Ohjelmoinnin harjoitustöiden arviointidatan analysointi ja visualisointi

Lappeenrannan–Lahden teknillinen yliopisto LUT

Tietotekniikan kandidaatintyö

Tietotekniikan kandidaatin ohjelma

2024

Kaapo Pylväläinen

Tarkastaja: DI Ilpo Kainulainen

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUTin insinööritieteiden tiedekunta

Tietotekniikka

Kaapo Pylväläinen

Ohjelmoinnin harjoitustöiden arviointidatan analysointi ja visualisointi

Tietotekniikan kandidaatintyö

2024

25 sivua, 7 kuvaa, 0 taulukkoa ja 0 liitettä

Tarkastaja(t): DI Ilpo Kainulainen

Avainsanat: Palautuksen arviointi, Virheiden analyysi, data-analyysi

Tässä opinnäytetyössä tutkittiin ja analysoitiin opiskelijoiden koodaus harjoitustöiden palautusten arviointeja kurssilta "Ohjelmoinnin perusteet" aikaisemmilta kurssi suorituksilta. Tavoitteena oli kehittää työkalu, joka helpottaisi datananalyysiä ja jolla voitaisiin visualisoida arviointi dataa. Tarkoituksena on tehostaa palauteprosessia ja helpottaa vaikeiden osaalueiden löytämistä kurssilta. Datana käytettiin yksittäisten opiskelijoiden palautustietoja, jotka sisälsivät mm. opiskelijan tekemät virheet ja palautteet virheellisistä kohdista. Analyysissä keskityttiin tunnistamaan yleisimpiä virheluokkia, ongelmallisia osa-alueita ja yleisiä virheitä palautteissa. Tutkimuksen tuloksena kehitettiin visualisointityökalu, joka auttaa opettajia ja oppimisavustajia ymmärtämään paremmin opiskelijoiden osaamistasoa ja oppimishaasteita. Työkalun avulla opettajat ja oppimisavustajat voivat helposti visualisoida opiskelijoiden arvioinnit ja löytää yleisimmät virheet. Opinnäytetyön tuloksia voidaan hyödyntää vastaavien kurssien opetuksen ja arvioinnin kehittämisessä. Työkalu tarjoaa opettajille ja oppimisavustajille erilaisen lähestymistavan analysoida opiskelijoiden tekemiä virheitä.

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

School of Engineering Science

Software Engineering

Kaapo Pylväläinen

Evaluation and visualisation of student coding exercise feedback

Bachelor’s thesis

2024

25 pages, 7 figures, 0 tables and 0 appendices

Examiners: DI Ilpo Kainulainen

Keywords: Return evaluation, Error analysis, Data analysis

This thesis studied and analysed the coding assignment feedback of students from the course "Fundamentals of Programming" from previous course iterations. The aim was to develop a tool to facilitate data analysis and to visualize the evaluation data. The purpose is to make the feedback process more efficient and to make it easier to learn what parts of the course the students struggled with. The data used was individual student return data, which included errors made by the student and feedback on incorrect items. The analysis focused on identifying the most common categories of errors, problematic areas and common errors in feedback. As a result of the study, a visualisation tool was developed to help teachers and learning assistants better understand students' proficiency levels and learning challenges. The tool allows teachers and learning assistants to easily visualise students' assessments and identify common errors. The results of the thesis can be used to develop teaching and assessment in similar courses. The tool provides teachers and learning assistants with a different approach to analysing errors made by students.

KIITOKSET/ ACKNOWLEDGEMENTS

Kiitokset kaikille, jotka auttoivat työssä oli se motivaation/parannuskohtien kanssa.

Sisällysluettelo

Tiivistelmä

Abstract

(Kiitokset)

1 Johdanto ... 7
1.1 Tausta ... 7
1.2 Tavoitteet ja rajaukset ... 7
1.3 Tutkimuskysymykset ... 7
1.4 Työn rakenne ... 8
2 Kirjallisuuskatsaus ... 9
2.1 Muu tutkimus ... 9
3 Työkalut ja analysointi ... 11
3.1.1 Python ja datan analysointi ... 11
3.1.2 Pandas ja Matplotlib ... 11
3.1.3 Jupyter ... 12
3.2 Analysointi ... 12
4 Design Artefact ... 13
4.1 Läpipääsy ... 13
4.2 Keskimääräiset virhepisteet ... 14
4.3 Virheluokkien jakauma ... 15
4.3.1 Ryhmittäin ... 16
4.3.2 Keskiarvot ... 17
5 Pohdinta ... 19
5.1 Läpipääsy ... 19
5.2 Keskimääräiset virhepisteet ... 19
5.3 Virheluokkien jakauma ... 19
5.3.1 Ryhmittäin ... 20
5.4 Johtopäätökset ... 21
5.5 Vastaukset tutkimuskysymyksiin ... 22
6 Yhteenveto ... 24

Lähteet 25

1 Johdanto

1.1 Tausta

Tämä työ käsittelee opiskelijoiden saamaa palautetta heidän tekemistään harjoitustöistä kurssilla *Ohjelmoinnin perusteet*. Kurssi kuuluu olennaisena osana monien tekniikan alan opsikelijoiden opintoihin LUT-yliopistossa ja sen suorittaminen edellyttää sekä aikaa että vaivaa. Harjoitustyön hyväksytty arvosana on yksi kurssin läpipääsy vaatimuksista ja kurssi on myös monen opiskelijan ensimmäinen kohtaaminen koodauksen kanssa, joten kurssin aikana tulee monille opiskelijoille täysin uutta materiaalia.

1.2 Tavoitteet ja rajaukset

Tässä työssä on tarkoitus analysoida ja visualisoida opiskelijoiden tekemiä virheitä heidän tekemästään harjoitustyöstä. Opiskelijatietojen arkaluonteisuuden vuoksi tässä työssä ei voida käyttää kaikkia kursilla tehtyjä harjoitustöitä. Tämän vuoksi tutkimus rajataan koskemaan vain anonymisoitua otosta kurssin harjoitustöistä. Työn päätavoitteena on luoda selkeä visualisointi yleisimmistä virhetyypeistä sekä kehittää työkalu, joka auttaa opetushenkilökuntaa tunnistamaan opiskelijoiden kohtaamia haasteita ohjelmoinnin oppimisessa.

1.3 Tutkimuskysymykset

Tämän kandidaatintyön päätutkimuskysymys on: ’Mitkä ovat yleisimmät opiskelijoiden tekemät virheet’ ja 'Miten ohjelmoinnin perusteet harjoitustöiden analyysiä voidaan hyödyntää kurssin kehittämisessä?' Harjoitustyö toimii kurssin kokoavana elementtinä, jossa opiskelijan tulee soveltaa kaikkia kurssin aikana opittuja taitoja. Täten harjoitustyön analysointi tarjoaa ainutlaatuisen näkökulman opiskelijoiden kokonaisvaltaiseen oppimiseen.

1.4 Työn rakenne

Työ koostuu kuudesta pääluvusta. Luku kaksi esittelee tutkimuksen teoreettisen taustan, aiemman tutkimuksen sekä käytetyt analyysimenetelmät. Luku kolme keskittyy käytettyjen analyysimenetelmien ja työkalujen esittämiseen. Luku neljä sisältää tuloksien esittämisen ja pohjaa itse suuremmalle pohdinnalle. Luku viisi keskittyy tuloksien syvempään pohdintaan ja tutkimuskysymyksien vastaamiseen. Luku kuusi tiivistää tutkielman pääpiirteet ja tavoitteet yhteen lukuun.

