Lappeenrannan teknillinen yliopisto
School of Engineering Science
Tietotekniikan koulutusohjelma

Kandidaatintyö

Niko Aurelma

Ohjelmoinnin perusteiden kurssin tuotosten staattinen analyysi

Työn tarkastaja(t): Uolevi Nikula

Työn ohjaaja(t): Uolevi Nikula

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
School of Engineering Science
Tietotekniikan koulutusohjelma

Niko Aurelma

Ohjelmoinnin perusteiden kurssin tuotosten staattinen analyysi

Kandidaatintyö

2019

21 sivua, 10 kuvaa.

Työn tarkastajat: Uolevi Nikula

Hakusanat: Ohjelmoinnin perusteet, analyysi
Keywords: CS1, Analysis.

Tässä työssä tutkittiin oppilaiden ohjelmointi palautuksia. Palautukset kerättiin Ohjelmoinnin perusteet -kurssilta (CS1), joka opetettiin Python kielen avulla. Tutkittavina asioina oli rivimäärät, kompleksisuus, halstead metriikat ja muuttujien nimien käyttö sekä niiden erot malli ratkaisuihin. Mikään tutkittu arvo ei noudata lineaarista kasvua kurssin edetessä. Datasta löytyi odottamattomia piikkejä, joiden syyt liittyivät kurssin sisäisiin ja ulkoisiin tekijöihin. Analyysiin kuuluu noin 400 opiskelijan palautukset 14 viikolta. Analyysin tavoitteena on tukea opetuksen kehitystä ja tämä tavoite saavutettiin.

ABSTRACT

Lappeenranta University of Technology
School of Engineering Science
Degree Program in Computer Science

Niko Aurelma

Static analysis of CS 1 course work

Bachelor's Thesis

21 pages, 10 figures.

Examiners: Uolevi Nikula

Keywords: CS1, Analysis.

This work examines the programming assignments of students. The assignments were gathered from CS 1 course, that used the Python programming language. The metrics used were: line numbers, complexity, Halstead metrics and usage of variable names. In addition the metrics were compared against the answers. None of the values followed a simple linear curve. The data had unexpected spikes that had explaining factors both inside and outside the course The analysis contained assignments from 400 students over 14 weeks. The goal of the analysis was to support course improvement and this goal was accomplished.

ALKUSANAT

Työ on tehty Lappeenrannan teknillisessä yliopistossa. Kiitän työkavereita jaetuista kokemuksista.

SISÄLLYSLUETTELO

1 JOHDANTO

1.1 Tausta

Tässä työssä tutkitaan CS1 kurssilla tuotettuihin ohjelmiin liittyviä metriikoita ja niiden muutoksia. Mitattaviksi metriikoiksi valittiin Haelstad metriikat ja yksinkertaiset lähdekooditiedoston rivimääriin liittyviä metriikoita. Näiden lisäksi ohjelmista analysoitiin myös muuttujien ja funktioiden nimeämiskäytäntöjä. Analysoitava data haettiin Viope palvelusta ja palautusten oikeellisuudesta ei ollut merkintää datassa.

1.2 Tavoitteet ja rajaukset

Tässä työssä on tarkoituksena tutkia miten oppilaiden ohjelmointi vastaa malliratkaisujen ohjelmointia. Alunperin tarkoituksena oli verrata viikkotehtäviä ja harjoitustöitä, mutta datan ongelmallisuuden takia analyysi rajoittui vain viikkotehtäviin. Datasta puuttui henkilön yksilöivä tieto ja jokaisessa palautuksessa oli tehtävän yksilöivä tunniste, mutta tunniste piti manuaalisesti liittää tehtävän numeroon. Tässä työssä analysoidaan vain ohjelmoinnin perusteiden viikkotehtävien palautuksia, jotka ovat kirjoitetty Python kielellä.

Data saatiin Viope palvelusta tietovaraston vientinä. Viope on internetissä oleva ohjelmointi tehtäviin keskittyvä opetus alusta, joka sallii ohjelmien suorittamisen hiekkalaatikossa ja sen tulosteen vertaamisen malliratkaisulla tuotettuun. Tutkittavalla kurssilla Viopesta käytettiin vain ohjelmien suoritus- ja validointitoimintoja.

1.3 Tutkimuskysymykset

Tutkimuksessa perusoletus kaikissa tutkittavissa arvoissa on, että ne pysyvät joko vakioina kurssin ajan tai ne kasvavat noin lineaarisesti kurssin edetessä. Nämä oletukset valittiin, koska ne ovat yksinkertaisimmat mallit ja oman kokemuksen perusteella kurssin vaatimustaso kasvaa suurin piirten lineaarisesti. Tiettyjen suureiden kohdalla, esimerkiksi kommenttirivien märällä, voidaan olettaa, että ne pysyvät vakiona kurssin aikana.

Tutkimuksessa keskityttiin palautuksien ja malliratkaisujen eroihin ja miten ne muuttuvat kurssin edetessä.

1.4 Työn rakenne

Luvussa 2 käsitellään muuta tutkimusta asiaan liittyen sekä analyysiin käytettyjä työkaluja. Luvussa 3 esitellään data, analyysi sekä tulokset. Luvussa 4 käsitellään tulevaisuutta ja luvussa 5 esitetään yhteenveto työstä.

2 MUU TUTKIMUS

2.1 Kirjallisuuskatsaus

Ahadi et al.(2017) tutkivat miten hyvin koodi tilannekuvien oikeellisuus kurssin edetessä pystyisi ennustamaan kurssin arvosanaa. He löysivät, että keskimääräinen oikeellisuus noudatti alaspäin aukeavaa parabelia, eli kurssin keskellä oli keskimäärin parhaimmat tulokset.

