OHJELMOINNIN PERUSTEET -KURSSIN VIRHEDATAN ANALYYSI JA VISUALISAATIO

Versio 1.01/17.4.2025

Lappeenrannan–Lahden teknillinen yliopisto LUT

Tietotekniikan koulutusohjelma, Kandidaatintyö

2025

Arttu Korpela

Tarkastajat: DI Ilpo Kainulainen

Tiivistelmä

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUTin insinööritieteiden tiedekunta

Tietotekniikan koulutusohjelma

Arttu Korpela

Ohjelmoinnin Perusteet -kurssin virhedatan analyysi ja visualisaatio
Versio 1.01/17.4.2025

Kandidaatintyö

2025

33 sivua, 9 kuvaa, 3 taulukkoa ja 0 liitettä

Tarkastajat: DI Ilpo Kainulainen

Avainsanat: Ohjelmoinnin perusteet, Abstrakti Syntaksipuu Analysoija, Uusien ohjelmoijien virheet

Tässä tutkimuksessa tarkastellaan ohjelmointivirheiden rakennetta ja kehittymistä LUT-yliopiston Ohjelmoinnin perusteet -kurssilla syksyllä 2024. Virhedata kerättiin Abstraktin syntaksipuun analysoijan (ASPA) kautta viikoilta 40–51, ja aineisto koostuu 5794 palautuksesta ja 22 997 opiskelijan tuottamasta virheestä. Virheet luokiteltiin automaattisesti kuuteen kategoriaan: basic, function, file_handling, data_structures, library ja exception_handling. Aineistoa visualisoitiin Pythonin, Pandasin ja Seabornin avulla Jupyter-ympäristössä.

Analyysi osoitti, että poikkeustenkäsittely- (exception_handling, yht. 9392, 40,8%) ja aliohjelmavirheitä (function, yht. 6871, 29,9%) esiintyi eniten, kun taas basic-virheitä oli vähiten (yth. 934, 4,1%). Viikoittaisten trendien tarkastelu paljasti virhemäärien piikit viikoilla 45 ja 46, jotka ajoittuivat kurssin harjoitustyön palautuksen yhteyteen. Peräkkäisten palautusten seuranta osoitti virhekeskiarvon laskevan 39,8% ensimmäisestä palautuksesta (2,78) kymmenenteen (1,67), jyrkimmin ensimmäisen ja toisen yrityksen välillä (–30,2%). Tulokset osoittavat, että välitön työkalupalaute edistää virheiden järjestelmällistä vähentämistä ja visualisointi voisi tarjota opettajille ajantasaista tietoa kurssin kohdennettuun kehittämiseen.

Abstract

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Science

Degree Programme in Software Engineering

Arttu Korpela

Analysis and visualization of programming mistakes made during CS1

Bachelor's thesis

2025

33 pages, 9 figures, 3 tables and 0 appendices

Examiners: DI Ilpo Kainulainen

Keywords: CS1, Abstract Synxtax Tree Analyzer, Novice programming mistakes

This study investigates the structure and evolution of programming errors made by students in the "Programming Basics" (CS1) course at LUT University during autumn 2024. Error data were collected via the Abstract Syntax Tree Analyzer (ASPA) from weeks 40 to 51, comprising 5794 submissions and 22 997 student-generated errors. Errors were automatically classified into six categories: basic, function, file_handling, data_structures, library, and exception_handling—and visualized using Python, Pandas, and Seaborn within a Jupyter environment.

The analysis revealed that exception_handling (9392 occurrences, 40.8%) and function-related errors (6871 occurrences, 29.9%) dominated the dataset, while basic errors were least frequent (934 occurrences, 4.1%). Weekly trends exhibited showed peaks in error counts during weeks 45 and 46, corresponding to the course's major project deadlines. Tracking consecutive submissions showed a 39.8% decrease in average errors—from 2.78 in the first submission to 1.67 by the tenth—with the steepest decline (30.2%) occurring between the first and second attempts. These findings demonstrate that immediate, tool-provided feedback fosters systematic error reduction and visualization can offer educators timely insights for targeted course improvements.

Sisällysluettelo

Tiivistelmä

Abstract

Sisällysluettelo 4

Kuva- ja taulukkoluettelot (valinnainen) 6

1 Johdanto 7

1.1 Tausta 7

1.2 Tavoitteet ja rajaukset 7

1.3 Tutkimuskysymykset 8

2 Kirjallisuuskatsaus 9

2.1 Ohjelmoinnin opettaminen yliopistossa ja virheiden mittaaminen 9

2.2 Muita ohjelmien rakennetta arvioivia työkaluja 9

2.3 Työkaluihin liittyvä tutkimus 10

2.4 Tutkimukset käyttöön sekä käytettävyyteen 10

2.5 Kirjallisuuskatsauksen yhteenveto 11

3 Menetelmät ja työkalut 12

3.1 Työkalut 12

3.1.1 IPython ja Jupyter 12

3.1.2 Pandas ja Seaborn 13

3.2 Menetelmät 13

3.2.1 Aineiston rakenne 13

3.2.2 Aineiston luokittelu ja muokkaus 14

3.2.3 Aineiston tarkastelu ja havainnollistaminen 16

4 Aineiston analyysi 17

4.1 Yleisimmät virhetyypit ja kategoriat 17

4.1.1 Yleisimmät virhe ID:t 17

4.2 Viikottainen virhemäärien tarkastelu 19

4.2.1 Viikottaiset virhemäärät kategorioittain 20

4.2.2 Yleiskuva palautuksien virhemääristä yleisesti sekä kategorioittain . 21
4.3 Virhemäärien muutokset peräkkäisissä palautuksissa 22
4.3.1 Kokonaisvirhemäärien muutokset peräkkäisten palautuksien aikana 22
4.3.2 Kategoriakohtaiset virhemäärien muutokset peräkkäisten palautuksien aikana 23
5 Keskustelu 24
5.1 Opiskelijoiden yleisimmät virheet 24
5.1.1 Poikkeustenkäsittelyyn liittyvät virheet 24
5.1.2 Aliohjelmiin liittyvät virheet 25
5.1.3 Muut virhekategoriat 25
5.2 Viikottaiset virhemäärät 26
5.2.1 Kokonaisvirhemäärä viikottain suhteessa palautuksien määrään . . . 26
5.2.2 Viikkottaiset virhemäärät kategorioittain 26
5.3 Muut palautuksien trendit 27
5.3.1 Peräkkäisten palautusten vaikutus keskimääräisiin virhemääriin . . 27
5.3.2 Kategoriakohtaiset virhemäärien muutokset peräkkäisissä palautuksissa 27
5.4 Virheet harjoitustyössä 27
5.5 Tutkimuskysymykset 28
5.6 Epätarkkuudet sekä jatkotutkimus 29
5.7 Yhteenveto 30
Lähdeluettelo 31

Kuvaluettelo

1 JSON-tiedoston tietueiden rakenne
2 Virheluokat sekä niiden sisältämät virhe ID:t
3 Yleisimmät virhe ID:t
4 Yleisimmät virhekategoriat
5 Viikottaiset virhe- ja palautusmäärät
6 Viikottaiset virhemäärät kategorioittain
7 Hajontakuvaaja virhemääristä ja yleisimmästä virhekategoriasta
8 Keskimääräinen virheiden määrä peräkkäisissä palautuksissa
9 Virheiden määrä kategorioittain kymmenessä ensimmäisessä peräkkäisessä palautuksessa

Taulukkoluettelo

1 Testien käyttöönotto viikoittain (alkaen viikko 40)
2 Yleisimmät kahdeksan virhettä ja niiden kuvaukset (suurimmasta pienimpään)
3 Virhekategoriat ja niiden esiintymismäärät (suurimmasta pienimpään)

1 Johdanto

1.1 Tausta

Abstrakti syntaksipuu analysaattori (ASPA) on LUT-yliopiston Ohjelmoinnin perusteet -kurssilla käytettävä palauteohjelma. Sen pääsääntöinen tarkoitus on auttaa opiskelijaa löytämään viikkotehtävistään tai lopputyöstään erilaisia virheitä sekä tyyliohjeen vastaisia rakenteita. ASPA on staattinen analyysityökalu, joka muodostaa opiskelijan Python-kooditiedostosta abstraktin syntaksipuun. Tällöin on mahdollista tutkia tarkemmin ohjelmiston rakennetta, kuten luokkia, aliohjelmia ja muuttujia. ASPA ei tarkastele ohjelman suoritteiden syötteitä tai palautteita, vain itse koodin rakennetta (Luukkainen et al., 2022).

Syksyn 2024 kurssilla työkalu on siirretty ladattavasta Python-ohjelmasta verkkopohjaiseksi, joka mahdollistaa analyysidatan keräämisen. Aikaisemmat tutkimukset kurssin palautusten virheistä ovat liittyneet automaattitestauksesta saatavaan dataan. Rajoitteena tässä metodissa on kuitenkin, että tarkastelu kohdistuu vain ohjelmien suorituksen aikana syntyviin virheisiin.

Tutkimuksessa hyödynnetään LUT-yliopiston syksyn 2024 Ohjelmoinnin perusteet -kurssin aikana ASPA:n käytöstä syntyvää dataa. Aineisto koostuu viikkojen 40 ja 51 välillä suoritettujen testien palautteista, jotka sisälsivät yhteensä 24196 tunnistettua virhettä 6193 palautuksessa. Tutkimusdata saadaan yksittäisenä Javacript Object Notation (JSON) -tiedostona, jota analysoimme Jupyter-alustalla hyödyntäen Pandas-kirjastoa. Visualisoinnissa hyödynnetään Seaborn-kirjastoa.

1.2 Tavoitteet ja rajaukset

Tutkimuksessa analysoidaan laajaa aineistoa opiskelijoiden tekemistä virheistä ensimmäisen ohjelmointikurssin viikkotehtävistä sekä harjoitustöistä. Tutkimuksen tavoite on uusien metriikoiden tunnistaminen ja visualisointi aineistosta. Visualisoinnissa painopisteet ovat luettavuus sekä automatisaatio. Toistettavuudella pyritään luomaan työkaluja assistenttien ja opettajien käyttöön. Työkalujen avulla opetuksen aikana voidaan huomata erilaisia trendejä palautuksissa ja reagoida niihin opetuksessa jo kurssin aikana.

Tutkimuksen tavoitteena ei ole selvittää opetuksen ja virheiden välistä syy-seuraussuhdetta. Se ei myöskään käsittele opetusmenetelmien tarkastelua tai niiden tehokkuuden analysointia. Lisäksi tutkimus ei tarkastele koodin virhemäärän ja opiskelijan loppuarvosanan välistä korrelaatiota tai tunnista ennakoivia tekijöitä, jotka voivat viitata kurssin keskeyttämiseen. Vaikka aineisto sisältää jokaisen palautuksen lähdekoodin, tulee tarkastelu keskitymään vain

ASPA:n merkkaamiin virheisiin.

1.3 Tutkimuskysymykset

Tutkimuskysymykset ovat seuraavat:

1. Millaisia virheitä opiskelijat tekevät Ohjelmoinnin perusteet -kurssin aikana?
2. Voiko ASPA:n virhedatan visualisaation avulla löytää poikkeamia ja trendejä?
3. Miten voimme hyödyntää ASPA:n virhedataa kurssin kehittämiseksi?

Tutkimus etenee virheiden sekä virhetyyppien kartoituksella aineistosta hyödyntämällä Jupyter-alustaa ja Python-ohjelmointikieltä sekä sen Pandas-kirjastoa. Kartoituksen jälkeen aloitetaan uusien metriikoiden muodostaminen vertailemalla palautusten määriä, virhetyyppejä ja muita aineistomme datapisteitä. Metriikoiden määrittämisen jälkeen siirrytään muodostamaan automatisoituja visualisointeja, joista tutkimme mahdollisia trendejä tai poikkeamia. Näitä tuloksia käyttämällä tutkimme lopuksi, voidaanko niitä hyödyntää opetuksen kannalta.