2 Kirjallisuuskatsaus

Oppimisanalytiikalla tarkoitetaan opiskelijadataan perustuvaa mittaamista, keräämistä, analysointia ja raportointia, jonka avulla pyritään ymmärtämään ja parantamaan oppimisprosesseja (Long & Siemens, 2011). Viberg ja Grönlund (2023) huomauttavat, että oppimisanalytiikka ei koske vain oppimistulosten parantamista, vaan myös opiskelijan kontekstin huomioimista, kuten sosiaalisia ja ympäristöllisiä tekijöitä. Ohjelmoinnin opetuksessa oppimisanalytiikan soveltaminen voi tarkoittaa esimerkiksi opiskelijan palautteen analysointia, jotta saataisiin ymmärrystä opiskelijoiden kohtaamista haasteista ja oppimisprosesseista.

2.1 Muu tutkimus

Erilaiset tapaustutkimukset ovat osoittautuneet hyödyllisiksi välineiksi kurssien kehittämisessä, erityisesti ohjelmoinnin opetuksessa. Esimerkiksi Niemenmaa (2023) kandidaatintutkielmassaan analysoi LUT-yliopiston *Ohjelmoinnin perusteet* -kurssin vuoden 2022 toteutuksen opiskelijapalautuksia. Analyysin kohteina olivat palautuksien määrät, palautuksien koodirivien pituudet sekä palautuksien saamat arvosanat. Tutkimuksessa pyrittiin vastaamaan kysymyksiin, kuten "Miten opiskelijoiden tehtäväpalautuksia voi tutkia ja hyödyntää kurssin kehittämiseksi?" ja "Mitä opiskelijapalautuksissa on tapahtunut?" Analysoimalla tehtäväkohtaisten palautusmäärien graafisia piikkejä sekä palautusaikojen ja rivimäärien vaihtelua kurssin aikana, tutkimus tarjosi suosituksia kurssin kehittämiseksi ja osoitti, kuinka tällainen analyysi voisi toimia pohjana jatkotutkimuksille. (Niemenmaa 2023).

Samansuuntaisia havaintoja teki myös Aurelma (2019) analysoidessaan LUT-yliopiston Ohjelmoinnin perusteet -kurssin Python-tehtävien palautuksia. Hänen tutkimuksessaan tarkasteltiin opiskelijoiden palautuksia koko kurssilta, keskittyen erityisesti koodin rivimääriin, kompleksisuuteen, Halstead-metriikoihin ja muuttujien nimeämiskäytäntöihin. Tutkimus osoitti, ettei mikään mitatuista suureista noudattanut lineaarista kasvua kurssin edetessä, vaan datassa havaittiin odottamattomia piikkejä, jotka liittyivät sekä kurssin sisäisiin että ulkoisiin tekijöihin. Yhtenä havainto oli että opiskelijoiden tuottama koodi sisälsi huomattavasti vähemmän kommenttirivejä kuin malliratkaisut. Tämä ero selittynee sillä, että malliratkaisut on suunniteltu opetustarkoitukseen ja niiden on tarkoitus olla helposti luettavia ja

ymmärrettäviä. Opiskelijoiden omat ratkaisut taas on tehty vain tehtävän läpäisemistä varten, eikä niitä ole tarkoitettu muiden luettavaksi tai uudelleenkäytettäväksi. (Aurelma 2019).

Ohjelmointivirheiden analyysi on myös olennainen osa oppimisanalytiikkaa. Tähän liittyen havaintoja tekivät Mase ja Nel (2022) tutkimuksessaan, jossa analysoitiin 38 aloittelevan ohjelmoijan tekemiä virheitä yhden lukukauden aikana. Tutkimuksessa tarkasteltiin 684 ohjelmointitehtävää C# ohjelmointi kielestä, joista löydettiin yhteensä 2396 virhettä. Virheet luokiteltiin neljään kategoriaan: syntaksivirheet (12,98 prosenttia), tyyppivirheet (32,85 prosenttia), semanttiset virheet (46,24 prosenttia) ja loogiset virheet (7,93 prosenttia). Merkittävimpänä havaintona oli, että semanttiset ja tyyppivirheet muodostivat valtaosan kaikista virheistä, mikä viittaa siihen, että aloittelevat ohjelmoijat kamppailevat erityisesti näiden virheiden tunnistamisen ja korjaamisen kanssa. Mase ja Nel korostavat, että opettajien tulisi kiinnittää erityistä huomiota näiden virheiden toistuvuuteen, sillä se voi toimia indikaattorina opiskelijan osaamisesta. (Mase ja Nel, 2022.)

Ohjelmointivirheitä on myös analysoitu aikaisemmin. Lahtinen et al. (2005) teki kansainvälistä kyselytutkimusta opiskelijoiden suurimmista vaikeuksista ohjelmointiin liittyen. Tutkimuksessa paljastui, että aloittelevilla ohjelmoijilla on usein rajoittunut ymmärrys ohjelmoinnista, mikä johtaa virheisiin toistolauseiden ja parametrien määrittelyssä. Vaikeuksia aiheutti myös ohjelmien toiminnallisuuden jakaminen aliohjelmiin sekä ohjelmien virheenkäsittely, jotka vaativat suurempaa ymmärrystä ohjelman kokonaisuudesta. (Lahtinen et al., 2005.)

Kristiansen et al. (2024) vertailivat automaattisen ja manuaalisen arvioinnin eroja ohjelmoinnin opetuksessa. 117 opiskelijan kokeessa verrattiin kolmea arviointitapaa, pelkkä automaattinen arviointi, pelkkä manuaalinen arviointi eli opetusassistenttien toteuttamana sekä näiden yhdistelmä. Tulokset osoittivat, että molempien arviointimuotojen yhdistelmä toimii parhaiten kokonaisuuden kannalta. Automaattinen arviointi edisti tehtävien tehokasta suorittamista, kun taas manuaalinen arviointi oli opiskelijoiden mieleen sen tarjoaman henkilökohtaisemman palautteen vuoksi. Tutkimuksen havainnot ovat relevantteja tämän tutkielman kannalta, sillä LUT-yliopiston ohjelmoinnin peruskurssilla käytetään vastaavaa yhdistelmää arviointimenetelmistä: CodeGrade-järjestelmä tarkistaa automaattisesti koodin toiminnallisuuden vertaamalla sitä mallivastaukseen, ASPA-työkalu analysoi koodin tyylivirheitä, ja opetusassistentit tarjoavat manuaalista arviointia tunnistaen virheitä ja antaen henkilökohtaista palautetta. (Kristiansen et al. 2024.)

3 Työkalut ja analysointi

Tämän tutkielman tavoitteena on analysoida ja visualisoida opiskelijoiden tekemiä virheitä Ohjelmoinnin perusteet -kurssin harjoitustöistä. Opiskelijat voivat valita kolmen harjoitustyö tason väliltä, minimi, perus ja tavoite. Tason noustessa nousee myös harjoitustyön vaikeus ja vaativuus, sillä opiskelijoiden tulee hyödyntää vaikeampia käsitteitä ja ratkoa vaikeampia ongelmia kurssilta. Minimitason työ kokoaa kurssilla käydyt pääpiirteet ja perus- ja tavoitetaso rakentavat tähän päälle. Tutkimuksen pääkysymykset ovat: ”Mitkä ovat yleisimmät opiskelijoiden tekemät virheet?” ja ”Miten Ohjelmoinnin perusteet -kurssin harjoitustöiden analyysiä voidaan hyödyntää kurssin kehittämiseksi.”

3.1.1 Python ja datan analysointi

Tämän tutkielman analysointi on suoritettu Python-ohjelmointikielellä. Tähän ratkaisuun päädyttiin sen helppokäyttöisyyden ja laajan kirjastovalikoiman takia. Python mahdollistaa monipuolisen ja tehokkaan data-analyysin useilla erikoistuneilla kirjastoilla, joista keskeisimpänä tässä työssä käytettiin Pandas-kirjastoa. Pandas-kirjaston avulla analysoitua dataa visualisoitiin tämän jälkeen käyttämällä Matplotlib-kirjastoa, joka antaa monipuoliset ja helppokäyttöiset työkalut kaavioiden luomiseen.