Assister et al. tutki voisiko tiivistelmien tekeminen auttaa ohjelmoinnin opettamisessa. Heidän tulokset viittaavat, että sanallinen selitys antaa paremman kuvan osaamisesta riippumatta saadusta pistemäärästä.

Jos opiskelijat saavat palauttaa työnsä usein ja saavat joka palautuksesta pistemäärän, niin se voi johtaa suureen määrään opiskelijoita ketkä hakevat puhtaasti parempia pisteitä. Tätän ilmiön nimi on bricolage. (Falkner et al., Mar 5, 2014)
Yleisesti ottaen suurempi koettu vaihteluväli arvosanoissa lisää opiskelijoiden saamaa arvosanaa. Vaikka opiskelijoiden lukumäärä kasvaa, niin huonosti suoriutuvien opiskelijoiden osuus on pysynyt tasaisena (Sahami, Piech, Feb 17, 2016).

Suurilla kursseilla on Bey et al. (2018) mukaan käytössä kahden tyyppisiä arvosteluohjelmistoja. Ensinnäkin on ohjelmia, jotka suorittavat opiskelijan ohjelman eri data seteillä ja vertaavat sen tulosta malliratkaisuun. Toinen vaihtoehto on staattinen analyysi, jossa arvosteltavaa ohjelmaa ei suoriteta, vaan siitä kerätään metriikoita ja niitä verrataan malliratkaisuun. Viope palvelu kuuluu ensimmäiseen kategoriaan.

Petit et al. analysoivat opiskelijoiden ohjelmointi tuotoksia käyttäen metriikoina SLOC, McCabe kompleksisuus, Haelstad metriikoita sekä tila-avaruutta muuttujien lukumäärän avulla.

2.2 Työkalujen kuvaus

2.2.1 Radon

Radon on Pythonin staattiseen analyysin kehitetty työkalu. Sillä pystyy laskea koodista rivimäärä metriikoita, kompleksisuus metriikoita, halstead metriikoita ja maintainability-indexin (Lacchila 2017). Työkalulla tuotettiin suurin osa analysoitavasta datasta.

2.2.2 Shell

Bash on käytetty shell kieli, sitä käytettiin sitomaan muut käytetyt työkalut yhteen. Bash toimi välikohtana datan siirtämisessä ohjelmalta toiselle ja muiden ohjelmien koordinoinnissa. Python on data-analyysiin käytetty ohjelmointikieli. Tutkittavat ohjelmat olivat kirjoitettu tällä kielellä ja osa analyysistä myös. Golang on ohjelmointikieli jota käytettiin datan siivoamisen. Jq on json formaatin käsittelyyn komentorivillä kehitetty työkalu. Sed on stream editor, jota käytettiin datan siivoamiseen.

2.2.3 Flex, Bison

Analyysissä oli tarkoitus käyttää Flex ja Bison työkaluja, mutta analyysin jatkuessa niiden rooli väheni, koska löytyi tähän tarkoitikseen valmiiksi kirjoitettu työkalu: Radon.

Flex on lexikaalinen analysaattri rakentaja. Sen avulla pystyy rakentamaan leksikaalisen analysaattorin. Tässä analyysissä käytetty muuttujien nimien hakemiseen. Bison on kääntäjien rakentamiseen käytetty työkalu.

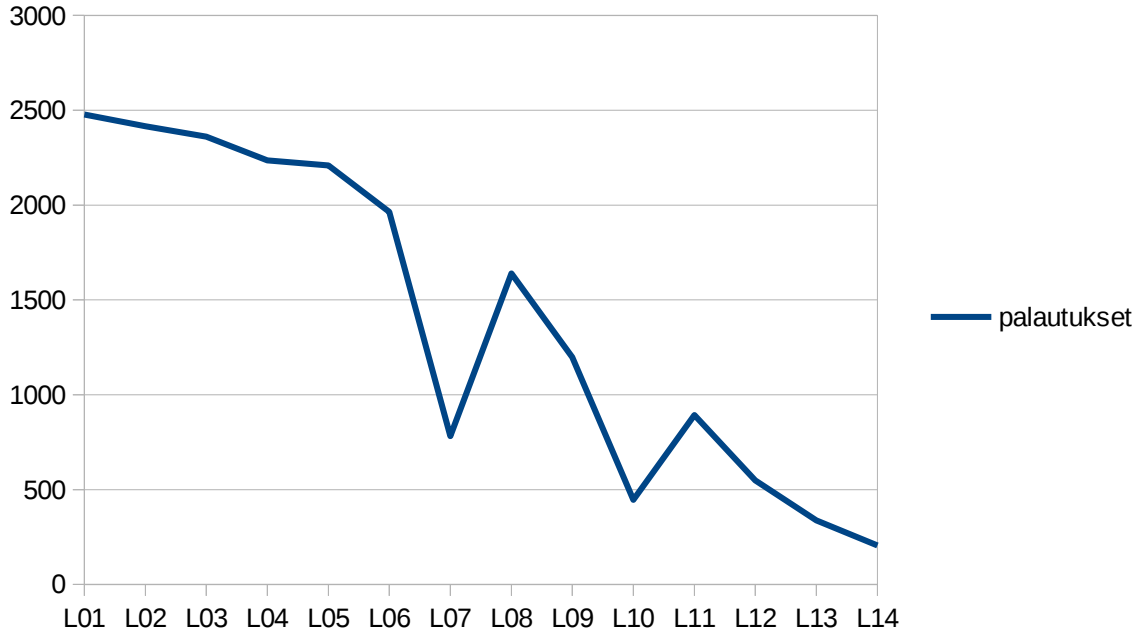
3 TULOKSET

3.1 Datan kuvaus

Datasetti saatiin Viopen kautta. Alkuperäinen oletus oli että kaikki saadut tuotokset olisivat saaneet pisteen tehtävästä, mutta myöhemmän katselmoinnin jälkeen todettin, että jotkin eivät olleet saaneet pistettä. Tämä johtui eräästä työstä, jossa oli vain yksi rivi ja se sisälsi kommentin kurssin pitäjälle, eikä ollut edes validia Python koodia. Tästä johtuen oletetaan, että saatu data sisälstää opiskelijan viimeisimmän tallennetun version eikä läpimenneitä palautuksia.

