jamk

Algoritminen ajattelu ohjelmoinnissa

Kuvaileva kirjallisuuskatsaus

Tuulia Nivalainen

Opinnäytetyö, AMK
Joulukuu 2023
Tietojenkäsittely ja tietoliikenne

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Nivalainen, Tuulia

Algoritminen ajattelu ohjelmoinnissa. Kuvaileva kirjallisuuskatsaus

Jyväskylä: Jyväskylän ammattikorkeakoulu. **Joulukuu 2023**, 34 sivua

Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Aikaisemmin ohjelmoinnissa keskityttiin teknisten taitojen hallitsemiseen, eikä huomiota kiinnitetty itse ohjelmointiprosessiin ja siinä esiintyvien vaiheiden ymmärtämiseen. Ohjelmointitaitojen hallitseminen vaatii kuitenkin teknisten taitojen lisäksi kykyä sisäistää ohjelmoinnin periaatteita ja prosesseja. Algoritmisen ajattelun näkökulman myötä katse on kiinnittynyt yksilön ajattelutaitoihin; algoritminen ajattelu tarjoaa erilaisia tapoja oppia ja kehittää ohjelmoinnissa tarvittavia ominaisuuksia sekä kykyjä innovatiivisesti ja yksilöllisesti. Lisäksi näkökulma tarjoaa tukea ohjelmoinnin teknisten taitojen hallitsemiseen.

Tutkimustyö tarkasteli algoritmisen ajattelun merkitystä ohjelmoinnissa. Työn tarkoituksena oli selvittää, miten algoritminen ajattelu tukee ja edistää yksilön ohjelmointitaitoja. Tavoitteena oli lisätä algoritmisen ajattelun merkitystä ohjelmoinnin taitamiseen ja kehittämiseen. Työn luonteen ja aiheen pohjalta tutkimusmenetelmäksi valikoitui traditionaalinen eli kuvaileva kirjallisuuskatsaus. Tutkimusmenetelmän avulla oli mahdollista saada tarkempaa tietoa algoritmisen ajattelun merkityksestä ohjelmoinnin taitamiseen sekä siihen vaikuttavista tekijöistä. Aineiston hakua varten määriteltiin aiheen kannalta keskeisimmät käsitteet ja hakusanat. Tutkimusaineisto haettiin kolmesta eri tietokannasta sekä manuaalisesti Google Scholarin avulla. Aineiston analysointi toteutettiin kuuden eri tutkimusartikkelin avulla. Valitut aineistoon sisällytetyt artikkelit keskittyivät ohjelmointikoulutuksen ja oppimisen kontekstei-hin, jonka pohjalta tuloksissa keskityttiin tarkastelemaan tätä yhteyttä.

Tutkimustulokset toivat esiin, että ohjelmointitaitojen oppiminen ja harjoittelu voi tapahtua erilaisin algoritmisen ajattelun menetelmin, kuten opiskelijakeskeisen ja ongelmakeskeisen oppimisen tai erilaisten älykkäiden oppimisympäristöjen avulla. Työn tulokset osoittivat, että algoritmisen ajattelun sisällyttäminen ohjelmointikoulutukseen ja -opetukseen tukee yksilöiden kykyä sisäistää ohjelmointiin tarvittavia ominaisuuksia ja taitoja. Algoritmisen ajattelun sisällyttäminen ohjelmointitaitojen kehittämiseen korostui erityisesti opiskelijoilla, joilla oli vähemmän ohjelmointikokemusta takanaan. Tutkimustulosten pohjalta voitiin todeta, että algoritminen ajattelu kiinnitti huomiota ennen kaikkea opiskelijan ajattelu- ja ongelmanratkaisutaitojen kehittämiseen teknisten ohjelmointitaitojen sijasta sekä tarjosi useita yksilöllisiä ja innovatiivisia ratkaisuja ohjelmoinnin eri haasteisiin.

Avainsanat (asiasanat)

Algoritminen ajattelu, ohjelmointitaidot, kirjallisuuskatsaus

Muut tiedot (salassa pidettävät liitteet)

-

jamk

Description

Nivalainen, Tuulia

Algorithmic thinking in programming. A descriptive literature review

Jyväskylä: JAMK University of Applied Sciences, December 2023, 34 pages

Degree Programme in Information and Communications Technology. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

Previously in programming, the focus was on mastering technical skills, and little attention was given to understanding the programming process and its stages. Mastering programming skills, however, requires the ability to internalize the principles and processes of programming in addition to technical skills. With the perspective of algorithmic thinking, attention has shifted to individual thinking skills; algorithmic thinking provides different ways to learn and develop the qualities and abilities needed in programming innovatively and individually. Moreover, the perspective provides support for mastering the technical skills of programming.

The research examined the significance of algorithmic thinking in programming. The purpose of the study was to determine how algorithmic thinking supports and enhances an individual's programming skills. The goal was to increase the importance of algorithmic thinking on mastering and developing programming skills. Based on the nature and topic of the study, the chosen research method was a traditional descriptive literature review. This method allowed for obtaining more detailed information about the significance of algorithmic thinking in mastering programming and the factors influencing it. Key concepts and keywords relevant to the topic were defined for data retrieval. The research articles were collected from three different databases and manually through Google Scholar. The analysis of the articles was carried out using six different research articles. The selected articles focused on the contexts of programming education and learning, forming the basis for examining this relationship in the results.

The research results revealed that learning and practicing programming skills can occur through various algorithmic thinking methods like student-based and problem-based learning, or different smart learning environments. The results of the study showed that including algorithmic thinking into programming education supports individuals' ability to adopt the qualities and skills needed for programming. The inclusion of algorithmic thinking in the development of programming skills was particularly emphasized for students with less programming experience. Based on the research results, it could be concluded that algorithmic thinking focused primarily on developing students' thinking and problem-solving skills instead of technical programming skills, offering several individual and innovative solutions to various challenges in programming.