3.1.2 Pandas ja Matplotlib

Pandas on avoimen lähdekoodin kirjasto, joka tarjoaa tehokkaita ja helppokäyttöisiä data-analyysityökaluja Python-ohjelmointikielelle (pandas 2024). Kirjasto on todella hyödyllinen jäsennellyn datan käsittelyssä kuten JSON ja YAML tiedostojen, ja tässä tutkielmassa keskeisiä ominaisuuksia ovat DataFrame-tietorakenne, joka luo kaksiulotteisen taulukon ja monipuoliset datan suodatus ja analysointi työkalut. Matplotlibissä on kattava kirjasto työkaluja, jotka helpottavat staattisten, animoitujen ja interaktiivisten visualisointien rakentamiseen (Matplotlib 2024). Tässä työssä on käytetty vain staattisia kaavioita analysoidun datan visualisointiin.

3.1.3 Jupyter

Jupyter Notebook on avoimen lähdekoodin web-sovellus, joka mahdollistaa ohjelmakoodin, tekstin ja visualisointien yhdistämisen samaan dokumenttiin (Project Jupyter 2024). Tässä tutkielmassa Jupyter Notebook toimi keskeisenä kehitysympäristönä, sillä se mahdollistaa datan analysoinnin vaiheittaisen kehittämisen ja tulosten välittömän tarkastelun. Jupyter Notebookin interaktiivinen luonne helpottaa data-analyysin iteratiivista kehittämistä, kun koodia voidaan suorittaa ja muokata pienissä osissa, ja tulokset ovat heti nähtävissä.

3.2 Analysointi

Tässä tutkielmassa analysoidaan opiskelijoiden tehtäväpalautusten virheitä ja suoriutumista YAML-tiedostosta, joka sisältää tietoja opiskelijoiden palautuksista, arvosanoista ja virhepisteistä. Kaikki analysoidut koodit ovat toiminnallisuudeltaan oikeita, sillä ne ovat läpäisseet CodeGrade-automaatiotarkastajan. Tarkastaja vertaa koodin tuottamia tuloksia mallivastauksiin, ja vain hyväksytyt palautukset etenevät manuaaliseen arviointiin. Vaikka tutkimuksen otos käsittää 119 opiskelijan palautukset, sen voidaan katsoa edustavan riittävän kattavaa otosta kurssin opiskelijoista ja näiden perusteella voidaan tehdä suuntaa antavia johtopäätöksiä.

Data kerättiin kahdesta eri palautuskierroksesta, ja se sisältää tietoja opiskelijoiden työn tasoista, virhetyypeistä sekä virheiden määristä. Jokainen palautus sisältää opiskelijan perustiedot (nimi, sähköposti, työn taso) sekä suoritukseen liittyvät tiedot kuten virhepisteet ja arvosana. Kaikki henkilökohtainen data on kuitenkin anonymisoitu, joten yksittäisen opiskelijan suorituksia ei voi selvittää. Virhepisteitä kertyy erilaisista virheistä, jotka voidaan kategorisoida vielä virhetyyppeihin. Kaikki virheet eivät ole saman arvoisia ja joistain virheistä saa enemmän virhepisteitä ja joistain vähemmän. Tehtävän läpäiseminen edellyttää alle kahta virhepistettä. Analyysissä tarkastellaan erityisesti ryhmäkohtaisia eroja virhepisteissä, läpäisyprosenteissa sekä virhetyyppien jakaumissa. Tulosten visualisointiin käytetään pylväsdiagrammeja, jotka havainnollistavat mm. ryhmäkohtaisia opiskelijamääriä, keskimääräisiä virhepisteitä ja virhetyyppien jakaumaa eri palautuskierroksilla.

4 Design Artefact

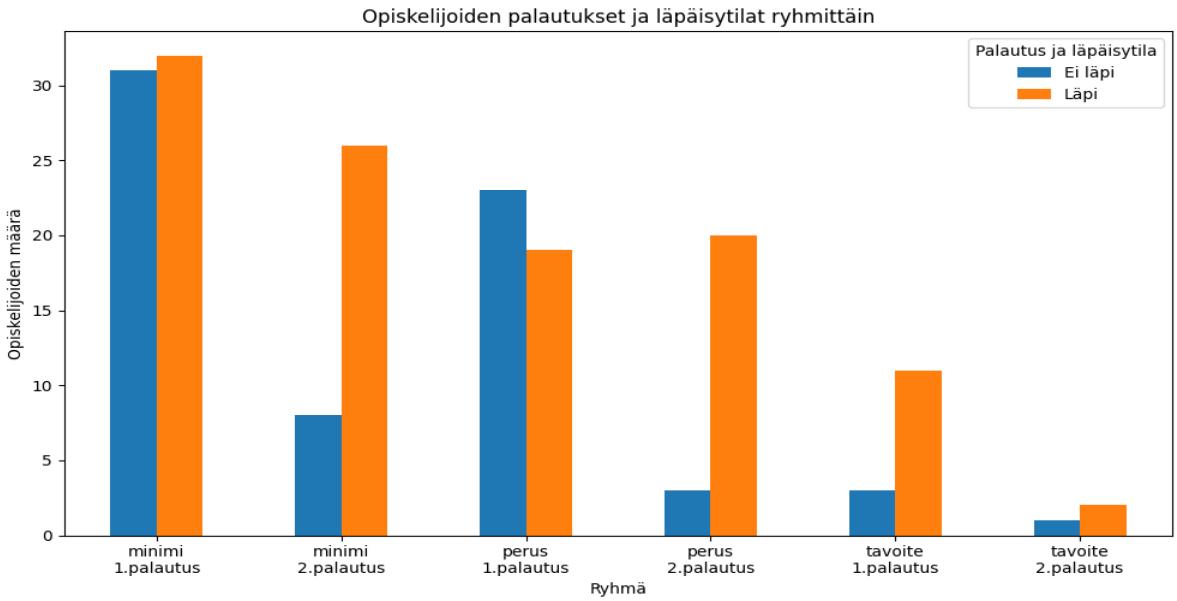
4.1 Läpipääsy

Kuvasta 1 voidaan tarkastella opiskelijoiden läpipääsyä eri tasoisten harjoitustöiden osalta molemmilta palautuskierroksilta. Minimitason työssä ensimmäisellä kierroksella läpipääsyprosentti oli noin 51 prosenttia, kun 63:sta opiskelijasta 32 pääsi läpi ja 31 ei päässyt läpi. Toisella palautuskierroksella läpipääsyprosentti oli noin 76 prosenttia, kun 34:stä opiskelijasta 26 pääsi läpi ja 8 ei päässyt.

Perustason työssä ensimmäisellä kierroksella läpipääsyprosentti oli noin 45 prosenttia, kun 42:sta opiskelijasta 19 pääsi läpi ja 23 ei päässyt. Toisella palautuskierroksella läpipääsyprosentti oli noin 87 prosenttia, kun 23:sta opiskelijasta 20 pääsi läpi ja 3 ei päässyt.

Tavoitetason työssä ensimmäisellä kierroksella läpipääsyprosentti oli 79 prosenttia, kun 14 opiskelijasta 11 pääsi läpi ja 3 ei päässyt. Toisella palautuskierroksella läpipääsyprosentti oli noin 66 prosenttia, kun 3:sta opiskelijasta 2 pääsi läpi ja 1 ei päässyt.