Datassa itsessään kerrotaan tehtävän numero, osaratkaisu vai koko ratkaisu, tiedoston nimi ja itse sisältö. Datassa ei pysty kertomaan kenen työ se on eikä saatua piste määrää. Joissakin töissä on otsikkotiedot, joista saisi selville tekijän tiedot, mutta tässä analyysissä nämä tiedot jätettiin pois.

Kuva 1: Palautusten lukumäärä



Kuvassa 1 on palautusten lukumäärä viikoittain. Kuvan laskeva sunta selittyy osittain tehtävien vähentyvällä määrällä; aluksi 5 per viikko ja sitten 4 ja lopuksi 3 viikolla. Kuvaajan suuret pudotukset liittyyvät viikkoihin, joissa oli normaalia vähemmän tehtäviä.

Viikolla 7 oli vain 2 palautusta samoin kuin viikolla 10. Viikolla 7 oli myös lomaa, joka selittää oppilaiden vähempiä palautuksia. Viikolla 10 oli harjoitus työn palautus ja opiskelijat ovat luultavasti keskittyneet siihen enemmän. Kuvaajan yleinen lasku selittyy myös opiskelijoiden vähenevällä mielenkiinnolla.

3.2 Analyysin kuvaus

Raakadatasta poistettiin aluksi kaikkien kommenttien sisällöt, pitäen kommenttimerkit paikallaan. Tämän tekeminen ei vaikuta ohjelman suorittamiseen ja komentit saattoivat sisältää yksilöiviä tietoja. Sitten raaka data purettiin yhdestä tiedostosta useaan tiedostoon siten että jokainen palautus on omassa tiedostossaan. Tämä tehtiin työkalujen käytön helpottamiseksi. Seuraavaksi jokainen palautus ajettiin työkalun läpi tuottaen haetut metriikat. Nämä yhdistettiin yhteen csv:hen per tehtävä, jonka jälkeen jokaisesta csv:stä laskettiin sarakekohtaiset keskiarvot ja tulos lisättiin uuteen csv:hen. Tätä yhdistettyä csv:tä käytettiin tuottaamaan lopulliset graafit.
Malliratkaisuista metriikat otettiin suoraan, sillä joka tehtävään on vain yksi malliratkaisu.

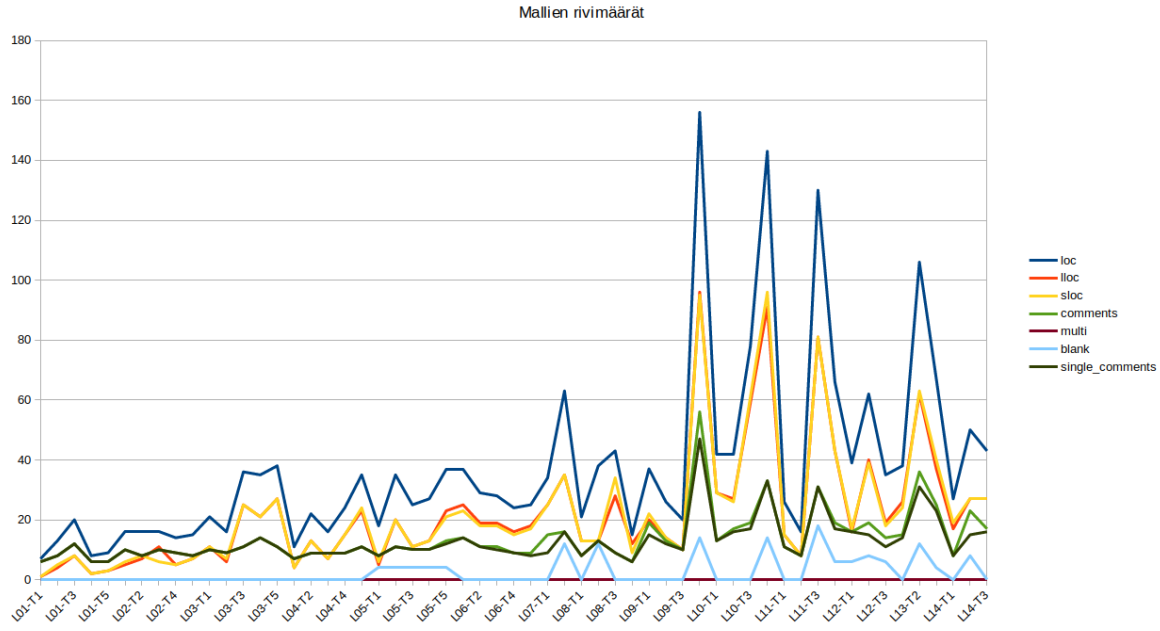
3.3 Analyysin tulokset

3.3.1 Rivimäärä analyysi

Analyysissä oli mukana 6 metriikkaa: LOC; LLOC, SLOC, comments, multi ja blank. LOC tarkoittaa lines of code eli koodirivien määrä. LLOC on logical lines of code eli koodirivien määrä, jossa on tasan yksi statement. SLOC on source lines of code eli lähdekoodirivien määrä. Comments eli kommentti rivien lukumäärä. Multi on monirivikommenttien määrä. Blank on tyhjien rivien määrä. Nämä metriikat valittiin, koska valittu työkalu tuotti nämä metriikat suoraan.