2 Kirjallisuuskatsaus

Tässä luvussa tutustutaan aikaisempiin tutkimuksiin, joissa on tutkittu opiskelijoiden tekemiä virheitä heidän opiskellessaan ohjelmointia. Kappale pyrkii antamaan tiivistelmän aikaisempien tutkimuksien tuloksista, historiasta sekä näkökulmista.

2.1 Ohjelmoinnin opettaminen yliopistossa ja virheiden mittaaminen

Yliopisto-opetuksessa opiskelijoiden ohjelmointitehtäväpalautuksien automatisaatio on alkanut jo 1960-luvulla Rensselarin polyteknisessä yiopistossa. Opettajat osoittivat jo tällöin kiinnostusta siihen miten automaatiotestaus vaikuttaa opetuksen laatuun. (Hollingsworth, 1960) Automaattitestauksesta on tullut yleisesti käytetty toimintatapa ohjelmoinnin opetuksessa ja sen tuoma mahdollisuus kerätä jatkuvasti dataa on luonut paljon kiinnostusta ohjelmoinnin opiskelun tutkimiseen. Kuitenkin datan kerääminen testausohjelmistoista on ollut haastavaa ennen internetin yleistymistä. Käänne tapahtui 2000-luvun puolessa välissä, jonka jälkeen tutkimukset testausohjelmista saatuun dataan ovat lähteneet suureen kasvuun. (Ihantola et al., 2015)

Lappeenrannassa otettiin myös vuonna 2001 automaattinen testaus käyttöön *Ohjelmoinnin perusteet* -kurssilla. Kuitenkin nykyinen kurssilla käytetty automaattitestaus ei huomio tai pisteytä itse ohjelman rakennetta. Testausohjelma vain tarkastelee ohjelman palautteita tietyillä syötteilllä, toisin sanoen se suorittavaa vain yksikkötestejä. Ongelmaksi muodostui, etteivät kurssin lopputyöt vastanneet rakenteeltaan ja tyyliltään kurssin tyyliohjetta. Ongelman ratkaisuksi muodostettiin ASPA, ohjelma, jonka avulla opiskelija pystyy tarkistamaan ohjelmastaan tyylivirheitä. (Luukkainen et al., 2022)

2.2 Muita ohjelmien rakennetta arvioivia työkaluja

ASPA ei ole ensimmäinen ohjelman rakennetta tutkiva testausohjelmisto opetuskäytössä. WebTA on vuonna 2015 Michiganin teknisessä yliopistossa käyttöön otettu ensimmäisen ohjelmointikurssin arviointityökalu. Perinteisen yksikkötestaustyyppisen automaattitestauksen lisäksi ohjelmaan on integroitu koodin rakennetta tutkiva ominaisuus. WebTA on suunniteltu käytettäväksi Java-ohjelmointikielen kanssa, joten se pääsääntöisesti keskittyy tarkastelemaan, että opiskelija noudattaa olio-ohjelmoinnin perusteita. (Ureel II & Wallace, 2019) Myös enemmän tyylivirheisiin keskittyviä työkaluja on julkaistu, esimerkiksi Hyperstyle on viime vuosina kehitetty ja otettu käyttöön suurilla MOOC-kursseilla. Hyperstyle on Stepik ja JetBrains Academy -alustoilla käytettävä työkalu, joka arvioi opiskelijoiden ohjelmointitehtäväpalautuksia. Se hyödyntää jo markkinoilla saatavilla olevia koodianalyysityökaluja, joita se hyödyntää antamaan kokonaisvaltaisemman ja ensikertalaiselle helposti ymmärret-

tävämmän palautteen tyylivirheistä. (Birillo et al., 2022)

George Fox:in yliopistossa on vuodesta 2021 otettu käyttöön kahdella ensimmäisellä ohjelmoinnin kurssilla työkalu, joka arvioi rakenteellisia virheitä. Tämäkin toteutus hyödyntää abstraktia syntaksipuuta. Sen kykenee arvioimaan Python- sekä Java -kielillä kirjoitettuja ohjelmia. (Walker, 2022)

2.3 Työkaluihin liittyvä tutkimus

Tutkimukset näiden työkalujen ympärillä ovat keskittyneet lähinnä oppimistuloksien muutokseen käyttöönoton jälkeen sekä opiskelijoiden mielipiteisiin niiden käytöstä. Hyperstyle:n käyttö nosti tyylivirheettömien tehtäväpalautuksien määrää nelinkertaisesti Python:illa tehdyillä palautuksilla. Java:lla kirjoitetuissa palautuksissa merkittävin muutos löytyi töistä, jotka sisälsivät yli kahdeksan virhettä. Tällöin kaikkien palautuksien kokonaisvirhemäärä laski 56:sta yhteen.(Birillo et al., 2022)

Myös Gearge Fox:in yliopiston työkalun vaikutuksia ohjelmointitehtävien virheiden määrään tarkasteltiin. Molemmissa, sekä Python että Java -tehtävissä nähtiin tilastollisesti merkittävää laskua virheiden määrissä. (Walker, 2022) WebTA:n kehittäjät olivat kiinnostuneita ohjelmoinnin rakennevirheistä ja he lisäsivät niitä jatkuvasti WebTA:n tietokantaan tutkimalla opiskelijoiden palautuksia. Jatkotutkimusta näihin tunnistettuihin rakennevirheisiin ei valitettavasti ole saatavilla. (Ureel II & Wallace, 2019)

Yksikään tarkastelluista rakenne- ja tyylivirheitä tunnistava automaattitestausohjelmista ei ole saanut ympärilleen tutkimusta, jossa olisi tarkasteltu tarkemmin kurssin opiskelijoiden virheitä. Tutkimukset keskittyivät lähinnä ohjelmien vaikutuksiin kokonaisvirhe määriin verrattuna aikaan ennen työkalujen käyttöönottoa. Tässä tutkimuksessa, halutaan selvittää, millaisia virheitä opiskelijat tekevät eri vaiheissa kurssia sekä saada laajemman ymmärryksen virhetyyppien yleisyydestä *Ohjelmoinnin perusteet* -kurssilla.

2.4 Tutkimukset käyttöön sekä käytettävyyteen

Tutkimusta työkalujen käytöstä sekä niihin liittyvistä mieltymyksistä on ASPA:n sekä Toronton yliopiston staattisen automaattitestaustyökalu PythonTA:n kanssa tehty. Luukkainen et al., 2022 havaitsivat, että suurin osa *Ohjelmoinin perusteet* -kurssin opiskelijoista käytti ASPA:a, kun 236:sta kyselyyn vastanneesta 205 vastasi käyttäneensä kurssin aikana työkalua. Myös sen käytön koettiin olevan helppoa, koska vähintään 106 206:sta koki käytön kaikissa kategorioissa erittäin helpoksi. (Luukkainen et al., 2022)

Myös PythonTA:n käyttö oli suosittua opiskelijoiden keskuudessa, kun 45.5% vastasi toi-

sessa kyselyssä käyttäneensä työkalua. Liu, Calver & Craig, 2024 havaitsi myös, että vastanneista 53.9% korjasivat kaikki PythonTA:n löytämät virheet ennen palautusta. Kaikki virheet korjanneiden opiskelijoiden määrä laski 9.8 prosenttiyksikköä ensimmäiseen kyselyyn verrattuna. Tutkijat veikkasivat tämän poikkeaman johtuvan kurssin työmäärään sekä vaikeustason noususta kurssin edetessä, jolloin opiskelijoilta yksinkertaisesti loppui aika tai he pelkäsivät korjauksien rikkovan ohjelmansa. (Liu, Calver & Craig, 2024)

2.5 Kirjallisuuskatsauksen yhteenveto

Rakennetta, sekä tyylivirheitä arvioivia työkaluja on tutkittu laajasti. Kuitenkin tutkimus on lähinnä keskittynyt näiden työkalujen käyttökokemuksen sekä tehokkuuden ympärille. Tämän tutkimuksen näkökulma hyödyntää ASPA:n automaattisesti luomaa virhedataa on uniikki ja tuore näkökulma.

3 Menetelmät ja työkalut

Tässä työssä käytämme ASPA:sta saatavaa tutkimusaineistoa havainnollistavien tilastollisten ja graafisten kuvausten luomiseen osana eksploratiivista data-analyysia (EDA). Exploraatiivinen data-analyysi eroaa perinteisestä tilastollisesta analyysistä sekä menetelmällisesti että ajattelutavaltaan. Sen keskiössä ovat skeptisyys ja avoimuus analyysin tulosten suhteen, kuten Hartwig & E.Dearling (1979) esittävät. Behrens (1997) korostaa, että EDA pyrkii ymmärtämään aineistoa syvällisemmin ja välttämään ylisovittamisen riskejä.

EDA:ssa skeptisyys tarkoittaa varovaisuutta yksinkertaistavia tilastollisia yhteenvedoja kohtaan, jotka voivat piilottaa merkittäviä poikkeamia. Graafisten esitysten, kuten laatikko- ja viiksidiagrammien, tiheysarvioiden ja jäännösanalyysien, avulla voidaan tuoda esiin piirteitä, joita perinteiset yhteenvedot eivät tavoita (Behrens, 1997). Tämän lisäksi avoimuus viittaa valmiuteen tutkia aineistosta nousevia odottamattomia kuvioita ilman ennalta määrättyjä hypoteeseja, mikä on erityisen tärkeää ASPA:n kaltaisten laajojen aineistojen analysoinnissa (Behrens, 1997). ASPA:n tarjoama monipuolinen data mahdollistaa trendien ja poikkeavuuksien havainnoinnin, tarjoten uutta tietoa oppimisprosessien ja haasteiden ymmärtämiseksi.

3.1 Työkalut

3.1.1 IPython ja Jupyter

Työssä hyödynnetään ensisijaisesti Python:ia. Se on erittäin yleisesti käytetty, soveltuva, tulkattu ohjelmointikieli, joka on viime vuosina laajan sekä vakinoituneen kirjastojensa ansiosta asettunut data-analyytikan työkaluissa vahvaan asemaan (Rajagopalan, 2021). Vaikka Pythonin komentotulkki soveltuu nopeaan ideoiden tutkimiseen, se on rajallinen pidempiaikaisessa interaktiivisessa työskentelyssä. IPython tarjoaa tähän ratkaisun yhdistämällä laajennetut ominaisuudet, kuten dynaamisen objektien tarkastelun, tehokkaan komento- ja kahden prosessin arkkitehtuurin, jossa Kernelin ja käyttäjät toimivat erillään. Tämä mahdollistaa usean käyttäjän samanaikaisen yhteyden kerneliin. (Perez & Granger, 2007)

IPython:in ominaisuuksien hyödyntämiseksi on sen rinnalla kehitetty Jupyter:ia. Project Jupyter on IPython Notebook:sta muodostunut avoimen lähdekoodin projekti, joka mahdollistaa IPython:in tuomien edistysaskelten käytön.(Perkel, 2021) Jupyter-alustan suosio on kasvanut merkittävästi viime vuosina, koska se tarjoaa mahdollisuuden yhdistää koodin suoritus, dokumentaatio ja visualisointi yhteen interaktiiviseen ympäristöön. Esimerkiksi Jupyternotebookeja käytetään laajasti eri tieteenaloilla datan analysointiin ja mallintamiseen. (Perkel, 2021) Jupyterin käyttö sulavoittaa ASPA:n virhedatan käsittelyä sekä luo hyvän perustan koodin jakamiselle sekä mahdollistaa samanaikaisen käytön useille käyttäjille.