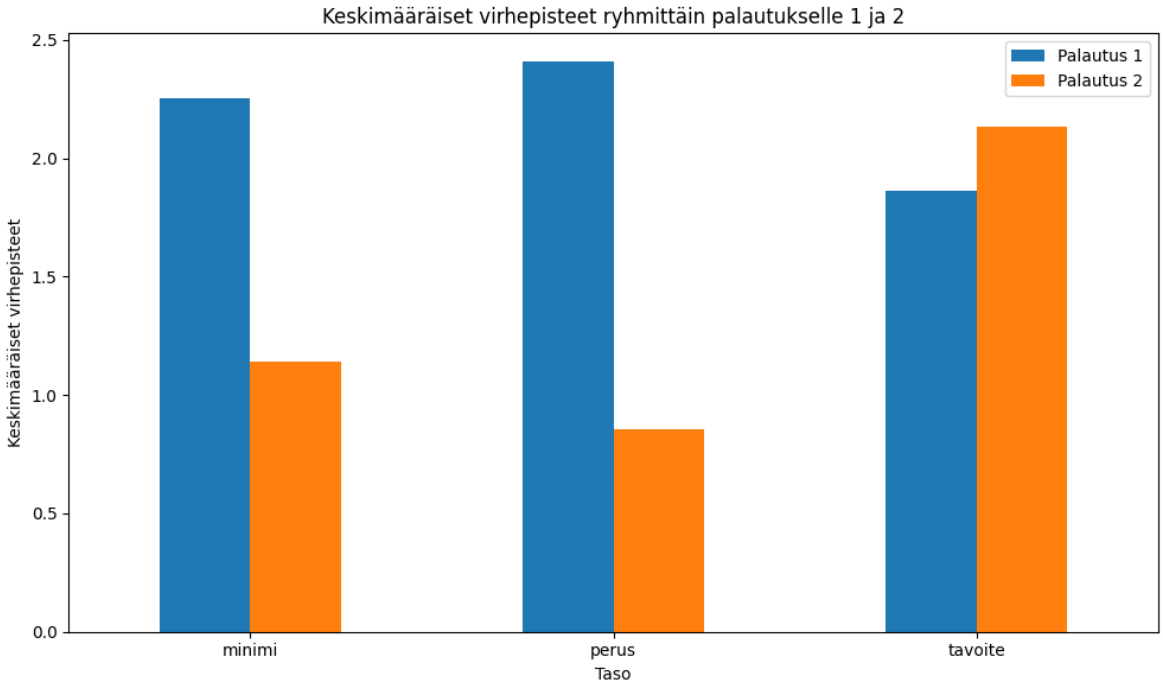
Havaittiin siis, että läpipääsyprosentti nousi toisella palautuskierroksella huomattavasti paitsi tavoitetason töissä, tämä voidaan kuitenkin selittää pienen otoskoon takia.



Kuva 1. Opiskelijoiden läpipääsy ryhmitetty eri työn tasojen (minimi, perus, tavoite) ja palautuskertojen mukaan

4.2 Keskimääräiset virhepisteet

Kuvasta 2 voidaan tarkastella opiskelijoiden keskimääräisiä virhepisteitä eri tasojen ja palautuskierrosten välillä. Minimitason työssä keskimääräinen virhepistemäärä oli ensimmäisellä palautuskierroksella 2,26 ja toisella kierroksella 1,14. Perustason työssä opiskelijat saivat ensimmäisellä kierroksella keskimäärin 2,41 virhepistettä ja toisella kierroksella 0,86 virhepistettä. Tavoitetason työssä keskimääräiset virhepisteet olivat ensimmäisellä palautuskierroksella 1,86 ja toisella kierroksella 2,13. Kuvasta havaitaan, virhepisteet ovat melko tasaiset minimi- ja perustason välillä, mutta niitä esiintyy enemmän perustason töissä. Tavoitetason töissä virheitä esiintyi vähiten mikä saattaa selittyä opiskelijoiden paremmalla motivaatiolla työtä kohtaan. Virhepisteet myös vähenivät yleisesti toisella palautuskierroksella, mutta tavoitetasolla virhepisteet kasvoivat hieman, mikä saattaa selittyä pienellä otoskoolla.



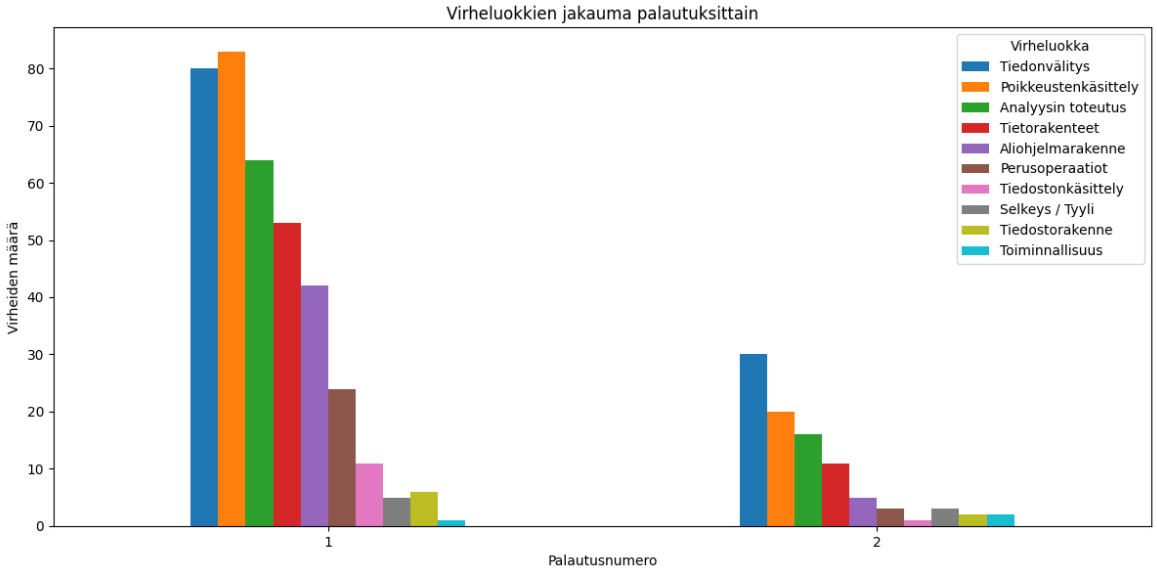
Kuva 2. Keskimääräiset virhepisteet ryhmitetty eri työn tasoille (minimi, perus ja tavoite) ja palautuskerroille.

4.3 Virheluokkien jakauma

Ensimmäisellä palautuskerralla virheitä havaittiin yhteensä 369 ja luvut sisältävät kaikki havaitut virheet, mukaan lukien tapaukset, joissa opiskelija toistaa saman virheen useampaan kertaan. Kuvasta 3. nähdään virheet jaettuna niiden virheluokkiin eri palautuskierroksilta, mikä antaa paremman kuvan siitä minkälaisia virheitä opsikelijat ovat tehneet. Ensimmäisellä palautuskierroksella eniten virheitä esiintyi Poikkeustenkäsittelyssä 83 virhettä, Tiedonvälityksessä 80 virhettä ja Analyysin toteutuksessa 64 virhettä. Aliohjelmarakenteessa havaittiin 42 virhettä ja Tietorakenteissa 53 virhettä. Vähiten virheitä esiintyi Selkeys/Tyylissä 5 virhettä, Tiedostorakenteessa kuusi virhettä ja Toiminnallisuudessa yksi virhe.

Toisella palautuskierroksella virheitä esiintyi yhteensä 93, mistä suurimmat virhemäärät olivat Tiedonvälityksessä 30 virhettä, Poikkeustenkäsittelyssä 20 virhettä ja Analyysin toteutuksessa 16 virhettä. Tietorakenteissa virheitä oli 11 ja Aliohjelmarakenteessa viisi. Vähiten virheitä löytyi Tiedostorakenteissa kaksi virhettä, Toiminnallisuudessa kaksi virhettä ja Tiedostonkäsittelyssä yksi virhe.