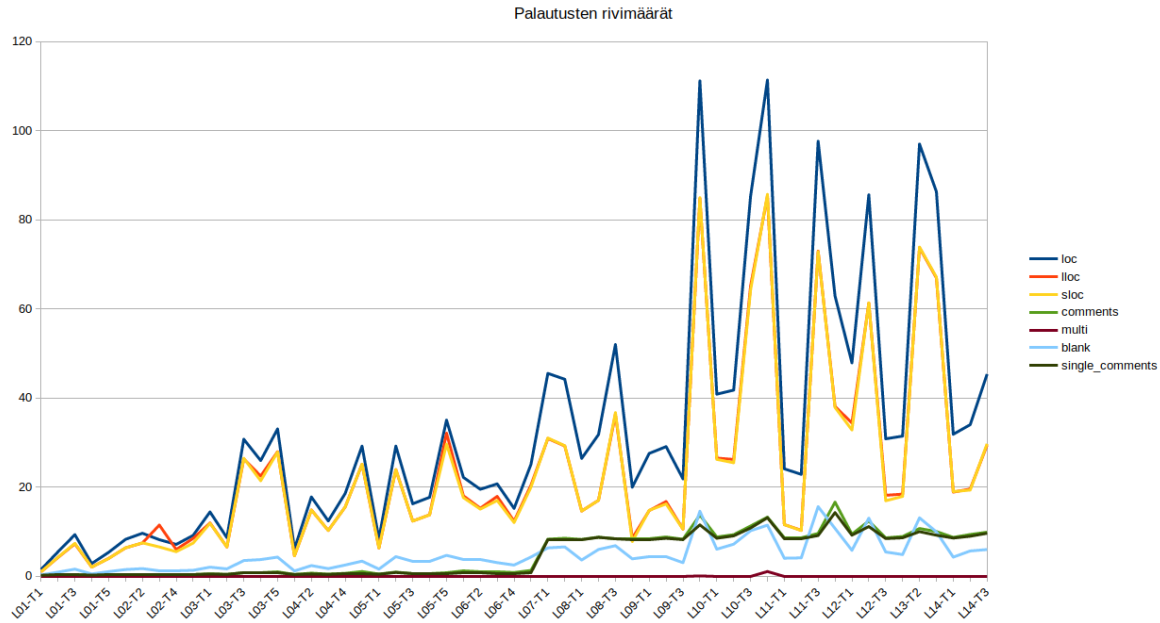
Kuvassa 2 esitetään malliratkaisujen rivimäärät, kuvan selite vastaa aikaisempaa määritelmää. Malli ratkaisujen rivimäärä pysyy suhteellisen tasaisena viikkoon 10 asti, senjälkeen tulee erittäin suuri hyppy ja sitten ne laskevat loppua kohti.

Kuva 2: Mallien rivimäärät



Kuvassa 3 on kuvaa 2 vastaavat arvot, mutta lähtökohtana on oppilaiden palautusten keskiarvot.