3.1.2 Pandas ja Seaborn

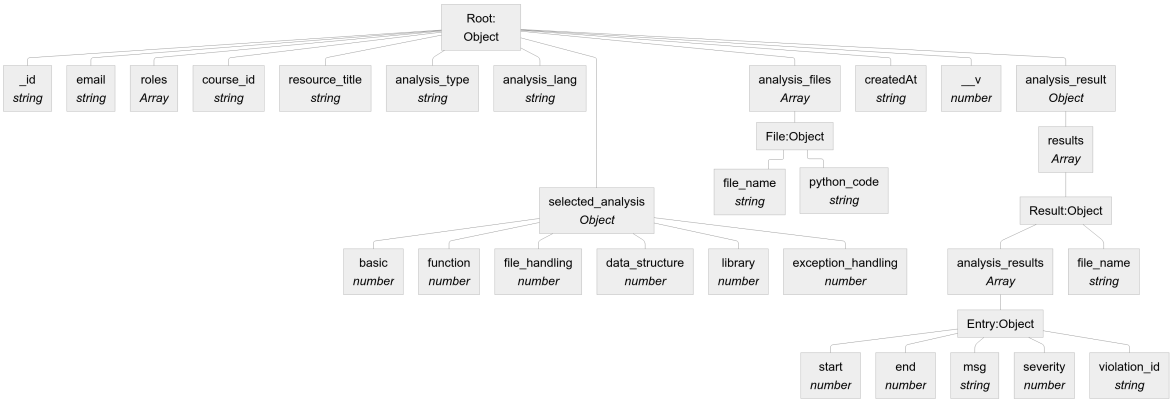
Python itsessään tarjoaa datastruktuureja kuten listoja (Array) sekä sanakirjoja (Dictionaries), jotka mahdollistavat yksinkertaisen data-analyysin suorittamisen. Kuitenkin kaikki nämä Python:in rakenteet ovat yksiuloitteisia eikä niille voi suorittaa suoraan martriisilaskutoimituksia.(Rajagopalan, 2021) Ratkaisu ongelmaan esitetäänkin jo *Ohjelmoinnin perusteet* -kurssilla. *NumPy* -kirjaston avulla pystymme luomaan useamman ulottuvuuden matriiseja sekä käsittelemään niitä tehokkaasti kirjoittamatta silmukoita (McKinney, 2017). Tässä työssä hyödynnämme epäsuorasti *NumPy*:n ominaisuuksia sekä sen luomaa tyylisuuntaa käyttämällä *Pandas*-kirjastoa. *NumPy*:n tavoin *Pandas* lisää Python:sta puuttuvia monidimensionaalisia datastruktuureja sekä funktioita niiden tehokkaaseen silmukattomaan käsittelyyn. Kuitenkin *Pandas* on suunniteltu heterogeenisen datan käsittelyyn toisin kuin *NumPy*, joka soveltuu paremmin puhtaaseen numeriseen analyysiin. (McKinney, 2017) ASPA:n virhedatan erilaiset datapisteet sisältävät merkkijonoja, kokonais- ja liukulukuja sekä aikaleimoja, näin ollen *Pandas* soveltuu paremmin tämän tutkimuksen tarpeisiin.

Eksploratiivisen tutkimuksen kannalta tärkeässä datan visualisaatiossa hyödynnetään *Seaborn*-kirjastoa, joka on suunniteltu erityisesti tilastollisen datan visualisointiin. *Seaborn* tarjoaa korkean tason rajapinnan Matplotlib-kirjastoon ja integroituu tiiviisti *Pandas*:in datastruktuureihin, tehden siitä luonnollisen valinnan heterogeenisen datan käsittelyyn ja visualisointiin. Kirjaston deklaratiivinen, datasetti-orientoitunut käyttöliittymä mahdollistaa monimutkaisten graafien luomisen yksinkertaisilla funktiokutsuilla. Esimerkiksi muuttujien automaattinen kartoitus graafisiin ominaisuuksiin, kuten väreihin, kokoihin tai tyyleihin vähentää merkittävästi manuaalista säätöä ja nopeuttaa analyysiprosessia.(Waskom, 2021) Tässä tutkimuksessa *Seaborn*:n tarjoamat monipuoliset graafiset työkalut, kuten histo-, hajonta- ja viivadiagrammit tukevat ASPA:n virhedatan visualisointia. Visualisoinnilla pyritään havainnollistamaan kurssin aikana tehtyjen virheiden jakaumat, kehityssuunnat ja mahdolliset poikkeamat.

3.2 Menetelmät

3.2.1 Aineiston rakenne

ASPA kerää kaikista ajetuista testeistä datan keskitetysti MongoDB tietokantaan (Nykänen, 2024). Tämä data saadaan JSON-tiedostona, jota tämä tutkimus käyttää aineistonaan. JSON-tiedosto sisältää listan tietueita, joista jokainen kuvaa suoritettua testiä. Näiden tietueiden sisältö voidaan nähdä kuvaajassa 1. Kaikista ylimmällä tasolla löytyvät testin uniikki id, käyttäjän sähköpostiosoite, jolla hän on kirjautunut moodleen, kurssin id, resurssin otsikko, analyysin tyyppi, käyttäjän valitsema kieli sekä aikaleima. Alemmalla tasolla löytyvät käyttäjän rooli (Moodle LTI integraatiota varten), valitut analyysit, analyysin tiedostot sekä analyysien



Kuva 1: JSON-tiedoston tietueiden rakenne

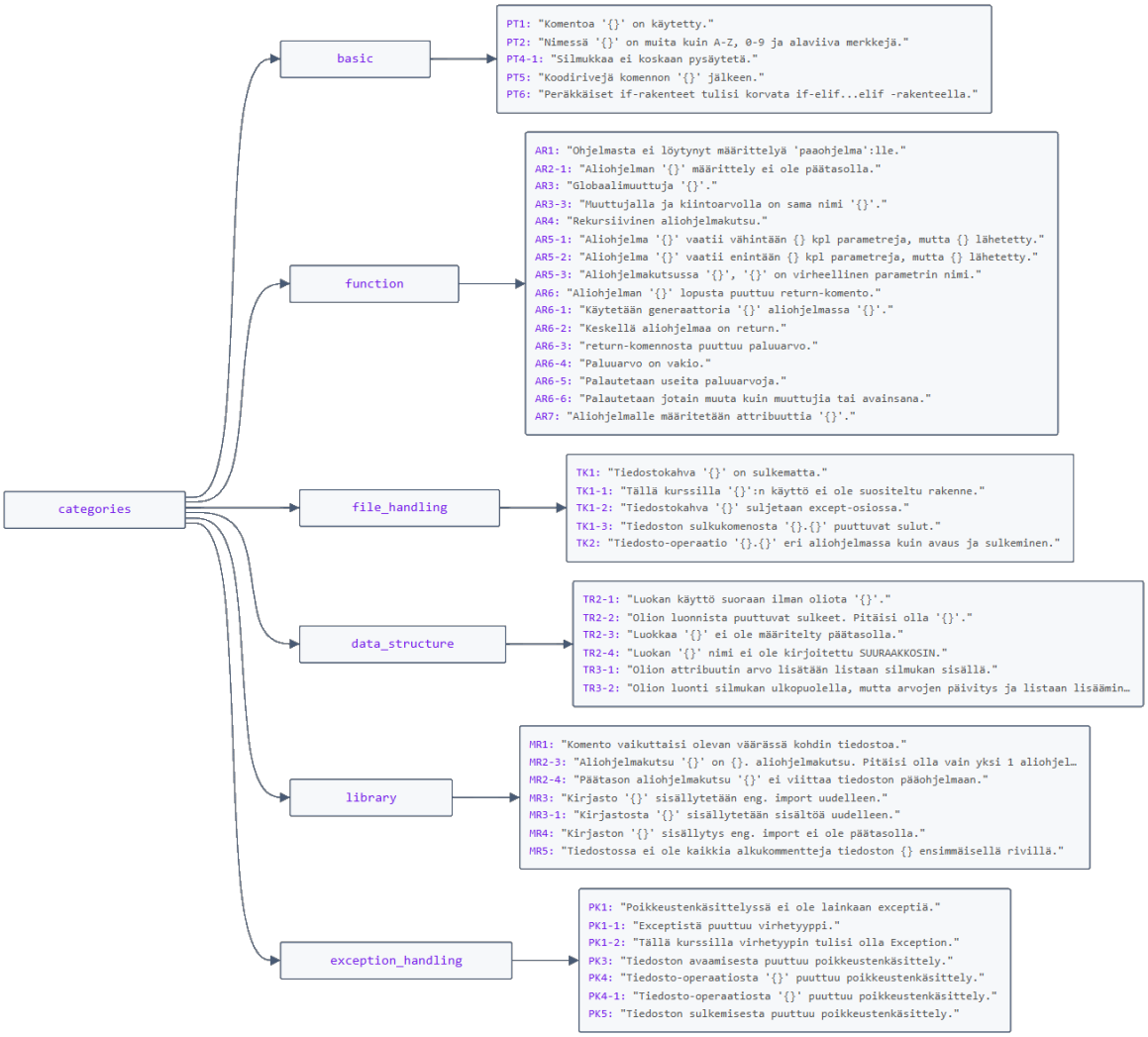
tulokset. Valitut analyysit -objekti sisältää kuusi avain-arvo -paria, jotka ovat virhekategoriat, joita testissä on tarkasteltu. Kuvaajasta 2 näemme nämä virheluokat sekä tarkemmat virhetyypit, jotka niihin sisältyvät. `analysis_files` takaa löytyy kokoelma tiedostoja, joita on tarkasteltu. Alkiot sisältävät tiedostojen nimen sekä koko tiedostosisällön. Tämän työn kannalta tärkein kohta on `analysis_result` -avain, jonka alta löytyy kokoelma tiedostoja sekä niistä löydetyt virheet.

3.2.2 Aineiston luokittelu ja muokkaus

ASPA luokittelee virheet kuuteen eri kategoriaan, joista käyttäjä pystyy valitsemaan analyysissä tutkittavat. Virheluokkia ovat `basic`, `function`, `file_handling`, `data_structures`, `library` ja `exception_handling`.

`Basic`-luokan virheet koostuvat lähinnä tyyliohjeen nimeämisohjeiden ja ehtorakenteiden sekä loputtomien silmukoiden rikkeistä. `Function`-luokka koostuu aliohjelmien ympärillä tapahtuvista virheistä, kuten niiden parametrimääristä sekä `return`-komennon käytöstä. `File_handling`-luokka sisältää tiedoston käsittelyyn liittyvät virheet, kuten tiedostokahvojen käsittelyn eri aliohjelmissa ja sulkemisen unohtamisen. Kurssin tyyliohje määrittelee tarkat ehdot tiedostojen avaamiseen, lukemiseen, kirjoittamiseen sekä sulkemiseen. `Data_structures`-luokan virheet muodostuvat luokkien nimeämisen, luomisen ja käyttämisen ympärille. `Library`-luokan virheet koostuvat kirjastojen tuonnin ja käytön tyyli- sekä syntaksivirheistä. `Exception_handling`-luokan virheet ympäröivät kaikkea poikkeustenkäsittelyn ympärillä. Erityisesti tiedosto-operaatioiden ympärille vaaditaan poikkeustenkäsittely.

JSON-tiedoston käsittely ja muokkaus analyysia varten toteutettiin Python-skriptien avulla, hyödyntäen kirjasto-tiedostossa määriteltyjä funktioita. Näiden funktioiden avulla data muokattiin helpommin analysoitavaan muotoon. Ensimmäisenä vaiheena oli tietueiden jäsentä-



Kuva 2: Virheluokat sekä niiden sisältämät virhe ID:t

minen ja kategorisointi. Tähän käytettiin `categorize_error_type` -funktiota, joka määrittelee virhetyypin niiden `violation_id` -avaimen perusteella. Virheet luokitellaan kuuteen kategoriaan: perustason virheet (`basic`) , funktiovirheet (`function`) , kirjastoihin liittyvät virheet (`library`) , poikkeuskäsittelyn virheet (`exception_handling`) , tiedostojen käsittelyn virheet (`file_handling`) ja tietorakenteisiin liittyvät virheet (`data_structures`) .

Seuraavaksi aineistosta muodostettiin tiivistettyjä tietueita `errors_per_return`-funktion avulla. Tämä funktio tuottaa jokaiselle käyttäjälle ja analyysille yhteenvedon analysoiduista tiedostoista sekä kategorisoitujen virheiden lukumääristä. Lisäksi funktio yhdistää tiedostonimet ja muuttaa aikaleiman (datetime) -muotoon, mikä helpottaa aikapohjaisen analyysin suorittamista Pandas:ssa. Toisessa vaiheessa, `extract_errors`-funktiolla, data pilkottiin hienojakoisemmin yksittäisiin virheisiin. Tällä tavalla jokainen virhe taltioitiin omaksi tietueekseen sisältäen tiedoston nimen, käyttäjän sähköpostin, virhe-ID:n, vakavuustason, virheilmoituksen sekä kategorian. Tämä lähestymistapa mahdollistaa tarkemman analyysin ja

tilastollisten jakaumien tarkastelun.