Ongelmalliset osa-alueet kuten poikkeustenkäsittely ja tiedonvälitys säilyivät suurimpin virhekategorioina molemmissa palautukissa. Virheiden kokonaismäärä on kuitenkin laskenut merkittävästi, mikä osoittaa, että opiskelijat ovat löytäneet ja korjanneet tekemiään virheitä.

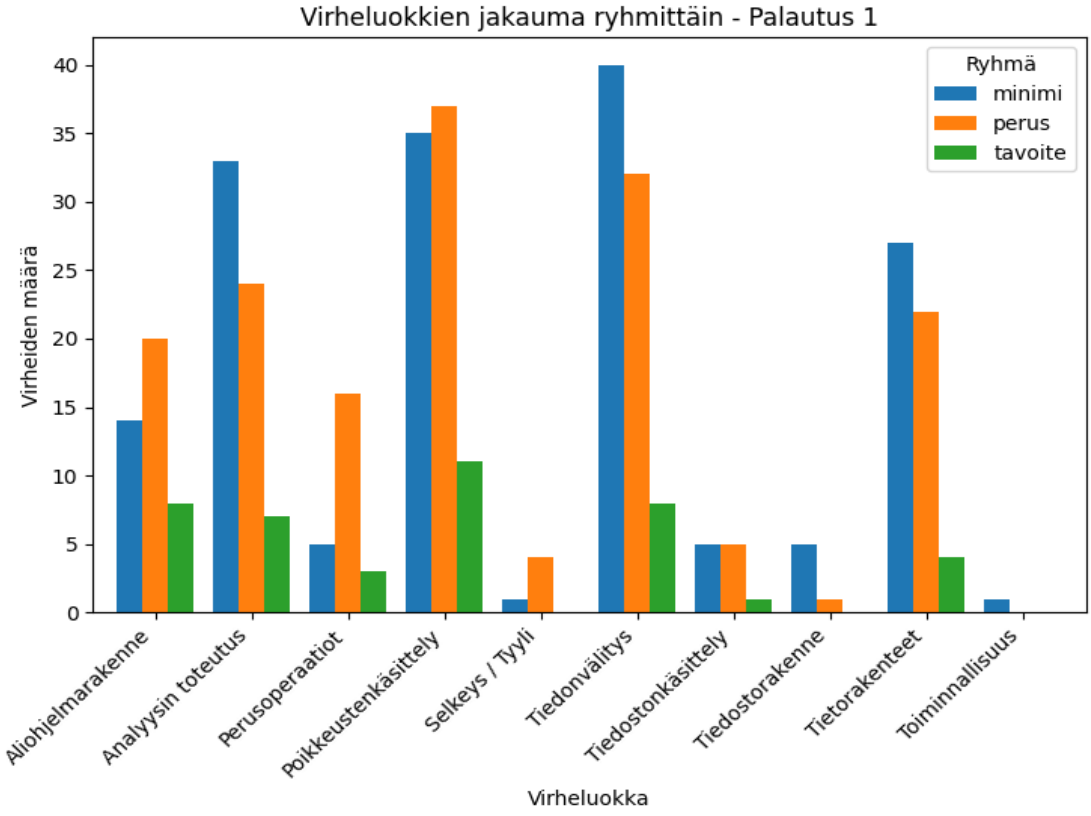


Kuva 3. Virheet ryhmiteltynä virheluokkiin eri palautuskerroilta

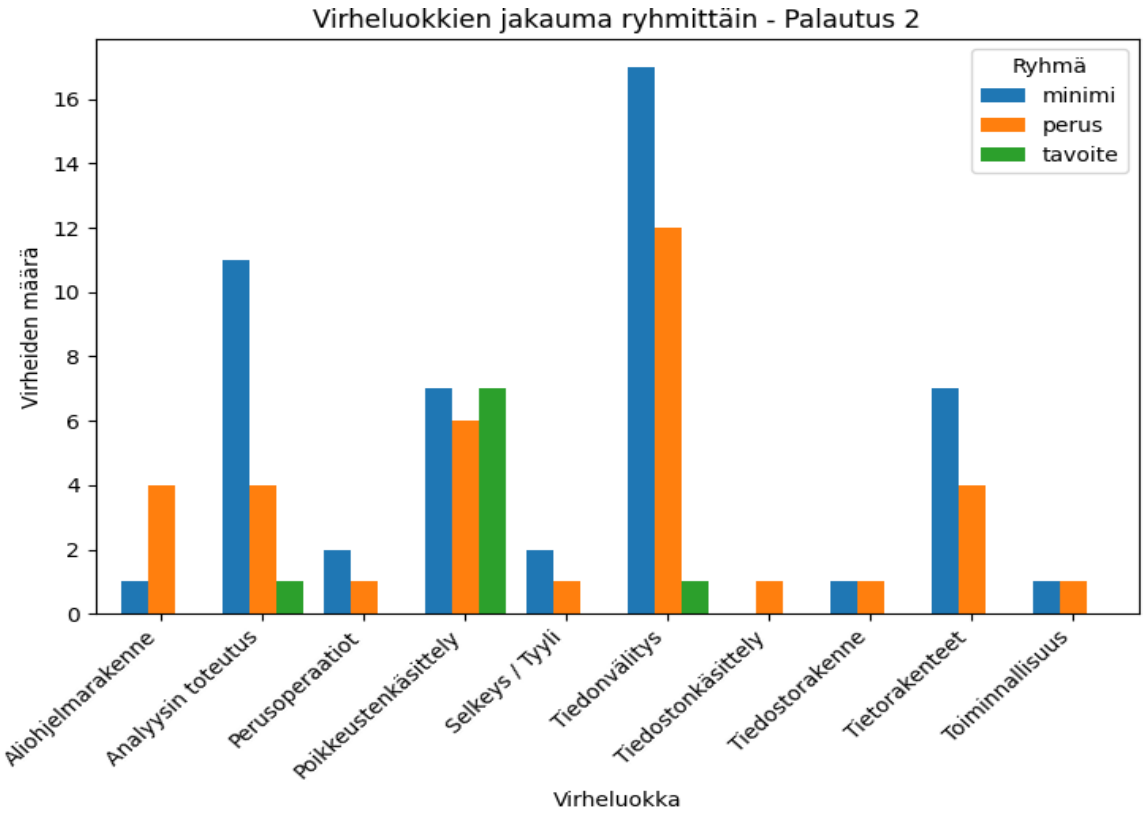
4.3.1 Ryhmittäin

Kuvasta 4 ja 5 voidaan tarkastella virheluokkien esiintymismääriä eri työn tasoilla ja eri palautus kerroilta. Kuvassa 4 on ensimmäisen palautuksen tulokset ja kuvassa 5 on toisen palautuksen. Ensimmäisessä palautuksessa minimitason töissä eniten virheitä esiintyi Tiedonvälityksessä 40 virhettä, Poikkeustenkäsittelyssä 35 virhettä ja Analyysin toteutuksessa 33 virhettä. Perustason töissä suurimmat virhemäärät olivat Poikkeustenkäsittelyssä 37 virhettä, Tiedonvälityksessä 32 virhettä ja Analyysin toteutuksessa 24 virhettä. Tavoitetasolla eniten virheitä tehtiin Poikkeustenkäsittelyssä 11 virhettä, Tiedonvälityksessä kahdeksan virhettä ja Aliohjelmarakenteessa kahdeksan virhettä.

Toisella palautuskierroksella minimitason töissä esiintyi eniten virheitä Tiedonvälityksessä 17 virhettä ja Analyysin toteutuksessa 11 virhettä. Perustason töissä Tiedonvälitys 12 virhettä ja Poikkeuksienkäsittelyssä kuusi virhettä. Tavoitetason töissä merkittävin virheluokka oli Poikkeustenkäsittely seitsemän virhettä ja palautuksien pienten määrän takia muissa kategorioissa oli vain yksittäisiä virheitä.



