiheuttaako työtehtävien kommentointi viiveellä työtehtävien valmistumista? 4. Vaikuttaako vanhojen työtehtävien määrä myöhemmin ilmestyvien työtehtävien määrään?

Näihin hypoteettisiin kysymyksiin liittyvää ohjelmistotuotantodataa on saatavilla lähes kaikista moderneista projekteista eikä datan kerääminen vaadi monimutkaista esikäsittelyä. Tällä hetkellä harva ohjelmistoprojekti hyödyntää matemaattista ennustamista osana suunnittelua. Jos näin alkeellisella datalla on mahdollista ennustaa ohjelmistoprojektin etenemistä, suuri osa vakavista ohjelmistoprojekteista voisi ennakoida resurssitarpeitaan nykyistä paremmin.

Tutkimus keskittyy ohjelmistotuotantodatan ennustuspotentiaalin etsimiseen viiveellisen yhteyden avulla eli kuinka muutos yhdessä aikasarjassa vaikuttaa viiveellä toiseen tai itseensä. Kymmentä avoimen lähdekoodin projektia visualisoidaan yleisten huomioiden tekemiseksi, ja tarkkaa ennustuspotentiaalia tutkitaan ristikorrelaatiolla ja impulssivastefunktiolla. Näillä signaalinkäsittelyn menetelmillä lasketaan kahden aikasarjan välisen viiveellisen yhteyden voimakkuutta. Lisäksi autokorrelaatiolla tutkitaan aikasarjan vanhojen arvojen vaikutusta uusiin arvoihin.

Tutkimuksessa jatkokehitettiin ensin valmista visualisointi- ja analyysiohjelmistoa, jolla valituista projekteista kerättiin dataa. Esitetyt hypoteettiset kysymykset tarkennettiin liittymään tiettyihin datapareihin. Projektien visualisaatioista tunnistettiin sopiva aikaväli viiveellisen yhteyden etsimiseen ja data ryhmiteltiin viikkotasolle. Tämän jälkeen jokaisen hypoteesin dataparista laskettiin ristikorrelaation ja impulssivastefunktion tulokset ja tulosten merkittävyyttä arvioitiin tilastollisella mittarilla. Lopuksi projektien tuloksia vertailemalla arvioitiin dataparien ennustuspotentiaalia.

Tuloksista kävi ilmi, että tutkituissa datapareissa on voimakasta yhteyttä yhden viikon aikajänteellä eli ilman, että aikasarjoja on viivästetty toisiinsa nähden ollenkaan. Erilaisesta datasta muodostetuissa datapareissa ei löytynyt yleistä viiveellistä yhteyttä yli viikon aikajänteellä. Työtehtäviin liittyy kuitenkin voimakasta autokorrelaatiota lähes kaikissa projekteissa, mikä mahdollistaa ennustamisen tietyin rajoituksin.

Tutkimus rakentuu seuraavanlaisesti. Luvuissa 2 ja 3 esitellään työhön liittyvä teoreettinen tausta ohjelmistotuotantodatan ja ennustamisen osalta. Luvussa 4 esitellään tutkimusongelma, tarkat hypoteesit ja merkitsevien tulosten kriteerit. Luvussa 5 taustoitetaan tutkimuksen etenemistä, esitellään käytetty matemaattinen menetelmä sekä käytetty tutkimustyökalu. Luvussa 6 esitellään tutkitut projektit ja näiden valintakriteerit. Luvussa 7 suoritetaan tutkimus ja esitellään tulosten yksityiskohdat. Luvussa 8 arvioidaan tuloksia sekä niiden merkitystä kokonaisuutena, ja luvussa 9 tehdään koko diplomityön yhteenveto.

2. OHJELMISTOTUOTANTODATA

2.1 Data ja datalähteet

Ohjelmistotuotantoprosessi tuottaa paljon tietoa tukemaan työntekoa ja sen johtamista erilaisia työkaluja ja prosesseja käytettäessä. Osa tiedosta on vain ihmisten muistissa, osa tietokoneilla ja osa molemmissa. Osa tiedosta tuotetaan tarkoituksellisesti ja osa prosessin sivutuotteena. Tietokoneille tallennetusta tiedosta käytetään selvyyden vuoksi termiä data erottamaan se projektihenkilöstön tiedoista. Ohjelmistotuotantodataa on tallessa muun muassa versiohallinta- (eng. version control), tehtävienhallinta- (eng. issue tracker), testaus- ja koontijärjestelmissä [29], jotka yhdessä ovat erilaisia datalähteitä. Ohjelmistotuotantodatalle on tyypillistä, että se kuvaa tapahtumia, joihin liittyy aikaleima, tekijä ja vaihteleva joukko teennöksiä [29]. Erityisesti aikaleimojen yleisyys tekee ohjelmistotuotantodatasta helposti käsiteltävää ja tutkittavaa aikasarjamenetelmillä.

Tehtävät (eng. issue) ovat tehtävienhallintajärjestelmään kirjattuja työkokonaisuuksia, jotka voivat esittää puhtaiden työtehtävien lisäksi muun muassa virheraportteja (eng. bug report), dokumentoinnin tarvetta tai toiveita uusiksi ominaisuuksiksi (eng. feature request). Tehtäviä on mahdollista merkitä (eng. label) näiden eri tehtävätyyppien mukaan. Tehtäviin voi liittyä kommentteja, kun kehittäjät keskustelevat esimerkiksi virheenkorjausvaihtoehdoista. Tehtäviin voi myös liittyä liitospyyntöjä (eng. pull request), jotka ovat ohjelmistokehityksen ja versionhallinnan työnkulkuun kuuluvia tapahtumia, joissa kehittäjä lähettää ohjelmistoprojektin ylläpitäjälle pyynnön lisätä tehdyt muutokset osaksi projektia. Muutosten kirjaukset (eng. commit), tai lyhyemmin kirjaukset, ovat pysyviä muutoksia versiohallintajärjestelmän hallinnoimaan tietovarastoon (eng. repository).

Tehtävät ovat erityisen kiinnostavaa dataa, koska tehtävien hallinnointi on keskeisessä osassa ohjelmistokehitystä [20] ja niiden tutkiminen voi auttaa ohjelmistoprojektin hallinnassa [42]. Ne liittyvät suoraan projektin laajuuteen ja sitä kautta kustannuksiin. Kaikki tehtävät eivät kuitenkaan ole selvitettävissä projektin alussa, joten parempi ymmärrys tehtävistä ja niiden kehityskulusta auttaa

projektipäällikön työtä.

Kirjaukset ovat harvoin kiinnostavaa dataa yksinään, koska ne muodostuvat kehitystyön sivutuotteena eivätkä itseisarvoisesti. Jos kirjausdataa yhdistetään tehtäviin, voidaan kuitenkin löytää kiinnostavia huomioita, koska näillä on vahva yhteys toisiinsa [3]. Kommentit voivat olla itsessäänkin kiinnostavia tietynlaisissa tutkimuksissa [5], mutta myös niitä yhdistämällä muihin dataluokkiin voidaan löytää kiinnostavia lisähuomioita.

Tehtäviä, kirjauksia ja kommentteja yhdistää kolme oleellista asiaa: ne ovat läsnä käytännössä kaikissa nykyaikaisissa ohjelmistoprojekteissa, ne yhdistetään usein samaan järjestelmään ja niillä on vahvat liitokset toisiinsa. Kirjauksia pitää tehdä, että tehtävät valmistuvat ja tehtäviä pitää välillä kommentoida, että tiedetään millaisia kirjauksia tarvitaan.

Versiohallintajärjestelmä ja tehtävienhallintajärjestelmä ovat tehtävien, kirjausten ja kommenttien pääasiallisia datalähteitä. Tehtävienhallintajärjestelmä on keskeinen työkalu, joka auttaa tehtävien ilmoittamisessa, jakamisessa kehittäjille, seurannassa, valmiiksi saamisessa ja arkistoinnissa sekä toimii yhteistyön ja kommunikoinnin alustana [5]. Vastaavasti versionhallintajärjestelmä mahdollistaa hajautetun ohjelmistokehityksen, pitää kirjaa projektin eri versioista ja toimii pysyvänä tietovarastona ohjelmiston välituotteille. Nämä järjestelmät voivat olla erillään tai yksi järjestelmä voi tarjota sekä versio- että tehtävienhallinnoinnin. Eri järjestelmät tallentavat dataa vaihtelevalla tarkkuudella ja eri muodoissa, mutta kaikista löytyy yhtenevä joukko perusdataa [20].

Monet tehtävienhallintajärjestelmät tarjoavat rajapinnan, jonka avulla voi ohjelmallisesti hakea sen sisältämää dataa [20]. Tämän ansiosta on mahdollista rakentaa automaattisesti toimivia järjestelmiä, joilla haetaan haluttu data tarvittaessa useastakin lähteestä [20], esikäsitellään data tarvittuun muotoon, tehdään analyysilaskenta ja esitetään tulokset käyttäjälle. Tällöin esimerkiksi projektipäällikkö voi saada syvällistä tietoa projektinsa tehtävien tilanteesta ja etenemissuunnasta käyttämättä siihen ylimääräistä aikaa.

2.2 Datan tutkiminen

Ohjelmistotuotantodata on harvoin kiinnostavaa raakamuodossa, mutta sen tarkemmalla tutkimisella voi löytää hyödyllisiä huomioita. Ohjelmistoanalytiikka (eng. software analytics) on ohjelmistotutkimuksen osa-alue, joka pyrkii ymmärtämään ohjelmistotuotantoa paremmin tutkimalla sen tuottamaa dataa. Sen tavoitteena on

hankkia ammatinharjoittajille hyödyllisiä huomioita jonkin tietyn tehtävän suorittamisesta siten, että huomioita voidaan jatkossa hyödyntää tehtävän parempaan suorittamiseen myös käytännössä. Paremmat suorituskäytännöt johtavat tuottavuuden, laadun ja käyttäjäkokemuksen paranemiseen. [43]

Menzies et al. painottavat analytiikan reaaliaikaisuuden tärkeyttä [31]. Koska analytiikan tulosten perusteella pitäisi pystyä tekemään parempia päätöksiä, analytiikan ja sen käyttämän datan tulee olla ajantasaista. Datan automaattinen kerääminen ja analyysin kohdistaminen meneillään olevaan projektiin auttaa tässä. Ohjelmistoanalytiikka on kasvanut räjähdysmäisesti viime aikoina [31]. Sillä on potentiaalia vaikuttaa laajasti käytännön ohjelmistotuotantoon, koska nykyiset avoimen lähdekoodin järjestelmät ovat täynnä ajantasaista dataa ja analytiikka on laajenemassa useille eri ohjelmistotuotannon osa-alueille [43].

Yksi tehokas analyysitekniikka on yhdistää ohjelmistotuotantodataa useasta eri datalähteestä yhteen paikkaan ja esittää data graafisessa muodossa eli visualisaationa [28]. Tällä tekniikalla on mahdollista tehdä uusia havaintoja sekä testata olemassa olevia hypoteeseja. Datan ei kuitenkaan ole pakko tulla useasta eri datalähteestä, vaikka analyysiin käytettäisiin useaa erilaista dataluokkaa.

Versiohallinta- ja tehtävienhallintajärjestelmien tallentamaa dataa on hyödynnetty useilla eri ohjelmistotutkimuksen osa-alueilla. Esimerkiksi ohjelmistojen evoluution tutkimusalalla D'Ambros et al. ovat tutkineet näistä lähteistä saatavaa dataa ohjelmistoarkeologia -tekniikan avulla [10], ja Fischer et al. ovat pyrkineet ymmärtämään ohjelmistojen muutosta taaksepäin katsovasta näkökulmasta [13]. Ohjelmistojen evoluutioon verrattain läheisesti liittyvä tulevaisuuden ennustaminen on myös yleinen tutkimuskohde, jota tarkastellaan tarkemmin luvussa 3.

Datan keräämistä ja analysointia varten on useita vaihtoehtoja riippuen siitä, mitä halutaan tutkia. Yksi keino on prosessien louhiminen (eng. process mining), jossa esimerkiksi liiketoimintajärjestelmästä haetaan maksuprosessiin liittyvää dataa. Prosessien louhinnalla yritetään ymmärtää paremmin yleistä prosessia tutkimalla yksittäisiä prosessitapauksia [33]. Esimerkin tapauksessa yritetään siis ymmärtää paremmin yleistä maksuprosessia tutkimalla tiettyyn yritykseen liittyviä yksittäisiä maksuprosesseja. Prosessien louhinnassa on usein ongelmana järjestelmien monimutkaisuus ja epäyhteensopivuus, minkä vuoksi louhintatyökalu pitää räätälöidä jokaista eri järjestelmää varten erikseen [40]. Prosessien louhimista ja siitä saatavaa dataa on perinteisesti käytetty liiketoiminta-analytiikassa.

Toinen keino on tietovarastojen louhiminen, jonka avulla yritetään löytää hyödyllistä tietoa ohjelmistoista, projekteista ja ohjelmistotuotannosta yleisesti. Tässä tek-

niikassa on perinteisesti yhdistetty datan esikäsittely läheisesti itse analyysiin, jolloin samaa analyysin laskentaosaa ei voida helposti käyttää muiden tietovarastojen tutkimiseen [33]. Tietovarastojen louhiminen on suosittua ja nykyään tutkijat hyödyntävät prosessien louhimisen tekniikoita myös ohjelmistoanalytiikassa [33][29]. Yksi esimerkki tästä on esikäsittelijän ja analytiikan eriyttäminen tehtävienhallintajärjestelmän louhinnassa.

Datan louhiminen on yleisesti ottaen tutkijoita tällä hetkellä kiinnostava menetelmä. Sitä käytetään ohjelmistoalalla muun muassa ennustusmenetelmissä, datan tutkimisessa, luokittelutyökaluissa, yhteyksien etsimisessä ja ohjelmistojen evoluution ymmärtämisessä sekä muissa harvinaisemmissa käyttötarkoituksissa. Usein louhittuja datalähteitä ovat muun muassa versiohallinta-, tehtävienhallinta- ja kommunikointijärjestelmät sekä erilaiset käyttölokit. Uusien juuri tätä tarkoitusta varten kehitettyjen louhintatyökalujen ja prototyyppien käyttäminen on hyvin yleinen tapa tehdä analytiikkaa, koska olemassa olevat raskaat louhintatyökalut ovat verrattain kalliita tutkijoille. [8]

2.3 GitHub ja avoimen lähdekoodin projektit

Ohjelmistotuotannon tutkijoilla on nykyisin helppo pääsy valtavaan määrään dataa erilaisten avoimen lähdekoodin projekteja isännöivien palveluiden kautta. Suosituin näistä on tällä hetkellä GitHub-verkkopalvelu, jonne kuka tahansa voi luoda oman git-versiohallintajärjestelmän tietovaraston ja tehdä tämän avulla hajautetusti yhteistyötä muiden kehittäjien kanssa. GitHubissa on tällä hetkellä yli 10 miljoonaa tietovarastoa erilaisille projekteille, joten valinnanvaraa aineiston valitsemiseen on paljon.

Yhteistyötä tukevat työkalut ovat GitHubin käyttäjille tärkeitä. Näitä ovat muun muassa kommentointi- ja keskusteluominaisuudet, sisäänrakennettu tehtävienhallintajärjestelmä, käyttäjien toiminnan seuraaminen sekä liitospyyntöihin perustuva työnkulku, joka mahdollistaa helpon koodikatselmoinnin. Ohjelmistotuotannon tutkijoita GitHubissa kiinnostaa sen suosio, laajuus, keskenään integroidut sosiaaliset ominaisuudet sekä metadata, johon pääsee käsiksi GitHubin tarjoaman rajapinnan avulla. [21]

GitHubin käyttöön datalähteenä liittyy myös vaaroja. Projektien valitsemisessa pitää olla tarkkana, koska esimerkiksi aktiivisten projektien tunnistaminen voi olla hankalaa, samaan projektiin voi liittyä useita tietovarastoja ja suuri osa GitHubin projekteista on pieniä ja henkilökohtaisia, eikä näistä tällöin voi vetää luotettavia johtopäätöksiä minkä tahansa tutkimuksen kontekstissa. Mikään ei takaa, etteikö

GitHubin rajapinta muuttuisi ja näyttäisi dataa odottamattomalla tavalla. GitHubissa ei välttämättä myöskään näy kaikki kehittäjien toimet, eikä esimerkiksi ulkopuolista viestintää voi jättää huomiotta, jos ollaan tutkimassa ohjelmistotuotannon sosiaalista puolta. [21]

Työn tutkiminen vaikuttaa työntekijöiden käyttäytymiseen toimistoympäristössä, mikä puolestaan vaikuttaa kerättävän datan luotettavuuteen. GitHub ei myöskään ole ongelmaton tässä suhteessa. Grier huomioi GitHubin muistuttavan merkittävästi sosiaalista mediaa, ja toteaa ettei palvelusta saa välttämättä luotettavampaa dataa verrattuna toimistoympäristöön [15]. Lisäksi pitää huomioida, että GitHubin tehtävienhallintajärjestelmä on ominaisuuksiltaan suppeampi kuin pelkkään tehtävien hallinnointiin tarkoitetuissa järjestelmissä [20], joten kaikki mahdollisesti hyödyllinen data ei ole tarjolla. Kaikista näistä huomioitavista seikoista huolimatta GitHub on erinomainen lähde monenlaisten tutkimusten käyttöön.

Myös avoimen lähdekoodin projekteilla, joita käytännössä kaikki laajemman tutkijakunnan saatavilla olevat GitHub-projektit ovat, on omat erityispiirteensä. Tyypillinen avoimen lähdekoodin projekti on laajuudeltaan pieni, yhden tai kahden kehittäjän hitaasti ylläpitämä ohjelmisto, jota käyttää pieni yleisö [6]. Tämän vuoksi keskiverto avoimen lähdekoodin projekti ei edusta hyvin esimerkiksi teollisuuden ohjelmistotuotantoa. Tutkittavien projektien valinnassa pitää siis ottaa huomioon myös projektien luonne.

Avoimen lähdekoodin projektien tuottama prosessidata ei ole välttämättä huonompaa kuin suljetun lähdekoodin projektien data. Bachmann et al. vertasivat avoimen ja suljetun lähdekoodin projektien prosessidatan laatua ja huomioivat eri projektien välillä olevan huomattavia eroja datan laadussa. Suljettu lähdekoodi ei kuitenkaan suoraan taannut parempaa laatua [3]. Turvallisinta on olettaa ettei missään projektityypissä ole välttämättä hyvälaatuista dataa tarjolla ja siksi kiinnittää huomiota datan laatuun kaikissa tapauksissa. Bachmann et al. tuloksen perusteella voidaan myös hieman luottavaisemmin soveltaa avoimen lähdekoodin prosessidataa hyödyntävien tutkimusten tuloksia teollisuuden tilanteisiin. Itse prosesseissa voi kuitenkin olla huomattavia eroja, joten tulosten soveltamisessa täytyy olla varovainen. Käytetyt kehitysprosessit ovat osittain projektikohtaisia ja myös avoimen lähdekoodin projektit voivat erota toisistaan paljon.

2.4 Ohjelmistotuotannon visualisointi

Kerättyä dataa voi tutkia monilla erilaisilla menetelmillä. Jos kyseessä on erityisesti suuri määrä abstraktia dataa, on visualisointi yksi voimakkaimmista vaihtoe-

hdoista analyysimenetelmäksi. Suuri osa ohjelmistotuotantoon liittyvästä tiedosta on jo valmiiksi tietokoneen ymmärtämässä muodossa, mikä lisää visualisointitekniikoiden yleisyyttä alalla. Yksi tuore, datalähteidensä puolesta kiinnostava ja visualisointia hyödyntävä esimerkki tästä on tutkimus, jossa Mattila et al. rakensivat aikajanavisualisaation ohjelmistoprojektin ominaisuuksien etenemisestä ja toimitusnopeudesta [28].

Visualisoinnin perusidea on esittää data graafisessa muodossa niin, että ihminen voi tehdä siitä tulkintoja ja olla vuorovaikutuksessa datan esitysmuodon kanssa. Visualisointi analyysitekniikkana sopii hyvin sekä uusien ideoiden ja hypoteesien etsimiseen, että näiden varmentamiseen, koska esitysmuodon muokkaaminen mahdollistaa tutkimuksen tavoitteiden ja fokuksen siirtämisen tarpeen mukaan. Visualisoinnin päähyödyt verrattuna tilastollisiin- ja koneoppimistekniikoihin ovat kyky käsitellä kohinaista ja epähomogeenistä dataa sekä intuitiivinen käyttötapa, joka ei vaadi tietämystä monimutkaisista kaavoista tai algoritmeista. [23]

Visuaalinen datan tutkiminen noudattaa pääsääntöisesti kolmivaiheista prosessia, jossa ensin tehdään yleiskatsaus ja tunnistetaan kiinnostavat muodot, sitten tarkennetaan huomio kiinnostaviin pisteisiin ja tarpeen mukaan noudetaan yksityiskohtia [23]. Kun tällainen visualisointiprosessi yhdistetään aikasarjadataan samassa työkalussa, ihminen pystyy intuitiivisesti täydentämään aikasarjan muotoja ja arvioida miten muoto voisi jatkua nykyhetkestä eteenpäin. Tästä on hyötyä ennustamisessa.

Tällä hetkellä yleisimmät visualisointia hyödyntävät ohjelmistotuotannon tutkimuskohteet liittyvät ohjelmistojen rakenteisiin, käytökseen ja evoluutioon. Projektinhallintaa, prosesseja ja viestintää sekä tehtävienhallintajärjestelmistä tulevaa dataa tutkitaan visuaalisesti vain vähän. Visualisointiin käytetään yleisimmin uusia tarkoitusta varten luotuja työkaluja valmiiden ratkaisujen sijaan. [27]

Tuoreita esimerkkejä ohjelmistoprojektin johtamista tukevista visualisoinneista ei löydy montaa. Näin ollen on luonnollista aloittaa perusyhteyksien tutkimisesta ja jatkaa myöhemmin tästä saatujen tulosten perusteella yksityiskohtaisempiin tutkimuksiin. Visualisointi tekniikkana vaatii tutkimuksen monimutkaisuudesta riippumatta täsmällistä tutkimusmenetelmää, ettei subjektiivinen tulkinta vääristä tuloksia.

3. OHJELMISTOTUOTANNON ENNUSTAMINEN

3.1 Ennustaminen

Ennustamisen perusajatuksena on saada arvausta parempi arvio jostain epävarmasta tulevaisuuden tilasta, jotta voitaisiin tehdä parempia päätöksiä nykyhetkessä. Siinä missä suunnittelu yrittää vaikuttaa siihen miltä tulevaisuuden pitäisi näyttää, ennustaminen pyrkii selvittäminen miltä tulevaisuus tulee näyttämään. Ennustuksen tuloksia on luonnollista käyttää suunnittelun tukena ja toisaalta suunnitelmien mahdolliset vaikutukset voi ottaa huomioon ennustuksessa. [2]

Kaikki ennustustilanteet eivät ole samanlaisia, joten on olemassa lukuisia erilaisia ennustusmenetelmiä. Menetelmän valinta vaikuttaa merkittävästi ennustuksen luotettavuuteen, mutta myös siihen millaisia johtopäätöksiä ennusteesta voi vetää. Esimerkiksi matemaattiset menetelmät tuottavat numeroita, joita voi jatkokäsitellä, kun taas ihmisten päätöksiä ennustava roolipelaamismenetelmä on lähempänä valistunutta arvausta. Ennustaminen ei lupaa varmoja vastauksia, mutta hyvin tehty ennustus on arvokkaampi kuin valistumaton arvaus. [2]

Ekstrapolaatiomenetelmät ja asiantuntija-arviot ovat kaksi kiinnostavaa ja toisiaan tukevaa ennustusmenetelmää. Ekstrapolaatiomenetelmien etuina ovat luotettavuus, objektiivisuus, halpuus, nopeus ja helppo automatisointi, mutta ne eivät toimi luotettavasti voimakkaasti muuttuvissa tilanteissa [2]. Näiden ominaisuuksien vuoksi ekstrapolointi on yleinen ohjelmistoilla toteutettava ennustusmenetelmä esimerkiksi varastotilanteiden ennakointiin. Asiantuntija-arviot sen sijaan ovat tehokas ennustuskeino, kun saatavilla ei ole hyvää numeerista dataa. Haittapuolena tässä ennustustavassa on alttius muun muassa ennakkoasenteiden ja huonon kysymyksenasettelun vaikutukselle [2].

Jos ennustamiseen käytetään apuna visualisaatiota, saadaan yhdistettyä ekstrapoloinnin ja asiantuntija-arvion hyviä puolia. Visualisaation lähtökohdaksi voidaan rajata asiantuntijan määrittelemä oleellinen datajoukko, rakentaa graafi tilastollisia menetelmiä hyödyntäen, ekstrapoloida se tulevaisuuteen ja tarjota loppu-

tulos asiantuntijalle tulkittavaksi. Tällainen yhdistelmä vähentää tilanteen muuttuvaisuudesta koituvia haittoja ja antaa asiantuntijalle objektiivisemman lähtökohdan arviointiin. Esimerkiksi trendikäyrän muutoksen arviointiin suositellaan asiantuntija-arviota pelkkien numeroiden käsittelyn sijaan, sillä pelkkä ekstrapolointi olettaa tilanteen jatkuvan samanlaisena kuin aiemminkin eikä siten kykene ennustamaan poikkeamia [2].

Ennustaminen on epävarmaa ja hyväkin ennuste on usein vain todennäköisyys, joten ennustusjärjestelmien toiminnan vertailu ei ole helppoa. Shepperd et al. tarjoavat kolme kysymystä ennustusjärjestelmien arvioimiseen: 1. Toimiiko järjestelmä paremmin kuin puhdas arvaus eli ennustaako se? 2. Onko vertailtavien järjestelmien välillä havaittu tarkkuusero tilastollisesti merkittävä? 3. Onko järjestelmien tarkkuuserolla käytännön merkitystä, vaikka ero olisi tilastollisesti merkittävä? [37]. Nämä kysymykset muistuttavat, ettei ennustettavuus ole vain joko mahdollista tai mahdotonta, vaan kyseessä on jatkumo hyvästä ennustettavuudesta huonoon. Esimerkiksi monia asioita on mahdollista ennustaa tutkimalla ennustavan muuttujan yhteyttä ennustettavaan muuttujaan eli seurata miten yhdessä muuttujassa ensin näkyvät muutokset vaikuttavat viiveellä toiseen muuttujaan. Eri muuttujien välisten yhteyksien voimakkuuksissa on kuitenkin eroja. Heikko yhteys voi teknisesti mahdollistaa ennustamisen, mutta käytännön tulokset voivat olla vaatimattomia vaikutuksen tai luotettavuuden osalta.