Näiden funktioiden avulla JSON-tiedoston sisältämä monimutkainen data muunnettiin selkeään ja järjestelmälliseen muotoon, mikä helpottaa sen käyttöä Pandas:in sekä visuaalisten että tilastollisten analyysien suorittamiseen.

3.2.3 Aineiston tarkastelu ja havainnollistaminen

Tulevissa analyyseissä tarkastellaan aineiston virheiden jakautumista ja kehitystä eri näkökulmista. Ensimmäisenä analysoidaan virheiden esiintymismäärät sekä `violation_id` -kohtaisesti että kategorioittain. Tämän jälkeen hyödynnämme pylväsdiagrammeja esiintymismäärien havainnollistamisessa. Lisäksi tutkitaan virheiden ajallista vaihtelua eri tunneittain ja viikoittain viivakaavioiden avulla.

Toisessa vaiheessa tarkastellaan lähetysjärjestyksen vaikutusta virhemääriin. Lähetykset järjestetään käyttäjän, tiedoston ja ajan perusteella, minkä jälkeen lasketaan keskimääräiset virheet per lähetysjärjestys niin kokonaisuutena kuin kategorioittain. Tämä mahdollistaa oppimiseen liittyvien trendien tunnistamisen.

Lopuksi analysoidaan yleisimmät virhekategoriat kussakin palautuksessa ja niiden määrät. Visualisointi scatterplot-kaaviona tarjoaa kokonaiskuvan virheiden jakautumisesta ajan ja yleisimpien virhetyyppien mukaan, mikä auttaa tunnistamaan erityisiä ongelma-alueita ja käyttäytymismalleja. Näiden analyysien avulla saadaan kattava ymmärrys virheiden luonteesta ja niiden kehityksestä.

4 Aineiston analyysi

Lopullinen aineisto kattaa syksyn 2024 kurssin viikot 4-16 (39-51). Työssä käytetään tästä edes ISO-standardin viikkonumerointia. Viikolla 40 opiskelijat saivat pääsyn ASPA:an, mutta eri kategorioiden testit alkoivat asteittain tulevina viikkoina. Aikatalun sekä mahdolliset poikkeukset löytyvät taulukosta 1. Aineisto koostuu yhteensä 5794 suoritetusta palutuksesta, jotka sisälsivät 23 768 virhettä. Tämä luku kuitenkin sisältää ASPA:n sisäiset `tool_error` -virheet, jolloin tunnistettujen virheiden kokonaismäärä on 22 997.

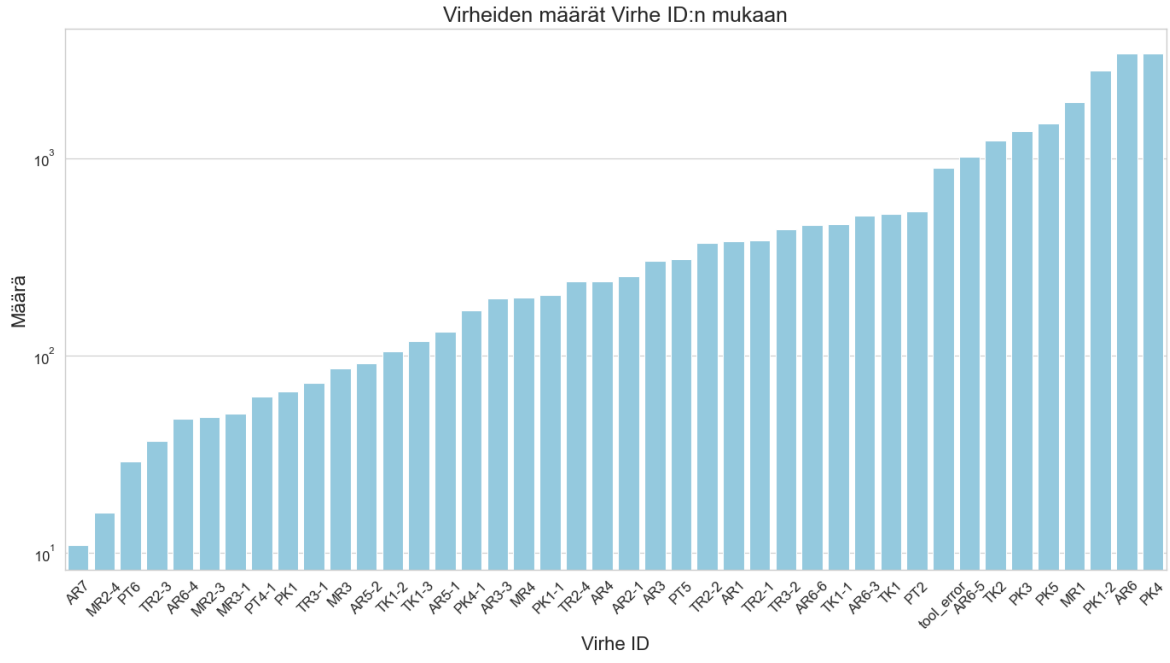
Viikko	Käytössä olevat testit	Huomiot
40	Aliohjelmat, perusominaisuudet	Alkaen viikko 40
41	Aliohjelmat, perusominaisuudet, tiedostonkäsittely	
42	Aliohjelmat, perusominaisuudet, tiedostonkäsittely, tietorakenteet	Palautusaikaa 2 viikkoa
43	Kaikki edelliset + tiedostorakenne (kirjastot)	
44-49	Kaikki analyysit (poikkeustenkäsittely lisätty)	Viikko 46: Ei omaa ASPA-linkkiä

Taulukko 1: Testien käyttöönotto viikoittain (alkaen viikko 40)

4.1 Yleisimmät virhetyypit ja kategoriat

Tämä ensimmäinen kappale keskittyy antamaan yleiskäsityksen yleisimmistä virheistä sekä niiden jakautumisesta eri kategorioihin. Tarkastelussa katsomme koko kurssia yhtenäisenä jaksona.

4.1.1 Yleisimmät virhe ID:t

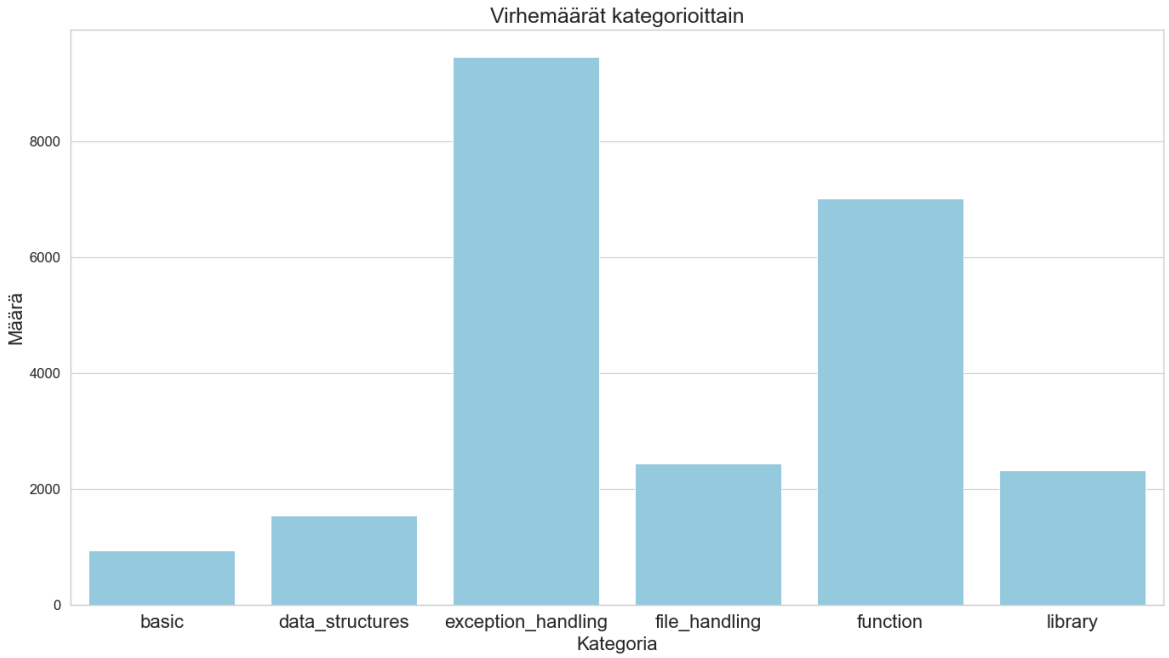


Kuva 3: Yleisimmät virhe ID:t

ASPA:n tunnistamien virheiden määrien varianssi on voimakasta, mikä näkyy selkeästi kuvaajasta 3. Virheiden määrien varianssi on 627,408.82. Keskihajonta, joka kuvaa keskimääräistä poikkeamaa, on 792.09. AR7-virhe ID esiintyi vain 11 kertaa, kun taas PK4 esiintyi peräti 3370 kertaa, mikä osoittaa merkittävän eron eri virheiden yleisyydessä. Kuvaajasta kahdeksan yleisintä virhe-ID:tä erottuvat selkeästi määrillään. Taulukosta 2 nähdään näiden tarkat esiintymismäärät sekä lyhyet kuvaukset.

Virhe ID	**Kuvaus**	**Määrä**
PK4	Tiedostonkäsittely	3370
AR6	`return`-komento puuttuu	3354
PK1-2	Ei käytetä yleistä `Exception`-poikkustyyppiä	2729
MR1	Tiedoston rakenne on tyyliohjeen vastainen	1747
PK5	Tiedoston sulkemisesta puuttuu poikkeustenkäsittely	1471
PK3	Tiedoston avaamisesta puuttuu poikkeustenkäsittely	1358
TK2	Tiedostonkäsittelystä kaikki ei tapahdu samassa aliohjelmassa	1151
AR6-5	Palautetaan useita paluuarvoja	977

Taulukko 2: Yleisimmät kahdeksan virhettä ja niiden kuvaukset (suurimmasta pienimpään)



Kuva 4: Yleisimmät virhekategoriat

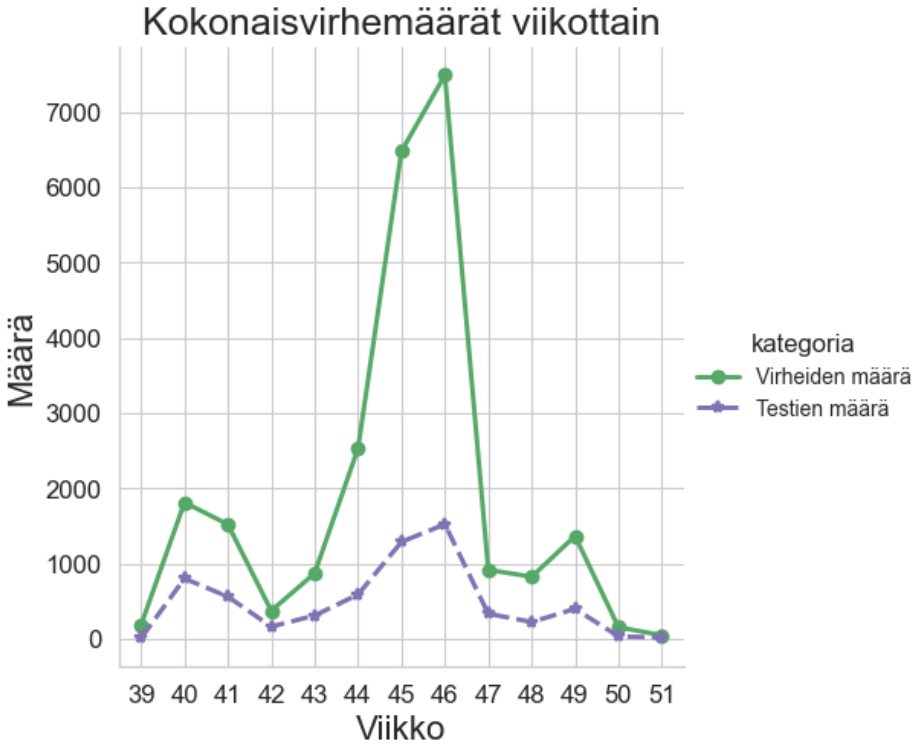
Kun tarkastellaan virheiden kategoriakohtaisia esiintymismääriä, varianssi on 8,703,499.06, mikä osoittaa edelleen merkittävää vaihtelua eri kategorioiden välillä. Keskihajonta on 2950.17, mikä tarkoittaa, että virheluokkien määrät poikkeavat keskiarvosta huomattavasti. `Basic`-luokan virheitä esiintyy kaikista vähiten, 934 kertaa, kun taas `exception_handling`-luokan

virheet dominoivat määrällisesti, esiintyen peräti 9392 kertaa. Kuvaajasta 4 selkeästi erottuvat kaksi yleistintä virheluokkaa ovat `exception_handling`- sekä `function`-luokat. ASPA:n sisäisen poikkeustenkäsittelyn tunnistamat virheet luokitellaan tähän `Tool_error` -kategoriaan. Ne eivät siis ole opiskelijoiden tekemiä virheitä, jolloin ne poisluetaan tässä tutkimuksessa. Luokkien tarkat esiintymismäärät löytyvät taulukosta.3.