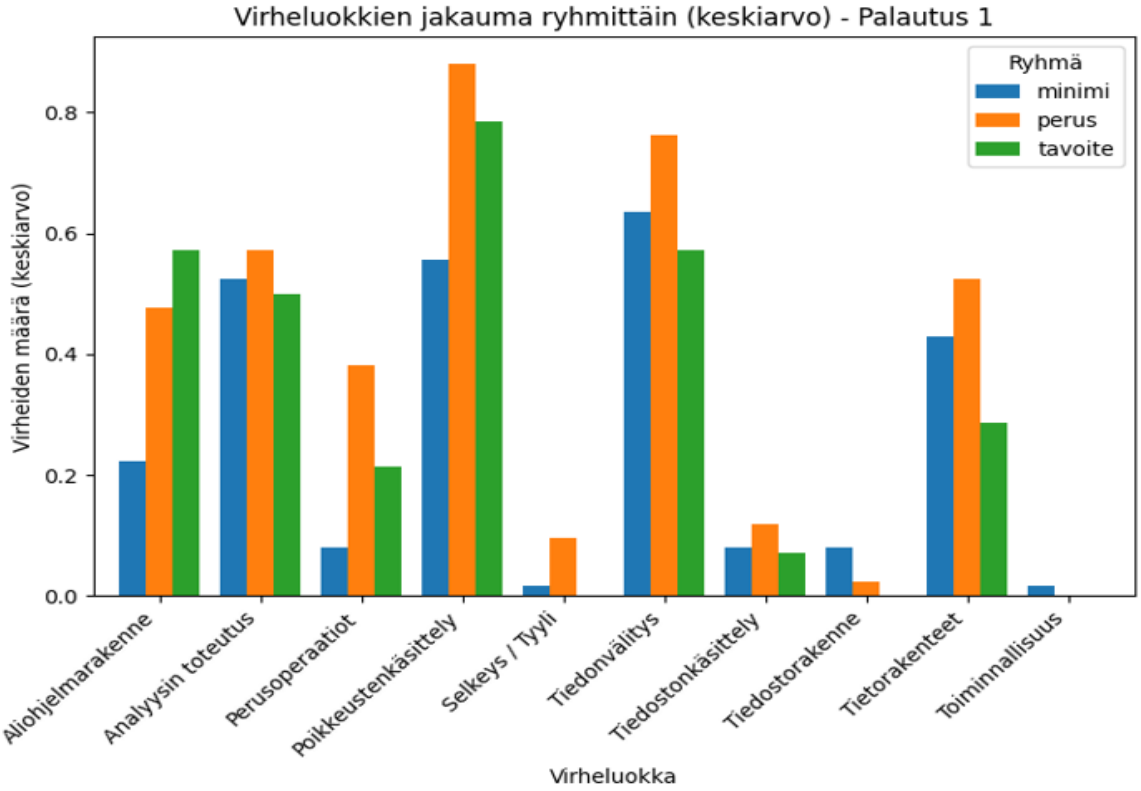
Kuva 4. Virheluokkien jakauma ryhmittäin palautukselle 1



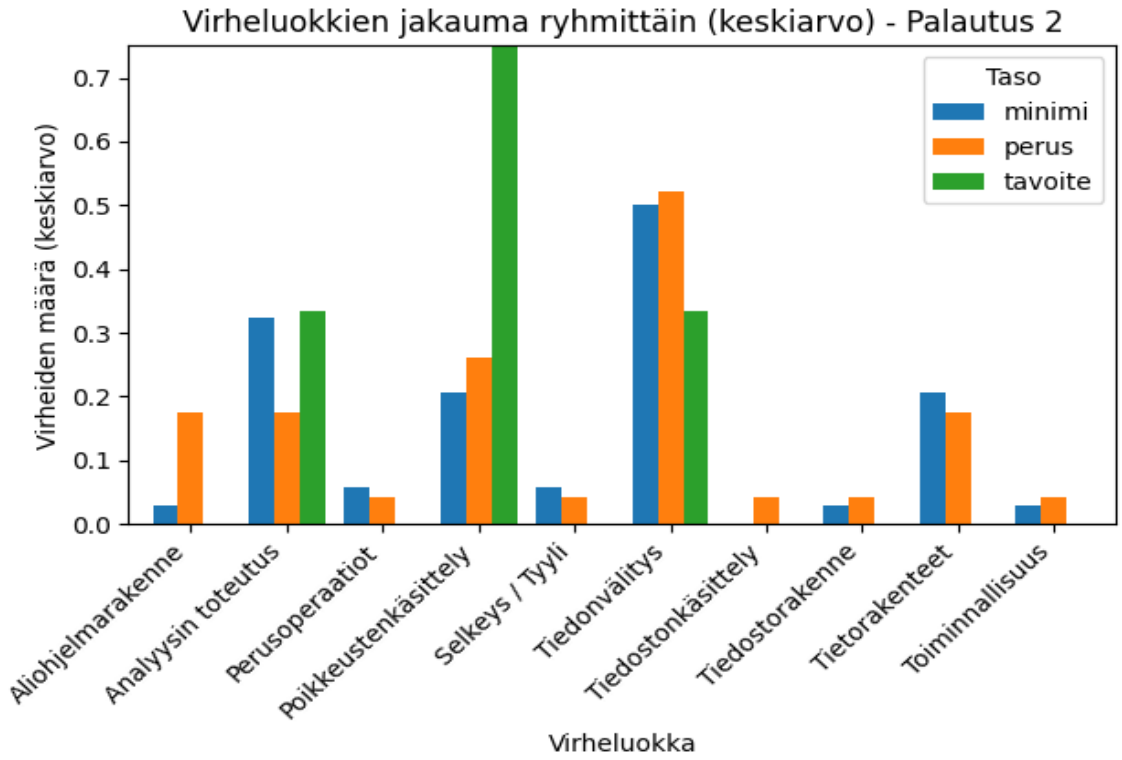
Kuva 5. Virheluokkien jakauma ryhmittäin palautukselle 2.

4.3.2 Keskiarvot

Koska tavoitetason töiden määrä oli huomattavasti pienempi kuin minimi- ja perustason töiden, absoluuttiset virhemäärät eivät anna vertailukelpoista kuvaa eri tasojen välisistä eroista. Tämän vuoksi kuvissa 6 ja 7 esitetään virheluokkien esiintyminen keskiarvoina opiskelijaa kohden palautuksille yksi ja kaksi. Toisen palautuksen kuvaaja on kuitenkin rajattu 0,75 virheeseen, koska tavoitetason pienen palautusmäärän takia poikkeustenkäsittelyyn liittyvät virheet kasvoivat 2.33 virheeseen ja näin vaikutti koko kuvaajan luettavuuteen. Keskiarvojen avulla voidaan paremmin näyttää virheiden suhteellista esiintymistä eri tasojen välillä, riippumatta palautusten kokonaismäärästä. Kuvasta voidaan huomata, että perus- ja tavoitetasolla nähdään enemmän virheitä poikkeustenkäsittelyssä ja aliohjelmarakenteessa, kuin minimitason töissä.



Kuva 6. Virheluokkien jakauma ryhmittäin keskiarvon perusteella palauttelle 1



Kuva 7. Virheluokkien jakauma ryhmittäin keskiarvon perusteella palauteelle 2, kuva rajattu 0.75 virheeseen ja ylimenevä arvo on 2.33

5 Pohdinta

5.1 Läpipääsy

Läpipääsyprosenttien merkittävä paraneminen toisella palautuskierroksella, minimitasolla 51 prosentista 76 prosenttiin ja perustasolla 45 prosentista 87 prosentiin viittaa siihen, että opiskelijat hyötyivät saadusta palautteesta ja korjausmahdollisuudesta. Tämä havainto tukee Kristiansen et al. (2024) tutkimuksen tuloksia, joissa todettiin automaattisen ja manuaalisen arvioinnin yhdistelmän olevan tehokkain tapa auttaa opiskelijoiden kehitystä. Erityisesti perustason töiden läpipääsyprosentin kasvaminen noin 42 prosenttilla kuvaa opiskelijiden kykyä hyödyntää saamiaan palautuskommentteja.