3.2 Ohjelmistotuotanto ja ennustaminen

Projektin tulevaisuudentilan ennustamista voidaan pitää insinööritieteiden kulmakivenä. Ohjelmistoprojektien kohdalla on pyritty ennustamaan muun muassa hintaa, aikataulua ja virheherkkyyttä yli 40 vuoden ajan, mutta tulokset ovat olleet harvoin yhdenmukaisia tai helposti tulkittavia. Kiinnostus ennustamista ja parempia menetelmiä kohtaan on suurta. Tutkittuja menetelmiä on paljon ja niihin kuuluu muun muassa perinteiset tilastolliset tekniikat sekä modernit oppivat järjestelmät. Tutkijoiden tekemät meta-analyysit kuitenkin osoittavat ettei mikään yksittäinen ennustusmenetelmä ole selvästi muita parempi, minkä vuoksi ammatinharjoittajille on vaikeaa antaa hyviä neuvoja. [37]

Keskiverto ohjelmistoprojektiin liittyy paljon erilaisia epävarmuustekijöitä heti rakennettavan tuotteen määrittelystä alkaen, joten ei ole yllättävää etteivät ennustusmenetelmät anna luotettavia tuloksia kautta linjan. Jokainen projeki on uniikki ja monesti monimutkaisempi ja abstraktimpi kuin esimerkiksi rakennusprojektit, joiden projektijohtamisen työkaluja sovelletaan ohjelmistoalallakin. Nykyisiltä työkaluilta ei siis voi odottaa erinomaista ennustustarkkuutta. Yksi hyödylli-

nen tutkimussuunta voikin olla löytää ne ohjelmistoprojektin osa-alueet, joilla on merkittävää ennustuspotentiaalia ja keskittyä kehittämään ennustustyökaluja näitä varten.

Kiinnostus ennustamista kohtaan on ymmärrettävää, kun ottaa huomioon miten arvokasta tietoa ennustaminen tuottaa esimerkiksi projektijohtamiselle. Jos projektipäällikkö pystyy kohtalaisen luotettavasti ennustamaan virheiden (eng. bug, error) esiintymisen, on mahdollista optimoida virheiden korjaamiseen sijoitettavia resursseja [35]. Virheet eivät ole ainoa esimerkki rahallisen hyödyn tuottamisesta. Sama periaate toimii muuallakin: mitä aiemmin ongelmaan voidaan varautua, sen halvempaa ongelman käsittely on.

Vuosikymmenien tutkimuksesta huolimatta 2000-luvun alkupuolella yleisesti tarjolla olleet ohjelmistoprojektien suorittamista tukevat visualisointiohjelmistot keskittyivät pääasiassa kokonaisuuksien hahmottamiseen eivätkä ennustamiseen [4]. Yksi tähän liittyvä ongelma on voinut olla epätarkkojen ennustusmenetelmien lisäksi ennustukseen käytettävän datan vaikea saatavuus, koska käytössä ei ole ollut standardia keinoa louhia dataa. Visualisoinnissa, jossa käytetään paljon samaa dataa kuin ennustamisessa, on myöskin huomattu lähes välttämättömäksi rakentaa uusi tarkoituksenmukainen työkalu jokaista visualisointitehtävää varten [32], mikä ei auta asiaa. Nykyiset suuret projektien isännöintipalvelut voivat olla apu tähän ongelmaan.

Lähivuosina ennustukseen liittyvää tutkimusta on tehty muun muassa virheiden ennakoinnin ja muutosten ennustamisen osa-alueilla. Vaikka versiohallintajärjestelmään tehtävät kirjaukset ovat yleisesti käytetty dataluokka ohjelmistotutkimuksessa, virheiden ennakointiin keskittyvä tutkimus ei ole käyttänyt tätä dataa paljoa [32]. Kirjauksia hyödyntävä virheiden ennustaminen voi siis olla potentiaalinen alue uusia löytöjä ajatellen.

Tutkijat ovat myös laittaneet rinnakkain vanhoja ja uusia menetelmiä näiden paremmuuden arvioimiseksi. Esimerkiksi eräs tutkimus vertasi yleisesti tunnettua COCOMO-mallia regressio- ja neuroverkkopohjaisiin laajuudenarviointimenetelmiin [11]. Neuroverkko osoitti kyvykkyytensä tässä tilanteessa kuten monissa muissakin tuoreissa tutkimuksissa. Erikoisempi esimerkki neuroverkoille annetusta syötteestä on projektin kehittäjien käymä keskustelu tutkimuksessa, jonka tavoitteena oli tunnistaa puutteita uusien ominaisuustoiveiden joukosta [14]. Oppivien järjestelmien menestyksestä huolimatta tällä tekniikalla on omat heikkoutensa kuten kallis hinta ja monimutkaisuus [35]. Ohjelmistoprojekteja on yritetty ennustaa myös paljon korkeammalla tasolla, esimerkiksi koko projektin onnistumisen näkökulmasta [16][17].

Näin laaja näkökulma tosin vaatii jatkotutkimuksia tarkkojen epäonnistumiseen johtaneiden syiden selvittämiseksi.

Raja et al. huomasivat tutkimuksessaan kaksi kiinnostavaa seikkaa: pelkällä virheraporttien aikasarjadatalla voi ennustaa tyydyttävästi tulevien virheiden esiintymistä, mutta ennustusmalli täytyy säätää organisaatiokohtaisesti [35]. Tämä viittaa siihen, ettei yleispätevää ennustusmenetelmää voi välttämättä rakentaa, vaan jokaisen organisaation, ellei jopa jokaisen projektin, on räätälöitävä ennustusmenetelmän parametrit organisaatioon sopiviksi. Tutkimus käytti ainoastaan dataa virheraporteista, joten kaikki merkitsevä data ei todennäköisesti ollut mukana ennustusmallissa. Tämä voi myös vaikuttaa tuloksiin ja ennustusmenetelmän organisaatiosidonnaisuuteen.

Ohjelmistoprojektien ennustaminen ei ole käyttökelpoinen tekniikka kaikissa tilanteissa. Luotettava aikasarjadataan pohjautuva ennustaminen vaatii paljon historiallista dataa [35], ja pienempien toimijoiden kannattaa keskittyä liiketoiminnan analysointiin ennen ohjelmistotuotannon ennustamista, koska siitä saa hintaan verrattuna suuremman hyödyn [36]. Tarkkaa rajaa riittävä ison toimijan määritelmälle ei kuitenkaan ole vielä kehitetty, ja hyödyt kasvanevatkin portaattomasti suurempaa kohti mentäessä.

3.3 Virheiden ennustaminen

Ohjelmistotuotannossa ja erityisesti testauksessa virheellä tarkoitetaan jotain tahatonta poikkeamaa ohjelman määrittelystä. Esimerkiksi väärin muotoiltu ehtolause lähdekoodissa on virhe. Vasta virheellisen ohjelman ajaminen voi johtaa vikoihin ja häiriöihin. Virheiden ennustamisella yritetään selvittää esimerkiksi paljonko löytymättömiä virheitä on vielä olemassa, paljonko niitä ilmestyy tulevaisuudessa ja mitkä ohjelmiston osat ovat virhealtteimpia. Ennustamisella ei pyritä selvittämään ohjelman ajonaikaista käytöstä.

Virheiden ennustaminen on erityisen houkuttelevaa, koska korjaus- ja ylläpitotoimet maksavat keskiverto ohjelmistoprojektin elinkaarella enemmän kuin mikään muu toimi. Paremmalla tiedolla voidaan tehdä rahallisia säästöjä. On hyvä huomioida, etteivät kaikki tehtävienhallintajärjestelmään luodut tehtävät ole virheraportteja, vaikka virheraportit voivat olla joissain projekteissa merkittävin yksittäinen tehtävätyyppi.

Virheitä voidaan ennustaa monella eri tavalla, tarkkuudella ja monimutkaisuusasteella. Lähestymistapoja ovat muun muassa muutoslokiin pohjautuvat

menetelmät, yhteen versioon pohjautuvat menetelmät sekä muut menetelmät. Muutoslokiin pohjautuvissa menetelmissä käytetään versiohallintajärjestelmästä saatavaa dataa olettamalla esimerkiksi uusissa tiedostoissa olevan enemmän virheitä kuin vanhoissa. Yhteen versioon pohjautuvissa menetelmissä historiadatan käyttämisen sijaan oletetaan ohjelman nykytilan kuten lähdekoodin monimutkaisuuden vaikuttavan tulevaisuudessa löytyvien virheiden määrään. Muissa menetelmissä voidaan yhdistellä kahta aiempaa lähestymistapaa tai esimerkiksi ennustaa virheitä tiedostojen viittaussuhteiden kautta. Ennustusalgoritmien syötteenä käytetään yleensä erilaisia lähdekoodiin perustuvia mittareita, prosessidataa sekä historiadataa löydetyistä virheistä. Ennustustulokset jakaantuvat arvioihin tulevaisuudessa ilmestyvistä virheistä ja arvioihin virheiden nykyisestä määrästä joko projekti-, tiedosto-, komponentti- tai luokkatasolla. [9]

Myös projektin tyypillä on vaikutusta virheiden ennustamiseen. Zou et al. huomasivat, etteivät suljetun lähdekoodin ohjelmistoille tarkoitetut luotettavuuden kasvumallit (eng. reliability growth model) sovellu suoraan avoimen lähdekoodin projektien arvioimiseen. Syynä tähän on muun muassa erilainen kehitystahti, koska avoimen lähdekoodin projekteissa on harvoin kokopäivätöissä kehittäjiä. Tämän vuoksi tuorein historiadata on avoimen lähdekoodin tilanteessa erityisen arvokasta ennustuskäyttöön. Tutkimuksessa ei todettu avoimen lähdekoodin tarkoittavan huonoa ennustettavuutta, mutta tulokset viittaavat siihen, ettei avoimen lähdekoodin projekteilla toimivia menetelmiä voi välttämättä suoraan soveltaa teollisuudessa. [44]

Virheiden ennustaminen kiinnostaa tutkijoita laajasti. Zanetti et al. lähestyivät aihetta luokittelun näkökulmasta ja jakoivat projektin virheraportit päteviin sekä käsittelykelvottomiin ilmoituksiin [42]. Ferenc puolestaan otti dataa lähdekoodista, versiohallintajärjestelmästä ja tehtävienhallintajärjestelmästä ja käytti koneoppimista ennustamaan mistä luokista löytyy virheitä nyt ja tulevaisuudessa [12]. Singh et al. vertasivat kahta lähdekoodimuutosten monimutkaisuuteen perustuvaa ennustusmallia ja yrittivät löytää avoimen lähdekoodin projekteista piilossa olevien virheiden kokonaismäärän [39]. Lähdekoodin monimutkaisuudella on ennustettu myös virheiden sijainteja yhdistämällä tekniikka regressiiviseen koneoppimiseen [22].

Parhaista virheiden ennustamiseen sopivista dataluokista ei ole tarkkaa tietoa, mutta arvioita löytyy. Shin et al. huomasivat ettei ennustustarkkuus parantunut merkittävästi, jos syötteeseen lisättiin historiadatan lisäksi muita yksinään hyvin ennustavia dataluokkia eli tässä tilanteessa kutsurakennedataa (eng. calling structure information) [38]. Tutkimuksen aineisto oli pieni, mutta tutkijat suosittelivat käyttämään historiadataa ennustamiseen aina, kun sitä on saatavilla. Yksi mahdollinen

syy vaatimattomaan tarkkuuden parantumiseen on korrelaatiot eri syötteiden välillä, mihin on syytä kiinnittää huomiota useampaa dataluokkaa käytettäessä.

Kaikkiaan virheitä on ennustettu onnistuneesti monilla eri dataluokilla. Yleisimpiä vaikuttaisivat tällä hetkellä olevan historiallinen data projektin virheistä sekä lähdekoodin muutoksiin pohjautuvat mittarit. Puhtaasti kirjauksiin tai projektin keskusteluun perustuvia ennustusmalleja on vähän, joten näiden tutkimisella on mahdollista löytää uusia kiinnostavia tuloksia.

4. ENNUSTUSONGELMA

4.1 Ongelman asettelu

Tämän tutkimuksen tavoitteena on selvittää voiko avoimen lähdekoodin ohjelmistoprojektin tehtävien määrän kasvua ennustaa käyttämällä automaattisesti louhittua kirjaus-, kommentti- ja tehtävädataa. Vaikka tutkimuksessa keskitytään avoimen lähdekoodin projekteihin, toiveena on saada suuntaa antavia tuloksia myös yleisen tilanteen ennustamiseen. Tulosten pohjalta voidaan harkita jatkotutkimusten mielekkyyttä eri datalähteillä.

Tehtävien muutoksia tarkastellaan kokonaismäärän avulla, koska määriä on käytännöllistä käsitellä aikasarjamenetelmillä ja koska automaattisella louhinnalla voidaan helposti kerätä tehtävien kokonaismääriä sekä aukeamis- ja sulkeutumishetkiä. Tehtävän aukeamisella tarkoitetaan hetkeä, jolloin tehtävä lisätään tehtävienhallintajärjestelmään ja vastaavasti sulkeutuminen tarkoittaa hetkeä, jolloin tehtävä merkitään järjestelmässä valmiiksi. Tehtävien määrä on myös ennustamisen kannalta sopivalla abstraktiotasolla, koska ennustamisen epävarmuus ei silloin riipu yksittäisten tehtävien sattumanvaraisista muutoksista. Lisäksi kokonaismäärä on riittävän käytännönläheinen mittari ohjelmistoprojektin johtamiseen.

Tehtävät kuvaavat yksittäisenä dataluokkana hyvin projektin tilannetta niin etenemisen kuin sen kohtaamien haasteiden näkökulmasta. Kuten luvussa 3 esiteltiin, tehtäviä on onnistuttu ennustamaan erilaisilla menetelmillä sekä pelkän historiallisen tehtävädatan avulla että muita ennustavia muuttujia hyödyntämällä. Ei kuitenkaan tiedetä löytyykö ennustuspotentiaalia pelkästä valitunkaltaisesta yksinkertaisesta ja yleisluontoisesta datasta.

Virheraportit ovat kohdan 3.3 mukaisesti yksi kiinnostavimpia ennustuskohteita. Tämä tutkimus ei kuitenkaan keskity ainoastaan niihin, koska puhtaista virheraporteista ei ole tarjolla riittävästi dataa kattavaan vertailuun. Projektin kaikkien tehtävien määrä on siksi pääasiallisessa tarkastelussa. Projekteissa, joissa on riittävä määrä virheraportteja ja ominaisuustoivetehtäviä, tutkitaan lisäksi näiden muutoksia.

Kuten kohdassa 2.1 mainitaan, automaattinen datan kerääminen on oleellista, ettei ennustamisen potentiaalisissa soveltamistilanteissa kulu tarpeettoman paljon resursseja ennustuksen tekemiseen. Kirjaus-, kommentti- ja tehtävädataan rajoittuminen puolestaan mahdollistaa tulosten soveltamisen käytännössä kaikissa moderneissa ohjelmistoprojekteissa.

Tällainen ennustuspotentiaalin tutkiminen on arvokasta useammasta syystä. Sen avulla tiedetään mihin tilanteisiin kannattaa rakentaa optimoituja ennustusalgoritmeja, joiden kehittäminen itsessään on kallista ja hidasta, ja minne puolestaan työpanoksen sijoittaminen on turhaa. Ennustuspotentiaali valaisee ohjelmistotuotannon perusprosessien välisiä yhteyksiä ja toisaalta eroavaisuuksia projektien välillä. Suurimman käytännöllisen hyödyn saavat kohdan 3.2 mukaisesti resurssien sijoituspäätöksiä tekevät projektipäälliköt, jotka voivat ennakoida tietoisemmin resurssitarpeita tehtävien määrän mukaan.

4.2 Hypoteesit

Tutkimuksessa oletetaan yhden muuttujan vaikuttavan toiseen muuttujaan tai itseensä viiveellä, mikä toimii ennustamisen ja ennustuspotentiaalin etsimisen pohjana. Jos muuttujilla on viiveellinen ja riittävän voimakas yhteys, voidaan ennustavassa muuttujassa tapahtuneesta tai odotetusta muutoksesta ennakoida ennustettavan muuttujan muutosta. Seuraavat hypoteesit pohjautuvat tähän oletukseen.

Hypoteesi 1 Kirjaukset aiheuttavat viiveellä uusia aukeavia tehtäviä.

Hypoteesi 2 Kirjaukset, jotka muuttavat pientä määrää koodirivejä, aiheuttavat viiveellä tehtävien sulkeutumista.

Hypoteesi 3 Tehtävien kommentoinnista seuraa viiveellä tehtävien sulkeutumista.

Hypoteesi 4 Tehtävien aukeamisella on vaikutusta tulevien tehtävien aukeamiseen.

Hypoteesissa 1 oletetaan, että huomattava osa kirjauksista edustaa uusien ominaisuuksien kehittämistä ja että uusimmissa osissa ohjelmistoa on eniten löytymättömiä virheitä. Hypoteesia tukee myös huomio, jonka mukaan myös pienissä kirjauksissa voi syntyä uusia virheitä järjestelmään [34]. Tämä hypoteesi valittiin, koska kirjausten ja tehtävien välinen yhteys on hyvin perustavanlaatuinen osa ohjelmistotuotantoprosessia. Jos pelkkien kirjausten avulla on mahdollista ennustaa tehtävien aukeamista, voi käytännössä mikä tahansa ohjelmistoprojekti ennakoida resurssitarpeitaan ennen tehtävien kasautumista.

Ihanteellisessa tilanteessa voitaisiin olettaa, että kirjaukset, jotka muuttavat suurta määrää rivejä, aiheuttaisivat vielä voimakkaammin tehtävien aukeamista. Tätä ei kuitenkaan voida tutkia luotettavasti, koska esimerkiksi koodin uudelleenjärjestelystä ja välilyöntien muuttamisesta tabulaatioksi seuraa valtavia rivimuutoksia, joilla ei ole vaikutusta virheisiin. Näiden tyhjien muutosten suodattaminen ei mahdu tämän tutkimuksen laajuuteen. Työläämmällä esikäsittelyllä voisi saada tarkempia tuloksia, mutta silloin aineiston helppo louhittavuus kärsii.

Hypoteesissa 2 oletetaan, että pienet täsmämuutokset ovat virheenkorjauksia ja että tehtävät suljetaan vasta, kun kirjaus on tehty ja erikseen testattu toimivaksi. Tehtävien aukeamista voidaan pitää sulkeutumista arvokkaampana tietona resurssien sijoittelun näkökulmasta, mutta jossain vaiheessa ylimääräisistä resursseista pitää myös luopua. Jos projektipäällikkö voi ennakoida tehtävien sulkeutumista luotettavasti, hän voi luvata resursseja ajoissa myös muihin tarkoituksiin.

Toisin kuin hypoteesissa 1, pienten kirjausten kohdalla ei ole syytä olettaa, että datassa on mukana yllättävää kohinaa, joten on parempi yrittää etsiä tarkkaa yhteyttä tarkalla ennustavalla datalla kuin heikkoa yhteyttä kaikella kirjausdatalla. Ohjelmistotuotannon tutkimus tukee tätä hypoteesia: Marzban et. al. huomasivat, että pientä tiedostomäärää muuttavat kirjaukset ovat todennäköisemmin virheenkorjauksia [26]. Myös muissa tutkimuksissa on todettu pienten kirjausten olevan enemmän korjaavia kuin ominaisuuksia lisääviä [19][1].

Hypoteesissa 3 oletetaan, että tehtäviä kommentoidaan, koska niitä ollaan suorittamassa ja että kyseessä on keskustelua vaativa ongelma. Kun keskustelulla on selvitetty ongelman yksityiskohdat, voidaan tehtävä toteuttaa ja kirjata versiohallintajärjestelmään, mikä tapahtuu keskusteluun nähden viiveellä. Tällä hypoteesilla on samoja käytännön hyötyjä kuin hypoteesilla 2 liittyen resurssien luovuttamiseen. Tätä tutkimalla voidaan saada myös parempaa käsitystä kommentoinnin vaikutuksesta ohjelmistotuotantoprosessiin.

Hypoteesissa 4 oletetaan, etteivät tehtävät aukea sattumanvaraisesti, vaan aukeamistahdissa on jatkuvuutta. Aiemmissa tutkimuksissa on onnistuttu ennustamaan ja luokittelemaan tehtäviä erilaisista näkökulmista pelkkää historiallista tehtävädataa käyttämällä. On siis mielenkiintoista verrata löytyykö pelkistä tehtävien aukeamisista riittävän voimakasta sisäistä yhteyttä ennustamiseen. Tällä hypoteesilla on samoja käytännön hyötyjä kuin hypoteesilla 1 resurssien hallinnassa.

Hypoteesit 1 ja 3 ovat vahvimmat kohteet viiveellisen ennustuspotentiaalin löytymiselle, koska näissä kuvatut yhteydet ovat vahvimpia ja yleisluontoisimpia. Esimerkiksi uusia virheitä ei voi tulla ilman uusia kirjauksia ja kommentointi on

suora osoitus tehtävän käsittelystä, joka lopulta johtaa tehtävän sulkeutumiseen. Hypoteesia 2 heikentää oletus tietynlaisesta työnkulusta, jossa tehtävä suljetaan viiveellä vasta testauksen jälkeen. Hypoteesia 4 on vaikea arvioida käytännön ohjelmistokehityskokemuksen perusteella, koska yhden dataluokan sisäisen yhteyden voimakkuus ei näy yhtä selvästi ulospäin kuin kahden eri dataluokan vuorovaikutus. Kuten luvussa 3 todettiin, aiempien tutkimusten perusteella sisäistä yhteyttä pitäisi kuitenkin olla olemassa.

4.3 Kriteerit onnistumiselle

Tämän tutkimuksen tavoite on ennustuspotentiaalin tunnistaminen, joten onnistumista arvioidaan havaitun ennustuspotentiaalin vahvuuden perusteella. Koska ennustuspotentiaalia yritetään etsiä aikasarjamenetelmillä, on luonnollista arvioida potentiaalin vahvuutta tilastollisilla mittareilla. Näin ollen ennustavan ja ennustettavan muuttujan korrelaation laskettua voimakkuutta testataan satunnaisten heittelyiden varalta yleisesti käytetyn 95% luottamusvälin avulla [7].

Luottamusväli määrittää välin $[a, b]$, jolle tietty määrä satunnaisotoksesta laskettuja lukuja asettuu. Luottamusvälin keskipiste on tulosten odotusarvo, ja esimerkiksi normaalijakaumassa kahden keskihajonnan etäisyys odotusarvosta molempiin suuntiin luo rajat noin 95% luottamusvälille. Luottamusvälin avulla voidaan arvioida kuinka todennäköisesti laskettu tulos johtuu satunnaisista vaihteluista datassa eikä oikeasta korrelaatiosta. Tässä tutkimuksessa tuloksia pidetään siis alustavasti merkittävinä, jos korrelaatiokerroin on luottamusvälin ulkopuolella eli tulos johtuu vain 5% todennäköisyydellä sattumasta.

Koska 95% luottamusvälillä keskimäärin yksi korrelaatiokerroin kahdestakymmenestä näyttää merkittävältä, vaikka todellisuudessa merkittävyys johtuu sattumasta, ei jokaiseen alustavasti merkittävään korrelaatiokertoimeen voida luottaa. Laskettu korrelaatiokerroin liittyy aina tiettyyn viiveeseen tutkittujen aikasarjojen välillä. Mielivaltaisilta viiveiltä löytyvät alustavasti merkittävät tulokset hylätään, ellei viiveitä voida selittää ohjelmistotuotannon normaaleihin prosesseihin kuten iteratiiviseen projektikehykseen vetoamalla.

Pääsääntöisesti lyhyen viiveen korrelaatiot ovat uskottavampia kuin yksittäiset pitkän viiveen tulokset, vaikka ne olisivat tilastollisesti yhtä merkittäviä. Käytännön kokemuksen perusteella tutkimukseen valittujen ennustavien muuttujien vaikutus ennustettavaan muuttujaan tapahtuu ennemmin lyhyellä kuin pitkällä viiveellä. Tietyt ohjelmistokehitysmallit voivat kuitenkin olla poikkeus tähän, joten pitkän

viiveen yhteydet arvioidaan tapauskohtaisesti. Näissä tilanteissa iteroivan kehitysmallin käyttö yritetään varmistaa muulla datalla.

Jokainen hypoteesi testataan erikseen ennustuspotentiaalin löytämiseksi, eivätkä hypoteesien tulokset vaikuta toisiinsa onnistumista arvioitaessa. Lisäksi on mahdollista, että osasta tutkittavista projekteista löytyy ennustuspotentiaalia tietylle hypoteesille ja osasta ei. Koska minkään projektin kaikkiin taustatietoihin, kuten työnkulkuun, ei ole pääsyä ei ole mahdollista arvioida löytyykö tietynlaisista projekteista ennustuspotentiaalia enemmän kuin muista vai johtuvatko havainnot jostain muusta syystä. Tällaista tilannetta pidetään alustavasti lupaavana kohteena ennustamiselle, mutta potentiaalin varmistaminen vaatii jatkotutkimusta.

5. MENETELMÄ JA TYÖKALUT

5.1 Esikartoitus

Tutkimuksen idea sai inspiraationsa Tampereen teknillisellä yliopistolla tehdyistä tutkimuksista [28][29][20], joten oli luonnollista perehtyä ensimmäisenä tutkimuksissa käytettyihin työkaluihin ja aineistoihin. Näitä tutkimuksia varten kehitetty Visu-työkalu, josta on kerrottu lisää kohdassa 5.3, tarjosi joukon hyödyllisiä ominaisuuksia. Se mahdollisti syvempään tutkimustavoitteeseen tähtäämisen, koska uuden työkalun rakentamiseen ei kuluisi aikaa. Tarjolla ei ollut muita varteenotettavia valmiita vaihtoehtoja, joita voisi myös laajentaa helposti, joten Visu otettiin tutkimuksen päätyökalun pohjaksi.