Virhekategoria	**Määrä**
Exception_handling	9392
Function	6871
File_handling	2437
Library	2145
Data_structures	1536
Basic	934
Tool_error	771

Taulukko 3: Virhekategoriat ja niiden esiintymismäärät (suurimmasta pienimpään)

4.2 Viikottainen virhemäärien tarkastelu

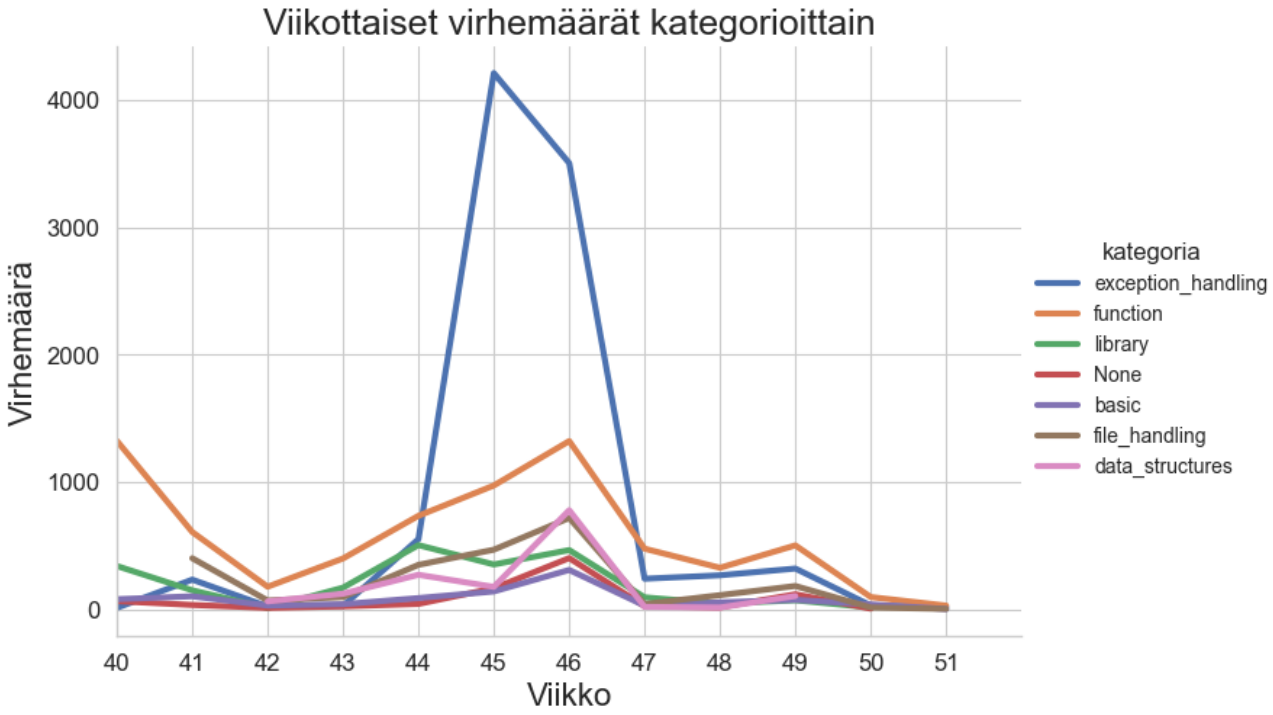


Kuva 5: Viikottaiset virhe- ja palautusmäärät

Kuvaajan 5 tarkastelussa kiinnittyy huomio erityisesti virhemäärien suureen nousuun viikoilla 45 ja 46. Piikkeihin ovat todennäköisesti vaikuttaneet poikkeuskäsittelyn lisäys testauksen

arsenaaliin sekä viikon 46 lopussa oleva harjoitustyön palautuksen sulkeutuminen.

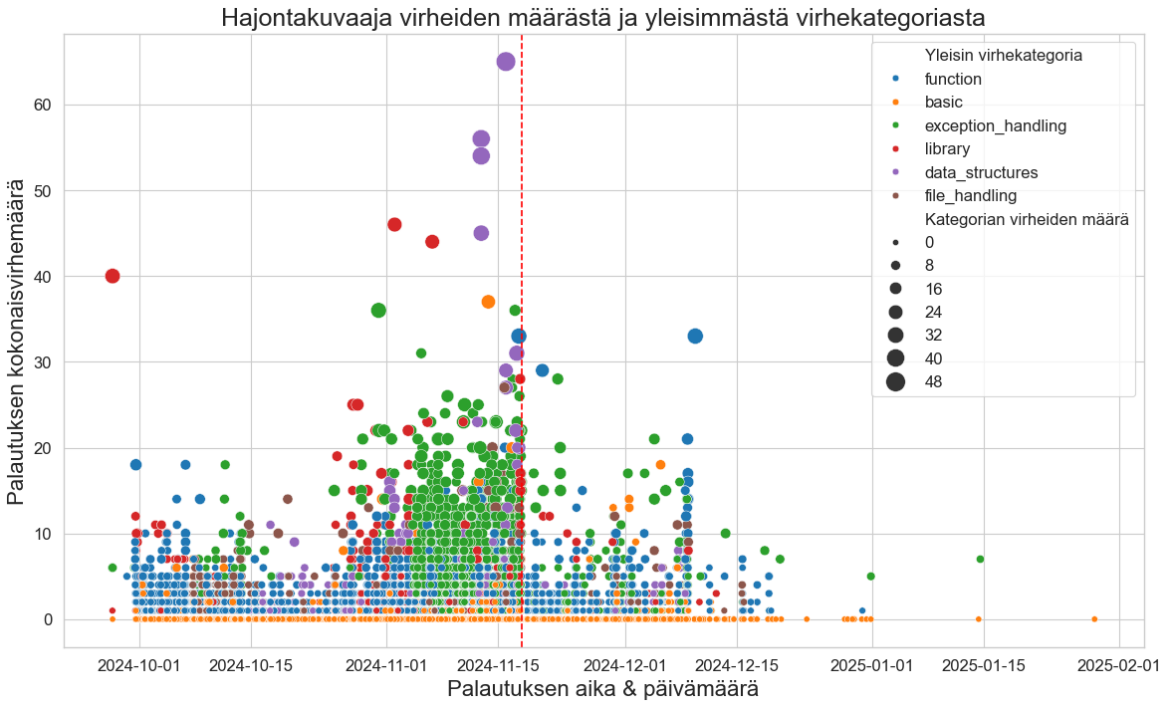
4.2.1 Viikottaiset virhemäärät kategorioittain



Kuva 6: Viikottaiset virhemäärät kategorioittain

Kuvaaja 6 vahvistaa oletusta poikkeusteskäsittelyn lisäämisen vaikuttaneen eniten kokonaisvirhemäärän nousuun juuri viikoilla 45 ja 46. Myös `function`-kategorian virheet nousevat näillä harjoitustyön palautusta edeltävillä viikoilla huippunsa, jääden kuitenkin selkeästi `exception_handling` -virheiden muodostaneen piikin varjoon. Aliohjelmakategorian virheet olivat yleisin virheluokka harjoitusttyötä edeltävien viikkojen ulkopuolella, myös harjoitustyön korjauspalautusta edeltävinä viikkoina. Muut virheluokat jäävät selkeästi aliedustetuiksi verrattuna näihin kahteen edeltämainittuun kategoriaan, mikä pystytään näkemään myös kuvaajasta 4. `Basic` -luokan virheet laskevat merkittävästi ensimmäisillä viikoilla, mutta lähtevät taas tasaiseen nousuun viikon 42 kohdalla ollen huipussa viikolla 46. Muut virheluokat kasvavat määrällisesti aina viikkoon 46 asti. `Library` -luokka on poikkeuksellisesti korkeimmillaan jo viikolla 44.

4.2.2 Yleiskuva palautuksien virhemääristä yleisesti sekä kategorioittain



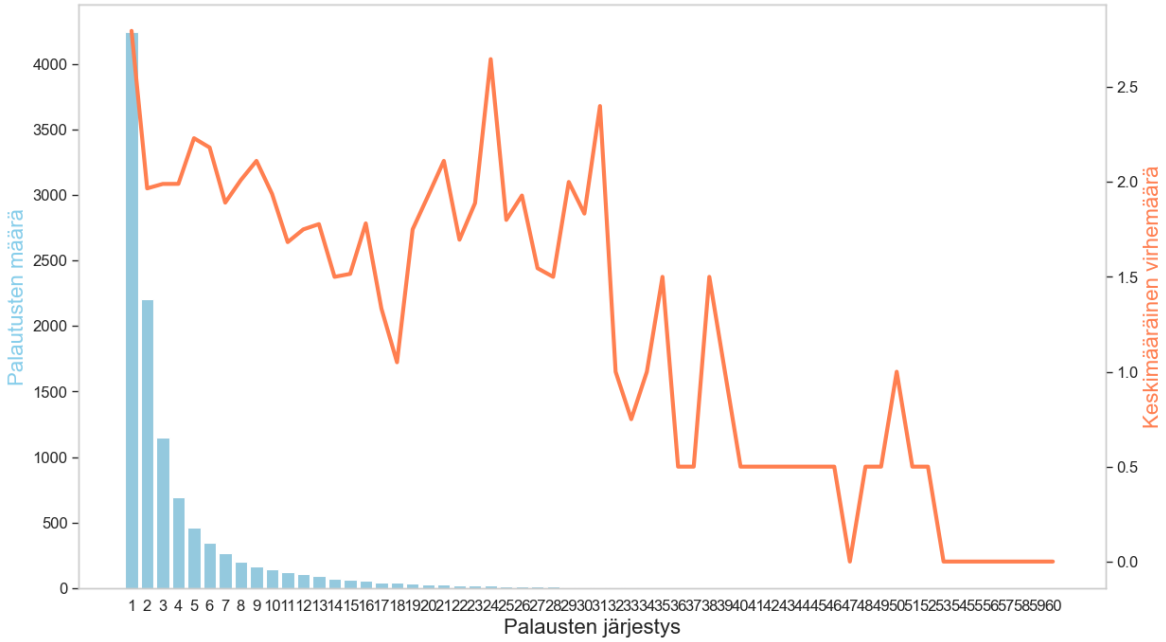
Kuva 7: Hajontakuvaaja virhemääristä ja yleisimmästä virhekategoriasta

Kuvaajassa 7 yksittäinen piste kuvaa ASPA:an tehtyä palautusta. Y-akseli kuvaa kokonaisvirhemäärää palautuksessa ja x-akseli aikaa. Pisteen väri ja koko määräytyvät yleisimmän virhekategorian sekä sen virhemäärän mukaan. Kuvaajasta 7 erottuu selkeästi harjoitustyön viimeinen palautuspäivä lähes välittömänä laskuna palautuksien määrässä (punainen katkoviiva). Myös harjoitustyön korjauspalautus viikolla 49 erottuu pienempänä palautuksien ryppäänä.