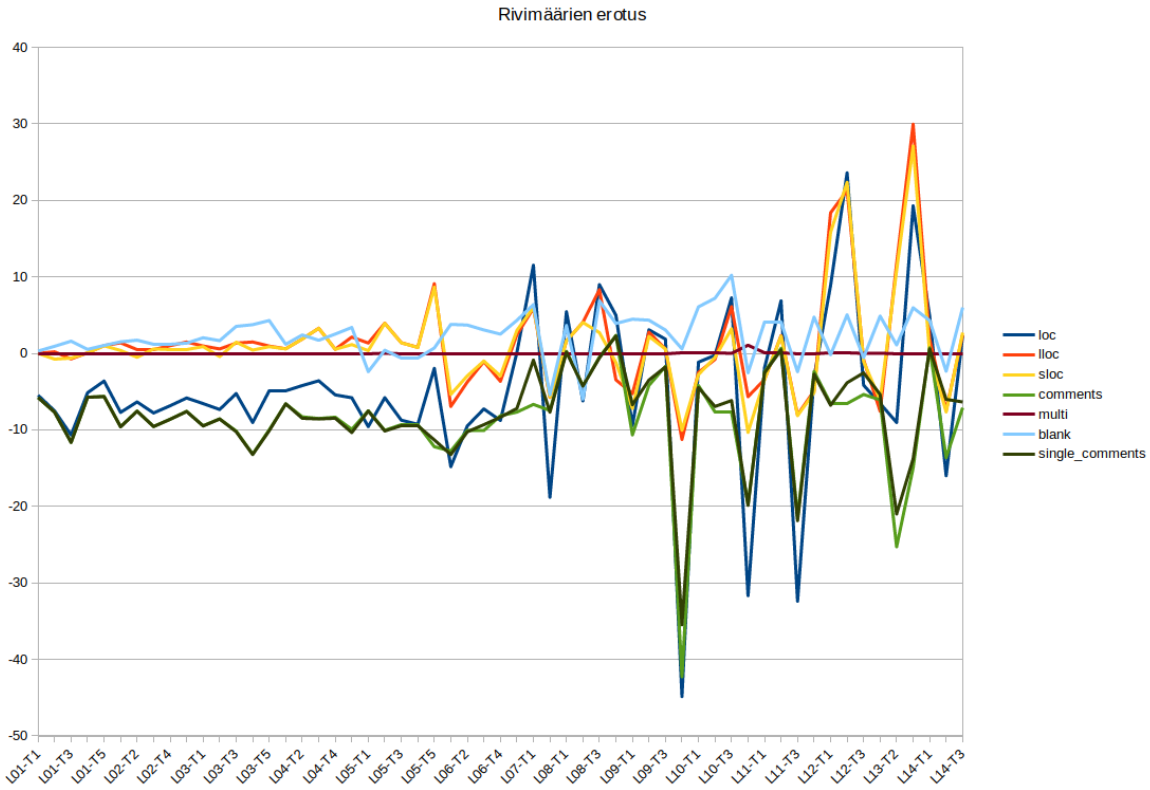
Kuva 3: Palautusten rivimääärät



Kuvassa 3 viikko 7 kohdalla oleva kommenttirivien hyppy selittyy kurssin rakenteen perusteella. Silloin annetaan ohjeistus lisätä otsikkotiedot jokaiseen tehtävään. Otsikko tiedot sisältävät tekijän nimen ja opiskelija numeron sekä yhteistyön tavan.

Kuvassa 4 on oppilaiden arvojen ja malliratkaisujen arvojen erotus. Eli jos malliratkaisussa oli jotakin suuretta enemmän, niin se saa negatiivisen arvon kuvaajassa.

Kuva 4: Rivimäärien erotus



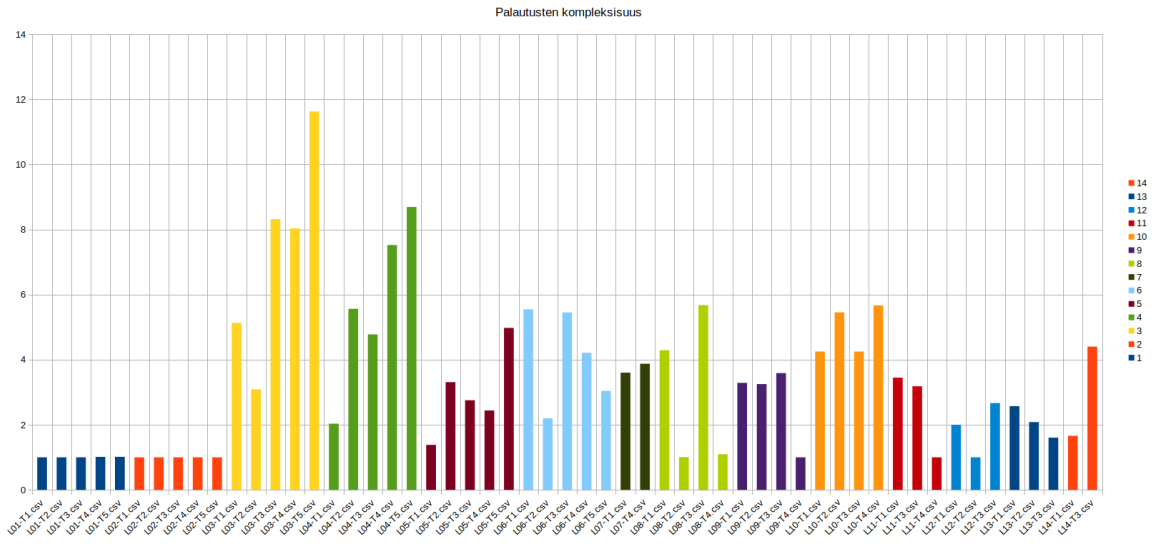
Kuvaajassa kommenttien lukumäärä on kokoajan malliratkaisujen puolella, mikä voi johtua tekijöiden asenteesta ja tavoitteesta. Malliratkaisu on tehty opetuskäyttöön ja sen tarkoitus on selittää mitä se tekee. Opiskelijatöissä tavoite on käytännössä Viopesta läpi pääsy, eli pisteen saaminen. Malliratkaisua on tarkoitus lukea uudelleen, palautuksia ei.
Opiskelijoilla on enemmän tyhjiä rivejä keskimäärin. Tämä voi johtua erilaisesta koodaustyylistä ja osittain myös tyhjistä ratkaisusista.
Opiskelijoilla on pääasiassa myös enemmän koodirivejä. Tämä saattaa johtua vähemmästä optimoinnista ja tehtävän koodaamisen lopettamiseta heti kun se menee arvosteluohjelmasta läpi.

3.3.2 Kompleksisuus analyysi

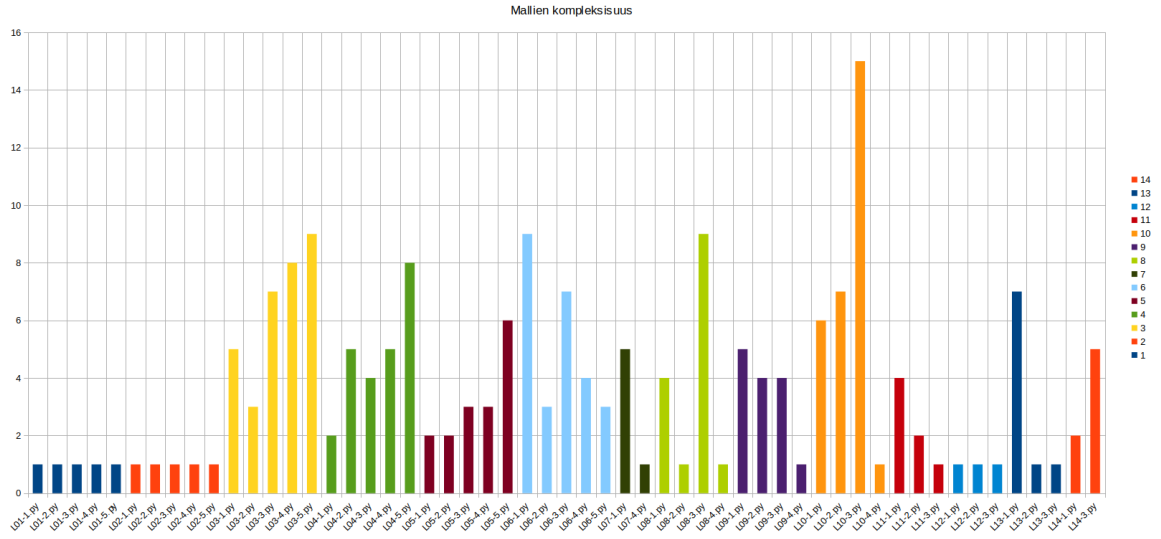
Syklomaattinen kompleksisuus on tässä laskettu summaamalla koodin päätös kohdat. Päätös kohtia Pyythonissa on if, elif, for, while, except, with, assert ja boolean operaattorit. Näihin kuuluu myös automaattinen listan tai sanakirjan täyttö, sillä ne vastaavat for looppia. Tämä vastaa lineeaarisesti riippumattomien koodipolkujen lukumäärää.