Työkaluvalinta rajasi datalähteen GitHubiin, koska siihen oli valmis rajapinta ja se tarjosi valtavasti aineistoa. Tarkemmaksi aineistoksi oltaisiin haluttu yritysten ja muiden suurten organisaatioiden avoimen lähdekoodin projektit, jotta tutkimustulokset olisivat paremmin yleistettävissä teollisuuteen. Muita kriteerejä olivat tauoton kehittäminen, laajuus, keskittyminen tuotekehitykseen ja samanlainen projektityyppi ja kieli. Tällaisia projekteja ei kuitenkaan löytynyt riittävästi, joten alustavana kompromissina päädyttiin avoimen lähdekoodin tekstinkäsittelyohjelmiin. Lopullisesta aineistosta ja sen valintaperusteista kerrotaan tarkemmin kohdassa 6.

Visu-työkalun GitHubista keräämää dataa tarkastelemalla päädyttiin valitsemaan kirjaus-, kommentti- ja tehtävädata tutkimuksen keskiöön, koska nämä ovat merkittävimpiä tarjolla olevia ohjelmistokehityksen tapahtumia, niistä on tarjolla riittävästi dataa, lähes mistä tahansa ohjelmistoprojektista voi louhia tämän datan ja niillä on selkeät yhteydet toisiinsa. Samalla ennustuskohteeksi valikoitui tehtävät, koska niiden etenemisen ennustaminen muulla datalla on projektijohtamisen kannalta hyödyllisintä.

Tutkimusmenetelmä muuttui vasta päätutkimusta suoritettaessa aikasarjapohjaiseksi samalla, kun tutkimustavoite siirtyi ennustamisesta ennustuspotentiaalin tutkimiseen. Alkuperäinen tavoite eli ennustaminen vaikutti lopullisen tutkimustavoitteen valintaan, koska ennustaminen siirrettiin mahdolliseksi jatkotutkimuksen

aiheeksi. Tutkimustavoitteen muutoksesta kerrotaan lisää kohdassa 5.2.1. Laajuus saatiin esikartoituksen pohjalta rajattua alustavasti sopivaksi, ja seuraavaksi siirryttiin varsinaiseen tutkimukseen. Huomionarvoista on, ettei tässä vaiheessa oltu perehdytty vielä kunnolla aiheeseen liittyvään muuhun tutkimukseen.

5.2 Päätutkimus

5.2.1 Yleiskatsaus

Varsinainen tutkimus aloitettiin Visu-työkalun laajentamisella. Tässä vaiheessa tutkimustavoitteena oli vielä ennustaminen. Ensimmäisenä prioriteettina oli saada rakennettua kuvaajat, että niistä tehtävät huomiot voisivat ohjata tutkimusta hyödylliseen suuntaan jatkossa. Visulla sai jo valmiiksi haettua tehtävä- ja kirjausdataa, mutta kommentit ja tarkemmat tiedot kirjauksista puuttuivat. Laajennuksen jälkeen työkalua käytettiin valittujen projektien louhintaan ja datan jatkokäsittelyyn visualisoitavaan muotoon. Tämän jälkeen rakennettiin visualisointiominaisuudet, joista on kerrottu lisää kohdassa 5.3.2, ja luotiin dataluokista kuvaajia varten tehtävä-kirjaus, tehtävä-kommentti ja tehtävä-tehtävä parit.

Aluksi kuvaajista tutkittiin silmämääräisesti tehtävien aukeamista ja sulkeutumista suhteessa kirjausten ja kommenttien määriin ja muutosnopeuksiin, viivettä näiden välillä sekä tehtävien määrän trendiä. Tähän käytettyjä kuvaajia on esitelty jokaisen projektin kohdalla luvussa 6. Vastoin alkuperäistä oletusta eri käyrissä ei näkynyt silmämääräisellä tarkastelulla viivettä lyhyellä aikavälillä tulleiden kirjausten ja aukeavien tehtävien välillä. Tilanne oli sama myös muiden hypoteesien kohdalla.

Kuvaajiin yritettiin sovittaa lineaarista regressiota ja numeerista derivaattaa ennustuksen pohjatyöksi, mutta alustavat tulokset näissä osoittivat ettei ennustamiseen soveltuvaa viiveellistä korrelaatiota löytyisi näin yksinkertaisilla menetelmillä. Tutkittavat dataluokkaparit seurasivat toistensa muutoksia olemattomalla viiveellä. Alkuperäinen ajatus ennustamisesta laitettiin väliaikaisesti syrjään ja tutkimuksessa siirryttiin kirjallisuusselvitykseen laajemman näkökulman saamiseksi.

Kirjallisuusselvityksessä keskityttiin ohjelmistotuotantodataan, avoimen lähdekoodin projekteihin datalähteenä, yleisen informaation ja erityisesti ohjelmistotuotannon visualisointiin sekä yleiseen ennustamiseen ja erityisesti ohjelmistotuotantoon liittyvään ennustamiseen. Osoittautui, että ohjelmistotuotannon tutkijat ovat ennustaneet muun muassa ohjelmistojen virheitä, laajuutta

ja hintaa. Käytetyt ennustusmenetelmät olivat tämän diplomityön laajuuteen nähden monimutkaisia ja toisaalta jo monimuotoisesti käsiteltyjä erilaisissa tutkimuksissa, joten tutkimukselle oli kannattavaa miettiä toisenlainen näkökulma.

Kirjallisuusselvityksessä ei löytynyt visualisointiin pohjautuvia aikasarjamenetelmiä, joissa ennustetaan yhden muuttujan ja viiveen avulla toisen muuttujan kehitystä käyttäen tehtävä-, kirjaus- ja kommenttidataa. Tutkimuksen kohde oli luonnollista siirtää ennustamisesta ennustuspotentiaalin etsimiseen, koska moderneilla tekniikoilla tapahtuva ennustaminen monimutkaisine algoritmeineen olisi mennyt yli tutkimuksen tavoitelaajuudesta. Tutkimustilanteeseen sopivat tekniikat löytyivät ennustamiseen perehtymisen yhteydessä ristikorrelaatiosta ja impulssivastefunktiosta, joista on kerrottu lisää kohdassa 5.2.2.

Vaikka hypoteesien mukaisia korrelaatioita ei löytynyt silmämääräisellä tutkimisella, pidettiin mahdollisena, että matemaattinen lähestymistapa pystyisi löytämään yhteyksiä, jotka jäävät silmältä piiloon. Toisaalta matemaattisella menetelmällä voi saada myös suuremman varmistuksen mikäli hypoteesien mukaisia korrelaatioita ei löydy. Menetelmän ja tutkimuskohteen iteroinnin jälkeen Visu-työkaluun lisättiin uudet tutkimuksen tarvitsemat ominaisuudet ja varsinainen tutkimus voitiin suorittaa.

5.2.2 Matemaattinen menetelmä

Tutkimuksessa käytetään kahta signaalinkäsittelyn menetelmää ennustuspotentiaalin etsimiseen: ristikorrelaatiota ja impulssivastefunktiota. Ristikorrelaatiolla tutkitaan kahden aikasarjan korrelaatiota ajan suhteen toisiinsa nähden siirrettynä [7]. Toisin sanoen ristikorrelaatio kertoo miten yhden aikasarjan muutokset näkyvät viiveellä toisessa aikasarjassa. Impulssivastefunktiolla puolestaan tutkitaan miten yksittäinen muutos yhdessä aikasarjassa vaikuttaa toisessa aikasarjassa usealla myöhemmällä ajanhetkellä [7]. Se kertoo siis miten yhden muutoksen vaikutus jakautuu useammalle ajanhetkelle ikään kuin sormella vedetty mustetahra paperilla. Siinä missä ristikorrelaatio vertaa vain yhtä ajanhetkeä toiseen kaikilla ajanhetkien yhdistelmillä, impulssivastefunktio kertoo miten kauaskantoisia vaikutuksia yksittäisillä muutoksilla on.

Mukaillen Chatfieldin[7] kuvausta ristikorrelaatiosta, oletetaan, että käsiteltävänä on kaksi N mittaista äärellistä, diskreettiä ja ajassa osittain sattumanvaraisesti etenevää prosessia, joista mitataan tasaisin aikavälein aikasarjat X_t ja Y_t. Oletetaan ettei mitatulla aikavälillä tapahdu muutosta aikasarjaprosessien keskiarvoissa, ristikovarianssifunktiossa ja autokovarianssifunktioissa, jotka kuvaavat kuinka

voimakkaasti kunkin aikasarjan aiemmat arvot vaikuttavat sen nykyisiin arvoihin. Ristikorrelaatio rakentuu seuraavanlaisia merkintöjä käyttäen:

Odotusarvofunktio E, jonka tulos on diskreetin tasaisen jakauman tilanteessa aikasarjan keskiarvo μ

$$E(X_t) = \sum_{t=1}^{N} \frac{X_t}{N} = \mu_x; \qquad E(Y_t) = \sum_{t=1}^{N} \frac{Y_t}{N} = \mu_y$$

Ristikovarianssifunktio

$$Cov(X_t, Y_{t+k}) = E[(X_t - \mu_x)(Y_t - \mu_y)] = \gamma_{xy}(k)$$

Ristikovarianssifunktion tulokset riippuvat aikasarjojen mittayksiköistä, joten yleisluontoista tulkintaa varten tulokset pitää standardoida. Tästä saadaan ristikorrelaatiofunktio

$$\rho_{xy}(k) = \frac{\gamma_{xy}(k)}{\sigma_x \sigma_y}$$

missä σ_x ja σ_y ovat aikasarjojen X_t ja Y_t keskihajonnat. Ristikorrelaatio mittaa korrelaatiota aikasarjojen ajanhetkien X_t ja Y_{t+k} välillä. Sillä on yhtenä oleellisena ominaisuutena

$$|\rho_{xy}(k)| \leq 1$$

minkä ansiosta tulosten tilastollista merkittävyyttä on helpompi arvioida. Kendall et al. [24] Chatfieldin [7] mukaan voidaan osoittaa, että itsenäisen satunnaismuuttujan odotusarvo ja varianssi ovat

$$E(r_k) \simeq -1/N; \qquad Var(r_k) \simeq 1/N$$

Satunnaismuuttujan keskihajonta on sen varianssin neliöjuuri, joten odotusarvon molemmille puolille kahden keskihajonnan verran ulottuvaksi väliksi saadaan

$$-1/N \pm 2/\sqrt{N}$$

mikä on noin 95% luottamusväli. Tämä voidaan vielä approksimoida yleisesti käytettyyn muotoon

$$\pm 2/\sqrt{N}$$

jota käytetään tutkimuksen tulosten tilastollisen merkittävyyden arviointiin.

Ristikorrelaatio ei kuitenkaan anna luotettavia tuloksia, jos tutkittavissa aikasarjoissa esiintyy autokorrelaatiota eli korrelaatiota aikasarjan nykyisten ja sen omien vanhojen arvojen suhteen. Tämän vuoksi aikasarjoista pitää puhdistaa mah-

dolliset autokorrelaatiot pois sovittamalla kumpaankin aikasarjaan erikseen korkean asteen autoregressiivinen malli ja vähentämällä tämä aikasarjasta. Kaava asteen n autoregressiiviselle mallille on

$$y_t - \mu = \phi_1(y_{t-1} - \mu) + ... + \phi_n(y_{t-n} - \mu) + \epsilon_t$$

missä μ on keskiarvo, ϕ_n on mallin arvioitu parametri n ja ϵ_t on mallin jäännösvirhe eli residuaali. Kun aikasarjasta vähennetään malli, jäljelle jää residuaali, jossa ei ole autokorrelaatiota. Tutkimuksessa käytetään valmista kirjastoa[25] autoregressiivisten mallien parametrien arvioimiseen. Puhdistetuista aikasarjoista voi laskea luotettavan ristikorrelaation.

Impulssivastefunktio ottaa järjestelmätason näkökulman kahden aikasarjan välisen yhteyden arvioimiseen. Toista aikasarjaa käsitellään dynaamisen lineaarisen järjestelmän syötteenä ja toista ulostulona. Dynaamisuus tarkoittaa tässä yhteydessä, että minkä tahansa ajanhetken ulostulo voi riippua myös vanhoista syötteen arvoista. Impulssivastefunktio kertoo järjestelmän hypoteettisen ulostulon, kun järjestelmään syötetään yksikköpulssi ajanhetkellä $t = 0$.

Impulssivastefunktio lasketaan ristikorrelaation avulla, mutta tutkittavat aikasarjat puhdistetaan eri tavalla kuin normaalisti ristikorrelaatiota laskettaessa. Syötteenä käsiteltävään aikasarjaan sovitetaan taas korkean asteen autoregressiivinen malli, jonka avulla syötteestä puhdistetaan autokorrelaatio pois. Ulostulona käsiteltävään aikasarjaan ei kuitenkaan soviteta omaa mallia, vaan siitä vähennetään sama malli, jolla syöte puhdistettiin. Tämän tuloksena ulostulosta ei jää jäljelle pelkkää jäännösvirhettä. Suodatuksen jälkeen aikasarjoista lasketaan ristikorrelaatio ja tulokseksi saadaan impulssivaste, jonka tilastollista merkittävyyttä voidaan arvioida samalla tavalla kuin normaalia ristikorrelaatiota. [7]

5.3 Visu-Forecaster työkalu

5.3.1 Esittely ja tausta

Visu-Forecaster on Tampereen teknillisellä yliopistolla kehitetyn Visu-työkalun laajennos, joka kehitettiin tätä diplomityötä varten. Visun kehitys alkoi vuonna 2014 TEKES-TIVIT:n Need-for-Speed projektia varten ja sen pääkehittäjiä ovat Antti Luoto, Anna-Liisa Mattila ja Henri Terho. Visu on Node.js pohjainen web-sovellus, jolla voi visualisoida kirjaus- ja tehtävädataa. Visualisoitava data voidaan hakea helposti lisättävien rajapintakuvausten perusteella eri datalähteistä kuten GitHubista,

minkä jälkeen data esikäsitellään ja tallennetaan paikalliseen tietokantaan. Visun tietokannan dataa voidaan hakea erillisen REST-rajapinnan avulla ja myös visualisaatiot hakevat datansa tämän kautta.

Visu tallentaa datan teennöksinä ja tapahtumina. Esimerkiksi tehtävä tallennetaan yhtenä teennöksenä, johon liittyy vähintään aukeamistapahtuma, mutta siihen voi liittyä myös sulkeutuminen, uudelleen avaaminen tai kommentti. Kirjaukset tallennetaan pelkästään tapahtumina.

Visu-Forecaster lisää Visuun ominaisuudet kommenttien ja tarkempien kirjaustietojen hakemiselle GitHubista, oman esikäsittelijän ja esikäsiteltyä dataa visualisoivan näkymän, jossa tehdään myös datan analysointi. Visu-Forecaster hyödyntää Visun web-palvelinta, tietokantaa, REST-rajapintaa ja datan keräämistä, mutta ei sen valmiita visualisaatioita ja analysaattoreita.

Visua on käytetty muissakin tutkimuksissa. Hylli et al. loivat yhtenäisen tietokantamallin tehtävädatalle ja todensivat tätä hakemalla Visulla tehtävädataa eri datalähteistä helposti lisättävien rajapintakuvausten perusteella [20]. Tutkimuksessa käytettiin Mattilan et al. kehittämää yhtenäistä ohjelmistotuotannon datamallia, jota myös he todensivat Visun avulla [29]. Lisäksi ohjelmistotuotannon prosessien ongelmia on tutkittu Visun avulla vertaamalla projektien suunniteltua prosessia visualisoituun toteumaan [30].

Keim luokittelee visualisointityökalut datatyypin, visualisointitekniikan ja interaktiotekniikan mukaan [23]. Tällä luokittelulla Visu-Forecaster visualisoi yksiulotteista aikasarjadataa kaksiulotteisiksi näkymiksi ja tarjoaa tarkennus- ja suodatusinteraktiota. Voinean et al. mukaan visuaalisen analyysityökalun kriittiset vaatimukset ovat visualisaation yksinkertaisuus, sen yleiskäyttöisyys ja integraatio datalähteeseen [41]. Visu-Forecaster vastaa näihin vaatimuksiin tyydyttävästi. Tarjotut visualisaatiot ovat yksinkertaisia ja helppoja ymmärtää. Visualisointitekniikkaa voi soveltaa mihin tahansa dataan, joskin data täytyy ensin erikseen esikäsitellä, mikä ei tapahdu automaattisesti. Integraatio GitHubiin on olemassa, mutta datan louhiminen ja päivittäminen pitää käynnistää manuaalisesti.

5.3.2 Datan esikäsittely ja visualisointi

Visu-Forecaster valitsee GitHubin tarjoamasta datasta visualisoitavan osajoukon ja tallentaa sen tietokantaan. Kaikkiin dataluokkiin eli kirjauksiin, kommentteihin ja tehtäviin liittyvistä tapahtumista otetaan talteen aikaleimat ja kahdesta jälkimmäisestä myös tieto siitä, liittyykö tapahtuma liitospyyntöön vai ei. Kirjauksista otetaan

talteen muutettujen rivien määrä ja tehtävistä sen kulloinenkin tila eli aukesiko tehtävä vai sulkeutuiko se. Monet GitHubia projektinhallintaan käyttävät projektit merkitsevät melko selkeällä tavalla tehtäville tyyppejä. Tutkimusta varten jokaisen projektin tehtävätyypit jaettiin kolmeen luokkaan: kaikki tehtävät, toiveet uusiksi ominaisuuksiksi ja virheraportit. Nämä merkinnät otettiin myös talteen.

Dataa esikäsiteltiin kumulatiivisia ja ryhmiteltyjä kuvaajia varten. Kumulatiiviset kuvaajat esittävät tutkittavan muuttujan kehitystä ajassa ja antavat yleiskatsauksen projektin luonteesta. Kumulatiivisissa kuvaajissa on useita käyriä, jotka on koottu erilaisten esisuodattimien avulla. Liitospyyntöjen vaikutusta ei osattu arvioida etukäteen, joten tehtävistä ja kommenteista on erotettu käyrät kaikille tapahtumille sekä tapahtumille, jotka eivät liity liitospyyntöihin. Tehtävistä ja niihin liittyvistä kommenteista on myös erotettu eri käyriksi kaikki data, toiveet uusiksi ominaisuuksiksi ja virheraportit. Näistäkin on erikseen liitospyynnöttömät käyrät. Kumulatiivisia kuvaajia on auenneille tehtäville, sulkeutuneille tehtäville, auki oleville tehtäville, sulkeutuneille tehtäville ja kommenteille, auenneille tehtäville ja kirjauksille, sulkeutuneille tehtäville ja pienille kirjauksille sekä sulkeutuneille tehtäville ja kaikille kirjauksille.

Ryhmitellyissä kuvaajissa tapahtumat on koottu viikkotasolle absoluuttisina lukuina, koska käytetty matemaattinen menetelmä vaatii etteivät aikasarjojen keskiarvot muutu järjestelmällisesti ajan kuluessa. Viikkotasoa pidettiin pienimpänä mielekkäänä ennustamisen aikajänteenä. Auki olevia tehtäviä lukuunottamatta kaikista kumulatiivisista kuvaajista on olemassa myös ryhmitelty versio.

Jokaisen projektin visualisaatioita voi tarkentaa ja selata Visu-Forecasterissa interaktiivisesti. Turhia käyriä voi piilottaa, jos halutaan yksityiskohtaisempaa tietoa tietystä ajankohdasta. Kuvaajista on mahdollista valita käyriä ja laskea tietyltä aikaväliltä lineaarinen regressio, korrelaatio ja kulmakerroin. Näitä käytettiin projektien alustavaan tutkimiseen. Lisäksi näkymistä voidaan valita kaksi käyrää ja laskea näiden ristikorrelaatio ja impulssivastefunktio kohdassa 5.2.2 esitetyllä tavalla, joka on tutkimuksen pääasiallinen kiinnostuksen kohde.

6. TUTKITUT PROJEKTIT

6.1 Yleispiirteet ja valintaperusteet

Tutkimuksessa käsiteltävä aineisto koostuu avoimen lähdekoodin projekteista, joista seitsemän tuottaa tekstinkäsittelyohjelmaa ja kolme ei. Kaikissa projekteissa on käytössä GitHubin tarjoama tehtävienhallintajärjestelmä. Projektien koot vaihtelevat paljon: siinä missä pienimmässä projektissa on n. 800 kirjausta, suurimmassa on n. 9000. Visu-Forecaster rajoitti osaltaan valittujen projektien kokoa, koska yli 10 000 kirjausta sisältävien projektien louhintaan kuluu kohtuuttomasti aikaa. Tämä johtuu siitä, että kommenttidata haetaan pienissä erissä yksitellen, ja kommenttidataa on suurissa projekteissa valtavia määriä. Tehtävien kokonaismäärät vaihtelevat muutamasta sadasta useaan tuhanteen ja vastaavasti kommenttien määrät vaihtelevat paljon. Osassa projekteista avoimien tehtävien määrä on hallinnassa ja osassa pahasti paisunut.

Myös projektien iät ja kehitystahdit vaihtelevat. Vanhin projekti on aloitettu vuonna 2007 ja nuorin vuonna 2014. Kaikki projektit eivät ole enää aktiivisessa kehityksessä tai kehityksessä on ollut taukoja, mutta kaikissa on vähintään yksi kuukausia kestänyt yhtäjaksoinen kehityskausi. Kahta poikkeusta lukuunottamatta projekteissa on pääasiallisia kehittäjiä 1-3 henkeä, minkä lisäksi liitospyyntöjä on tehnyt kymmenet muut kehittäjät. Projektit on kehitetty eri teknologioilla ja kielillä, ja kaikista projekteista on julkaistu toimiva tuote.

Tekstinkäsittelyohjelmat valittiin tutkittavaksi useista syistä. Samanlainen ohjelmatyyppi ja tuotekehitysluonne nähtiin tulosten vertailukelpoisuutta edistävänä tekijänä. GitHubissa on myös riittävästi kilpailevia ja riittävän pitkään kehitettyjä tekstinkäsittelyohjelmia, että niiden välillä voidaan tehdä vertailua. Löydetyissä projekteissa on ollut riittävän tiheä kehitystahti ja riittävästi aineistoa tilastolliseen tutkimiseen. Suurin osa ohjelmista on aktiivisessa kuluttajaluontoisessa käytössä, joten voidaan olettaa, että virheitä ilmoitetaan normaalin kuluttajatuotteen tapaisesti. Ne myös muistuttavat teollisuudessa kehitettäviä käyttäjäkeskeisiä tuotteita paremmin kuin moni muu avoimen lähdekoodin projekti.

Tutkimus keskittyy pääasiassa kaikkien tehtävien ennustuspotentiaalin tutkimiseen, koska suurempi datamäärä tarjoaa luotettavampia tuloksia. Aineistoon jouduttiin lisäämään kolme tekstinkäsittelystä poikkeavaa ohjelmistoprojektia, koska alkuperäisessä aineistossa ei ollut riittävää määrää tyyppimerkittyjä tehtäviä pelkkien virheraporttien tai ominaisuustoiveiden ennustuspotentiaalin tutkimiseen. Kolme lisättyä projektia valittiin kirjausmäärän, tasaisen kehitystahdin ja tyyppimerkittyjen tehtävien määrän perusteella. Ne eivät edusta samaa tuotetyyppiä.

On hyvä huomioida, että tehtävien tyyppimerkkaus ei ole täysin luotettavaa, koska dataa suodattamaan valitut tehtävämerkinnät on valittu käsin tulkitsemalla tehtävämerkintöjen nimiä. Ei voida myöskään tietää miten täsmällisesti projektit käyttävät tehtävämerkintöjä. Ihanteellisessa tilanteessa tutkittaisiin erikseen eri tehtävätyyppien ennustuspotentiaaleja tarkempien ja paremmin käytännön projektijohtamisessa hyödynnettävien tulosten takia. Tässä tilanteessa joudutaan kuitenkin tyytymään tehtävätyyppien osalta suuntaa antaviin tuloksiin, joista voi olla hyötyä jatkotutkimuksia harkittaessa.

Projekien hallinnointitietoihin ei ole pääsyä, joten esimerkiksi mahdollista syklisyyttä kehityksessä ei voida varmistaa. Ei ole myöskään varmuutta miten johdonmukaisesti GitHubin projektinhallintatyökaluja on käytetty. Etenkin pienemmissä projekteissa on mahdollista, etteivät GitHubiin luodut tehtävät edusta kaikkia tehtäviä, jos esimerkiksi yksittäinen pääkehittäjä on pitänyt erillistä listaa itse löytämistään virheistä ja suunnitelluista ominaisuuksista.

Avoimen lähdekoodin projektit eivät vastaa teollisuuden projekteja, joihin tulokset olisi hyödyllistä yleistää. Esimerkiksi avoimen lähdekoodin projekteissa yksittäisiä liitospyyntöjä voi tulla tuntemattomilta kehittäjiltä. Ilman aineistoon liittyviä puutteita tuloksista voitaisiin saada tarkempia. Esikartoituksessa kuitenkin todettiin, ettei GitHubista löytynyt riittävää määrää samankaltaisia ja muilta osin sopivia yritysten tai muiden suurten organisaatioiden projekteja tutkimusta varten.

6.2 Projektit

6.2.1 Esitystapa

Seuraavissa kohdissa esitellään tutkimukseen valitut kymmenen projektia. Tulokset ovat projektikohtaisia, joten yksittäisten projektien erityispiirteiden ymmärtäminen on hyödyllistä tuloksia tulkittaessa. Jokaisesta projektista on esitetty samat tiedot, joten tiiviyden ja vertailukelpoisuuden vuoksi tiedot on esitetty aina samalla tavalla.

Ensimmäisenä jokainen projekti esitellään taustatietojen, iän, GitHub tunnusten ja kehittämisen näkökulmista. Toisena esittelyssä on taulukko, johon on kerätty tietoa louhituista datamääristä. Taulukoissa on seuraavat tiedot:

Kirjaukset	Kaikki kirjaukset.
Pienet kirjaukset	Kaikki kirjaukset, jotka muuttavat yhteensä alle 15 riviä projektitiedostoissa.
Kommentit	Kaikki kommentit, jotka eivät liity liitospyyntöihin.
Auenneet tehtävät	Kaikki auenneet tehtävät, jotka eivät ole liitospyyntöjä.
Auenneet virheet	Kaikki tehtävät, jotka on merkitty virheraporteiksi ja eivät ole liitospyyntöjä. Projektin virheraportteihin liittyvät merkinnät löytyvät seuraavasta taulukosta.
Auenneet toiveet	Kaikki tehtävät, jotka on merkitty ominaisuustoiveiksi ja eivät ole liitospyyntöjä. Projektin ominaisuustoiveisiin liittyvät merkinnät löytyvät seuraavasta taulukosta.