Palautukset, joissa on vain muutamia virheitä koostuvat lähinnä `basic` -kategorian virheistä. `Function`-kategorian virheet olivat yleisimmillään palautuksissa, joissa kokonaisvirhemäärä oli kahden ja kahdeksan välillä. `Library`, `data_structures`, `file_handling` -virheet yleistyvät tämän jälkeen. Huomaamme selkeästi `exception_handling` -virheitä eniten sisältäneiden palautuksien määrän kasvun viikkoina, kun opiskelijat työstivät harjoitustöitään.

4.3 Virhemäärien muutokset peräkkäisissä palautuksissa

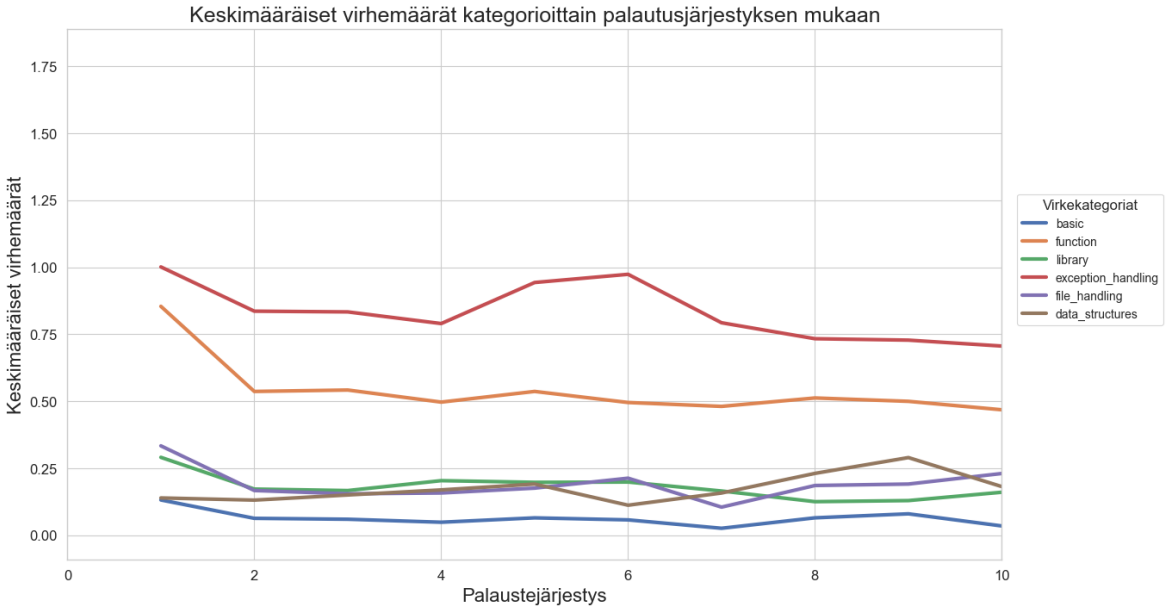
4.3.1 Kokonaisvirhemäärien muutokset peräkkäisten palautuksien aikana



Kuva 8: Keskimääräinen virheiden määrä peräkkäisissä palautuksissa

Kuvaajassa havainnollistetaan kokonaisvirhemäärien muutoksia opiskelijan palauttaessa saman nimiset tiedostot useaan kertaan peräkkäin. Lisäksi kuvaaja vertaa virhemäärää palautuksien määrään, jotka jokaisessa palautuskerrassa huomioidaan. Kuvaaja 8 esittää 60. palautukseen asti virheiden kokonaiskeskiarvoa. Palautusmääristä huomataan, että 92.1% kaikista opiskelijoiden suorittamista perättäisistä testeistä tapahtuivat ensimmäisen kymmenen palautuskerran sisällä. Tänä aikana virheiden keskiarvo palautuksessa laskee 2,778:sta 1,672:teen. Laskeva trendi on havaittavissa 20. palautuskertaan asti. Virhekeskiarvon lasku ensimmäisen ja toisen palautuskerran välillä on $2{,}778 - 1{,}94 = 0{,}838$, mikä vastaa 30,17% suhteellista laskua . Tällöin voidaan havaita, että ensimmäisen palautuskerran jälkeen opiskelijat korjaavat merkittävimmissä määrin virheitä.

4.3.2 Kategoriakohtaiset virhemäärien muutokset peräkkäisten palautuksien aikana



Kuva 9: Virheiden määrä kategorioittain kymmenessä ensimmäisessä peräkkäisessä palautuksessa

Kuvaaja 9 tarkastelee ensimmäisessä kymmenessä palautuskerrassa kategoriakohtaisesti virheiden keskimäärien muutoksia. Kuvaajassa 8 tunnistetu vahva lasku virheiden määrissä ensimmäisen ja toisen palautuskerran välillä on huomattavissa, kun kaikkien virhekategorioiden keskiarvot yhtenäisesti laskevat myös tällä välillä. Tämän jälkeen yhtenäinen laskeva trendi päättyy ja vaihtelevuus lisääntyy. Edelleen `function`- sekä `exception_handling` kategoriat ovat yleisimpiä virhekategorioita ja molemmat erottuvat selkeästi muita korkeammilla määrillä koko järjestyksen läpi. `basic`-luokan virheet pysyvät vähiten edustettuna. Muiden kategorioiden virhemäärät laskevat ja nousevat toistensa edelle ilman selkeää trendiä.

5 Keskustelu

5.1 Opiskelijoiden yleisimmät virheet

Merkittävimmät virhekategoriat olivat tiedostonkäsittely sekä aliohjelmat. Nämä kaksi virhekategoriaa muodostivat yli 3/4 kaikista virheistä ja poikkeustenkäsittely itsesään melkein puolet. Varianssi oli merkittävää kategorioiden välillä. Muut neljä kategoriaa olivat suhteellisesti paljon lähempänä toisiaan.

5.1.1 Poikkeustenkäsittelyyn liittyvät virheet

Yksi syy tälle poikkeustenkäsittelyyn liittyvien virheiden suurelle määrälle voi olla ohjelmointityyli, jossa itse ohjelmointivaiheessa poikkeustenkäsittely jätetään lähes kokonaan huomioimatta. Shah, Gorg & Harrold, 2010 esittävät tälläisen tyylin olevan yleistä kokemattomilla ohjelmoijilla, jolloin poikkeistenkäsittelyä hyödynnetään vain törmätessä odottamattomaan ohjelman toimintaan. Poikkeustenkäsittelyä siis käytetään tällöin lähinnä virheenkorjaustyökaluna ohjelmoinnin aikana. Rashkovits & Lavy, 2011 havaitsivat omassa tutkimuksessaan, että tarkastelluista opiskelijoden palautuksista 41%:sta puuttui poikeustenkäsittely kokonaan sekä 45% palautuksista osoitti vain hyvin vähäistä ymmärrystä poikkeustekäsittelystä. Tutkimus selitti tätä opiskelijoiden ohjelmointityylillä, jossa keskitytään vain teknisten vaatimusten täyttymiseen, jolloin ylläpidettävyyys sekä modulaarisuus kärsivät. Muita esitettyjä syitä olivat opiskelijoiden väärinkäsitys poikkeustenkäsittelyn tarkoituksesta. Opiskelijat näkivät poikkeustenkäsittelyn vain työkaluna, jolla estetään kriittisen systeemitason virheitä, kuten muisti sekä I/O -virheet. (Rashkovits & Lavy, 2011)

Aikaisemmat tutkimukset ovat myös osoittaneet poikkeustenkäsittelyn tuottavan haasteita opiskelijoille, ja sen näi nollen usein jäävän taka-alalle harjoitustehtävissä. Samanlainen trendi voidaan tunnistaa tämän tutkimuksen aineistosta. LUT Yliopiston *Ohjelmoinnin perusteet* -kurssin ohjelmointiopas painottaakin vahvasti poikkeustenkäsittelyä ja sen tärkeyttä virheiden ennaltaehkäisyssä, selvittämisessä sekä niistä toipumisesta (Vanhala & Nikula, 2020). Jos ohjelmointityyli palautuksia tehdessä painottaa eniten vain tehtävänannon toimintojen täyttymistä, mutta tyyliohje taas laadukkasta sekä vakaata koodia, syntyy välttämättä ristiriitaa. Tämä voidaan nähdä siinä, että syntyy suuri määrä juuri tiedostonkäsittelyn virheitä.

PK1-2 (Ei käytetä yleistä `Exception`-poikkeustyyppiä) oli kolmanneksi yleisin aineistossa. Shah, Gorg & Harrold, 2010 esittivät yhdeksi ongelmalliseksi strategiaksi poikkeustenkäsittelyssä yleistämisen, jolloin eri poikkeustyyppejä ei tunnisteta ja käsitellä erikseen. Kuitenkin kurssin ohjelmointiopas sekä tyyliohje vaativat Python:in `Exception`-poikkeustyypin

käyttöä, jonka alle jäävät kaikki poikkeukset `SystemExit` sekä `KeyboardInterrupt` poislukien. Ohjelmointiopas käy läpi syvällisemmin poikkeustenkäsittelyä Python:ssa, mutta tarkentaa lopuksi, että lyhyissä ohjelmissa kuten kurssin töissä sen voi jättää sivurooliin. Opas mainitsee tavoitteekseen esitellä lukijalle poikkeustenkäsittelyn sekä osoittaa sen toteuttaminen tärkeimpien käskyjen kohdalla. (Vanhala & Nikula, 2020) Poikkeustenkäsittelystä on tällöin erilaisia käsityksiä sekä vaatimuksia kurssitöissä eri instituutioissa. Opiskelijalle voi tällöin jäädä väärä käsitys, miten se tulisi toteuttaa. Aikaisemmat kokemukset tai ulkoiset lähteet voivat ohajata erilaisiin toteutuksiin poikkeuksienkäsittelyssä, jotka ovat ristiriidassa LUT Yliopiston tyyliohjeen kanssa.

5.1.2 Aliohjelmiin liittyvät virheet

Aliohjelmavirheet olivat toiseksi yleisin virhekategoria. Taulukosta 2 nähdään, että merkittävimmät yksittäiset aliohjelmavirheet olivat AR6 (return-komennon puuttuminen) sekä AR6-5 (Palautetaan useita paluuarvoja). Kurssin tyyliohje vaatii kaikkien aliohjelmien päättyvän `return`-komentoon sekä vaatii vain yksittäiset paluuarvot. Jos aliohjelmasta tulee palauttaa useita arvoja tulisi käyttää listaa tai oliota. Python suorittaa automaattisesti aliohjelman päättyessä `return`-komennon palauttaen `None` -arvon. Jos ohjelmoija ei tarkenna `return`-komentoon palautusarvoa, myös tässä tapauksessa Python palauttaa automaattisesti tyhjän palautusarvon. Koska Python on erittäin anteeksiantava `return`-komennon käytön kanssa, voi samanlainen poikkeustenkäsittelyssä tunnistettu vain olennaisiin tehtävänannon ominaisuuksiin keskittyvä ohjelmointityyli johtaa komennon käytön unohtumiseen.

5.1.3 Muut virhekategoriat

Neljänneksi yleisin virhe aineistossa oli tyyliohjeen vastainen tiedoston rakenne (MR1). Kurssin aikana palautettujen tiedostojen tulisi aina noudattaa kaavaa, joka noudattaa seuraavaa järjestystä: alkukommentti, standardi- sekä omien kirjastojen sisällytykset, kiintoarvojen ja luokkien määritys, aliohjelmien sekä pääohjelman määritys, ja viimeisenä pääohjelman kutsuminen. Kun ohjelma jaetaan useampaan tiedostoon, tulee pieniä muutoksia tähän kaavaan, mutta rakenne pysyy tällöinkin käytännössä samana. MR1:n alle jää hyvin laaja kirjo erilaisia rakennevirheitä, jotka kattavat niin tiedoston yleisen jäsentelyn kuin osien sisäisen loogisuuden epäselvyyksiä. Tällöin virheet, jotka luokitellaan MR1 virhe ID:n alle saattavat poiketa merkittävästi toisistaan. Esimerkiksi, jos tiedostossa luokka määritellään aliohjelmassa tai sisäiset kirjastot esitellään omien kirjastojen jälkeen, ASPA palauttaa analyysissä saman virhe ID:n. Tälläinen yleistävä virheluokka saattaa selittää sen yleisyyttä aineistossa.