5.2 Keskimääräiset virhepisteet

Keskimääräisten virhepisteiden analysoinnissa huomattiin suuria muutoksia eri palautus kierrosten välillä. Minimitasolla ensimmäisellä palautuksella virhepisteitä oli keskimäärin 2.26 ja toisella palautus kerralla 1.14. Perustasolla havaitiin samankaltaista paranusta, kun ensimmäisellä palautuskerralla oli keskimäärin 2.41 virhepistettä ja toisella palutuksella oli 0.86 virhepistettä. Yleisenä trendinä siis havaittiin, että virhepisteet noin puolittuivat ensimmäisen palautuksen jälkeen ja virhepisteet eri tasojen välillä olivat lähellä toisaan. Poikkeuksena oli kuitenkin tavoitetaso, missä virhepisteet kasvoivat ensimmäisen palautuksen jälkeen 1.86 virhepisteestä 2.13 virhepisteeseen. Tämän voidaan kuitenkin selittää toisen palautukserran pienellä otoskoolla, mikä laski jo alhaisesta 14 opiskelijasta kolmeen opiskelijaan. Tämän takia jo pienetkin poikkeamat tilastoissa näkyvät suuresti itse luvuissa.

5.3 Virheluokkien jakauma

Virheluokkien jakauma paljastaa, että suurimmat haasteet tutkielmassa liittyivät poikkeustenkäsittelyyn, tiedonvälitykseen ja analyysin toteutukseen. Opetusassistenttien kommenttien perusteella poikkeustenkäsittelyyn liittyvät virheet johtuivat väränlaisesta poikkeustenkäsittelystä tai sen puuttumisesta kokonaan. Tiedonvälitykseen liittyvät virheet

liittyivät aliohjelmien parametrien ja paluuarvojen puuttumiseen ja analyysin toteutuksessa virheet johtuivat vääränlaisesta valintarakenteesta analyysin toteuttamisessa tai alkuarvojen vääränlaisesta alustamisesta.

Havaitut tulokset hieman poikkeavat Masen ja Nelin (2022) tutkimuksesta, jossa semanttiset virheet muodostivat noin 46 prosentin osan kaikista virheistä. Käytetty aineisto on kuitenkin läpäissyt CodeGrade-automaattitarkastuksen, joten kaikki analysoidut palautukset ovat toiminallisesti oikeita. Semanttisten virheiden lisäksi tyyppivirheet muodostivat noin 33 prosenttia virheistä, syntaksivirheet 13 prosenttia ja loogiset virheet kahdeksan prosenttia. Tämän tutkielman aineistossa virheet liittyivät enemmän logiikka- ja rakenteellisiin ongelmiin kuin tyyppi- ja syntaksivirheisiin. Tämä ero voi selittyä esimerkiksi käytetyn ohjelmointikielen, opetusmenetelmien tai tehtävänantojen takia. Lahtinen et al. (2005) puolestaan havaitsi, että aloittelevat ohjelmoijat löysivät toistorakenteet, parametrien välitykset ja poikkeustenkäsittelyt vaikeiksi konsepteiksi, koska ne vaativat syvempää ymmärrystä ohjelman kokonaisuudesta. Tässä tutkielmassa havaittiin samankaltaisia havaintoja, sillä vaikeimmiksi osa-alueiksi nousivat juuri poikkeustenkäsittely ja tiedonvälitys.

Toisella kierroksella virheiden kokonaismäärä väheni huomattavasti 369 virheestä 93 virheeseen, mutta virheluokkien suhteelliset osuudet pysyivät samankaltaisina. Tiedonvälitys ja poikkeustenkäsittely säilyivät suurimpina haasteina, mikä viittaa näiden konseptien haasteellisuuteen aloitteleville ohjelmoijille.

5.3.1 Ryhmittäin

Kuvasta 4 ja 5 voidaan huomata, että minimitason töissä olisi eniten ongelmia kaikissa kategorioissa paitsi poikkeusten käsittelyssä, tämä on kuitenkin hieman harhaan johtavaa. Minimitason palautuksia oli eniten, joten on loogista, että minimitason töissä tapahtuu eniten virheitä suurimmassa osassa virheluokkia. Kuvasta 6 ja 7 voidaan katsoa virheluokkien jakaumaa opiskelijoita kohden mikä paremmin heijastaa minkä virheluokan kanssa eri tasojen töissä on ollut ongelmia. Aliohjelma rakenteen ja poikkeustenkäsittelyssä on huomattavasti enemmän virheitä perus- ja tavoitetason töissä. Tämä voidaan selittää sillä, että perus- ja tavoitetason työt vaativat vaikeampi aliohjelmarakenteita ja lisää poikkeustenkäsittelyä kuin

minimitason työt. Muiden virheluokkien kohdalla tasojen väliset erot ovat minimaaliset paitsi perusoperaatioiden kohdalla nähdään suuri kasvu minimitason töistä perustaston töihin. Opetusassistenttien kommentteja lukien yleisin virhe perusoperaatioissa liittyi tietorakenteen puuttumiseen tehtävässä. Tehtävänannon epäselvyys saattaa olla yksi syy siihen, miksi moni opiskelija teki kyseisen virheen. Kaikki opiskelijat eivät kuitenkaan tehneet tätä kyseistä virhettä, joten tehtävänannon epäselvyyden lisäksi on todennäköisesti muita tekijöitä. Yksi mahdollinen syy opiskelijoiden virheisiin on se, että he ovat lukeneet vain tehtävänannon, mutta eivät tyyliohjetta. Tyyliohje on tehtävänannosta erillinen tiedosto, joka määrittelee, miten tehtävä tulisi muotoilla ja mitkä ohjelmointikäytännöt ovat oikeaoppisia. Tämä tyyliohje voi kuitenkin olla vaikea ymmärtää, mikä tekee tiettyjen ohjeiden löytämisen siitä haasteelliseksi. Tämän vuoksi monet opiskelijat saattavat jättää tyyliohjeen huomiotta tai eivät huomioi sen vaatimuksia riittävällä tarkkuudella, mikä johtaa virheisiin tehtävien toteutuksessa.

5.4 Johtopäätökset

Kuvasta 1. selivisi, että kurssilla käytössä olevilla menetelmillä, kuten toisella palautuskerralla ja virheiden esittämisellä opiskelijoille on selkesti positiivinen vaikutus oppimiseen ja suoritusten parantumiseen. Tämä voidaan nähdä suoraan läpipääsyprosenttien parantumisella toisessa palautuksessa.

Samalla kuvasta 3. voidaan nähdä haasteellisia osa-alueita, joissa opiskelijat tekivät toistuvasti virheitä. Poikkeustenkäsittelyssä opiskelijat tekivät joko vääränlaista poikkeustenkäsittelyä tai eivät tehneet ollenkaan, tiedonvälityksessä opiskelijoiden virheet liittyivät aliohjelmien parametrien ja paluuarvojen puuttumiseen ja analyysin toteutuksessa opiskelijat käyttivät vääränlaista valintarakennetta analyysin toteuttamisessa ja eivät alustaneet alkuarvoja analysoitavan datan ensimmäisillä luvuilla. Virheiden kokonaismäärä väheni toisella palautuskerralla, mutta poikkeustenkäsittelyyn, tiedonvälitykseen ja analyysin toteuttamiseen liittyvät virheet säilyivät kuitenkin suhteellisesti suurimpina.