GitHub käsittelee liitospyyntöjä tehtävinä, mutta lisää liitospyyntötehtäville erillisen totuusarvon tiedoksi asiasta. Liitospyynnöt on suodatettu pois esiteltävistä datamääristä, koska dataa myös tutkitaan ilman liitospyyntöjä. Datan tutkimisesta ja liitospyyntöjen suodattamisesta on kerrottu lisää kohdassa 7.1.

Kolmantena esittelyssä on taulukko, jossa on esitetty projektin virheraporttien ja ominaisuustoiveiden merkinnät. Nämä merkinnät on valittu käsin kaikkien merkintöjen joukosta ja niihin liitetyt tehtävät on tulkittu edustavan virheraportteja ja ominaisuustoiveita. Projektissa voi olla myös muita merkintöjä, joita ei tulkittu virheraportteihin tai ominaisuustoiveisiin liittyväksi. Viimeisenä projektin etenemistä esitellään Visu-Forecasterin tuottaman kuvaajan avulla. Kuvaajassa näkyy ajan funktiona kirjaukset ja kaikki auenneet tehtävät ilman liitospyyntöjä.

6.2.2 Spacemacs

Spacemacs on beta-vaiheessa oleva yhteisöllisesti kehitetty versio Emacs-tekstinkäsittelyohjelmasta. Projekti on aloitettu 16.12.2012 ja se löytyy GitHubista nimellä spacemacs käyttäjältä syl20bnr. Tehtyjen kirjausten perusteella projektia kehittää n. 2 pääkehittäjää ja n. 520 avustajaa. Projektia kehitetään Emacs Lispillä. Taulukkoon 6.1 on kerätty tiedot louhitusta datasta.

Taulukko 6.1 *Projektista Spacemacs louhitun datan määrä.*

Kirjaukset	Pienet kirjaukset	Kommentit	Auenneet tehtävät	Auenneet virheet	Auenneet toiveet
5500	3300	18000	3100	1000	75

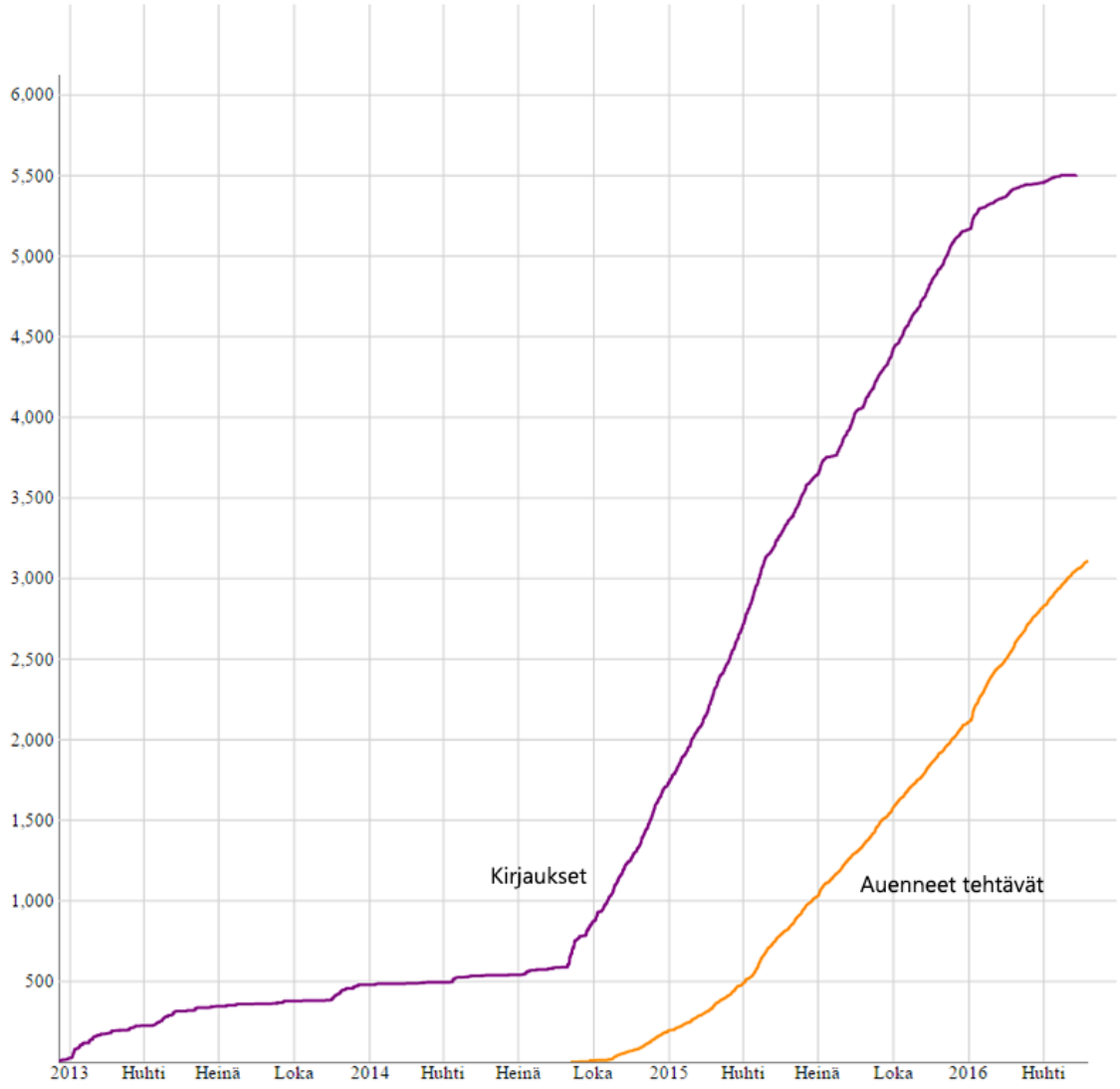
Spacemacs on kirjausten, tehtävien ja kommenttien puolesta suurimpia tutkimukseen valittuja projekteja. Auenneita virheraportteja on hyvin paljon, mutta ominaisuustoiveita mitättömästi. Taulukosta 6.2 löytyy projektin merkintätiedot.

Taulukko 6.2 *Projektin Spacemacs virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät.*

Virheet	Ominaisuustoiveet
Bug,tracker - Bug :-(<=(- UBO -)=> Regression :-(Unresponsive Need repro steps Not reproducible x_x	Feature request Proposal Enhancement

Spacemacsissa on enemmän erilaisia merkintöjä kuin muissa projekteissa ja osa niistä on enemmän kuvaavia kuin luokittelevia. Kuvaukset on kuitenkin mahdollista yhdistää sopivaan tehtävätyyppiin. Merkinnöistä erikoisin on <=(- UBO -)=>, joka on lyhenne sanoista Unidentified Bug Object eli tunnistamaton virhe-esine. Kuvassa 6.1 on esitetty projektin kehitys ajassa.

Hitaan puolentoista vuoden kehityksen jälkeen Spacemacsia on kehitetty hyvin lineaarisesti eteenpäin. Vuonna 2016 kirjaustahti on hidastunut, mutta aukeavien tehtävien määrä ei ole hidastunut yhtä voimakkaasti.



Kuva 6.1 *Projektin Spacemacs kirjaukset ja auenneet tehtävät ajan funktiona.*

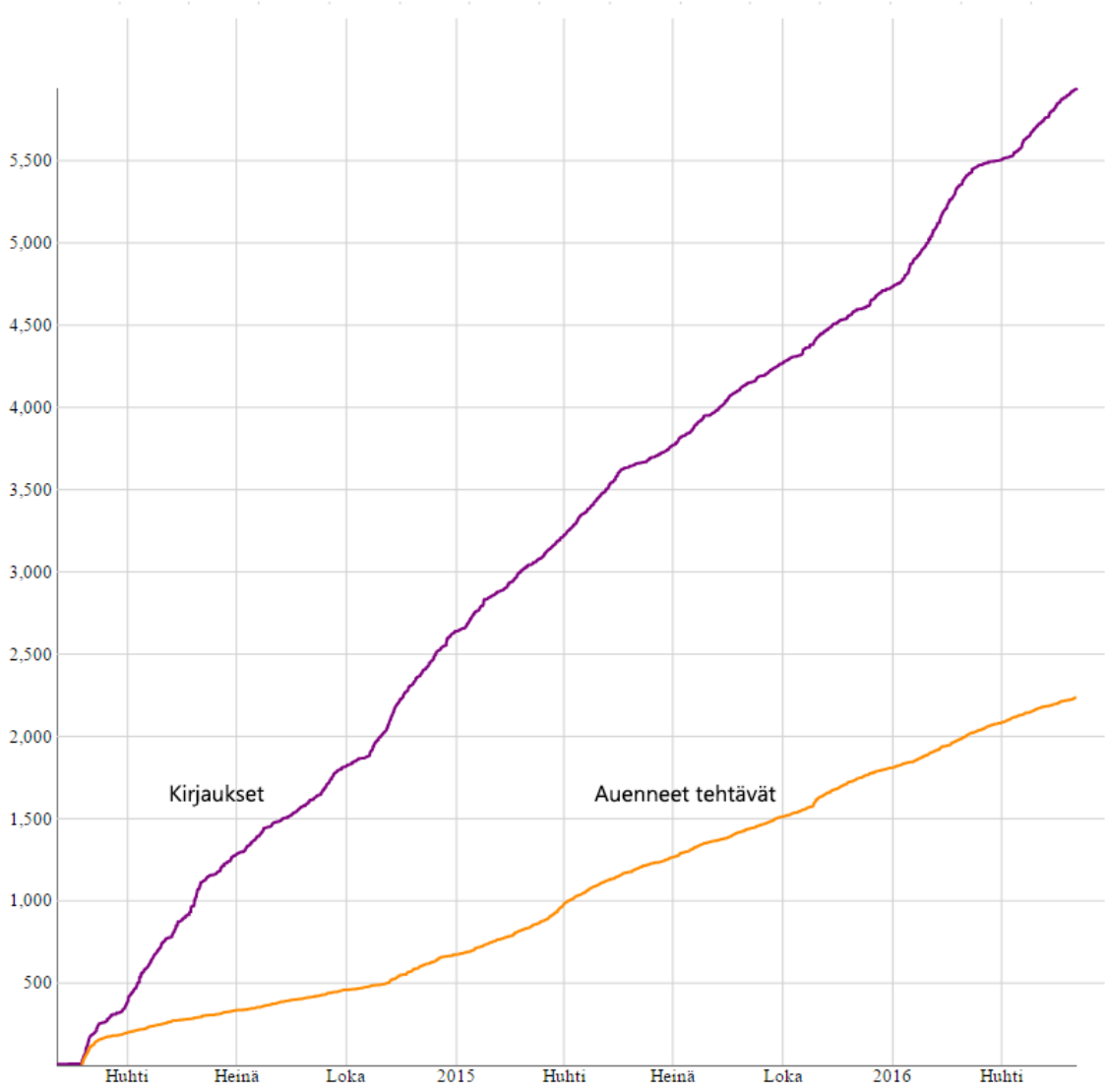
6.2.3 Neovim

Neovim on yhteisöllisesti kehitetty versio Vim-tekstinkäsittelyohjelmasta. Projektin tavoitteena on parantaa alkuperäisen Vimin laajennettavuutta ja käytettävyyttä. Projektista on tehty yksi vakaa julkaisu. Neovim on aloitettu 26.1.2014 ja se löytyy GitHubista nimellä neovim käyttäjältä neovim. Tehtyjen kirjausten perusteella projektia kehittää n. 10 pääkehittäjää ja n. 280 avustajaa. Projektia kehitetään C-kielellä. Taulukkoon 6.3 on koottu tiedot louhinnasta.

Taulukko 6.3 *Projektista Neovim louhitun datan määrä.*

Kirjaukset	Pienet kirjaukset	Kommentit	Auenneet tehtävät	Auenneet virheet	Auenneet toiveet
5900	2500	17500	2200	450	150

Neovim on suurimpia tutkimukseen valittuja projekteja kirjausten, tehtävien ja kommenttien puolesta. Virheraportteja on paljon, mutta ominaisuustoivetehtäviä melko vähän suhteessa kirjausten määrään. Taulukosta 6.4 löytyy projektin merkintätiedot. Projektin merkinnät ovat selkeät ja helposti tulkittavat. Kuvassa 6.2 on esitetty projektin kehitys ajassa.



Kuva 6.2 *Projektin Neovim kirjaukset ja auenneet tehtävät ajan funktiona.*

Taulukko 6.4 *Projektin Neovim virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät.*

Virheet	Ominaisuustoiveet
bug	enhancement
bug-vim	
bug-security	
blocked:needs-repro	

Projektia on kehitetty käytännössä alusta asti hyvin tasaisella tahdilla eikä tästä löydy ensimmäisen viikon lisäksi poikkeuksia.

6.2.4 TextMate

TextMate on graafinen tekstinkäsittelyohjelma, josta on julkaistu useita vakaita versioita. Projekti on aloitettu 24.6.2012 ja se löytyy GitHubista nimellä textmate käyttäjältä textmate. Tehtyjen kirjausten perusteella projektia kehittää 1 pääkehittäjä ja n. 70 avustajaa. Projektia kehitetään C++ kielellä. Taulukkoon 6.5 on koottu tiedot louhinnasta.

Taulukko 6.5 *Projektista TextMate louhitun datan määrä.*

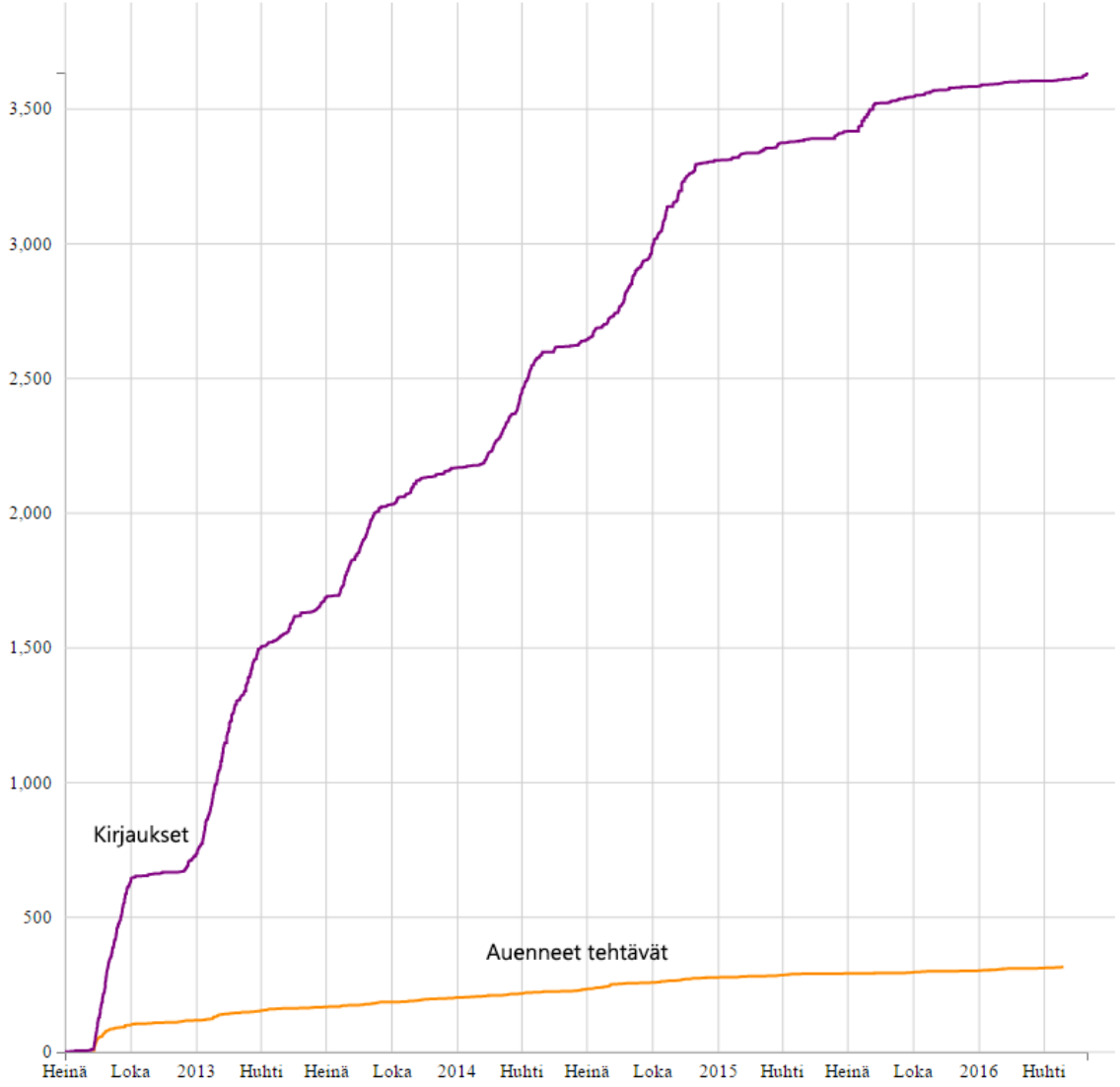
Kirjaukset	Pienet kirjaukset	Kommentit	Auenneet tehtävät	Auenneet virheet	Auenneet toiveet
3600	2100	900	300	0	0

TextMate on kirjausten puolesta keskikokoinen projekti, mutta auenneita tehtäviä ja kommentteja on suhteessa vähän. Virheraportteja ja ominaisuustoivetehtäviä ei ole ollenkaan. Tämä voi johtua siitä, että projekti on pääasiassa yhden henkilön ylläpitämä eikä projektinhallintatyökaluille ole tarvetta. Taulukosta 6.6 löytyy projektin GitHubissa olevat merkintätiedot.

Taulukko 6.6 *Projektin TextMate virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät.*

Virheet	Ominaisuustoiveet
bug	enhancement

Erikoisesti projektilla on perusmerkinnät virheraporteille ja ominaisuustoiveille vaikka louhinnasta ei saatu yhtään näiden tyyppistä tehtävää. Kuvassa 6.3 on esitetty projektin kehitys ajassa.



Kuva 6.3 *Projektin TextMate kirjaukset ja auenneet tehtävät ajan funktiona.*

Projektin kehitys on ollut syklistä, mutta tämä voi yhtä hyvin johtua ylläpitäjän omista aikatauluista ja kiireistä kuin iteratiivisesta projektirakenteesta. Avautuneiden tehtävien määrä on kehittynyt hitaasti, mutta lineaarisesti, joten suuri osa niistä voi olla käyttäjien luomia.

6.2.5 Lime

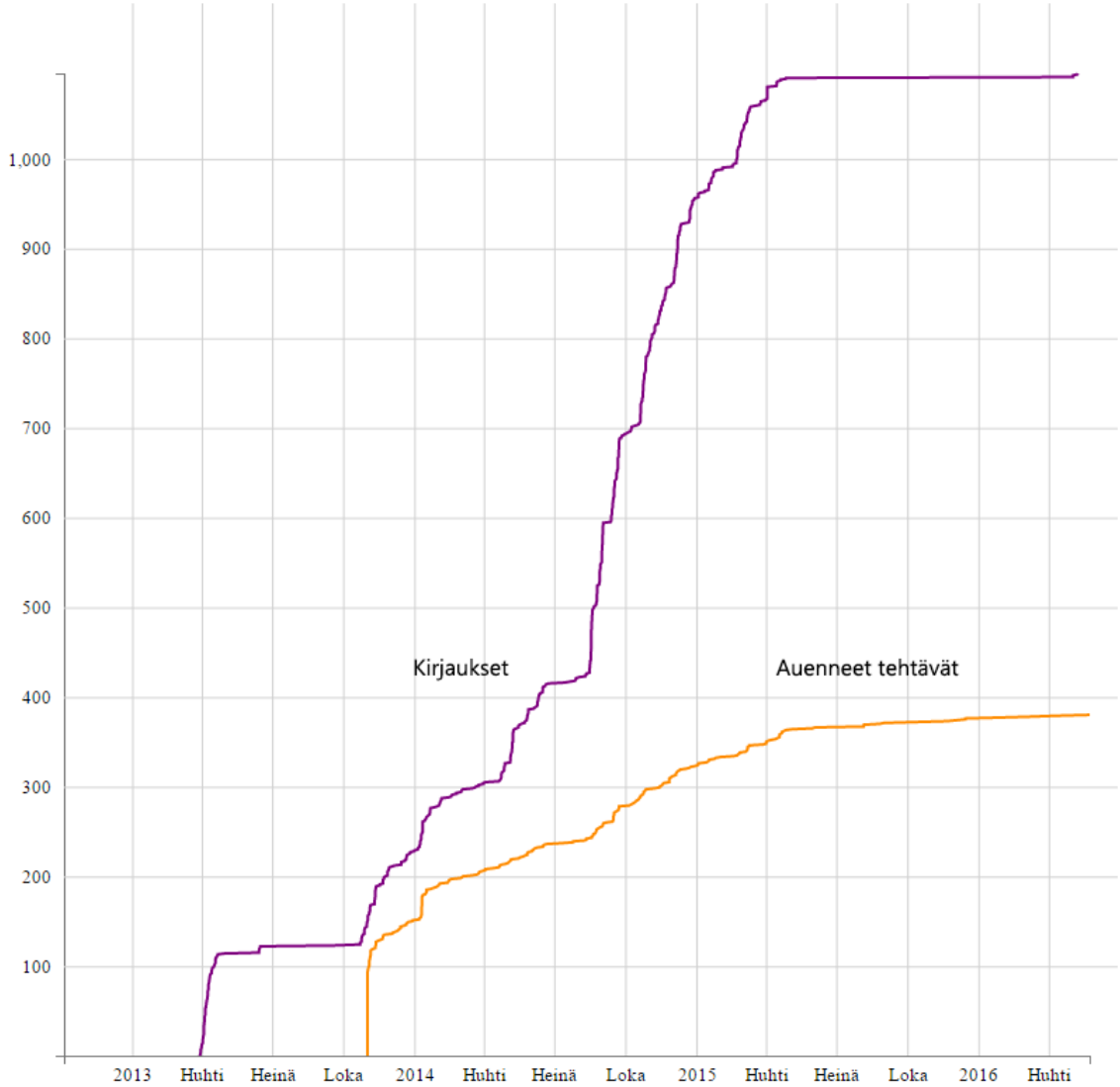
Lime on avoimen lähdekoodin vaihtoehto Sublime Text nimiselle tekstinkäsittelyohjelmalle ja vaikka sitä voi jo käyttää, se ei ole vielä valmis korvaamaan esikuvaansa. Projekti on aloitettu 24.3.2013 ja se löytyy GitHubista nimellä lime käyttäjältä limetext. Tehtyjen kirjausten perusteella projektia kehittää n. 3 pääkehittäjää ja n. 50

avustajaa. Projektia kehitetään Go-kielellä. Taulukkoon 6.7 on kerätty tiedot louhitusta datasta.

Taulukko 6.7 *Projektista Lime louhitun datan määrä.*

Kirjaukset	Pienet kirjaukset	Kommentit	Auenneet tehtävät	Auenneet virheet	Auenneet toiveet
1100	400	1600	300	25	75

Lime on pienimpiä tutkittavia projekteja kirjaus-, tehtävä- ja kommenttidatan puolesta. Virheraportteja ja ominaisuustoivetehtäviä on vähän. Taulukosta 6.8 löytyy projektin merkintätiedot. Limen merkintätiedot ovat selkeät ja suppeat. Kuvassa 6.4 on esitetty projektin kehitys ajassa.



Kuva 6.4 *Projektin Lime kirjaukset ja auenneet tehtävät ajan funktiona.*

Taulukko 6.8 *Projektin Lime virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät.*

Virheet	Ominaisuustoiveet
bug crash	enhancement

Projektin kehitys on ollut syklistä ja vuoden 2014 puolesta välistä vuoden 2015 alkupuolelle nopeaa. Projekti ei edistynyt heti perustamisensa jälkeen, ja se on tällä hetkellä selvästi tauolla. Auenneet tehtävät noudattavat hyvin kirjaustentekotahtia.

6.2.6 LightTable

LightTable on joukkorahoitettu ohjelmointiympäristö ja tekstinkäsittelyohjelma, josta on julkaistu useita vakaita versioita. Projekti on aloitettu 30.6.2013 ja se löytyy GitHubista nimellä LightTable käyttäjältä LightTable. Tehtyjen kirjausten perusteella projektia kehittää n. 2 pääkehittäjää ja n. 70 avustajaa. Projektia kehitetään ClojureScript-kielellä. Taulukkoon 6.9 on kerätty tiedot louhitusta datasta.