5.2 Viikottaiset virhemäärät

5.2.1 Kokonaisvirhemäärä viikottain suhteessa palautuksien määrään

Kun viikottaisten palautuksien sekä virheiden määrää tutkitaan klassisen tilastotieteen näkökulmasta hyödyntämällä Pearsonin korrelaatiokerrointa, saamme arvon:

$$r = 0,965364197$$

joka viittaa vahvaan lineaariseen yhteyteen testien sekä virheiden määrän välillä. Tämä tarkoittaa, että keskimääräinen virhemäärä pysyi suhteellisen vakiona koko kurssin läpi. Kuitenkin tarkastellessa kuvaajaa 5, huomaamme viikkojen 43-46 välillä suoritettujen testien määrän melkein viisinkertaistuvan 309:stä 1521:en. Samalla aikavälillä tunnistettujen virheiden määrä nousi 870:stä 7488:n kasvaen yhdeksänkertaisesti. Viikon 46 jälkeen palautuksien sekä virheiden määrät palautuivat takaisin ensimmäisten viikkojen tasolle, poislukien viikon 49, jolloin viimeinen huipputrendi oli nähtävissä. Palautuksien määrän huipputrendit selittyvät kurssin harjoitustyön palautuspäivien avulla, jotka asettuvat viikkojen 46 sekä 49 loppuun. Kuitenkin kuvaajaa 5 tarkastellessa huomataan näillä huippuviikoilla esiintyneen keskimäärin enemmän virheitä kuin aikaisemmilla viikoilla.

5.2.2 Viikkottaiset virhemäärät kategorioittain

Kuvaajasta 6 pystymme erottelemaan kategorioittain virheiden määrät. `Function`-kategorian virheet ovat yleisin virhekategoria kurssin aikana poislukien viikot 45 ja 46. Jo tunnistettu virhemäärän kasvu näiltä kyseisiltä viikoilta on myös nähtävissä Kuvaajasta 5, kun lähes kaikki kategoriat saavat huippunsa viikolla 46. Suurin muutos näkyy `exception_handling`-virheissä, kun ne hyppäävät viikoilla 45 ja 46 ylivoimaisesti yleisimmäksi virhekategoriaksi. Tämä massiivinen piikki lähes yksin selittää jo aikaisemmassa kappaleessa tunnistetun poikkeaman virheiden keskimääräisestä kasvamisesta palautuksissa. Kuvaajasta huomataan myös, että näiden kahden viikon aikana suoritetut palautukset sisälsivät suurimman osan kaikista poikkeustenkäsittelyn virheistä.

Samat poikkeustenkäsittelyn trendit ovat nähtävissä kuvaajassa 7. Vihreällä merkityt pisteet kuvaavat palautuksia, joiden yleisin virhekategoria on poikkeustenkäsittely. Kaksi viikkoa ennen harjoitustyön palautuspäivää sisältävät eniten juurikin vihreää. Kuvaajasta huomaamme myös tasaisen `function`-kategorian virheiden määrän koko kurssin läpi. Palautukset, joissa virheuden kokonaismäärä vaihteli 2-10 välillä sisälsivät eniten aliohjelmiin liittyviä virheitä. Palautuksissa, jotka sisälsivät vain muutaman virheen, ne olivat yleisimmin `basic`-kategoriaa. Muut virhekategoriat esiintyivät yleisyydeltään vaihtelevammin palautuksissa. Muutamat palautukset juuri ennen harjoitustyön eräpäivää sisälsivät suuria määriä data_

structures -kategorian virheitä, mutta todennäköisesti nämä ovat yksittäisten opiskelijoiden peräkkäisiä palautuksia.

5.3 Muut palautuksien trendit

5.3.1 Peräkkäisten palautusten vaikutus keskimääräisiin virhemääriin

Kuvaajassa 8 tarkasteltiin opiskelijoiden peräkkäisiä ASPA-palautuksia siten, että saman sähköpostin ja tiedostonimen palautukset ryhmiteltiin peräkkäisiksi suorituksiksi. Huomionarvoista on, että 92,1% kaikista peräkkäisistä testeistä tapahtui kymmenen ensimmäisen palautuskerran sisällä. Tänä aikana virheiden keskimääräinen määrä putosi 2,778:sta 1,672:een, mikä tarkoittaa 39.8% absoluuttista laskua kymmenenteen palautukseen mennessä. Erityisen terävä lasku nähtiin ensimmäisen ja toisen palautuksen välillä, kun keskiarvo aleni 2,778:sta 1,940:een (–0,838, –30,2%). Laskeva trendi jatkui vielä 20. palautuskertaan asti, jonka jälkeen keskiarvo stabiloitui. Tämä selkeä virheiden määrän väheneminen osoittaa, että ASPA:a hyödyntäen tehty välitön korjaaminen ja uudelleentestaaminen tukee opiskelijoita virheiden systemaattisessa hallinnassa ja vähentämisessä.

5.3.2 Kategoriakohtaiset virhemäärien muutokset peräkkäisissä palautuksissa

Kuva 9 syventää analyysiä kymmenen ensimmäisen palautuskerran osalta ja tarkastelee virheiden keskiarvoja kategorioittain. Kaikissa kategorioissa havaittiin selkeä pudotus ensimmäisen ja toisen palautuksen välillä, mikä korostaa opiskelijoiden ensisijaisia korjaustoimenpiteitä heti virheiden havaitsemisen jälkeen. Tämän jälkeen laskeva trendi heikkeni ja kategorioiden välinen vaihtelu lisääntyi. `Function`- ja `exception_handling`-virheet pysyivät läpi peräkkäisten palautusten selvästi yleisimpinä, kun taas `basic`-luokan virheiden osuus jäi vähäiseksi. Muiden kategoriakohtaisesti mitattujen virheiden määrissä ei erottunut pitkää yhtenäistä trendiä, vaan ne nousivat ja laskivat eri palautuskerroilla, mikä viittaa siihen, että tiettyjen virhetyyppien korjaaminen vaatii toistuvampaa tarkistusta ja harjoittelua.

5.4 Virheet harjoitustyössä

Pylväläinen, 2024 tutki kandidaatin työssään saman *Ohjelmoinnin perusteiden* -kurssin harjoitustöiden arviointeja. Tutkimuksessa tarkasteltiin myös virheiden jakaumaa eri palautustasoilla sekä kokonaisuudessaan. Pylväinen havaitsi myös suurimmaksi haasteksi virheiden osalta opiskelijoilla poikkeustenkäsittelyn sekä aliohjelmien tiedonvälityksen, jotka esiintyivät vahvasti ASPA:n virhedatassa. Myös tarkasteltaessa kuvaajaa 6 nähdään, että yleisimmät kaksi virhekategoriaa ASPA:n harjoitustyötä edeltävinä viikkoina olivat juuri kyseiset kategoriat. Vaikka ASPA tunnisti erityisen paljon näitä virhetyyppejä, ja virheiden määrien nähtiin olevan laskussa toistuvilla palautuksilla, jatkuivat virhekategorioiden trendit siitä huoli-

matta myös palautetuissa harjoitustöissä. Tarkemmin juuri vääränlainen tai kokonaan puuttuva poikkeustenkäsittely, aliohjelmien parametrien sekä palautusarvojen puuttuminen, sekä tietorakenteiden alustus aiheuttivat haasteita opiskelijoille.

Tälläiset virheet eivät lokeroidu huomiovirheiksi, vaan ne osoittavat laajempia haasteita ohjelmoinnin perusteiden sisäistämisessä sekä aukkoja opiskelijoiden ymmärykssä kurssin vaatimuksista. ASPA ei korjaa tai selitä virhettä syvemmin, vaan se ainoastaan osoittaa opiskelijalle, että ohjelmassa on todennäköisesti virhe merkatussa koodin kohdassa. Jos opiskelija ei ymmärrä ASPA:n osoittamaa virhettä, hän tuskin osaa korjata sitä. Kuitenkin huolestuttavinta Pylväläinen, 2024 löydöksissä on, että nämä yleisimmät virhekategoriat esiintyivät keskimäärin yhtä usein sekä minimitason että tavoitetason töissä. Kurssin harjoitustyön voi suorittaa kolmella eri tasolla, minimi, perus, ja tavoite, jotka korreloivat vastaavasti arvosanaan. Tämä on huolestuttava trendi, mikä tarkoittaisi laajojen poikkeustenkäsittelyn sekä aliohjelmien tiedonvälityksen liittyvien väärinkäsityksien esiintyvän koko kurssia suorittavien joukossa riippumatta opiskelijan tavoittelamasta arvosanasta. Kuitenkin hän huomauttaa ohjelman kompleksisuuden nousun eri tasoilla osaltaan selittävän tätä trendiä, kun laajempi ja monimutkaisempi ohjelma ajaa opiskelijan tekemään helpommin virheitä. Tutkimus ei myöskään huomioi tavoietason vähäisistä palautusmääristä johtuvia mahdollisia tapauksia, joissa yksi suuren määrän samoja virheitä sisältävä suoritus vaikuttaa merkittävästi lopulliseen keskiarvoon.

5.5 Tutkimuskysymykset

Työn tavoitteena on vastata seuraaviin tutkimuskysymyksiin: *"Millaisia virheitä opiskelijat tekevät Ohjelmoinnin perusteet -kurssin aikana?"*, *"Voiko ASPA:n virhedatan visualisaation avulla löytää poikkeamia ja trendejä?"*, sekä *"Miten voimme hyödyntää ASPA:n virhedataa kurssin kehittämiseksi?"*.

Ensimmäinen tutkimuskysymys on kaikista yksiuloitteisin. Virhedatasta saatiin kerättyä kaikki ASPA:n tunnistamat rakenteelliset virheet opiskelijoiden palautuksista. Tämä tutkimus ei käsittele syntaksivirheitä eikä muita virhetyyppyjä ASPA:n tunnistamien ulkopuolelta. Aineistosta erottuivat selkeästi aliohjelma- sekä poikkeustenkäsittelyn virhekategoriat. Aliohjelmavirheet olivat yleisin virhekategoria lähes koko kurssin ajan, mutta harjoitustöitä edeltävinä viikkoina poikkeustenkäsittelyn virheet ottivat selkeän johtoaseman. Tarkemmassa virhe ID:en PK4 nousi yleisimmäksi yksittäiseksi virhetyypiksi. Se on samanlainen useita virheitä alleen kattava tunniste, kuin MR1. Sen alle lukeutuvat kaikki muut tiedosto-operaatiot paitsi sulkeminen ja avautuminen, jotka saavat omat virhe ID:nsä. Opiskelijoiden palautuksista myös puuttui usein `return`-komento aliohjelmien lopusta. Useiden palautusarvojen käyttäminen nousi myös esiin aineistosta.

Poikkeamia ja trendejä löydettiin myös kategorioiden ulkopuolelta. Peräkkäisillä palautuksilla virheiden keskimäärät lähtivät laskemaan. Tämä osoittaa, että ASPA:n käyttö ainakin osittain auttaa opiskelijoita vähentämään erilaisia rakennevirheitä palautuksissaan. Selkeä poikkeama virheiden kokonaismäärän sekä palautuksien määrän välillä harjoitustöitä edeltävinä viikkoina todettiin visualisaation avulla, vaikka korrelaatiokerroin osoitti toisin. Tämä poikkeama koostui lähes täysin poikkeustenkäsittelyn virheistä. Kuitenkin jo mainitut poikkeustenkäsittelyn sekä aliohjelmien virheet esiintyivät myös yliedustetusti harjoitustöiden lopullisissa palautuksissa. ASPA ei siis suurista tunnistusmääristä huolimatta saanut estettyä tämän trendin jatkumista. Huomasimme myös palautuksien, jotka sisälsivät vain yhdestä kolmeen virhettä sisältävän yleisimmin `basic`-kategorian virheitä. Aliohjelmavirheet olivat yleisimmillään palautuksissa, jotka sisälsivät kolmesta kymmeneen virhettä. Muut virhekategoriat esiintyivät paljon laajemmalla hajonnalla, eikä niistä pystynyt tunnistamaan samanlaisia trendejä.