Kuvassa 5 on palautusten kompleksisuus. Tehtävässä L03T5 piti antaa käyttäjälle painoindeksi. Tämä on malliratkaisussa tehty usealla peräkkäisellä if lauseella, joten ei ole yllättävää että palautuksissa on vastaava piikki.

Kuva 5: Palautusten kompleksisuus

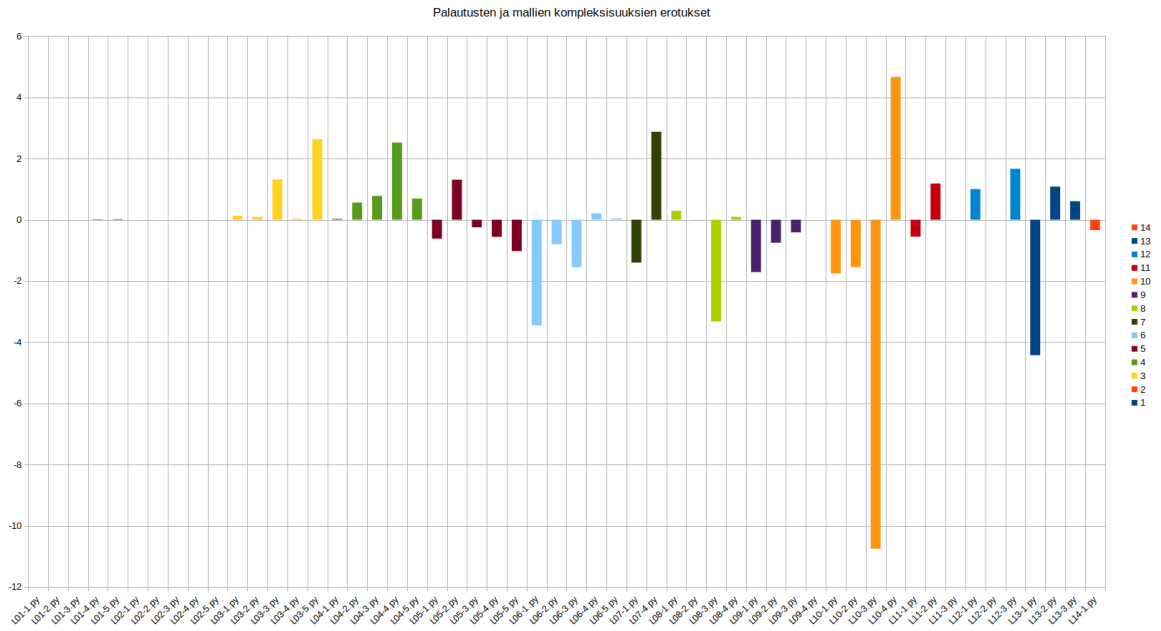


Kuva 6: Mallien kompleksisuus



Palautusten ja mallin erotus

Kuva 7: Kompleksisuuksien erotus



Viikon 3 tehtävässä 5 piti laskea painoindeksi ja tulostaa sen perusteella käyttäjälle sanallinen kuvaus tästä. Malliratkaisussa se on toteutettu peräkkäisillä if lausella, joissa aina testataan, onko arvo alle seuraavan ylärajan. Opiskelijoilla on enemmän testejä, joten he luultavasti ovat testanneet jokakerta ylä- ja alarajan kaikille tulostuksille.

Viikon 10 tehtävät ovat valikkopohjaisia tiedoston luku tehtäviä. Tehtävässä 5 on vain 15 palautusta, kun yleensä palutuksia useampi sata. Tehtävän pieni palautusmäärä voi johtua tehtävän hankaluudesta, mutta kurssin pitäjän perusteella tämän tehtävän palutusviikolla oli myös harjoitustyön palautuspäivä ja opiskelija ovat keskittyneet siihen. Tämä selittää kompleksisuuden hyppyä, koska vain korkeimmalle tähtäävät ovat suorittaneet tehtävän ja he tuottavat monimutkaisempaa koodia kuin muut.