Taulukko 6.9 *Projektista LightTable louhitun datan määrä.*

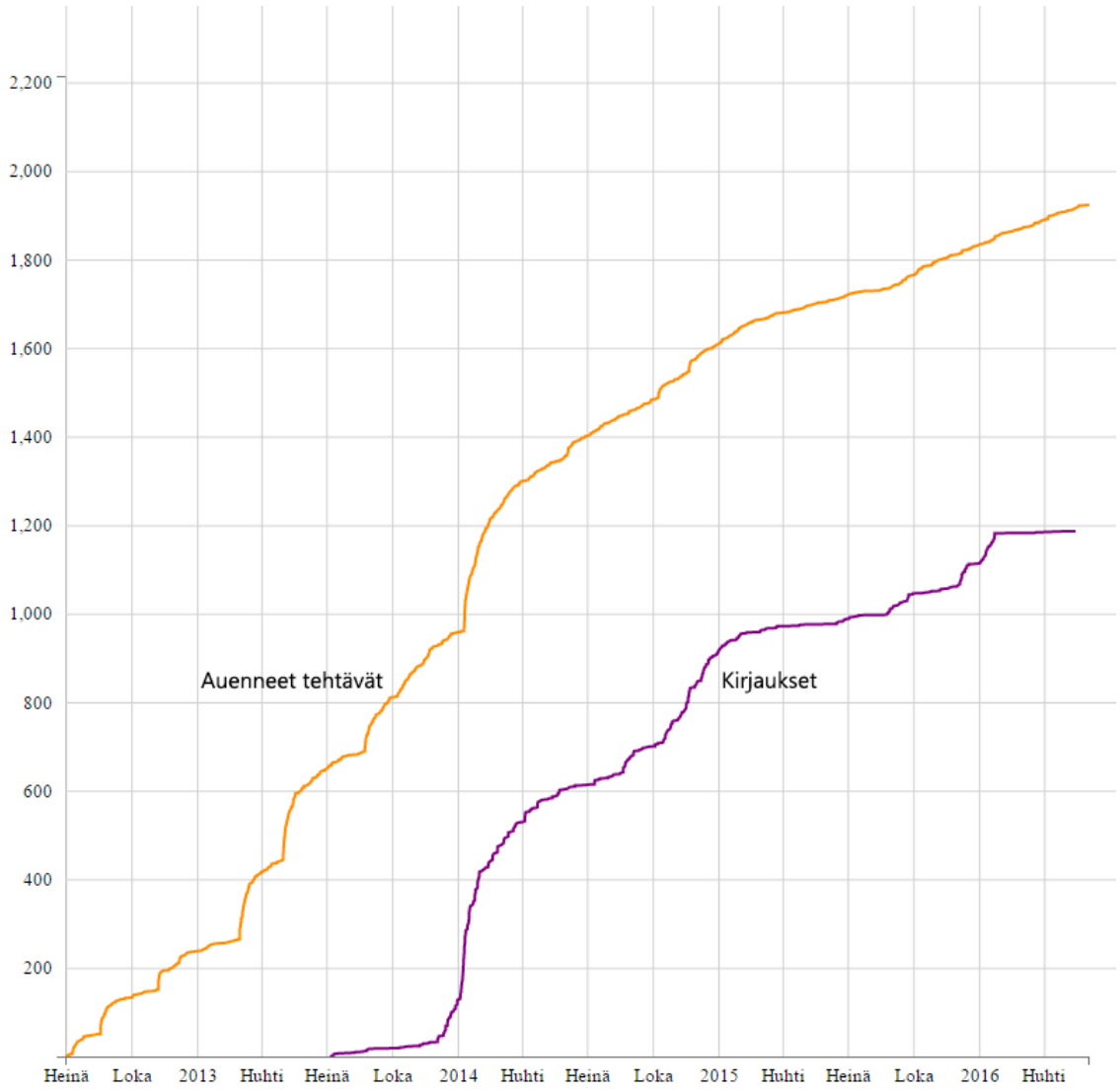
Kirjaukset	Pienet kirjaukset	Kommentit	Auenneet tehtävät	Auenneet virheet	Auenneet toiveet
1200	700	7400	1900	100	50

LightTable on kirjausten perusteella pienehkö projekti, mutta kommenttien ja tehtävien näkökulmasta keskikokoinen. Kaikista tehtävistä virheraportteja ja ominaisuustoivetehtäviä on vähän. Taulukosta 6.10 löytyy projektin merkintätiedot.

Taulukko 6.10 *Projektin LightTable virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät.*

Virheet	Ominaisuustoiveet
bug not-reproducible crash	feature enhancement

Projektissa on keskiverto määrä merkintöjä ja ne ovat selkeitä. Kuvassa 6.5 on esitetty projektin kehitys ajassa.



Kuva 6.5 *Projektin LightTable kirjaukset ja auenneet tehtävät ajan funktiona.*

Auenneiden tehtävien perusteella projektissa on voitu käyttää iteratiivista projektirakennetta, mutta kehitysjaksojen pituudet ovat venyneet. Kirjausten osalta projektissa vuorottelevat nopean kehityksen jaksot ja tauot kuukausien aikaskaalassa, joten iteratiivisuus ei ole varmaa. Auenneiden tehtävien käyrä voi selittyä kehityksessä olleen version valmistumisella ja sitä seuraavan version määrittelyllä.

6.2.7 Caret

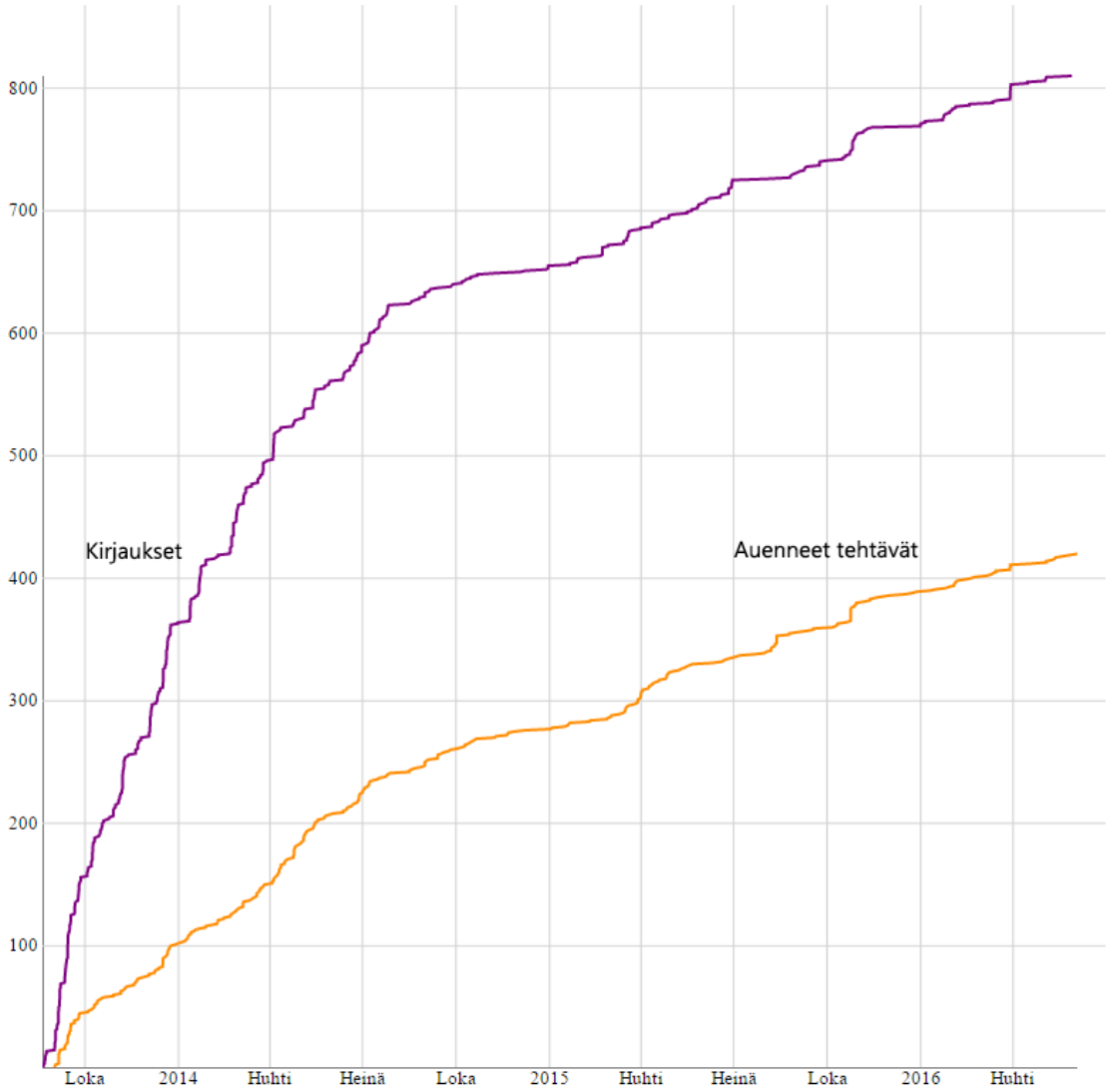
Caret on Sublime Text nimisen ohjelman inspiroima verkkoselainpohjainen tekstinkäsittelyohjelma, josta on julkaistu vakaita versioita. Projekti on aloitettu 18.8.2013 ja se löytyy GitHubista nimellä Caret käyttäjältä thomaswilburn. Tehtyjen kirjausten perusteella projektia kehittää 1 pääkehittäjä ja n. 30 avustajaa. Projektia

kehitetään Javascriptillä. Taulukkoon 6.11 on kerätty tiedot louhitusta datasta.

Taulukko 6.11 *Projektista Caret louhitun datan määrä.*

Kirjaukset	Pienet kirjaukset	Kommentit	Auenneet tehtävät	Auenneet virheet	Auenneet toiveet
800	400	1100	400	50	75

Caret on kerätyn datan puolesta pieni projekti. Virheraportteja ja ominaisuustoivetehtäviä on vähän, mutta suhteessa kirjausmäärään kohtalaisesti. Taulukosta 6.12 löytyy projektin merkintätiedot. Caretilla on tyypillisen suppeat, mutta selkeät merkintätiedot. Kuvassa 6.6 on esitetty projektin kehitys ajassa.



Kuva 6.6 *Projektin Caret kirjaukset ja auenneet tehtävät ajan funktiona.*

Taulukko 6.12 *Projektin Caret virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät.*

Virheet	Ominaisuustoiveet
bug	enhancement

Projektilla on selvästi kaksi lineaarisen kehityksen jaksoa, joista jälkimmäinen on huomattavasti hitaampi. Auenneiden tehtävien käyrä noudattaa samaa muotoa eikä merkittäviä poikkeuksia ole.

6.2.8 Zed

Zed on minimalistinen verkkoselaimessa toimiva tekstinkäsittelyohjelma, joka on saavuttanut vakaan version. Projekti on aloitettu 24.2.2013 ja se löytyy GitHubista nimellä zed käyttäjältä zedapp. Tehtyjen kirjausten perusteella projektia kehittää 1 pääkehittäjä ja n. 40 avustajaa. Projektia kehitetään pääasiassa Javascriptillä. Taulukkoon 6.13 on kerätty tiedot louhitusta datasta.

Taulukko 6.13 *Projektista Zed louhitun datan määrä.*

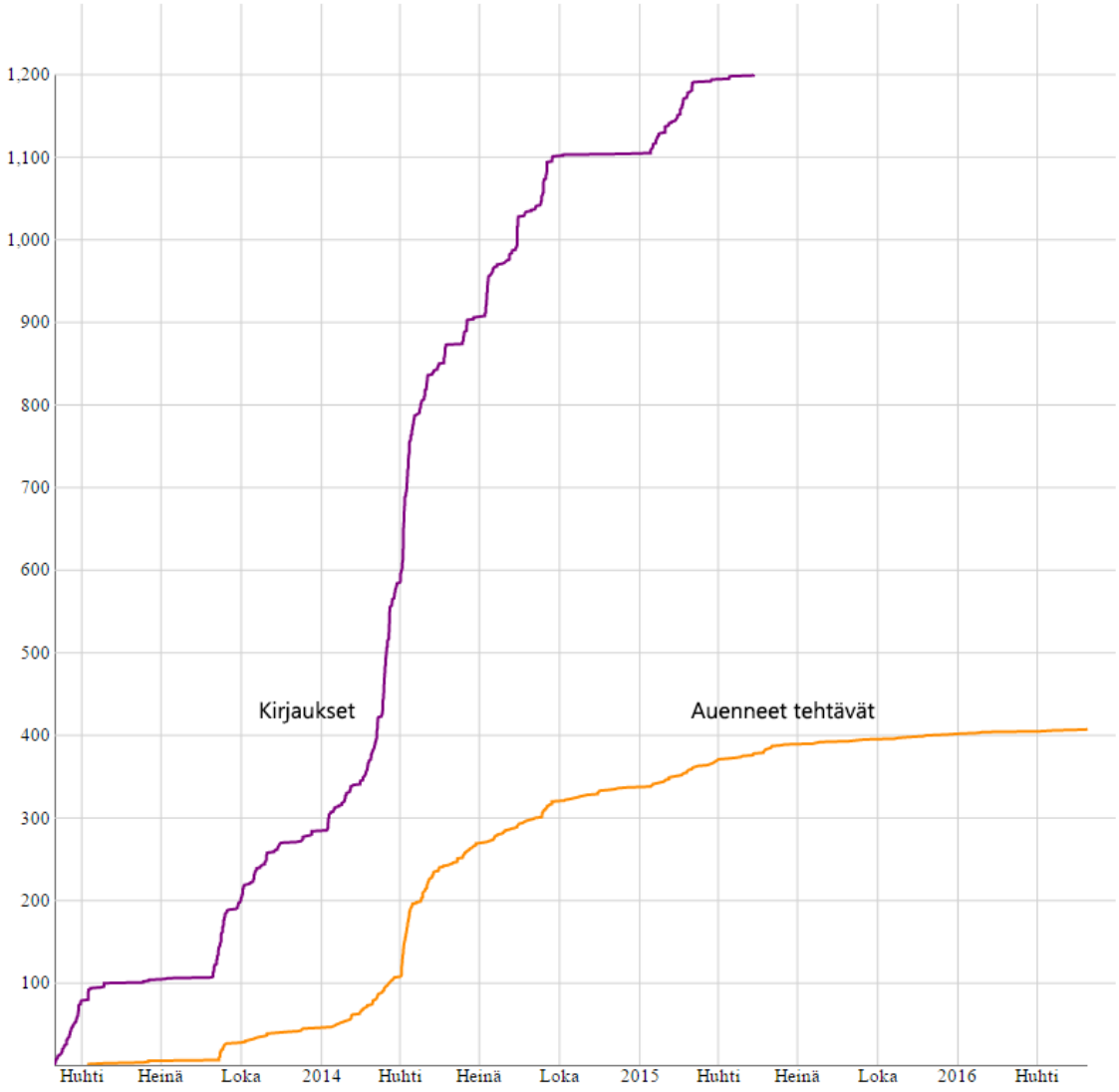
Kirjaukset	Pienet kirjaukset	Kommentit	Auenneet tehtävät	Auenneet virheet	Auenneet toiveet
1200	500	1300	400	25	50

Zed on kirjausten, tehtävien ja kommenttien puolesta pieni projekti. Virheraportteja ja ominaisuustoivetehtäviä on vähän. Taulukosta 6.14 löytyy projektin merkintätiedot.

Taulukko 6.14 *Projektin Zed virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät.*

Virheet	Ominaisuustoiveet
bug ACE Bug Chrome Bug	proposal enhancement

Zed käyttää keskivertoa määrää selkeitä merkintöjä. Kuvassa 6.7 on esitetty projektin kehitys ajassa.



Kuva 6.7 *Projektin Zed kirjaukset ja auenneet tehtävät ajan funktiona.*

Alkuvaiheen kehitys-tauko-kehitys jatkumon jälkeen projektissa on ollut molempien käyrien näkökulmasta yksi hyvin nopean kehityksen kausi, jonka jälkeen kehitys on hiipunut.

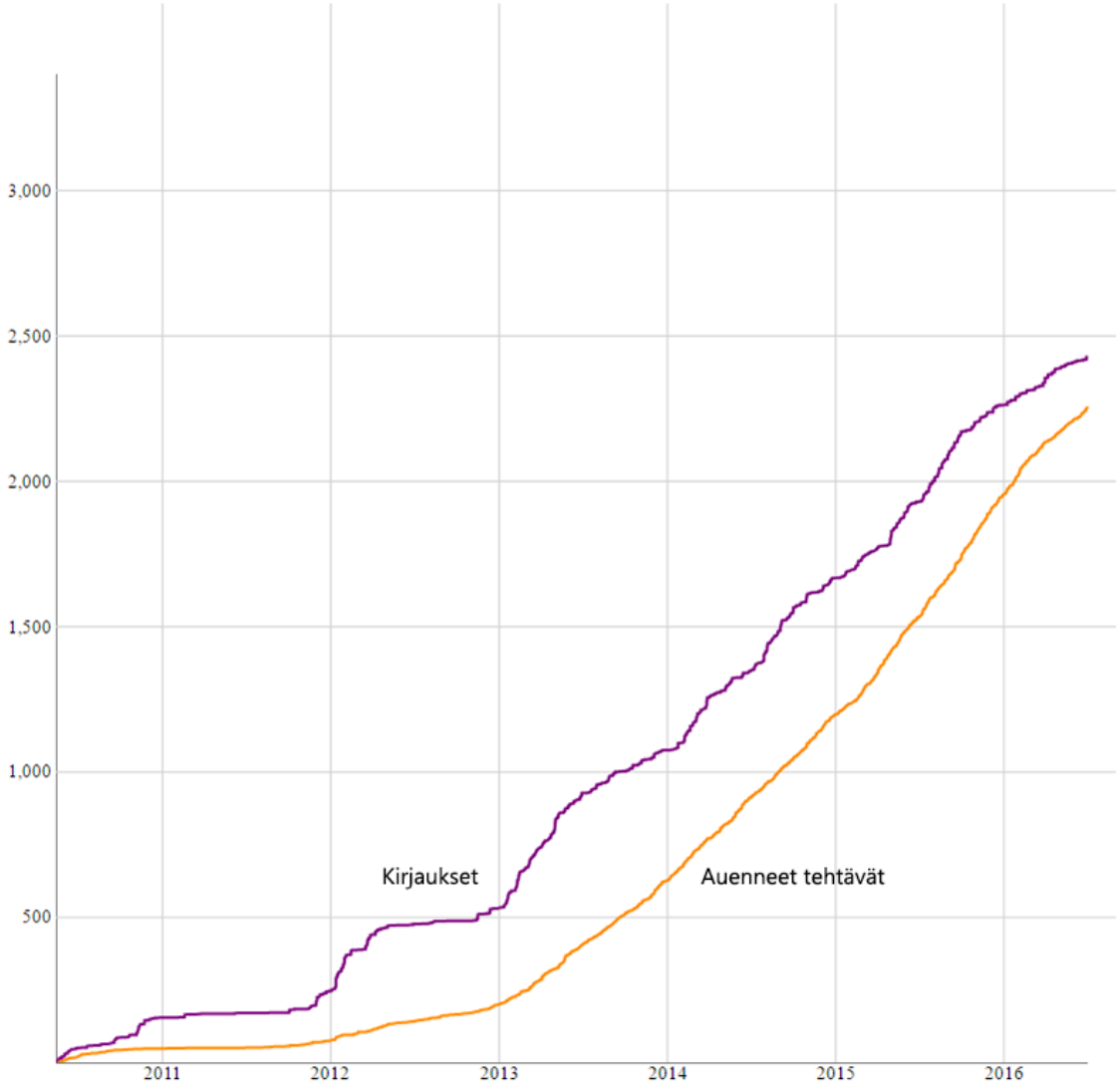
6.2.9 Video.js

Video.js on avoimen lähdekoodin videotoistin, joka toimii verkkoselaimessa. Se on laajasti käytetty ja siitä on tehty useita vakaita julkaisuja. Projekti on aloitettu 9.5.2010 ja se löytyy GitHubista nimellä video.js käyttäjältä videojs. Tehtyjen kirjausten perusteella projektia kehittää n. 3 pääkehittäjää ja n. 250 avustajaa. Projektia kehitetään pääasiassa Javascriptillä. Taulukkoon 6.15 on kerätty tiedot louhitusta datasta.

Taulukko 6.15 *Projektista Video.js louhitun datan määrä.*

Kirjaukset	Pienet kirjaukset	Kommentit	Auenneet tehtävät	Auenneet virheet	Auenneet toiveet
2400	1400	9800	2300	325	225

Video.js on kirjausten, tehtävien ja kommenttien perusteella keskikokoinen projekti. Virheraportteja ja ominaisuustoivetehtäviä on paljon, minkä vuoksi projekti valittiin tutkimukseen. Taulukosta 6.16 löytyy projektin merkintätiedot. Projektin merkintätiedot ovat minimalistiset, selkeät ja datamäärien perusteella johdonmukaisesti käytetyt. Kuvassa 6.8 on esitetty projektin kehitys ajassa.



Kuva 6.8 *Projektin Video.js kirjaukset ja auenneet tehtävät ajan funktiona.*

Taulukko 6.16 *Projektin Video.js virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät.*

Virheet	Ominaisuustoiveet
bug browser bug	enhancement

Kirjausten näkökulmasta projekti on päätynyt alun kehitys-tauko-kehitys-tauko jatkumon jälkeen monivuotiseen lineaarisen kehityksen kauteen. Avautuneiden tehtävien käyrä tukee tätä tulkintaa.

6.2.10 Fabric

Fabric on ohjelmointikieli Pythonille tarkoitettu kirjasto ja komentorivityökalu, jolla voi suoraviivaistaa järjestelmävalvojan tehtäviä. Kirjastosta on julkaistu useita vakaita versioita. Projekti on aloitettu 30.12.2007 ja se löytyy GitHubista nimellä fabric käyttäjältä fabric. Tehtyjen kirjausten perusteella projektia kehittää n. 1 pääkehittäjä ja n. 120 avustajaa. Projektia kehitetään luonnollisesti Pythonilla. Taulukkoon 6.17 on kerätty tiedot louhitusta datasta.

Taulukko 6.17 *Projektista Fabric louhitun datan määrä.*

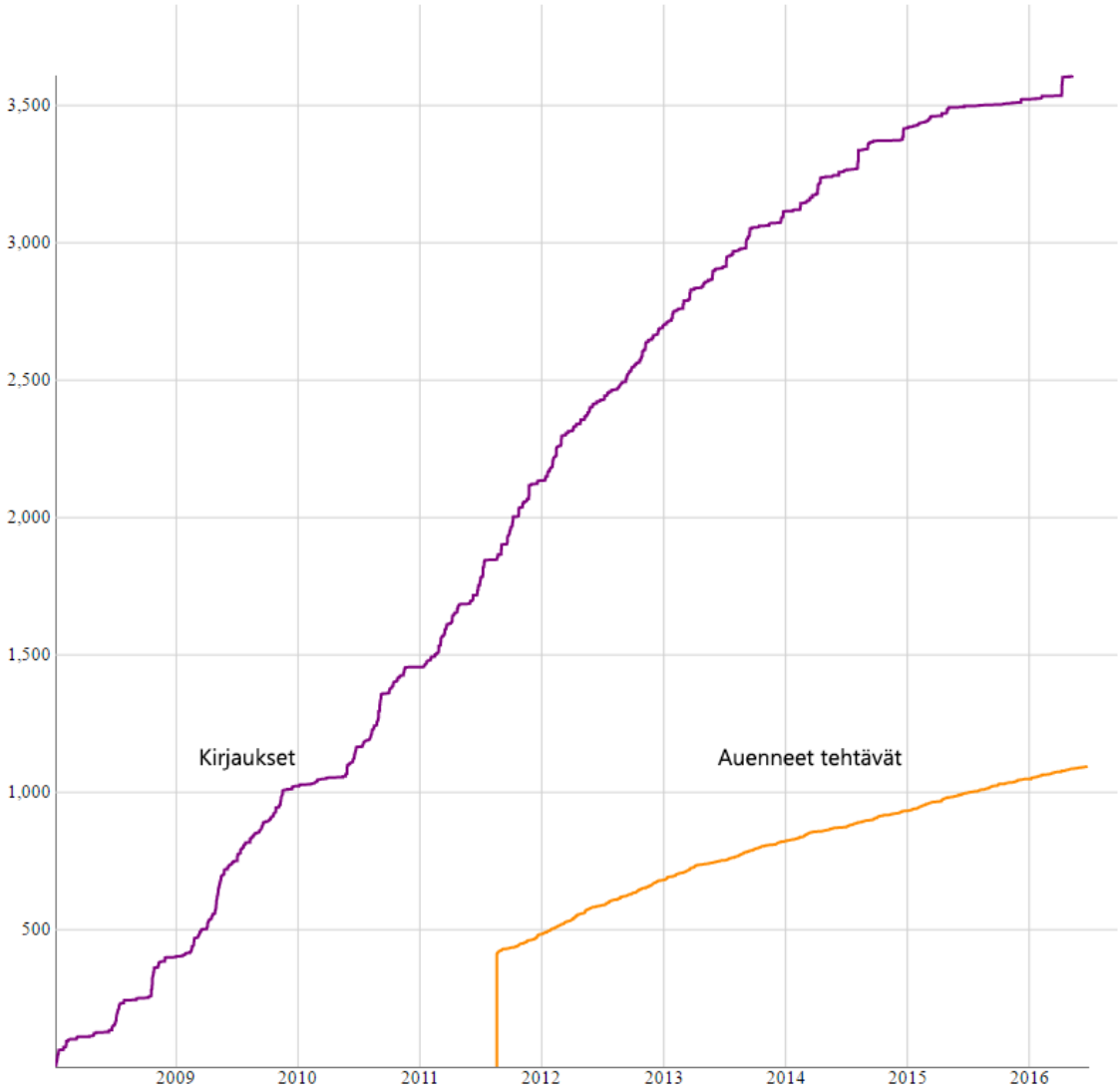
Kirjaukset	Pienet kirjaukset	Kommentit	Auenneet tehtävät	Auenneet virheet	Auenneet toiveet
3600	2200	4000	1100	325	225

Fabric on kirjausten puolesta suurehko projekti, mutta tehtävien ja kommenttien näkökulmasta vain keskikokoinen suhteessa muihin projekteihin. Virheraportteja ja ominaisuustoivetehtäviä on paljon sekä absoluuttisesti että suhteessa kaikkiin tehtäviin, minkä vuoksi projekti valittiin tutkimukseen. Taulukosta 6.18 löytyy projektin merkintätiedot.

Taulukko 6.18 *Projektin Fabric virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät.*

Virheet	Ominaisuustoiveet
Bug	Feature

Projektin merkintätiedot ovat minimalistiset, selkeät ja datamäärien perusteella johdonmukaisesti käytetyt. Kuvassa 6.9 on esitetty projektin kehitys ajassa.



Kuva 6.9 *Projektin Fabric kirjaukset ja auenneet tehtävät ajan funktiona.*

Projektia on kehitetty vuositasolla pitkään lineaarisesti eteenpäin, mutta vuoden 2015 taitteen kohdalla on havaittavissa hidastumista. Auenneissa tehtävissä on erikoinen piikki, joka saattaa johtua tehtävätietojen siirtämisestä toisesta palvelusta GitHubiin. Muutoin tehtäväkäyräkin on lineaarinen.