Analyysiin käytetyt työkalut on suunniteltu sekä toteutettu, että niitä voidaan hyödyntää tulevien kurssien tarkasteluun. Työkalun avulla voidaan jatkuvasti analysoida, miten opetus vaikuttaa ASPA:n tunnistamiin virheisiin sekä vertailla sitä aikaisempien kurssien trendeihin. Tällöin on mahdollista tunnistaa, tutkia sekä korjata opiskelijoiden haasteita sekä väärinkäsityksiä jo kurssin aikana. Tämän kurssin pohjalta huolestuttavinta oli poikkeustenkäsittely sekä aliohjelmien tietovirrat, jotka tuottivat haasteita kurssin aikana sekä lopullisissa harjoitustöissä, tavoitellusta arvosanasta riippumatta. Tämä trendi tulisi jatkossa huomioida opetuksessa esimerkiksi lisäämällä ASPA:n osaksi viikkotehtävien automaattitarkastusta. Tämä pakottaisi opiskelijat keskittymään pitemmällä aikavälillä tarkemmin kurssin vaatimuksiin liittyen juuri näihin kipukohtiin.

5.6 Epätarkkuudet sekä jatkotutkimus

Tämän tutkimuksen aineistona on käytetty ASPA:n tuottamaa virhedataa. Tämän vuoksi ASPA:n analyysin aikana tekemät virheet vaikuttavat myös tämän tutkimuksen tuloksiin. Yksittäiset opiskelijat voivat vaikuttaa merkittävästi tuloksiin erityisesti kurssin loppuviikkoina, sillä palautusten määrä vähenee huomattavasti harjoitustyön palauttamisen jälkeen. Lisäksi tutkimuksessa ei ole huomioitu niitä opiskelijoita, jotka eivät käyttäneet ASPA:a kurssin aikana. Luukkainen et al., 2022 kuitenkin huomauttaa omassa tutkimuksessaan, että vain noin 5,5 % opiskelijoista jätti työkalun käyttämättä vuoden 2022 kurssilla. Työkalua käytettäessä on suositeltavaa tarkistaa aineisto suurten poikkeamien varalta ja tarvittaessa poistaa ne analyysistä. Lisäksi työkalu ei automaattisesti erottele henkilökunnan tekemiä palautuksia opiskelijoiden palautuksista, mutta tällä tekijällä on tutkimuksen tuloksiin vain vähäinen vaikutus suuren palautusmäärän vuoksi.

Jatkossa tutkimusta voisi laajentaa vertaamalla opetuksessa tehtyjen muutosten vaikutuksia ASPA:n tunnistamiin virhemääriin. Tällöin voitaisiin tutkia esimerkiksi, kuinka muutokset opetuksen sisällössä tai menetelmissä vaikuttavat opiskelijoiden tekemiin yleisiin virheisiin. Lisäksi voisi selvittää, kuinka opiskelijoiden ymmärrys virheiden luonteesta kehittyy kurssin aikana ja miten ASPA:n palautteen tarkkuutta ja selkeyttä voisi edelleen parantaa tukemaan paremmin oppimista.

5.7 Yhteenveto

Tutkimuksessa tarkasteltiin LUT-yliopiston Ohjelmoinnin perusteet -kurssin vuoden 2024 opiskelijoiden ASPA-työkalulla tuotettua virhedataa. Keskeiset tutkimuskohteet olivat virheiden tyypit ja niiden esiintyvyys opiskelijoiden tehtäväpalautuksissa. Tutkimuskysymyksinä olivat: ”Millaisia virheitä opiskelijat tekevät Ohjelmoinnin perusteet -kurssilla?”, ”Voiko ASPA:n virhedatan visualisoinnilla tunnistaa trendejä ja poikkeamia?” sekä ”Kuinka ASPA:n virhedataa voidaan hyödyntää kurssin kehittämisessä?” Aineistoa analysoitiin tilastollisesti sekä visualisoitiin graafisesti käyttäen Python-ohjelmointikieltä sekä Pandas- ja Seaborn-kirjastoja.

Analyysin tulokset osoittavat, että yleisimpiä virheitä olivat poikkeustenkäsittelyyn sekä aliohjelmien käyttöön liittyvät virheet. Erityisesti poikkeustenkäsittelyn virheet lisääntyivät huomattavasti harjoitustöiden palautuspäivän lähestyessä. Lisäksi havaittiin, että toistuvat palautukset vähensivät selkeästi virheiden määrää. Tutkimuksessa tunnistettiin myös opetukseen liittyviä kehityskohteita, kuten tarve opettaa poikkeustenkäsittelyä ja aliohjelmia syvällisemmin.

Tutkimuksen tuloksia voidaan hyödyntää Ohjelmoinnin perusteet -kurssin jatkokehityksessä, erityisesti opetuksen kohdentamisessa tunnistettuihin yleisiin virheisiin. Lisäksi tutkimus tarjoaa pohjan jatkotutkimukselle, jossa voitaisiin tarkastella opetuksessa tehtyjen muutosten vaikutuksia virheiden esiintymiseen ja opiskelijoiden ymmärryksen kehittymiseen.

Lähdeluettelo

Behrens, J. T. (1997). Principles and procedures of exploratory data analysis. *Psychological Methods* 2, s. 131–160. URL: https://api.semanticscholar.org/CorpusID:7114895.

Birillo, A., Vlasov, I., Burylov, A., Selishchev, V., Goncharov, A., Tikhomirova, E., Vyahhi, N. & Bryksin, T. (helmikuu 2022). Hyperstyle: A Tool for Assessing the Code Quality of Solutions to Programming Assignments. Teoksessa: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*. Vol. 1. SIGCSE 2022. New York, NY, USA: Association for Computing Machinery, s. 307–313. ISBN: 978-1-4503-9070-5. DOI: 10.1145/3478431.3499294. URL: https://dl.acm.org/doi/10.1145/3478431.3499294 (viitattu 27. 10. 2024).

Hartwig, F. & E.Dearling, B. (1979). *Exploratory Data Analysis*. en. SAGE Publications Inc. ISBN: 978-1-4129-8423-2. DOI: 10.4135/9781412984232. URL: https://methods.sagepub.com/book/exploratory-data-analysis (viitattu 21. 11. 2024).

Hollingsworth, J. (lokakuu 1960). Automatic graders for programming classes. *Commun. ACM* 3(10), s. 528–529. ISSN: 0001-0782. DOI: 10.1145/367415.367422. URL: https://dl.acm.org/doi/10.1145/367415.367422 (viitattu 25. 10. 2024).

Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S. H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M. Á., Sheard, J., Skupas, B., Spacco, J., Szabo, C. & Toll, D. (heinäkuu 2015). Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. en. Teoksessa: *Proceedings of the 2015 ITiCSE on Working Group Reports*. Vilnius Lithuania: ACM, s. 41–63. ISBN: 978-1-4503-4146-2. DOI: 10.1145/2858796.2858798. URL: https://dl.acm.org/doi/10.1145/2858796.2858798 (viitattu 26. 10. 2024).

Liu, D., Calver, J. & Craig, M. (2024). A Static Analysis Tool in CS1: Student Usage and Perceptions of PythonTA. Teoksessa: *Proceedings of the 26th Australasian Computing Education Conference*. ACE '24. Sydney, NSW, Australia: Association for Computing Machinery, s. 172–181. ISBN: 9798400716195. DOI: 10.1145/3636243.3636262. URL: https://doi.org/10.1145/3636243.3636262.

Luukkainen, R., Kasurinen, J., Nikula, U. & Lenarduzzi, V. (lokakuu 2022). ASPA: A static analyser to support learning and continuous feedback on programming courses. An empirical validation. Teoksessa: *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training*. ICSE-SEET '22. New York, NY, USA: Association for Computing Machinery, s. 29–39. ISBN: 978-1-4503-9225-9. DOI: 10.1145/3510456.3514149. URL: https://dl.acm.org/doi/10.1145/3510456.3514149 (viitattu 22. 10. 2024).

McKinney, W. (syyskuu 2017). *Python for data analysis: data wrangling with Pandas, NumPy, and IPython*. O'REILLY. ISBN: 978-1-4919-5763-9.

Nykänen, J. (2024). Moodle-LTI integration of ASPA using modern web technologies. eng. Accepted: 2024-05-02T05:15:34Z. URL: https://lutpub.lut.fi/handle/10024/167285 (viitattu 25. 10. 2024).

Perez, F. & Granger, B. E. (2007). IPython: A System for Interactive Scientific Computing. *Computing in Science & Engineering* 9(3), s. 21–29. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.53. URL: http://ieeexplore.ieee.org/document/4160251/ (viitattu 23. 11. 2024).

Perkel, J. M. (tammikuu 2021). Ten computer codes that transformed science. en. *Nature* 589(7842). Bandiera_abtest: a Cg_type: News Feature Publisher: Nature Publishing Group Subject_term: Computer science, Mathematics and computing, Software, Technology, s. 344–348. DOI: 10.1038/d41586-021-00075-2. URL: https://www.nature.com/articles/d41586-021-00075-2 (viitattu 25. 11. 2024).

Pylväläinen, K. (2024). Ohjelmoinnin harjoitustöiden arviointidatan analysointi ja visualisointi. fin. Accepted: 2024-12-13T05:21:22Z. URL: https://lutpub.lut.fi/handle/10024/168673 (viitattu 26. 03. 2025).

Rajagopalan, G. (2021). *Python Data Analyst's Toolkit - Learn Python and Python-Based Libraries with Applications in Data Analysis and Statistics*. Apress, an imprint of Springer Nature. ISBN: 978-1-4842-6398-3. URL: https://app.knovel.com/hotlink/toc/id:kpGZ1ISVT2/python-data-analysts/python-data-analysts.

Rashkovits, R. & Lavy, I. (2011). Students' Strategies for Exception Handling. English. *Journal of Information Technology Education. Research* 10. Num Pages: 183-207 Place: Santa Rosa, United States Publisher: Informing Science Institute, s. 183–207. ISSN: 15479714. DOI: 10.28945/1500. URL: https://www.proquest.com/docview/2616664368/citation/78A5CE07EBFA4ACEPQ/1 (viitattu 20. 02. 2025).

Shah, H., Gorg, C. & Harrold, M. (maaliskuu 2010). Understanding Exception Handling: Viewpoints of Novices and Experts. *Software Engineering, IEEE Transactions on* 36, s. 150–161. DOI: 10.1109/TSE.2010.7.

Ureel II, L. C. & Wallace, C. (helmikuu 2019). Automated Critique of Early Programming Antipatterns. Teoksessa: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE '19. New York, NY, USA: Association for Computing Machinery, s. 738–744. ISBN: 978-1-4503-5890-3. DOI: 10.1145/3287324.3287463. URL: https://dl.acm.org/doi/10.1145/3287324.3287463 (viitattu 26. 10. 2024).

Walker, O. (2022). *Hyperstyle*. en. DOI: 10.1145/3478431.3499294. URL: https://dl.acm.org/doi/epdf/10.1145/3478431.3499294 (viitattu 27. 10. 2024).

Vanhala, E. & Nikula, U. (2020). *Python 3 – ohjelmointiopas versio 1.2.1*. fin. Accepted: 2021-01-04T06:59:03Z. LUT-yliopisto. ISBN: 978-952-335-622-1. URL: https://lutpub.lut.fi/handle/10024/162088 (viitattu 20. 02. 2025).

Waskom, M. L. (huhtikuu 2021). seaborn: statistical data visualization. en. *Journal of Open Source Software* 6(60), s. 3021. ISSN: 2475-9066. DOI: 10.21105/joss.03021. URL: https://joss.theoj.org/papers/10.21105/joss.03021 (viitattu 26. 11. 2024).