Kuvista 6. ja 7. voidaan nähdä töiden eri tasojen välisiä eroja virheiden määrissä ja niden jakautumisessa. Perus- ja tavoitetason töissä nähtiin enemmän virheitä aliohjelmarakenteiden ja poikkeustenkäsittelyn kohdalla, mikä voidaan selittää näiden tasojen vaativimmilla vaatimuksilla. Minimitason työn helpompi aliohjelmarakenne ei puolestaan aiheuttanut yhtä

paljon ongelmia. Minimitason töissä hankalimmaksi osa-alueeksi osoittautui tiedonvälitys aliohjelmien välillä, mutta poikkeustenkäsittelyssä havaittiin myös paljon virheitä, kuitenkin vähemmän kuin perus- ja tavoitetasoilla.

5.5 Vastaukset tutkimuskysymyksiin

Tutkimuksen tavoitteena oli vastata kahteen päätutkimuskysymykseen: "Mitkä ovat yleisimmät opiskelijoiden tekemät virheet?" ja "Miten ohjelmoinnin perusteet harjoitustöiden analyysiä voidaan hyödyntää kurssin kehittämisessä?" Analyysin perusteella yleisimmät opsikelijoiden tekemät virheet liittyivät poikkeustenkäsittelyyn, tiedonvälitykseen ja analyysin toteutukseen. Näiden osa-alueiden toistuvat virheet osoittavat, että ne ovat haastavia aloitteleville ohjelmoijille.

Lisäksi virheanalyysi paljasti, miten virhekategoriat jakautuivat eri palautuksien ja työn tasojen mukaan. Ensimmäisellä palautuskerralla vaikeimmaksi osa-alueeksi osoittautui poikkeustenkäsittely, tiedonvälitys ja analyysin toteutus. Toisella palautuskerralla virheiden kokonaismäärä väheni huomattavasti, mutta poikkeustenkäsittelyyn, tiedonvälitykseen ja analyysin toteutukseen liittyvät virheet säilyivät silti suurimpina virhekategorioina.

Virhekategorioiden analyysi eri työn tasojen perusteella paljasti merkittäviä eroja virheiden jakautumisessa. Perus- ja tavoitetason töissä havaittiin keskimääräistä enemmän virheitä aliohjelmarakenteessa ja poikkeustenkäsittelyssä. Tämä voidaan selittä näiden töiden monimutkaisemmista vaatimuksista, mitkä vaativat enemmän aliohjelmia ja poikkeustenkäsittelyä. Minimitason töissä puolestaan virheet painottuivat enemmän yksinkertaisempiin aiheisiin, kuten tiedonvälitykseen ohjelman sisällä ja poikkeustenkäsittelyyn. Minimitason töissä on paljon vähemmän vaadittavaa poikkeustenkäsittelyä kuin perus- ja tavoitetason töissä, mutta se osoittautui silti haastavaksi konseptiksi.

Vastauksena toiseen tutkimuskysymykseen analyysin perusteella voidaan tehdä selkeitä suuntaviivoja kurssin kehittämiseksi. Haastaviksi osoittautuneisiin aiheisiin, kuten poikkeustenkäsittelyyn, tiedonvälitykseen ja analyysin toteutukseen, olisi hyvä panostaa enemmän opetuksessa esimerkiksi konkreettisten harjoitusten ja selkeiden esimerkkien avulla. Lisäksi tehtävänantojen selkeyteen olisi syytä kiinnittää huomiota ja harjoitustöiden tehtävänannoissa voisi olla maininta kyseisestä tyyliohjeen kohdasta, jotta opiskelijat

ymmärtäisivät paremmin vaatimukset ja voisivat välttää väärinkäsityksistä johtuvia virheitä. Yksi käytännön kehitysehdotus on, että kurssin opetuskirjassa olevat koodiesimerkit koottaisiin yhteen erilliseen kansioon, mikä helpottaisi opiskelijoiden pääsyä esimerkkeihin ja tukisi heidän kykyään soveltaa opittuja konsepteja.

6 Yhteenveto

Tässä tutkielmassa tutkittiin opiskelijoiden harjoitustöistä saamia palautteita Ohjelmoinnin perusteet -kurssilta aikaisemmilta kurssi suorituksilta. Tutkimuksen kohteena olivat opiskelijoiden tekemät virheet ja näiden kategoriat sekä läpipääsyprosentti ja näiden vertaaminen eri harjoitustöiden tasojen välillä. Aineistona käytettiin anonymisoitua otosta, joten kaikkia palautuksai ei voitu arviodia. Tästä aineistosta tutkitiin opsikelijoiden virhepisteitä, virheluokkaa ja niiden määrää ja harjoitustyön tasoa. Tutkimuksen tavoitteena oli visualisoinnin avulla helpottaa data-analyysiä ja tämän perusteella löytää opiskelijoille vaikeita osa-alueita.

Tutkimuksen keskeiset kysymykset ovat ”Mitkä ovat yleisimmät opiskelijoiden tekemät virheet?” ja ”Miten ohjelmoinnin perusteet harjoitustöiden analyysiä voidaan hyödyntää kurssin kehittämisessä”. Kysymyksiin vastataan vain käytetyn opiskelija otoksen perusteella ja tulokset vastaavat vain Ohjelmoinnin perusteet -kurssia. Tutkimuksessa selitetään käytetty analysointi tapa ja mitä tuloksia kyseisellä analysoinnilla saatiin. Saatuja tuloksia pohditaan ja mahdollisia kurssin kehitysehdotuksia esitetään puolestaan keskusteluosiossa.

Tämän tutkimusken tuloksia voidaan hyödyntää lähtökohtana jatkotutkimuksille, jos niin nähdään tarpeen. Esimerkiksi voitaisiin analysoida opiskelijoiden viikkotehtävien arvosanoja ja verrata näitä heidän saamaan harjoitustyön arvosanaan ja tutkia mahdollisia yhtäläisyyksiä tai vastakohtia.

Lähteet

Aurelma, N., 2019. Ohjelmoinnin perusteiden kurssin tuotosten staattinen analyysi. Static analysis of CS 1 course work.

Kristiansen, N.G., Nicolajsen, S.M., Brabrand, C., 2024. Feedback on Student Programming Assignments: Teaching Assistants vs Automated Assessment Tool, in: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research, Koli Calling '23. Association for Computing Machinery, New York, NY, USA, pp. 1–10. https://doi.org/10.1145/3631802.3631804

Lahtinen, E., Ala-Mutka, K., Järvinen, H.-M., 2005. A study of the difficulties of novice programmers. ACM, New York, NY, USA, pp. 14–18. https://doi.org/10.1145/1067445.1067453

Long, P., 2011. Proceedings of the 1st International Conference on Learning Analytics and Knowledge, ACM Digital Library. ACM, New York, NY. https://doi.org/10.1145/2090116

Mase, M.B., Nel, L., 2022. Common Code Writing Errors Made by Novice Programmers: Implications for the Teaching of Introductory Programming, in: Leung, W.S., Coetzee, M., Coulter, D., Cotterrell, D. (Eds.), ICT Education. Springer International Publishing, Cham, pp. 102–117. https://doi.org/10.1007/978-3-030-95003-3_7

Matplotlib — Visualization with Python [WWW Document], n.d. URL https://matplotlib.org/ (accessed 11.15.24).

Niemenmaa, E., 2023. Ohjelmoinnin perusteet -kurssin tehtäväpalautusten analyysi. Analysis of student submissions in the first computer science course.

pandas documentation — pandas 2.2.3 documentation [WWW Document], n.d. URL https://pandas.pydata.org/docs/ (accessed 11.10.24).

Project Jupyter [WWW Document], n.d. URL https://jupyter.org (accessed 11.10.24).

Viberg, O., Grönlund, Å., 2023. Practicable Learning Analytics, 1st ed. 2023. ed, Advances in Analytics for Learning and Teaching. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-031-27646-0