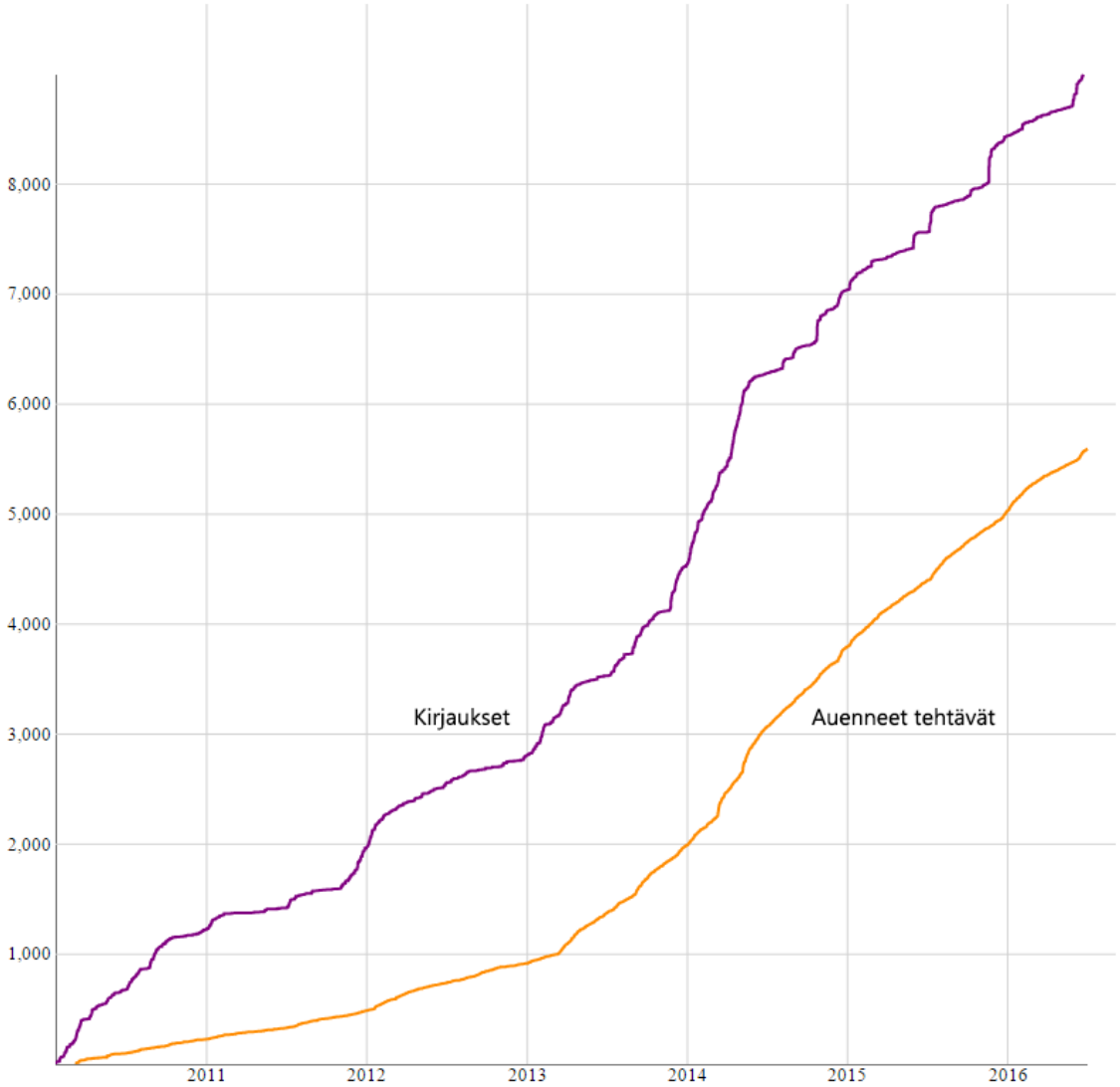
6.2.11 Vagrant

Vagrant on laajasti käytetty työkalu kehitysympäristöjen rakentamiseen ja jakeluun. Projektista on julkaistu useita vakaita versioita. Projekti on aloitettu 17.1.2010 ja se löytyy GitHubista nimellä vagrant käyttäjältä mitchellh. Tehtyjen kirjausten perusteella projektia kehittää n. 2 pääkehittäjää ja n. 730 avustajaa. Projektia kehitetään Ruby-kielellä. Taulukkoon 6.19 on kerätty tiedot louhitusta datasta.

Taulukko 6.19 *Projektista Vagrant louhitun datan määrä.*

Kirjaukset	Pienet kirjaukset	Kommentit	Auenneet tehtävät	Auenneet virheet	Auenneet toiveet
9000	5300	Puuttuu	5600	1175	300

Vagrant on kirjausten ja tehtävien osalta suurin louhittu projekti. Kommenttidataa ei saatu kerättyä liian suuren määrän vuoksi. Projektia tutkitaan silti muista näkökulmista. Virheraportteja on hyvin paljon ja ominaisuustoiveita absoluuttisessa määrässä paljon, mutta suhteessa kaikkiin tehtäviin vähän. Projekti valittiin tutkimukseen virheraporttien suuren määrän takia. Taulukosta 6.20 löytyy projektin merkintätiedot. Projektin merkintätiedot ovat suppeat ja datamäärien perusteella johdonmukaisesti käytetyt. Kuvassa 6.10 on esitetty projektin kehitys ajassa.



Kuva 6.10 *Projektin Vagrant kirjaukset ja auenneet tehtävät ajan funktiona.*

Taulukko 6.20 *Projektin Vagrant virheraportteja ja ominaisuustoiveita edustavien tehtävien merkinnät.*

Virheet	Ominaisuustoiveet
bug	enhancement

Projektin kehitystahti kiihtyi molempien käyrien näkökulmasta pitkään eksponentiaalisesti, kunnes vuonna 2014 kiihtyminen hidastui ja kehitys jatkui lineaarisesti eteenpäin. Kuukausitasolla kirjauskäyrässä nähdään myös taukoja.

6.3 Yhteenveto projekteista

Projekteja vertailemalla huomataan, että niillä on selvästi erilaisia kehityskaaria. Osa on hyvin lineaarisia ja tauottomia, osa on syklisiä tai niissä on taukoja. Osassa pitkistä projekteista syklisyys katoaa vuositasolla tarkasteltuna, mutta näkyy kuukausitasolla. Kirjauskäyrät ja avautuneiden tehtävien käyrät noudattavat pääsääntöisesti samankaltaista kehityskulkua toisiinsa verrattuna. Käyrien muutoksissa ei ole havaittavissa silmämääräisesti viivettä vuosi- tai kuukausitasolla. Vaikuttaa, että kirjaukset ja auenneet tehtävät liittyvät toisiinsa voimakkaasti. Vertailun helpottamiseksi projektien perustiedot on kerätty taulukkoon 6.21.

Taulukko 6.21 *Yhteenveto projektien datamääristä ja perusluonteista.*

Projekti	Tyyppi	Kirjaukset	Pienet kirj.	Kommentit	Auen. tehtävät	Auen. virheet	Auen. toiveet	Kehit. / Avust.
Spacemacs	Tekstink.	5500	3300	18000	3100	1000	75	2 / 520
Neovim	Tekstink.	5900	2500	17500	2200	450	150	10 / 280
TextMate	Tekstink.	3600	2100	900	300	0	0	1 / 70
Lime	Tekstink.	1100	400	1600	300	25	75	3 / 50
LightTable	Tekstink.	1200	700	7400	1900	100	50	2 / 70
Caret	Tekstink.	800	400	1100	400	50	75	1 / 30
Zed	Tekstink.	1200	500	1300	400	25	50	1 / 40
Video.js	Muu	2400	1400	9800	2300	325	225	3 / 250
Fabric	Muu	3600	2200	4000	1100	325	225	1 / 120
Vagrant	Muu	9000	5300	Puuttuu	5600	1175	300	2 / 730

Valittu projektijoukko on datamäärien puolesta monipuolinen, mutta myös vertailukelpoinen. Projekteja voidaan luonnehtia kirjaus-, kommentti- ja tehtävämääriltään suuriksi, keskikokoisiksi ja pieniksi. Pieniä kirjauksia on kaikissa projekteissa karkeasti puolet kaikista kirjauksista eikä suuria poikkeuksia ole. Vain puolessa projekteista on riittävästi virheraportteja ja ominaisuustoive-tehtävädataa, että näitä voidaan harkita tutkimukseen. Neljässä projektissa on vain yksi pääkehittäjä ja kaikissa on huomattava määrä avustajia, mikä tekee projekteista henkilöstön osalta erilaisia teollisuuden tuotekehitykseen verrattuna.

7. DATAN TUTKIMINEN

7.1 Yleisiä huomioita

Hypoteesien pätevyyttä tutkitaan kuhunkin liittyvällä dataparilla eli joko tehtäväkirjaus, tehtävä-kommentti tai tehtävä-tehtävä parilla. Dataparien tutkimisen yleinen esittely sekä tulosten esitysmuoto kuvataan tässä kohdassa. Jokaisen hypoteesin käsittely on jaettu omaan alilukuunsa, mutta kaikkien tulosten arviointi tehdään erikseen luvussa 8.

Datasta suodatettiin tutkimusta varten pois kaikki liitospyyntötyyppiset tehtävät, koska nämä tehtävät eivät edusta lisätyön tarvetta samalla tavalla kuin normaalit tehtävät. Tehtävien halutaan kuvastavan projektinhallinnallisesta näkökulmasta resurssien tarvetta, kun taas liitospyynnöt ovat lähempänä valmista työtä kuin tulevaa työtä. Ristikorrelaation laskemisessa molemmista aikasarjoista puhdistettiin autokorrelaatiot pois, mikä tuottaa luotettavampia tuloksia. Kaikkien auenneiden tehtävien autokorrelaatio on tästä luonnollisesti poikkeus, koska autokorrelaatio on tällöin tutkimuksen alla.

Tutkittavaksi aikaväliksi valittiin mahdollisimman pitkä yhtenäinen kausi, jolla kaikki tutkittavat dataparit kehittyivät. Pidempi aikaväli tuottaa tilastollisesti luotettavampia tuloksia ja saman aikavälin tarkastelu tuottaa vertailukelpoisempia tuloksia projektin sisällä. Virheraporttien ja ominaisuustoivetehtävien tutkiminen osoittautui haastavaksi, koska niissäkin projekteissa, joissa näitä tehtäviä oli hyvä määrä ei välttämättä ollut yhtäjaksoista kautta, jolloin tehtävät aukesivat tai sulkeutuivat. Siksi projektin Vagrant kohdalla tutkittiin kahta eri aikaväliä, jotta virheraportteja pystyttiin tutkimaan järkevästi.

Kuten kohdassa 5.3.2 kuvailtiin, dataa tutkitaan viikkotasolle koottuna. Tämän vuoksi tutkittujen tilastopisteiden määrä vastaa tutkitun ajanjakson pituutta viikkoina. Lyhyemmällä aikavälillä kohina on liian suurta ja pidemmällä aikavälillä hyödyllisen mittaiset viiveet katoavat datan koonnissa.

Suurimmassa osassa projekteista otos on yli 100 viikkoa, mitä pidetään tyydyttävänä otoskokona tässä tutkimuksessa. Monessa ennustusta potentiaalisesti soveltavassa

projektissakaan ei ole tätä pidempää historiaa tutkittavana. Dataparien viivettä tutkittiin siirtämällä aikasarjoja molempiin suuntiin askeleittain 16 viikon verran, minkä pitäisi olla enemmän kuin tarpeeksi luotettavan ennustuspotentiaalin etsimiseen. Mitä pidempiä viiveitä käsitellään, sen epätodennäköisempi yhteys on.

Yksi oleellinen huomio, joka tehtiin kohdassa 6.3 kirjausten ja auenneiden tehtävien osalta toistuu myös muiden dataparien kohdalla: dataparien vastaavissa käyrissä ei näy viivettä silmämääräisesti tarkasteltuna. Ristikorrelaatiota ja impulssivastefunktiota käyttämällä löytyy kuitenkin silmämääräistä arviota tarkempia tuloksia.

Kaikissa datapareja tutkivissa kohdissa on oma tulostaulukkonsa ristikorrelaatiolle ja impulssivastefunktiolle. Taulukoissa on esitetty jokaisesta projektista tutkittu aikaväli, otoskoko N, 95% luottamusrajan korrelaatiokerroin ja tulokset kaikkien tehtävien, virheraporttien ja ominaisuustoivetehtävien osalta. Luottamusraja perustuu kohdassa 4.3 esiteltyyn luottamusväliin ja kohdan 5.2.2 matematiikkaan, johon myös tulosten laskenta perustuu. Analysoitavan dataparin ennustava puoli määräytyy tarkastellun hypoteesin mukaan ja pysyy kohdan sisällä aina samana. Dataparin ennustettava puoli eli tehtävä puoli määräytyy taulukon sarakkeen mukaan.

Ristikorrelaatiota ja impulssivastefunktiota laskettaessa tutkittavia aikasarjoja siirretään toisiinsa nähden askeleittain ja jokaista viiveaskelta kohti lasketaan aikasarjojen korrelaatiokerroin. Korrelaatiokerroin kertoo kuinka voimakas yhteys aikasarjojen välillä on eli kuinka samanlaiset ne ovat. Ristikorrelaation tulokset kuvastavat kuinka voimakas yhteys on pelkkiä tiettyjä viivearvoja tarkasteltaessa, kun taas impulssivastefunktion tulokset kuvastavat miten ennustavan aikasarjan vaikutus jakaantuu usealle viivearvolle ennustettavassa aikasarjassa.

Tulokset esitetään viive-korrelaatiokerroin pareina, jossa viive esitetään ensin ja sitä vastaava korrelaatiokerroin esitetään suluissa. Taulukoissa yhden projektin yksittäinen tulossolu voi sisältää yhden, useita tai ei yhtäkään viive-korrelaatiokerroin paria. Tuloksiin on koottu kaikki viiveet, joiden korrelaatiokerroin ylittää luottamusrajan. Jos esimerkiksi kirjauksia ja auenneita tehtäviä tarkasteltaessa jonkin projektin tuloksissa näkyy viive 2, tämä tarkoittaa, että nykyhetkessä tehtyjen kirjausten tahti on samansuuntainen kuin tehtävien aukeamistahti kahden viikon päästä.

Luottamusvälillä on ala- sekä yläraja ja tässä tutkimuksessa näiden rajojen väliin osuu noin 95% tuloksista. Luottamusraja tarkoittaa tulostaulukoissa luottamusvälin ylärajaa, koska negatiivista korrelaatiota ei ole odotettavissa. Luottamusrajaa suuremmat korrelaatiokertoimet ovat alustavasti merkittäviä eli ne johtuvat vain 5% todennäköisyydellä sattumasta. Aikasarjojen välisen yhteyden merkittävyyden arviointi tietyllä viiveellä perustuu siis viivettä vastaavan korrelaatiokertoimen ja

luottamusrajan vertailuun.

Muista projekteista eroten projektissa Vagrant virheraporttien tutkimiseen käytetty aikaväli on merkitty (V):llä ja tätä vastaavalla rivillä on aikavälin otoskoon ja 95% luottamusrajan arvot. Tuloksissa on selkeästi sattumasta johtuvia luottamusrajan ylityksiä, joten nämä on merkitty yhdellä tai kahdella *-merkillä. Yksi tähti tarkoittaa todennäköisesti sattumasta johtuvaa tulosta ja kaksi tähteä hyvin varmasti sattumasta johtuvaa tulosta. Huutomerkki tarkoittaa luottamusrajan alittavaa, mutta kontekstissaan kiinnostavaa tulosta. Jos luottamusrajan ylittäviä tuloksia ei ole yhtään, merkintänä on —, mutta jos dataa ei ole ollut tarpeeksi tulosten laskentaan, merkintänä on piste. Dataa voi olla liian vähän absoluuttisessa mielessä, mutta myös siinä tapauksessa, jos data ei muodosta yhtenäistä tarkasteltavaa ajanjaksoa.

7.2 Kirjaukset ja auenneet tehtävät

Kirjausten ja auenneiden tehtävien dataparilla tutkitaan hypoteesia 1 eli arvioitavana on kehityksestä viiveellä seuraavat uudet tehtävät. Tämä datapari on tutkituista datapareista yleisluontoisin yhdistelmä viiveellisen yhteyden etsimiseen, koska pari edustaa ohjelmistotuotantoprosessin kahta päätuotosta ja ne liittyvät työskentelyssä voimakkaasti toisiinsa. Kohdasta 6 löytyvät kuvaajat esittelevät tässä hypoteesissa tutkittua dataa. Taulukkoon 7.1 on koottu tulokset ristikorrelaation osalta.

Taulukko 7.1 *Kirjausten ja auenneiden tehtävien ristikorrelaation tulokset*

Projekti	Aikaväli	N	95%	Kaikki	Virheet	Toiveet
Spacemacs	1.11.2014 - 1.4.2016	73	0.23	0 (0.44)	0 (0.35)	.
Neovim	1.4.2014 - 1.5.2016	108	0.19	0 (0.21)	—	.
TextMate	1.10.2012 - 1.2.2016	173	0.15	-15 (0.16)** -14 (0.22)** 0 (0.20) 3 (0.18)	.	.
Lime	1.12.2013 - 1.4.2015	63	0.24	0 (0.36)	.	.
LightTable	1.12.2013 - 1.1.2016	107	0.19	0 (0.64) 1 (0.29) 3 (0.19)	.	.
Caret	1.10.2013 - 1.5.2016	133	0.17	-15 (0.17)** 0 (0.39) 6 (0.33)* 13 (0.20)**	.	.
Zed	1.10.2013 - 1.10.2014	51	0.28	-14 (0.29)** 0 (0.48) 3 (0.47) 4 (0.30)	.	.
Video.js	1.1.2013 - 1.10.2015	142	0.17	11 (0.19)**	5 (0.18)* 11 (0.18)**	-4 (0.18)* 0 (0.18) 15 (0.17)**
Fabric	1.9.2011 - 1.5.2015	190	0.15	—	.	.
Vagrant	1.11.2011 - 1.6.2016, 1.10.2014 - 1.6.2016(V)	238 86	0.13 0.22	0 (0.23) 1 (0.14) 7 (0.25)**	7 (0.23)**	.

Taulukosta nähdään, että kahdeksassa projektissa on merkittävää tai hyvin merkittävää korrelaatiota viiveellä 0 eli alle viikon viiveellä. Merkittävä korrelaatio tarkoittaa, että viiveen korrelaatiokerroin ylittää luottamusrajan ja vastavasti hyvin merkittävässä korrelaatiossa luottamusraja ylittyy huomattavasti. Vain neljässä projektissa on korrelaatiota viiveillä 1-4, jota voidaan pitää luotettavimpana varsi-

naisena viivevälinä viiveen 0 lisäksi. Vain yhdessä neljästä projektista on korrelaatiota kirjausten ja virheraporttien välillä. Erikoisesti ainoassa ominaisuustoivetarkastelussa on korrelaatiota viiveellä 0, vaikka samassa projektissa ei ole luotettavaa korrelaatiota kaikkien tehtävien kohdalla. Taulukkoon 7.2 on koottu tulokset impulssivastefunktion osalta.

Taulukko 7.2 *Kirjausten ja auenneiden tehtävien impulssivastefunktion tulokset*

Projekti	**Aikaväli**	**N**	**95%**	**Kaikki**	**Virheet**	**Toiveet**
Spacemacs	1.11.2014 - 1.4.2016	73	0.23	0 (0.44)	0 (0.42)	.
Neovim	1.4.2014 - 1.5.2016	108	0.19	0 (0.19)	—	.
TextMate	1.10.2012 - 1.2.2016	173	0.15	-14 (0.15)** -7 (0.18)** 0 (0.23) 10 (0.17)**	.	.
Lime	1.12.2013 - 1.4.2015	63	0.24	0 (0.42)	.	.
LightTable	1.12.2013 - 1.1.2016	107	0.19	0 (0.68) 1 (0.34) 2 (0.18)!	.	.
Caret	1.10.2013 - 1.5.2016	133	0.17	-15 (0.19)** -10 (0.17)** 0 (0.38) 6 (0.29)* 13 (0.19)**	.	.
Zed	1.10.2013 - 1.10.2014	51	0.28	0 (0.52) 4 (0.29)	.	.
Video.js	1.1.2013 - 1.10.2015	142	0.17	0 (0.19)	5 (0.19)* 11 (0.24)**	-4 (0.21)*
Fabric	1.9.2011 - 1.5.2015	190	0.15	—	.	.
Vagrant	1.11.2011 - 1.6.2016, 1.10.2014 - 1.6.2016(V)	238 86	0.13 0.22	0 (0.21) 1 (0.17) 4 (0.13)	—	.

Impulssivastefunktion tulokset ovat linjassa ristikorrelaation tulosten kanssa. Vii-

veellä 0 löytyy korrelaatiota kaikista paitsi yhdestä projektista, josta ei löytynyt mitään ristikorrelaatiollakaan. Vain kolmessa projektissa kirjausten vaikutus ulottuu viivevälille 1-4. Yhdessäkään projektissa ei ole kolmea peräkkäistä luottamusrajan ylittävää tulosta, mutta LightTable pääsee lähelle. Myös impulssivastefunktion puolella virheraportit ja ominaisuustoivetehtävät korreloivat heikommin kirjausten kanssa kuin kaikki tehtävät.

Näiden tulosten perusteella hypoteesi 1 ei pidä yleisesti paikkaansa, mutta se voi pitää heikosti paikkansa yksittäisten projektien kohdalla. Tähän vaikuttanee projektin käyttämä kehitysprosessi. Tulokset voivat viitata avoimen lähdekoodin kehittäjien nopeaan reagointiin tai tiheään julkaisutahtiin. Jos kirjaukset aiheuttavat uusia tehtäviä, ne avataan pääsääntöisesti saman viikon aikana eikä ongelmien ilmestymiseen kulu merkittävästi aikaa. Todennäköisempää voi kuitenkin olla, että aktiivisessa kehityksessä tehdään sekä kirjauksia että uusia tehtäviä ilman, että nämä liittyvät suoraan toisiinsa. Näin käy esimerkiksi, kun kehittäjä ensin tekee suunnitellut kirjauksensa ja siirtyy sitten lisäämään kirjauksiin liittymättömiä tehtäviä järjestelmään osana normaalia työrutiiniaan.

7.3 Pienet kirjaukset ja sulkeutuneet tehtävät

Pienten kirjausten ja sulkeutuneiden tehtävien dataparilla tutkitaan hypoteesia 2 eli arvioitavana on korjaavaksi oletettujen kirjausten vaikutus tehtävien sulkeutumiseen. Erityisesti virheraporttien sulkeutuminen on tämän dataparin kohdalla kiinnostavaa. Tämän dataparin kanssa tehdään yksityiskohtaisempaa tarkastelua verrattuna kaikkien kirjausten ja tehtävien tutkimiseen, mutta pohjalla on silti yleisluontoinen suhde kirjausten ja valmistuvien tehtävien välillä.

Pienten kirjausten ja sulkeutuneiden tehtävien kuvaajat noudattavat pääsääntöisesti samaa kehityskaarta kuin kirjausten ja auenneiden tehtävien kuvaajat, mutta pieniä kirjauksia on luonnollisesti vähemmän. Sulkeutuneiden tehtävien käyrä on portaisempi verrattuna auenneiden tehtävien tasaisuuteen. Joissain projekteissa on suuri piikki sulkeutuneissa tehtävissä, mikä vääristää dataa. LightTable on otettu pois tarkastelusta tämän vuoksi, ja muissa aikaväli on valittu niin, ettei piikki osu tarkasteluvälille. Taulukkoon 7.3 on koottu tulokset ristikorrelaation osalta.

Taulukko 7.3 *Pienten kirjausten ja sulkeutuneiden tehtävien ristikorrelaation tulokset*

Projekti	Aikaväli	N	95%	Kaikki	Virheet	Toiveet
Spacemacs	1.11.2014 - 1.4.2016	73	0.23	0 (0.43)	0 (0.45)	.
Neovim	1.4.2014 - 1.5.2016	108	0.19	-8 (0.20)** -1 (0.19) 1 (0.22)	—	.
TextMate	1.10.2012 - 1.2.2016	173	0.15	-14 (0.21)** -11 (0.15)** 0 (0.17) 3 (0.23)	.	.
Lime	1.12.2013 - 1.4.2015	63	0.24	0 (0.39)	.	.
Caret	1.10.2013 - 1.5.2016	133	0.17	-10 (0.22)** 0 (0.49)	.	.
Zed	1.10.2013 - 1.10.2014	51	0.28	0 (0.57)	.	.
Video.js	1.1.2013 - 1.10.2015	142	0.17	0 (0.27) 11 (0.20)** 15 (0.21)**	0 (0.34) 12 (0.19)**	-3 (0.18)* 0 (0.33)
Fabric	1.9.2011 - 1.5.2015	190	0.15	0 (0.56) 14 (0.16)**	.	.
Vagrant	1.11.2011 - 1.6.2016, 1.10.2014 - 1.6.2016(V)	238 86	0.13 0.22	0 (0.83)	—	.

Kaikissa paitsi yhdessä projektissa löytyy merkittävää korrelaatiota viiveellä 0, kun tarkastellaan kaikkia tehtäviä. Vain kahdessa projektissa yhdeksästä on korrelaatiota viiveillä 1-4. Mielenkiintoisesti Neovimissä näkyy korrelaatiota negatiivisella viiveellä eli sulkeutuneiden tehtävien jälkeen ilmestyy pieniä kirjauksia. Tämä datapiste on juuri luottamusrajalla, joten kyseessä voi olla sattuma. Puolessa virheraportillisista projekteista löytyy korrelaatiota, mutta nämäkin viiveellä 0. Ominaisuustoiveissakin viive 0 on edustettuna. Taulukkoon 7.4 on koottu tulokset impulssivastefunktion osalta.

Taulukko 7.4 *Pienten kirjausten ja sulkeutuneiden tehtävien impulssivastefunktion tulokset*

Projekti	Aikaväli	N	95%	Kaikki	Virheet	Toiveet
Spacemacs	1.11.2014 - 1.4.2016	73	0.23	0 (0.47) 12 (0.27)**	0 (0.48) 12 (0.23)**	.
Neovim	1.4.2014 - 1.5.2016	108	0.19	-8 (0.20)**	—	.
TextMate	1.10.2012 - 1.2.2016	173	0.15	-14 (0.19)** 0 (0.16) 3 (0.15) 11 (0.16)**	.	.
Lime	1.12.2013 - 1.4.2015	63	0.24	0 (0.50)	.	.
Caret	1.10.2013 - 1.5.2016	133	0.17	-10 (0.22)** 0 (0.48) 6 (0.17)*	.	.
Zed	1.10.2013 - 1.10.2014	51	0.28	0 (0.64)	.	.
Video.js	1.1.2013 - 1.10.2015	142	0.17	-3 (0.17)* 0 (0.28) 11 (0.20)** 15 (0.18)**	-3 (0.18)* 0 (0.36) 12 (0.19)**	-3 (0.18)* 0 (0.35)
Fabric	1.9.2011 - 1.5.2015	190	0.15	0 (0.55) 14 (0.17)**	.	.
Vagrant	1.11.2011 - 1.6.2016, 1.10.2014 - 1.6.2016(V)	238 86	0.13 0.22	-11 (0.14)** 0 (0.82)	0 (0.84)	.

Impulssivastefunktiolla tutkittuna viive 0 korreloi samoissa projekteissa kuin ristikorrelaatiossakin, mutta viiveet 1-4 ainoastaan yhdessä projektissa. Ristikorrelaatiosta poiketen Neovimin tuloksissa ei näy negatiivista viivettä. Virheraportit korreloivat tässä taulukossa puolestaan yhtä projektia enemmän verrattuna ristikorrelaatiotaulukkoon, mutta tässäkin tilanteessa viiveellä 0. Tilanne on sama ominaisuustoiveissa.