3.3.3 Halstead metriikat

Halsted metriikat lasketaan neljän luvun avulla: n1, n2, N1 ja N2. Näistä lasketaan muut metriikat, joita ei tähän työhön laskettu.

n1 = uniikkien oparaattorien lukumäärä

n2 = uniikkien operandien lukumäärä

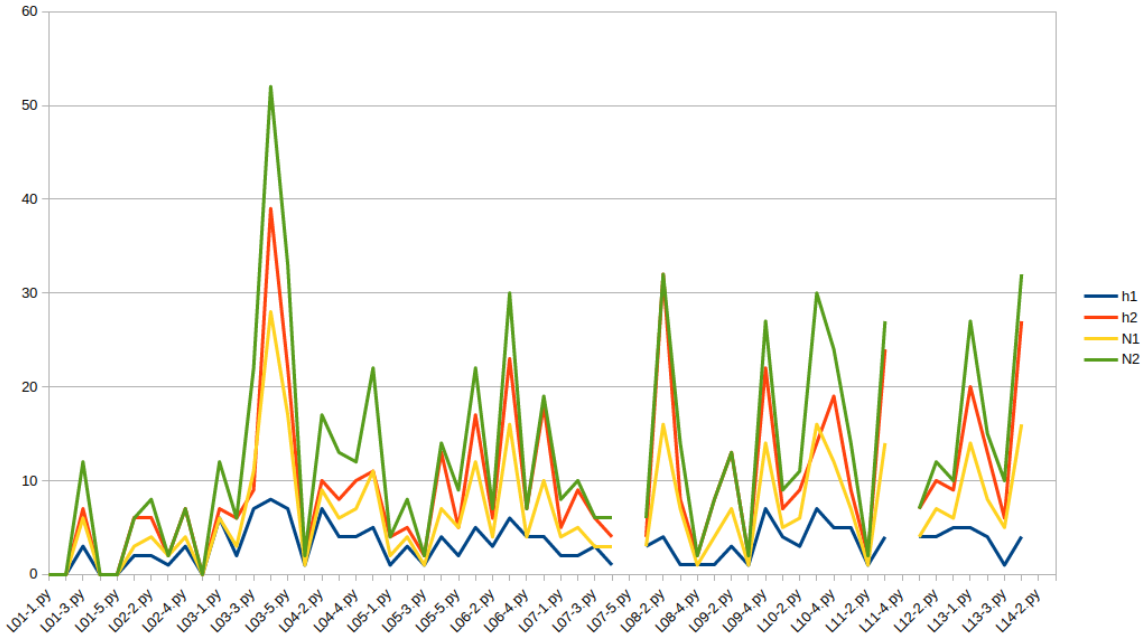
N1 = kaikkien operaattorien yhteislukumäärä

N2 = kaikkien operandien yhteislukumäärä

(Hariprasad et al., 2017)

Kuvassa 8 on esitetty nämä neljä perusmetriikkaa.

Kuva 8: Halsted metriikat

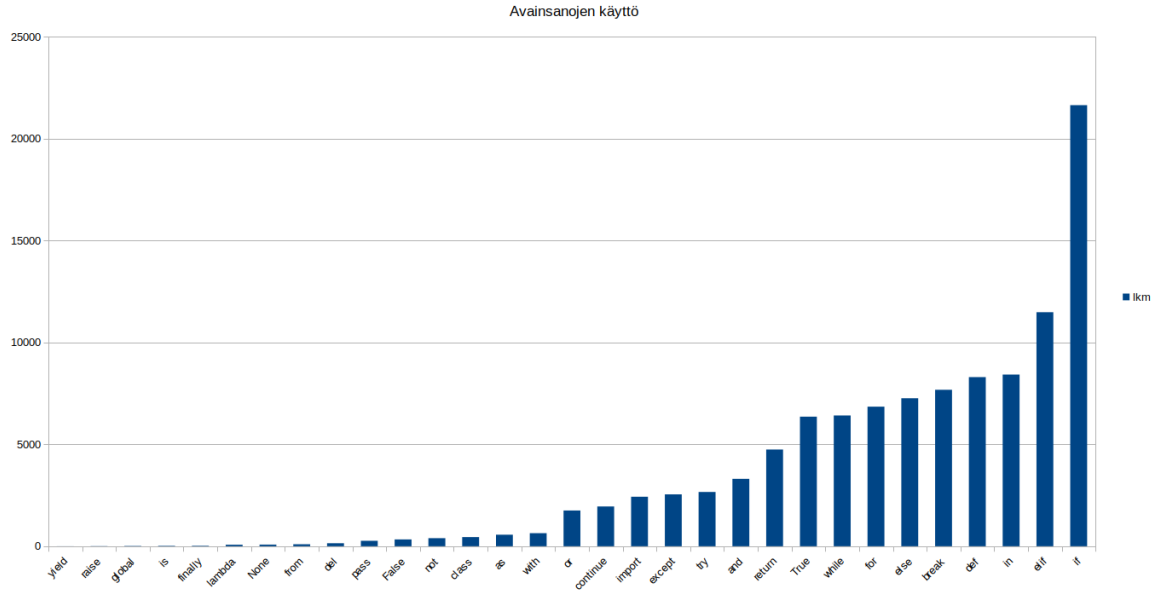


Kuvaajan katkot johtuvat työkalun antamista virheistä. Nämä virheet tulevat joko virheellisestä koodista tai puutteellisesta työkalusta. Eli kielessä saattaa olla joitain ominaisuuksia, jota työkalu ei pysty käsittelemään oikein tai aikaisemmat käsittely vaiheet muuttivat koodin rakenneta liikaa.

3.3.4 Nimianalyysi

Kuvassa 9 on avainsanojen käyttökerrat koko kurssin ajalta. Suurimmaksi arvoksi tuli if, mikä ei ollut ollenkaan yllättävää kurssin rakenteen takia.

Kuva 9: Avainsanat

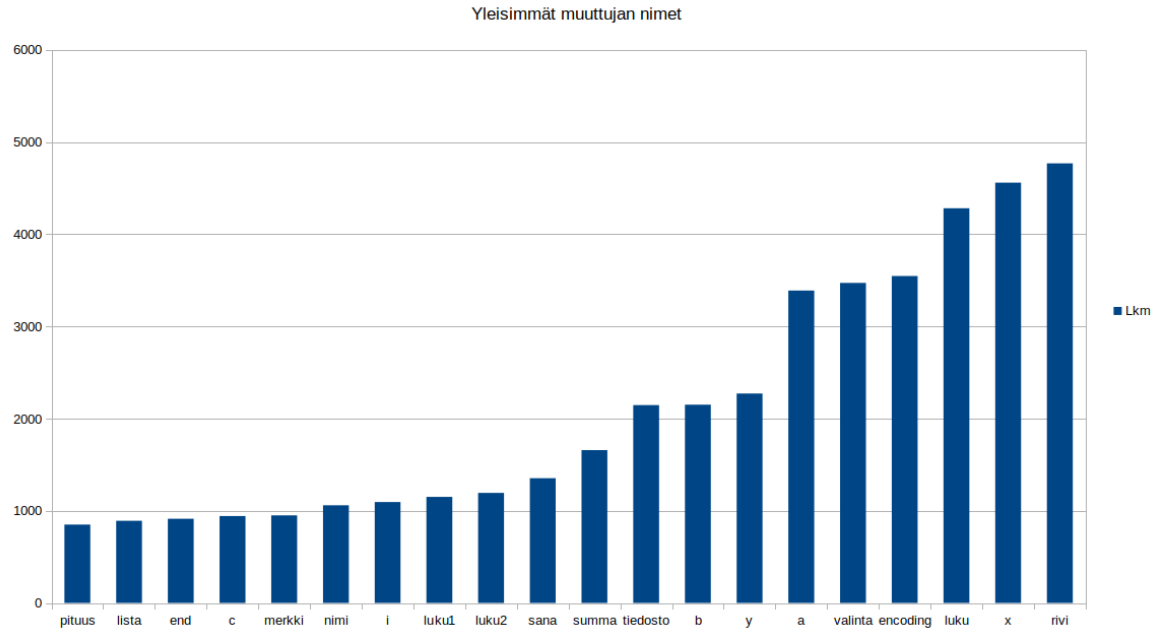


Avainsanaa ”yield” esiintyi koko kurssin aikana vain kerran. Se ei kuulu oppimateriaaliin eikä edes ohjelmoinnin perusteisiin, joten sen käyttäjä joko osasi Pythonia etukäteen jo tai löysi sen jostain muusta oppaasta. Palautuksen perusteella hän osasi sitä etukäteen.

Avainsanan ”global” käyttö oli yllättävää, koska kurssilla painotetaan että globaalien muuttujien käyttö on kielletty tai ainakin huonoa koodaus tyyliä ja sen avainsanan käyttöön ei ole muuta syytä kuin globaalin muuttujan käyttö funktion sisällä.

Palautuksissa oli noin 6000 uniikkia muuttujan nimeä, joista 50% käytettiin vain kerran. Kuvassa 10 on 20 yleisimmän muuttujan nimet. Nimet haettiin etsimällä kaikki kohdat, joissa muuttujalle asetetaan arvo. For loopin arvoja ja usean muutttujan arvon asettamista ei otettu huomioon datan tuotossa.

Kuva 10: Muuttujien yleisimmät nimet



Yleisin muuttujan nimi on rivi, koska kurssilla luetaan paljon tiedostoja ja niitä luetaan rivikerrallaan. Vastaavasti encoding saa paljon osumia, koska joka tiedostoa avatessa se asetetaan. Datassa yllätti luku2 suurempi käyttömäärä kuin luku1, mutta ero saattaa tulla nimien laskutavasta jälleen.

4 POHDINTA JA TULEVAISUUS

Seuaavassa Viope datassa pitää pyytää palatuksen pistemäärä, tehtävän nimi ja tekijän yksilöivä tunnus. Yksilöivää tunnusta tarvitaan vain jos ollaan tekemessä ennustavaa työtä, ei muuten. Tehtävän nimi on pääasiassa vain analyysiä helpottava tieto, sen pystyy hakea manuaalisestikkin.

5 YHTEENVETO

Tässä työssä tutkittiin oppilaiden ohjelmointi tehtävien tuotoksia ja lopputuloksena selvisi, että mikään metriikka ei noudata yksinkertaisinta regressiomallia. Vaan ne kasvavat kurssin alkuun, mutta kurssin loppupuolella ne pysyvät käytännössä muuttumattomina.

LÄHTEET

Ahadi, A., Lister, R., Lal, S., Leinonen, J. & Hellas, A. Jan 31, 2017, "Performance and Consistency in Learning to Program", ACM, , pp. 11.

Anis Bey, Patrick Jermann & Pierre Dillenbourg 2018, "A Comparison between Two Automatic Assessment Approaches for Programming", Journal of Educational Technology & Society, vol. 21, no. 2, pp. 259-272.

Assiter, K. Oct 2010, "Work in progress - Programming assignment summary for analysis, qualitative assessment and continuous improvement in CS1 - CS3", IEEE, , pp. T1.

Falkner, N., Vivian, R., Piper, D. & Falkner, K. Mar 5, 2014, "Increasing the effectiveness of automated assessment by increasing marking granularity and feedback units", ACM, , pp. 9.

T. Hariprasad, G. Vidhyagaran, K. Seenu & C. Thirumalai 2017, Software complexity analysis using halstead metrics.

Lacchia M. 2017 Various code metrics for Python code [verkkodokumentti]. [Viitattu 20.7.2019]. Saatavilla https://github.com/rubik/radon

Pettit, R., Homer, J., Gee, R., Mengel, S. & Starbuck, A. Feb 24, 2015, "An Empirical Study of Iterative Improvement in Programming Assignments", ACM, , pp. 410.

Sahami, M. & Piech, C. Feb 17, 2016, "As CS Enrollments Grow, Are We Attracting Weaker Students?", ACM, , pp. 54.