Tulokset ovat hyvin samanlaisia kuin kirjausten ja auenneiden tehtävien dataparissa, eikä oletus paremmasta ennustuspotentiaalista tarkemmalla dataparilla pidä

paikkaansa. Jos pienet kirjaukset aiheuttavat sulkeutuvia tehtäviä, tämä tapahtuu pääasiassa viikon aikajänteellä. Viiveen 0 korostuminen voi viitata siihen, että korjattavaan virheeseen liittyvä virheraportti suljetaan nopeasti kirjauksen tekemisen jälkeen. Tällöin kirjaus ja suljettava tehtävä liittyvät toisiinsa todennäköisemmin kuin hypoteesissa 1, koska tehtävän sulkemiseen liittyy lähes aina kirjaus toisin kuin tehtävän aukeamiseen.

7.4 Kommentit ja sulkeutuneet tehtävät

Kommenttien ja sulkeutuneiden tehtävien dataparilla tutkitaan hypoteesia 3 eli arvioitavana on keskustelun vaikutus tehtävien sulkeutumiseen. Kommenttien ja sulkeutuneiden tehtävien yhteys on epäsuorempi kuin kahdessa ensimmäisessä dataparissa, koska ohjelmistotuotantoprosessin näkökulmasta tehtyjen kommenttien ja sulkeutuneiden tehtävien välissä on todennäköisesti tehty myös kirjauksia. Yhteys kommentoinnin ja tehtävien sulkeutuminen välillä on kuitenkin intuitiivinen. Kommenttien kuvaajat noudattavat samaa kehityskaarta kuin auenneiden kirjausten käyrä, mutta kommentteja on paljon enemmän. LightTablen lisäksi tarkastelusta on otettu pois Vagrant, josta puuttuu kommenttidata. Taulukkoon 7.5 on koottu tulokset ristikorrelaation osalta.

Taulukko 7.5 *Kommenttien ja sulkeutuneiden tehtävien ristikorrelaation tulokset*

Projekti	**Aikaväli**	**N**	**95%**	**Kaikki**	**Virheet**	**Toiveet**
Spacemacs	1.11.2014 - 1.4.2016	73	0.23	-14 (0.30)** 0 (0.70)	-14 (0.33)** 0 (0.64)	.
Neovim	1.4.2014 - 1.5.2016	108	0.19	0 (0.59)	0 (0.30)	.
TextMate	1.10.2012 - 1.2.2016	173	0.15	-14 (0.15)** -1 (0.15) 0 (0.52)	.	.
Lime	1.12.2013 - 1.4.2015	63	0.24	-16 (0.31)** 0 (0.47)	.	.
Caret	1.10.2013 - 1.5.2016	133	0.17	0 (0.56) 15 (0.26)**	.	.
Zed	1.10.2013 - 1.10.2014	51	0.28	0 (0.70)	.	.
Video.js	1.1.2013 - 1.10.2015	142	0.17	0 (0.70)	0 (0.56)	0 (0.42)
Fabric	1.9.2011 - 1.5.2015	190	0.15	0 (0.68) 1 (0.25)	.	.

Kaikissa projekteissa näkyy voimakasta korrelaatiota viiveellä 0 kaikkien tehtävien kohdalla, mutta vain yhdessä näkyy korrelaatiota viiveillä 1-4. TextMatessa näkyy korrelaatiota negatiivisella viiveellä -1, mutta tälläkin kerralla korrelaatiokerroin on tarkalleen luottamusrajalla. Kaikissa tutkituissa virheraportteja ja ominaisuustoivetehtäviä sisältävissä projekteissa näkyy viive 0 vahvana, mutta mikään muu viive ei nouse luotettavasti esiin. Taulukkoon 7.6 on koottu tulokset impulssivastefunktion osalta.

Taulukko 7.6 *Kommenttien ja sulkeutuneiden tehtävien impulssivastefunktion tulokset*

Projekti	**Aikaväli**	**N**	**95%**	**Kaikki**	**Virheet**	**Toiveet**
Spacemacs	1.11.2014 - 1.4.2016	73	0.23	-14 (0.24)** 0 (0.62) 8 (0.23)**	-14 (0.26)** 0 (0.54) 8 (0.25)**	.
Neovim	1.4.2014 - 1.5.2016	108	0.19	0 (0.60)	-8 (0.19)* -4 (0.19)* 0 (0.34) 14 (0.20)**	.
TextMate	1.10.2012 - 1.2.2016	173	0.15	-14 (0.16)** 0 (0.54)	.	.
Lime	1.12.2013 - 1.4.2015	63	0.24	-16 (0.25)** 0 (0.54)	.	.
Caret	1.10.2013 - 1.5.2016	133	0.17	-10 (0.17)** 0 (0.60) 15 (0.27)**	.	.
Zed	1.10.2013 - 1.10.2014	51	0.28	0 (0.70)	.	.
Video.js	1.1.2013 - 1.10.2015	142	0.17	0 (0.69)	0 (0.57)	0 (0.44)
Fabric	1.9.2011 - 1.5.2015	190	0.15	0 (0.67)	.	.

Myös impulssivastefunktion puolella jokaisen projektin jokaisessa tutkitussa kohdassa erottuu viiveen 0 korrelaatio, mutta ei mitään muuta luotettavaa. TextMaten viiveellä -1 ei löydy korrelaatiota impulssivastefunktiolla tutkittuna toisin kuin ristikorrelaatiossa. Jos tehtävien kommentointi johtaa tehtävien sulkeutumiseen, tämä tapahtuu saman viikon sisällä eikä merkittävä osa tehtävistä vaadi tätä pidempää keskustelua. Kommentoitavat tehtävät eivät kuitenkaan välttämättä liity saman viikon aikana suljettaviin tehtäviin, vaan kyse voi olla yleisesti aktiivisesta kehityksestä.

Usean tehtävän rinnakkainen kommentoiminen on yleensä helpompaa kuin usean korjaavan kirjauksen tekeminen rinnakkain.

7.5 Auenneet tehtävät

Auenneiden tehtävien dataparilla tutkitaan hypoteesia 4 eli miten aiempien tehtävien aukeaminen liittyy uusien tehtävien aukeamiseen. Tämä on tutkituista datapareista alkeellisin, koska kyseessä on vain yksi dataluokka. Auenneita tehtäviä tutkitaan autokorrelaation, ristikorrelaation ja impulssivastefunktion näkökulmista. Kaikkien tehtävien sarakkeessa dataparin molemmat puolet muodostaa sama aikasarja eli kyseessä on autokorrelaatio. Virheraporttien ja ominaisuustoiveiden sarakkeissa kyseessä on taulukosta riippuen ristikorrelaatio tai impulssivastefunktio, koska dataparin puolet ovat erilaiset.

Autokorrelaation tulokset täytyy tulkita eri tavalla verrattuna muihin tuloksiin, vaikka käytetty laskukaava on sama kuin ristikorrelaatiolla. Autokorrelaatio kuvastaa pysyvyyttä tarkastellussa aikasarjassa eli mitä vähemmän autokorrelaatiota on, sitä sattumanvaraisemmin aikasarja etenee. Autokorrelaatiosta ei voi vetää yhtä luotettavia syy-seuraus epäilyjä kuin aidosta ristikorrelaatiosta, mutta vahva autokorrelaatio viittaa silti vahvaan ennususpotentiaaliin.

Virheraporttien ja ominaisuustoivetehtävien käyrät noudattavat samaa kehityskulkua kuin kaikki auenneet tehtävät paitsi silloin, kun niitä ei ole käytetty johdonmukaisesti projektissa. Projekteissa, joissa on tutkittu virheraportteja ja ominaisuustoiveita, on valittu aikaväli, jolla kehitys on ollut tasaista myös näiden osalta.

Autokorrelaatio tuottaa viiveellä 0 aina korrelaatiokertoimen 1.00, joten tulosten tulkinnassa ei keskitytä viiveeseen 0. Autokorrelaatio on myöskin symmetrinen molempiin suuntiin aika-akselia, joten vain positiiviset viiveet on merkitty. Taulukkoon 7.7 on koottu tulokset kaikkien tehtävien autokorrelaatiosta sekä kaikkien tehtävien ja virheraporttien sekä ominaisuustoivetehtävien välisestä ristikorrelaatiosta.

Taulukko 7.7 *Auenneiden tehtävien autokorrelaation ja ristikorrelaation tulokset*

Projekti	Aikaväli	N	95%	Kaikki	Virheet	Toiveet
Spacemacs	1.11.2014 - 1.4.2016	73	0.23	0 (1.00), 1 (0.57), 2 (0.47), 3 (0.37), 4 (0.37), 5 (0.41), 6 (0.37), 7 (0.38), 8 (0.31), 9 (0.33)	0 (0.65)	.
Neovim	1.4.2014 - 1.5.2016	108	0.19	0 (1.00), 1 (0.55), 2 (0.46), 3 (0.40), 4 (0.32), 5 (0.34), 6 (0.23)	0 (0.57)	.
TextMate	1.10.2012 - 1.2.2016	173	0.15	0 (1.00), 1 (0.23)	.	.
Lime	1.12.2013 - 1.4.2015	63	0.24	0 (1.00)	.	.
LightTable	1.12.2013 - 1.1.2016	107	0.19	0 (1.00), 1 (0.47), 2 (0.40), 3 (0.35), 4 (0.22), 5 (0.19)	.	.
Caret	1.10.2013 - 1.5.2016	133	0.17	0 (1.00), 1 (0.31), 2 (0.29), 3 (0.27), 6 (0.19)* 10 (0.24)**	.	.
Zed	1.10.2013 - 1.10.2014	51	0.28	0 (1.00), 1 (0.50), 3 (0.31), 4 (0.29)	.	.
Video.js	1.1.2013 - 1.10.2015	142	0.17	0 (1.00), 1 (0.22), 2 (0.23), 3 (0.18), 4 (0.29), 5 (0.24), 7 (0.21)*	0 (0.38) 13 (0.19)**	-14 (0.18)** 0 (0.44)
Fabric	1.9.2011 - 1.5.2015	190	0.15	0 (1.00), 1 (0.15), 9 (0.15)**	.	.
Vagrant	1.11.2011 - 1.6.2016, 1.10.2014 - 1.6.2016(V)	238 86	0.13 0.22	0 (1.00), 1 (0.63), 2 (0.50), 3 (0.44), 4 (0.50), 5 (0.38), 6 (0.39), 7 (0.40), 8 (0.50), 9 (0.44), 10 (0.36), 11 (0.29), 12 (0.34), 13 (0.32), 14 (0.28), 15 (0.26), 16 (0.25)	0 (0.71)	.

Taulukossa on jouduttu tiivistämään tuloksia kaksi yhdelle riville, että kaikki tulokset mahtuvat yhteen taulukkoon. Seitsemässä projektissa kymmenestä on luotettavaa autokorrelaatiota vähintään kolmella nollasta poikkeavalla viiveellä, kun tarkastellaan kaikkia tehtäviä. TextMatessa ja Fabricissa on lisäksi merkittävää autokorrelaatiota viiveellä 1. Vain yhdessä projektissa ei ole autokorrelaatiota, mutta silmämääräinen käyrän tutkiminen ei auta tulkitsemaan puutteen syytä.

Autokorrelaatio ulottuu puolessa projekteista yli kuukauden päähän ja kahdessa näistä yli kahteen kuukauteen. Tutkimalla kumulatiivisia käyriä silmämääräisesti vaikuttaa, että projektin pituus ja kehitystyön tasaisuus lisäävät autokorrelaation voimakkuutta. Tämä sopii autokorrelaation oletuksiin. Fabric on tästä ainoa poikkeus.

Siinä missä kaikki tehtävät autokorreloivat keskenään voimakkaasti, kaikkien tehtävien ja virheraporttien sekä ominaisuustoiveiden ristikorrelaatiot eivät nouse esiin merkittävästi muilla viiveillä kuin nollalla. Taulukkoon 7.8 on koottu tulokset impulssivastefunktion osalta.

Taulukko 7.8 *Auenneiden tehtävien impulssivastefunktion tulokset*

Projekti	**Aikaväli**	**N**	**95%**	**Kaikki**	**Virheet**	**Toiveet**
Spacemacs	1.11.2014 - 1.4.2016	73	0.23	0 (1.00)	0 (0.66)	.
Neovim	1.4.2014 - 1.5.2016	108	0.19	0 (1.00)	0 (0.55)	.
Video.js	1.1.2013 - 1.10.2015	142	0.17	0 (1.00)	0 (0.42) 13 (0.17)**	-14 (0.17)** 0 (0.48)
Vagrant	1.11.2011 - 1.6.2016, 1.10.2014 - 1.6.2016(V)	238 86	0.13 0.22	0 (1.00)	0 (0.71)	.

Impulssivastefunktion tuloksista on poistettu kaikki projektit, joissa ei tutkittu virheraportteja ja ominaisuustoiveita. Aikasarjan autokorrelaatio poistuu laskettaessa impulssivastefunktiota samasta aikasarjasta, joten kaikkien tehtävien sarake ei tuota järkeviä tuloksia eikä sitä voi tulkita. Impulssivastefunktion tulokset virheraporttien ja ominaisuustoivetehtävien osalta vastaavat ristikorrelaation tuloksia eli ainoastaan viive 0 on edustettuna. Tämä voi viitata siihen, etteivät aukeavat tehtävät aiheuta uusia aukeavia tehtäviä, vaan kaikkien tehtävien autokorrelaatio johtuu vakaana pysyvästä kehityskulusta.

Jos aiempien tehtävien aukeaminen liittyy uusiin aukeaviin virheraportteihin tai ominaisuustoiveisiin, yhteys on korkeintaan viikon aikajänteellä. Yhden viikon sisällä aukeaviin tehtäviin ei todennäköisesti liity syy-seuraussuhdetta, koska nopealla kehitystahdilla avataan joka tapauksessa paljon erilaisia tehtäviä.

Kaikkien tehtävien autokorrelaatiolla on selvästi voimakkain ennustuspotentiaali kaikista tutkituista datapareista. Sen avulla ei kuitenkaan voida ennustaa yllättäviä muutoksia tehtävien kehityskulussa, koska yllättävät muutokset rikkovat suoraan autokorrelaation pysyvyyttä.

8. TULOSTEN ARVIOINTI

8.1 Yhteenveto tuloksista

Kaikki tulokset on koottu kahteen taulukkoon vertailun helpottamiseksi ja kokonaiskuvan tulkitsemiseksi. Ristikorrelaation tulokset on koottu yhteen taulukkoon ja impulssivastefunktion tulokset toiseen. Tuloksiin on hyväksytty vain luotettavaksi tulkitut luottamusrajan ylittävät tulokset eli kaikki huutomerkeillä ja tähdillä merkityt tulokset on poistettu. Myös korrelaatiokertoimet on jätetty pois, joten jäljellä on vain merkitsevät viivearvot. Taulukon tiivistämiseksi sarakkeiden otsikot on merkitty seuraavilla lyhenteillä:

K/A Kirjaukset ja kaikki auenneet tehtävät.

K/Av Kirjaukset ja auenneet virheraportit.

K/At Kirjaukset ja auenneet ominaisuustoivetehtävät

pK/S Pienet kirjaukset ja kaikki sulkeutuneet tehtävät.

pK/Sv Pienet kirjaukset ja sulkeutuneet virheraportit.

pK/St Pienet kirjaukset ja sulkeutuneet ominaisuustoivetehtävät.

K/S Kommentit ja kaikki sulkeutuneet tehtävät.

K/Sv Kommentit ja sulkeutuneet virheraportit.

K/St Kommentit ja sulkeutuneet ominaisuustoivetehtävät.

A/A Kaikkien auenneiden tehtävien autokorrelaatio.

A/Av Kaikki auenneet tehtävät ja auenneet virheraportit.

A/At Kaikki auenneet tehtävät ja auenneet ominaisuustoivetehtävät.

Myös aikavälit, otoskoot ja luottamusrajat on jätetty pois. Taulukoissa — tarkoittaa ettei korrelaatiokerroin ylittänyt millään viivearvolla luottamusrajaa. Vastaavasti piste tarkoittaa ettei tulosta ole voitu laskea, koska dataa oli joko absoluuttisesti liian vähän tai data ei muodostanut yhtenäistä tarkasteltavaa ajanjaksoa. Taulukkoon 8.1 on koottu kaikki ristikorrelaation tulokset.

Taulukko 8.1 *Yhteenveto ristikorrelaation merkitsevistä tuloksista*

Projekti	K/A	K/Av	K/At	pK/S	pK/Sv	pK/St	K/S	K/Sv	K/St	A/A	A/Av	A/At
Spacemacs	0	0	.	0	0	.	0	0	.	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0	.
Neovim	0	—	.	-1 1	—	.	0	0	.	0, 1, 2, 3, 4, 5, 6	0	.
TextMate	0 3	.	.	0 3	.	.	-1 0	.	.	0, 1	.	.
Lime	0	.	.	0	.	.	0	.	.	0	.	.
LightTable	0 1 3	.	.	.	.	.	.	.	.	0, 1, 2, 3, 4, 5	.	.
Caret	0	.	.	0	.	.	0	.	.	0, 1, 2, 3	.	.
Zed	0 3 4	.	.	0	.	.	0	.	.	0, 1, 3, 4	.	.
Video.js	—	—	0	0	0	0	0	0	0	0, 1, 2, 3, 4, 5	0	0
Fabric	—	.	.	0	.	.	0 1	.	.	0, 1	.	.
Vagrant	0 1	—	.	0	0	.	.	.	.	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16	0	.

Kaikkien auenneiden tehtävien autokorrelaation tulokset nousevat ylivoimaisesti selkeimmin esiin, kun verrataan sarakkeita toisiinsa. Autokorrelaation merkitsevät viivearvot on tämän vuoksi jouduttu tiivistämään niin, että yhdellä rivillä voi olla kaksi merkitsevää viivearvoa. Minkään muun sarakkeen tulos ei näyttäisi olevan johdonmukaisesti yhteinen niille projekteille, joilla näkyy voimakasta autokorrelaatiota. Kirjausten ja kaikkien auenneiden tehtävien tuloksissa on toiseksi eniten nollasta poikkeavia viivearvoja.

Kun verrataan yksittäisten projektien kaikkia tuloksia toisiinsa, ei vaikuta, että yhdessä sarakkeessa esiintyvä viivearvo näkyisi säännönmukaisesti kaikissa saman projektin sarakkeissa. Viive 0 on tästä ainoa poikkeus. Esimerkiksi vaikka TextMatessa viive 3 nousee esiin sekä kaikilla kirjauksilla että pienillä kirjauksilla, Zedissä viiveet 3 ja 4 sekä Vagrantissa viive 1 eivät näy vastaavalla tavalla muissa sarakkeissa. Taulukkoon 8.2 on koottu vastaavasti impulssivastefunktion tulokset.

Taulukko 8.2 *Yhteenveto impulssivastefunktion merkitsevistä tuloksista*

Projekti	K/A	K/Av	K/At	pK/S	pK/Sv	pK/St	K/S	K/Sv	K/St	A/A	A/Av	A/At
Spacemacs	0	0	.	0	0	.	0	0	.	0	0	.
Neovim	0	—	.	—	—	.	0	0	.	0	0	.
TextMate	0	.	.	0 3	.	.	0	.	.	0	.	.
Lime	0	.	.	0	.	.	0	.	.	0	.	.
LightTable	0 1	.	.	.	.	.	.	.	.	0	.	.
Caret	0	.	.	0	.	.	0	.	.	0	.	.
Zed	0 4	.	.	0	.	.	0	.	.	0	.	.
Video.js	0	—	—	0	0	0	0	0	0	0	0	0
Fabric	—	.	.	0	.	.	0	.	.	0	.	.
Vagrant	0 1 4	—	.	0	0	.	.	.	.	0	0	.

Impulssivastefunktion puolella tilanne on samanlainen kuin ristikorrelaatiossa eli

nollasta poikkeavia viiveitä ei näy järjestelmällisesti projektien sisällä. Nollasta poikkeavia viiveitä on vain neljässä taulukon solussa eli puolet vähemmän kuin ristikorrelaatiossa, jos autokorrelaation tuloksia ei lasketa. Soluja, joissa ei ole ollenkaan merkitseviä tuloksia on saman suuntainen määrä eli impulssivastefunktiolla 7 ja ristikorrelaatiolla 6.

Ennustuspotentiaalin löytyminen tehtävien autokorrelaatiosta ei yllätä, koska virheraporttien autokorrelaatio on osoitettu toimivaksi ennustuksen pohjaksi erilaisten projektien kohdalla. Raja et al. tutkivat kymmentä ohjelmistoprojektia ARIMA-malliin (eng. autoregressive integrated moving average model) perustuvalla ennustusmenetelmällä ja huomasivat, että kaikissa tutkituissa projekteissa oli mahdollista ennustaa uusien virheraporttien ilmestymistä vanhan virheraporttidatan avulla [35]. Tutkimus keskittyi virheraportteihin, mutta vaikuttaa, että ARIMA-mallilla voisi olla mahdollista ennustaa myös kaikkien tehtävien aukeamista.

Zou et al. ennustivat virheraporttien ilmestymistä erilaisten parametrittömien tekniikoiden avulla myöskin käyttämällä datana vanhoja virheraportteja. Tutkimuksen mukaan GAM (eng. generalized additive model) ja ES (eng. exponential smoothing) tekniikoilla on mahdollista ennustaa uusien virheraporttien ilmestymistä ainakin joissain avoimen lähdekoodin projekteissa. [44]

Myös Shin et al. suosittelevat historiadatan käyttämistä ennustamiseen. He vertasivat virheraporttien ennustamisen tarkkuutta kahdella eri datalla: virheraporttien historiadatalla ja koodin kutsurakennedatalla. Tutkimuksessa pelkällä historiadatalla ennustaminen oli käytännössä yhtä tarkkaa kuin kutsurakennedatan ja historiadatan yhdistelmällä ennustaminen. Pelkän kutsurakennedatan käyttäminen virheraporttien ennustamiseen oli epätarkinta. [38]

On jossain määrin yllättävää kuinka heikko ennustuspotentiaali on käytettäessä muita dataluokkia ennustavana muuttujana. Tehtäviä on aiemmin onnistuttu ennustamaan käyttämällä esimerkiksi lähdekoodin muutosten monimutkaisuutta ennustavana muuttujana [39]. Tutkimuksia, joissa käytetään suoraan esimerkiksi kirjausmääriä ennustavana muuttujana ei kuitenkaan löytynyt, joten tarkkaa vertailua ei voida tehdä. Voi olla, että kirjausdata on liian yleisluontoista, että sitä voisi käyttää ennustamiseen.

Toinen yllättävä tulos on kuinka vahvaksi viiveen 0 korrelaatio osoittautui suurimmassa osassa projekteja ja datapareja. Pienten kirjausten ja sulkeutuneiden tehtävien kohdalla tulos on odotettavissa, jos virheenkorjauksia ei testata erikseen ja korjaavan kirjauksen jälkeen virheraportti suljetaan heti. Muiden dataparien kohdalla ei ole vastaavanlaista välitöntä yhteyttä puoliskojen välillä. Silti tuloksissa

viiveen 0 yhteys vaikuttaa yleisluontoiselta.

Viiveen 0 yleisyyden taustalla voi olla eri ilmiöitä, joita voidaan spekuloida, mutta ei varmistaa tämän tutkimuksen puitteissa. Vaihtoehtoisia selityksiä on kaksi: joko tarkasteltujen aikasarjojen välillä on viivettä yhden viikon sisällä tai sitten kirjauksia, kommentteja ja tehtäviä käsitellään saman työrupeaman aikana eli ilman mitään viivettä. Käsitellyt tehtävät voivat olla liitoksissa samaan aikaan tehtyihin kirjauksiin ja kommentteihin, tai ne voivat olla erillisiä. Myös molempia tilanteita voi esiintyä. Pelkillä kirjausten, kommenttien ja tehtävien kokonaismäärillä ei voida tehdä yksityiskohtaisempia johtopäätöksiä.

Tuloksia voidaan pitää yleisesti ottaen luotettavina, koska tutkittuja projekteja oli yhteensä kymmenen. Tietyt virhelähteet voivat kuitenkin vaikuttaa tuloksiin vaihtelevassa määrin. Suurin teoriassa mahdollinen virhelähde on väärin toteutettu matemaattinen menetelmä. Ristikorrelaation algoritmi sekä puhdistukseen käytetty autoregressiivinen malli tulivat kolmannelta osapuolelta ja muu matemaattinen lähdekoodi kirjoitettiin itse. Tulokset kuitenkin näyttivät testeissä uskottavilta, joten tämän virheen todennäköisyys on pieni.

Toinen virhelähde löytyy tutkittujen aikavälien valinnasta. Osassa projekteissa tapahtuu matemaattisen menetelmän näkökulmasta huomattavia muutoksia aikasarjojen keskiarvoissa kehitystahdin kiihtyessä ja hidastuessa, millä voi olla vaikutusta tuloksiin. Tätä kuitenkin tapahtuu myös oikeissa ennustettavissa projekteissa eikä aikaväliä voi jättää liian lyhyeksikään.

Kolmas virhelähde liittyy käsin valikoituihin virheraporttien ja ominaisuustoiveiden merkintöihin, jotka eivät välttämättä vastaa todellista tilannetta. Sekä käsin valitsemisessa että projektihenkilöstön merkitsemiskäytännöissä voi olla ongelmia. Tämä vaikuttaa vain virheraporttien ja ominaisuustoivetehtävien tutkimiseen.

8.2 Johtopäätökset

Taulukkoon 8.3 on koottu tulokset jokaisen hypoteesin päädataparista sekä ristikorrelaatiolla että impulssivastefunktiolla laskettuna. Taulukon prosenttiosuus ilmaisee, kuinka suuressa osassa tutkituista projekteista löytyy merkittävää yhteyttä dataparin välillä. Taulukkoon on valittu viive 0, koska se osoittautui merkittävimmäksi yksittäiseksi viivearvoksi sekä viiveväli 1-4, koska ennustaminen on luotettavinta lyhyellä aikavälillä. Taulukossa viiveen 1-4 prosenttiosuus sisältää kaikki projektit, joissa on vähintään yhdellä näistä viiveistä merkittävää yhteyttä dataparin välillä.

Taulukkoon on koottu vain kaikkiin auenneisiin ja kaikkiin sulkeutuneisiin tehtäviin liittyvät tulokset, koska näitä tutkittiin suurimmalla määrällä projekteja ja siten tulosten luotettavuus on suurin. Hypoteesia 4 vastaavasta dataparista on jätetty pois triviaalit tulokset.

Taulukko 8.3 *Merkittäviä tuloksia sisältävien projektien osuus tutkituista projekteista*

	Ristikorr. Viive 0	**Impulssiv. Viive 0**	**Ristikorr. Viive 1-4**	**Impulssiv. Viive 1-4**
Kirjaukset / Auenneet tehtävät (Hypoteesi 1)	80%	90%	40%	30%
Pienet kirjaukset / Sulkeutuneet tehtävät (Hypoteesi 2)	89%	89%	22%	11%
Kommentit / Sulkeutuneet tehtävät (Hypoteesi 3)	100%	100%	13%	0%
Auenneet tehtävät / Auenneet tehtävät (Hypoteesi 4)	—	—	90%	—

Tulokset ovat erilaisia eri hypoteesien kohdalla. Hypoteesien 1-3 eli kirjaus-tehtävä ja kommentti-tehtävä parien väliltä ei löydy yleisluontoista viiveellistä yhteyttä, jonka pohjalta voisi ennustaa tehtävien muutoksia viikkojen aikaskaalalla. Hypoteesin 1 kohdalla näyttäisi, että kehitystyö ei vaikuta uusien virheraporttien ja muiden tehtävien aukeamiseen yhtä paljon kuin jokin muu asia tai yhdistelmä asioita. Kehityksen seurauksena muodostuneiden virheiden löytyminen saattaa jakaantua paljon pidemmällä aikavälillä, jolloin mikään yksittäinen hetki ei nouse esiin. Hypoteesin 2 kohdalla vaikuttaa, että virheitä korjaavat kehittäjät sulkevat virheraportin tai muun pientä lähdekoodimuutosta vaativan tehtävän heti kirjauksen teon jälkeen, jolloin viivettä ei synny.

Hypoteesista 3 voidaan epäillä, että suurin osa tehtävistä ei vaadi ajallisesti pitkää keskustelua. Kun tehtävä otetaan käsittelyyn, siitä todennäköisesti keskustellaan nopeasti, jonka jälkeen tehtävä saadaan valmiiksi. Hypoteesin 4 tutkiminen kuitenkin osoittaa auenneiden tehtävien autokorrelaation olevan riittävän voimakas, että useimmissa projekteissa sitä voidaan käyttää ennustamiseen. Luvussa 7 esitettyjen yksityiskohtaisten tulosten perusteella ennustettavuudessa on selviä projekti-

kohtaisia eroja.

Auenneiden tehtävien autokorrelaation käyttämisessä ennustamiseen on rajoitteensa. Autokorrelaatio ei kykene ennustamaan yllättäviä muutoksia aikasarjassa, koska autokorrelaatio perustuu aikasarjan vakauteen. Tämän vuoksi auenneiden tehtävien avulla voidaan arvioida projektin tulevaisuuden tilannetta vain, jos tilanne jatkuu samanlaisena. Muutoksiin pitää varautua asiantuntija-arvion avulla eli esimerkiksi projektipäällikön arviolla.

Kaikkiin muutoksiin ei voida varautua samalla tavalla, koska tehtäviä voi projektista riippuen avata joko kehittäjät, käyttäjät tai molemmat. Tietoisen kehitysresurssien lisäämisen tai poistamisen vaikutus on helppo ennakoida, mutta välillä kehitysryhmässä tapahtuu myös onnettomuuksia tai muita yllättäviä muutoksia. Käyttäjien vaikutusta on vaikeampi ennakoida, koska käyttäjien määrään ja aktiivisuuteen ei voida vaikuttaa helposti. Monessa projektissa pitäisi kuitenkin olla mahdollista ennakoida tilanteen muutoksia tyydyttävällä tasolla.

Ohjelmistoprojektien johtajien on hyödyllistä ymmärtää alkeellisten ohjelmistotuotannon välituotteiden kuten tehtävien, kirjausten ja kommenttien välisiä yhteyksiä ja ottaa tämä huomioon suunnittelutyössä. Viiveen 0 korrelaatiosta kirjausten ja tehtävien välillä voidaan vetää yleisluontoinen johtopäätös, jonka voi myös yhdistää autokorrelaation avulla ennustamiseen: projektipäällikkö voi ennustaa millä tahdilla uusia tehtäviä ilmaantuu käyttämällä suunniteltua kehitystahtia ennustavana muuttujana. Tämä toimii, koska auenneiden tehtävien käyrä noudattaa kirjausten käyrän muutoksia.

Jos työtahti eli kirjaustentekotahti pysyy samana, myös tehtävien aukeamistahti pysyy todennäköisesti samana. Jos työtahti hidastuu, tehtäviäkin aukeaa vähemmän ja päinvastoin työtahdin kiihtyessä. Myös muut tutkijat suosittelevat vastaavankaltaista numeroiden ja asiantuntija-arvion yhdistämistä ohjelmistotuotannon ennustamiseen [18]. Muutos tehtävien aukeamisessa on pienempi kuin kirjauksissa ja tarkka vaikutus lienee projektikohtainen. Voi kuitenkin olla mahdollista arvioida tietyn suuruisen työtahdin kiihtymisen vaikutus aukeaviin tehtäviin. Tässä on potentiaalia jatkotutkimukselle.

Työtahdin muutosten ennakoinnin pitäisi olla helpompaa kuin aukeavien tehtävien tilanteen arvioinnin, koska käyttäjät eivät tee kirjauksia. Projektin valmiit kehityssuunnitelmat ja aikataulut ovat myös ennakoinnin apuna. Voikin olla hyödyllistä arvioida ylläpitoon ja korjauksiin kuluvaa aikaa jo projektin alkuvaiheen suunnittelussa käyttämällä suunniteltua kehitystahtia suunnanantajana. Tämän aihepiirin jatkotutkimuksessa olisi myös kiinnostavaa yrittää ennustaa pelkän projekti-

suunnitelman perusteella ylläpitoon ja korjaukseen kuluvia resursseja sekä työn jakautumista aika-akselilla.

Auenneiden tehtävien autokorrelaation avulla on mahdollista ennustaa projektin etenemistä ilman monimutkaista dataa tai algoritmeja. Tätä hyödyntämällä GitHub ja muut tehtävienhallintajärjestelmiä tarjoavat palvelut voisivat tarjota suuntaa antavan arvion tehtävien tulevasta kehityksestä. Esimerkiksi GitHubissa on jo nyt erilaisia yksinkertaisia tilastonäkymiä tehdystä työstä. Näiden ominaisuuksien rinnalle sopisi myös tulevaisuuden tarkastelu.

Ohjelmistoprojektit käyttävät harvoin ulkopuolisia ennustustyökaluja, joten ennustuksen integroiminen tehtävienhallintajärjestelmään olisi helpoin keino saada ennustus osaksi ohjelmistotuotannon työtapoja. Tämän avulla projektipäällikkö pystyisi esimerkiksi arvioimaan kannattaako seuraava kehitysjakso käyttää virheiden korjaamiseen vai uusien ominaisuuksien kehittämiseen, ja miten tämä vaikuttaa auki olevien tehtävien määrään myöhemmin.

Voimakas autokorrelaatio viittaa aikasarjan vakauteen eli autokorreloivan aikasarjan eteneminen ei ole sattumanvaraista. Tehtävien aukeamiseen liittyvän jatkuvuuden syitä ei ole mahdollista selvittää tämän tutkimuksen keinoilla, mutta aiheesta olisi mahdollista tehdä jatkotutkimus. Vaikka tästä ei välttämättä olisi hyötyä keskitetylle projektipäällikölle, parempi ymmärrys tehtävien aukeamiskäytännöistä voisi auttaa hajautetusti johdettuja kehitysryhmiä. Tällöin yhdellä henkilöllä ei ole kokonaisvaltaista projektikohtaista näkemystä projektin etenemisestä, jolloin yleisten aukeamiskäytäntöjen ymmärtäminen auttaa koko kehitysryhmää.

Oletettavasti tehtävien aukeamistahtiin voi vaikuttaa kehitystahdin lisäksi muun muassa kehitysvaihe, käyttäjien määrä ja aktiivisuustaso, kehitysryhmän koko, lähdekoodin laatu sekä projektikäytännöt. Mikään näistä ei yleensä muutu jatkuvasti, nopealla tahdilla tai yllättäen, mikä voisi selittää auenneiden tehtävien aikasarjan vakautta.

Vaikka näin alkeellisella datalla ei voida ennustaa yhden dataluokan pohjalta toisen muutoksia, on mahdollista, että tarkemmalla datalla siihen pystyttäisiin. Esimerkiksi olemassa olevilla virheraporttien luokittelumenetelmillä[42] olisi mahdollista saada parempi käsitys siitä, mitkä tehtävät ovat oikeita virheraportteja. Tämä olisi taas mahdollista yhdistää tietoon siitä oliko kirjaus lisäävä, poistava vai muokkaava kuten aiemmin on jo tehty toisenlaisen tutkimuksen kontekstissa [34]. Näin voisi esimerkiksi yrittää ennustaa lähdekoodirivejä lisäävien kirjausten vaikutusta varmojen virheraporttien aukeamiseen.

Kiinnostavampi suunta jatkotutkimukselle olisi kuitenkin teollisuuden suljetun lähdekoodin projektien analysointi. Tällöin olisi käytössä varma tieto muun muassa käytetyistä projektikehyksistä ja tehtävien merkinnöistä. Lisäksi teollisuuden projektien kehitystahti on todennäköisesti tasaisempi ja siten helpompi ennustaa verrattuna avoimen lähdekoodin harrasteprojekteihin. Tällaisen tutkimuksen tulokset voisivat valaista myös avoimen ja suljetun lähdekoodin projektien välillä olevia kehitystapaeroja.

9. YHTEENVETO

Ohjelmistoprojektien käyttämät työkalut keräävät paljon dataa ohjelmistoprosessista normaalin käytön sivutuotteena. Tällaista dataa ovat muun muassa versiohallintaan tehdyt kirjaukset, erilaiset tehtävät sekä näiden kommentit. Ohjelmistotuotantoprosessiin liittyvää dataa voidaan kerätä useista datalähteistä kuten versio- ja tehtävienhallintajärjestelmistä. Näitä järjestelmiä tarjoavat esimerkiksi avoimen lähdekoodin projekteja isännöivät palvelut. Yhteen tietokantaan kerättyä dataa voidaan visualisoida uusien havaintojen tekemiseksi ja olemassa olevien hypoteesien varmistamiseksi.

Erilaiset aikasarjamenetelmät sopivat tilastollisen tutkimisen lisäksi ohjelmistotuotannon ennustamiseen. Yleisiä ennustuskohteita ovat muun muassa projektin eteneminen ja kustannukset sekä ohjelmiston virheet. Ennustaminen ei ole vielä vakiintunut käytäntö ohjelmistotuotannossa, eivätkä monet ennusteet ole kovinkaan tarkkoja. Siksi on oleellista päättää, mitä dataa yritetään käyttää ennustamiseen.

Tämän tutkimuksen tavoitteena oli selvittää voiko avoimen lähdekoodin ohjelmistoprojektin tehtävien määrän kasvua ennustaa käyttämällä automaattisesti louhittua kirjaus-, kommentti- ja tehtävädataa. Kymmenestä avoimen lähdekoodin projektista päätettiin etsiä ennustuspotentiaalia viiveellisen yhteyden avulla, koska aiempi ohjelmistoanalytiikan tutkimus ei ole keskittynyt sellaisen etsimiseen alkeellisilla datapareilla. Ennustuspotentiaalin etsiminen jaettiin neljään hypoteesiin, joita tutkittiin viiveellisen yhteyden löytämiseksi kirjaus-tehtävä, kommentti-tehtävä ja tehtävä-tehtävä pareilla. Tehtävä-tehtävä paria käytettiin myös autokorrelaation etsimiseen.

Tutkimuksessa käytettiin apuna projektien datan visualisointia, jonka avulla huomattiin ettei datakäyrissä näy silmämääräistä ennustukseen soveltuvaa viivettä. Tarkempaa tutkimusta varten valittiin kaksi signaalinkäsittelyn menetelmää: ristikorrelaatio ja impulssivastefunktio. Näiden menetelmien tuloksia voidaan käsitellä tilastollisesti, joten tulosten merkittävyyttä tutkittiin luottamusvälin avulla. Tutkimuksessa käytettiin 95% luottamusväliä eli tulokset johtuvat sattumasta korkeintaan 5% todennäköisyydellä.

Tutkitut kymmenen projektia valittiin tuotekehitysluonteen, vertailukelpoisuuden sekä riittävän yhtäjaksoisen kehityksen perusteella. Valittujen projektien tuotetyypit lisäävät vertailukelpoisuutta, koska seitsemän kymmenestä projektista tuottaa tekstinkäsittelyohjelmistoa. Kolme muuta projektia valittiin virheraporttien ja ominaisuustoiveiden määrän perusteella, koska alkuperäisissä projekteissa ei ollut riittävästi dataa näiden tutkimiseen. Virheraporttien ja ominaisuustoiveiden tutkimista haittasi myös niiden ilmestymistahti, joka oli harvoin yhtäjaksoinen. Niiden tutkiminen jäi toivottua suppeammaksi.

Tutkittava data koottiin viikkotasolle eli tutkimuksessa yksi viiveaskel vastaa yhtä kehitysviikkoa. Hypoteeseihin liittyviä datapareja tutkittiin kaikkien tehtävien, virheraporttien ja ominaisuustoiveiden pohjalta ristikorrelaatiolla ja impulssivastefunktiolla. Tuloksissa kävi ilmi, ettei eri dataa sisältävissä datapareissa ole yleistä viiveellistä yhteyttä. Parhaimmankin dataparin kohdalla vain 40 prosentissa tutkituista projekteista löytyi merkittävää yhteyttä yhdestä neljään viikkoon ulottuvalla aikajänteellä. Tätä viiveväliä 1-4 pidettiin riittävän lyhyenä aikavälinä luotettavaan ennustamiseen.

Suurimmassa osassa projekteista löytyi kuitenkin välitöntä korrelaatiota dataparien välillä, mikä näkyy tuloksissa viiveellä 0. Tutkimuksessa käsiteltävillä datan kokonaismäärillä ei ole mahdollista selvittää, onko dataparin välillä viivettä alle viikon aikajänteellä vai käsitelläänkö viikon sisällä dataparin molempia osapuolia ilman, että ne liittyvät toisiinsa. Tämän perusteella projektipäällikön on kuitenkin mahdollista ennustaa projektin etenemistä suunnitellun etenemisen pohjalta: jos projektin kehitystahti kiihtyy, todennäköisesti myös tehtävien aukeamistahti kiihtyy.

Lisäksi tehtävien aukeamisesta löytyi voimakasta autokorrelaatiota 90 prosentissa tutkituista projekteista. Tämä tarkoittaa, ettei tehtävien aukeamistahti ole sattumanvarainen, vaan siihen liittyy tavalla tai toisella jatkuvuutta. Tässä tutkimuksessa ei kuitenkaan ollut mahdollista selvittää, mistä jatkuvuus johtuu. Autokorrelaation tulosten perusteella ei ole mahdollista ennustaa yllättäviä muutoksia projektin etenemisessä, mutta sillä saadaan suuntaa antava arvio samalla kehitystahdilla jatkuvalle projektille. Yksinkertaisen luonteensa vuoksi autokorrelaatioon perustuva ennustaminen olisi mahdollista integroida osaksi avoimen lähdekoodin projekteja isännöiviä palveluita.

Kokonaisuudessaan tutkimus onnistui ja antoi vastauksia tutkimuskysymykseen, vaikka hypoteesien tarkkoja syy-seuraussuhteita ei tässä tutkimuksessa voitu varmistaa. Projektin etenemisen ennustamiseen kannattaa harkita yksityiskohtaisempaa dataa kuin projektin kaikkia kirjauksia ja kommentteja, vaikka näillä onkin

yhteyttä tehtäviin. Koska tehtävät eivät tulosten perusteella aukea sattumanvaraisella tahdilla, tahtiin vaikuttava säännönmukaisuuden lähde voi olla mahdollista löytää tulevissa tutkimuksissa.

LÄHTEET

[1] O. Arafat ja D. Riehle, "The commit size distribution of open source software," *2009 42nd Hawaii International Conference on System Sciences*, Tammikuu 2009, ss. 1–8.

[2] J. S. Armstrong, *Principles of Forecasting: A Handbook for Researchers and Practitioners*, 1st ed. Springer Science & Business Media, 2001, vol. 30, 850 s.

[3] A. Bachmann ja A. Bernstein, "Software process data quality and characteristics: A historical view on open and closed source projects," *Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops*, ser. IWPSE-Evol '09. New York, NY, USA: ACM, 2009, ss. 119–128. Saatavilla: http://doi.acm.org/10.1145/1595808.1595830

[4] S. Bassil ja R. K. Keller, "Software visualization tools: survey and analysis," *Proceedings 9th International Workshop on Program Comprehension. IWPC 2001*, 2001, ss. 7–17.

[5] D. Bertram, A. Voida, S. Greenberg, ja R. Walker, "Communication, collaboration, and bugs: The social nature of issue tracking in small, collocated teams," *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '10. New York, NY, USA: ACM, 2010, ss. 291–300. Saatavilla: http://doi.acm.org/10.1145/1718918.1718972

[6] A. Capiluppi, P. Lago, ja M. Morisio, "Characteristics of open source projects," *Seventh European Conference on Software Maintenance and Reengineering, 2003. Proceedings.*, Maaliskuu 2003, ss. 317–327.

[7] C. Chatfield, *The Analysis of Time Series: An Introduction*, 6th ed. Florida, US: CRC Press, 2004, 352 s.

[8] K. K. Chaturvedi, V. B. Sing, ja P. Singh, "Tools in mining software repositories," *2013 13th International Conference on Computational Science and Its Applications*, Kesäkuu 2013, ss. 89–98.

[9] M. D'Ambros, M. Lanza, ja R. Robbes, "An extensive comparison of bug prediction approaches," *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, Toukokuu 2010, ss. 31–41.

[10] M. D'Ambros ja M. Lanza, "Visual software evolution reconstruction," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 21, no. 3, ss. 217–232, 2009. Saatavilla: http://dx.doi.org/10.1002/smr.407

[11] I. F. de Barcelos Tronto, J. D. S. da Silva, ja N. Sant'Anna, "An investigation of artificial neural networks based prediction systems in software project management," *Journal of Systems and Software*, vol. 81, no. 3, ss. 356 – 367, 2008, selected Papers from the 2006 Brazilian Symposia on Databases and on Software Engineering. Saatavilla: http://www.sciencedirect.com/science/article/pii/S016412120700132X

[12] R. Ferenc, *Bug Forecast: A Method for Automatic Bug Prediction.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ss. 283–295. Saatavilla: http://dx.doi.org/10.1007/978-3-642-17578-7_28

[13] M. Fischer, M. Pinzger, ja H. Gall, "Populating a release history database from version control and bug tracking systems," *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.*, Syyskuu 2003, ss. 23–32.

[14] C. Fitzgerald, E. Letier, ja A. Finkelstein, "Early failure prediction in feature request management systems: an extended study," *Requirements Engineering*, vol. 17, no. 2, ss. 117–132, 2012. Saatavilla: http://dx.doi.org/10.1007/s00766-012-0150-7

[15] D. A. Grier, "The github effect," *Computer*, vol. 48, no. 5, ss. 116–116, Toukokuu 2015.

[16] G. Guillaume-Joseph ja J. S. Wasek, "Improving software project outcomes through predictive analytics: Part 1," *IEEE Engineering Management Review*, vol. 43, no. 3, ss. 26–38, Maaliskuu 2015.

[17] ——, "Improving software project outcomes through predictive analytics: Part 2," *IEEE Engineering Management Review*, vol. 43, no. 3, ss. 39–49, Maaliskuu 2015.

[18] A. E. Hassan, A. Hindle, P. Runeson, M. Shepperd, P. Devanbu, ja S. Kim, "Roundtable: What's next in software analytics," *IEEE Software*, vol. 30, no. 4, ss. 53–56, Heinäkuu 2013.

[19] A. Hindle, D. M. German, ja R. Holt, "What do large commits tell us?: A taxonomical study of large commits," *Proceedings of the 2008 International Working Conference on Mining Software Repositories*, ser. MSR '08. New York, NY, USA: ACM, 2008, ss. 99–108. Saatavilla: http://doi.acm.org/10.1145/1370750.1370773

[20] O. Hylli, A.-L. Mattila, ja K. Systä, *Collecting issue management data for analysis with a unified model and API descriptions*, ser. CEUR Workshop Proceedings, 10 2015, ss. 251–265.

[21] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, ja D. Damian, "An in-depth study of the promises and perils of mining github," *Empirical Software Engineering*, vol. 21, no. 5, ss. 2035–2071, 2016. Saatavilla: http://dx.doi.org/10.1007/s10664-015-9393-5

[22] A. Kaur, K. Kaur, ja D. Chopra, "An empirical study of software entropy based bug prediction using machine learning," *International Journal of System Assurance Engineering and Management*, ss. 1–18, 2016. Saatavilla: http://dx.doi.org/10.1007/s13198-016-0479-2

[23] D. A. Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, ss. 1–8, Tammikuu 2002.

[24] M. Kendall, A. Stuart, ja J. Ord, *The Advanced Theory of Statistics*, 4th ed. London: Griffin, 1983, vol. 3.

[25] J. Loutre, "Timeseries analysis module," 2015, Saatavilla: https://www.npmjs.com/package/timeseries-analysis [Viitattu 2.2.2017].

[26] M. Marzban, Z. Khoshmanesh, ja A. Sami, *Cohesion Between Size of Commit and Type of Commit*. Dordrecht: Springer Netherlands, 2012, ss. 231–239. Saatavilla: http://dx.doi.org/10.1007/978-94-007-2792-2_22

[27] A.-L. Mattila, P. Ihantola, T. Kilamo, A. Luoto, M. Nurminen, ja H. Väätäjä, "Software visualization today: Systematic literature review," *Proceedings of the 20th International Academic Mindtrek Conference*, ser. AcademicMindtrek '16. New York, NY, USA: ACM, 2016, ss. 262–271. Saatavilla: http://doi.acm.org/10.1145/2994310.2994327

[28] A.-L. Mattila, T. Lehtonen, H. Terho, T. Mikkonen, ja K. Systä, *Mashing Up Software Issue Management, Development, and Usage Data*. The Institute of Electrical and Electronics Engineers, Inc., 7 2015, ss. 26–29.

[29] A.-L. Mattila, A. Luoto, H. Terho, J. Hylli, O. Sievi-Korte, ja K. Systä, *Unified Model for Software Engineering Data*. IEEE, 9 2015, ss. 150–154.

[30] A.-L. Mattila, K. Systä, O. Sievi-Korte, M. Leppänen, ja T. Mikkonen, *Discovering Software Process Deviations Using Visualizations*, 5 2017, ss. 1–8.

[31] T. Menzies ja T. Zimmermann, "Software analytics: So what?" *IEEE Software*, vol. 30, no. 4, ss. 31–37, Heinäkuu 2013.

[32] R. L. Novais, A. Torres, T. S. Mendes, M. Mendonça, ja N. Zazworka, "Software evolution visualization: A systematic mapping study," *Information and Software Technology*, vol. 55, no. 11, ss. 1860 – 1883, 2013. Saatavilla: http://www.sciencedirect.com/science/article/pii/S0950584913001298

[33] W. Poncin, A. Serebrenik, ja M. v. d. Brand, "Process mining software repositories," *2011 15th European Conference on Software Maintenance and Reengineering*, Maaliskuu 2011, ss. 5–14.

[34] R. Purushothaman ja D. E. Perry, "Toward understanding the rhetoric of small source code changes," *IEEE Transactions on Software Engineering*, vol. 31, no. 6, ss. 511–526, Kesäkuu 2005.

[35] U. Raja, J. Hale, ja D. Hale, "Temporal patterns of software evolution defects: A comparative analysis of open source and closed source projects," *Journal of Software Engineering and Applications*, vol. 4, no. 8, ss. 497–511, 2011.

[36] R. Robbes, R. Vidal, ja M. C. Bastarrica, "Are software analytics efforts worthwhile for small companies? the case of amisoft," *IEEE Software*, vol. 30, no. 5, ss. 46–53, Syyskuu 2013.

[37] M. Shepperd ja S. MacDonell, "Evaluating prediction systems in software project estimation," *Information and Software Technology*, vol. 54, no. 8, ss. 820 – 827, 2012, special Issue: Voice of the Editorial BoardSpecial Issue: Voice of the Editorial Board. Saatavilla: http://www.sciencedirect.com/science/article/pii/S095058491200002X

[38] Y. Shin, R. Bell, T. Ostrand, ja E. Weyuker, "Does calling structure information improve the accuracy of fault prediction?" *2009 6th IEEE International Working Conference on Mining Software Repositories*, Toukokuu 2009, ss. 61–70.

[39] V. B. Singh, K. K. Chaturvedi, S. K. Khatri, ja V. Kumar, "Bug prediction modeling using complexity of code changes," *International Journal of System Assurance Engineering and Management*, vol. 6, no. 1, ss. 44–60, 2015. Saatavilla: http://dx.doi.org/10.1007/s13198-014-0242-5

[40] B. F. van Dongen ja W. Aalst, "A meta model for process mining data," *Proceedings of the CAiSE'05 Workshops (EMOI-INTEROP Workshop)*, J. Castro ja E. Teniente, Eds., vol. 160, FEUP Edições. FEUP Edições, 2005, ss. 309–320.

[41] L. Voinea ja A. Telea, "Visual querying and analysis of large software repositories," *Empirical Software Engineering*, vol. 14, no. 3, ss. 316–340, 2009. Saatavilla: http://dx.doi.org/10.1007/s10664-008-9068-6

[42] M. S. Zanetti, I. Scholtes, C. J. Tessone, ja F. Schweitzer, "Categorizing bugs with social networks: A case study on four open source software communities," *2013 35th International Conference on Software Engineering (ICSE)*, Toukokuu 2013, ss. 1032–1041.

[43] D. Zhang, S. Han, Y. Dang, J. G. Lou, H. Zhang, ja T. Xie, "Software analytics in practice," *IEEE Software*, vol. 30, no. 5, ss. 30–37, Syyskuu 2013.

[44] F. Zou ja J. Davis, "Analyzing and modeling open source software bug report data," *19th Australian Conference on Software Engineering (aswec 2008)*, Maaliskuu 2008, ss. 461–469.