Keywords/tags (subjects)

Algorithmic thinking, programming skills, descriptive literature review

Miscellaneous (Confidential information)

-

Sisältö

Kuviot

Taulukot

1 Algoritminen ajattelu – uudenlainen tapa ajatella ohjelmointia

”Miksi on niin, että jotkut ohjelmistosuunnittelijat ja tietojenkäsittelytieteilijät pystyvät luomaan selkeitä ja elegantteja suunnitelmia ja ohjelmia, kun taas toiset eivät? Onko kyse pelkästään älykkyydestä? Onko mahdollista parantaa vähemmän kyvykkäiden taitoja ja kykyjä jatkuvan koulutuksen ja harjoittelun avulla?” (Kramer 2007, 1.)

Kramerin (2007, 1) määritelmään viitaten ohjelmointitaidot vaativat tietynlaista osaamista ja kyvykkyyttä. Mitä nämä taidot sitten ovat ja ovatko taidot opittavissa? Ohjelmointiosaaminen pitää sisällään ohjelmointikielen, eri rakenteiden ja toimintojen hahmottamisen eli tietokoneen teknisen ja toiminnallisen osuuden. Ilman edellä kuvattuja toiminnallisia taitoja, ohjelmoinnin oppiminen voi muodostua erittäin haastavaksi. Ohjelmointi kattaa kuitenkin myös erilaisia ominaisuuksia, jotka tukevat edellä kuvattuja ohjelmoinnin teknisiä taitoja ja mahdollistavat syvällisemmän ja tehokkaamman tavan tukea ohjelmoinnin käytännön soveltamista. Taidot keskittyvät enemmän yksilön ajattelutaitoihin ja tapaan ratkoa vastaantulevia ongelmia sekä haasteita. Kingsley-Hughesin & Kingsley-Hughesin (2005) mukaan ohjelmointi on suurimmaksi osaksi ongelmanratkaisua. Vaikka ohjelmoinnin hallitsemiseen tarvitaan sekä teknisiä taitoja ja osaamista, toimivat taidot vain ohjelmoinnin välineenä. (Kingsley-Hughes ym. 2005, 185.) Esimerkiksi ohjelmoinnissa yksilön on kyettävä jakamaan vastaantulevat tehtävät pienempiin osiin sekä tunnistettava niiden välisiä suhteita ja rakenteita.

Tässä työssä ohjelmoinnin taitoja ja niiden sisäistämistä tutkitaan algoritmisen ajattelun näkökulmasta. Yleisenä määritelmänä algoritmisella ajattelulla luonnehditaan tuttuja ongelmanratkaisun menetelmiä ja erilaisia ajattelun taitoja, jotka yhdessä edistävät yksilön kykyä ratkaista monimutkaisia ja laaja-alaisiakin algoritmisia ongelmia (Wing 2011; Selby & Woollard 2013). Ajattelun taitoihin lukeutuvat niin kyky ajatella abstraktisesti, arvioivasti, algoritmisesti ja yleistävästi kuin ongelman pilkkominen pienempiin osaongelmiin (Selby ym. 2013). Ohjelmoinnin näkökulmasta algoritmisen ajattelun voidaan katsoa olevan keskeinen taito, joka mahdollistaa tehokkaiden toimintaohjeiden, eli algoritmien, suunnittelun ja toteutuksen edellä kuvattuihin ajattelun taitoihin pohjautuen. Nämä ajatteluntaidot mahdollistavat ongelmanratkaisun systemaattisesti ja loogisesti vaihe vaiheelta.

Algoritmisen ajattelun merkitys on noussut esiin Jeannette Wingin vuonna 2006 julkaiseman artikkelin "Computational Thinking" jälkeen. Artikkelissaan Wing (2006) kuvaa algoritmista ajattelua yhtenä nykypäivän tärkeimmistä taidoista. Algoritmisen ajattelun tärkeys on korostunut entisestään nykypäivän digitalisoituneessa maailmassa; se auttaa ihmisiä ratkaisemaan monimutkaisia ongelmia tehokkaasti ja luomaan innovatiivisia ratkaisuja erilaisiin haasteisiin. (Wing 2006, 33–35.) Vaikka lähtökohtana algoritmiseen ajatteluun voidaan katsoa olevan tietojenkäsittelytieteen ala, voidaan algoritmisen ajattelun taitoja soveltaa myös hyvin arkipäiväisissä toiminnoissa, kuten ruuanlaitossa tai leipomisessa.

Aikaisemmin ohjelmoinnissa keskityttiin lähinnä ohjelmoinnin teknisten taitojen, kuten ohjelmointikielen syntaksin eli lauseopin sekä eri rakenteiden hallitsemiseen, eikä huomiota kiinnitetty niinkään itse ohjelmointiprosessin ja sen vaiheiden ymmärtämiseen (Jenkins 2001, 41). Algoritmisen ajattelun myötä katse on kiinnittynyt yksilön ajattelutaitoihin; algoritminen ajattelu tarjoaa erilaisia tapoja oppia ja kehittää ohjelmoinnissa tarvittavia ominaisuuksia ja kykyjä innovatiivisesti ja yksilöllisesti. Lokkila, Rajala, Veersamy, Enges-Pyykkönen, Laakso & Salakoski (2016, 1555–1556) tuovat esiin, että ohjelmointiprosessiin keskittyvä lähestymistapa auttaa oppilaita sisäistämään paremmin ohjelmointiin tarvittavien toimintojen merkityksen, sen sijaan että opetus rajoittuisi pelkästään valmiin koodin lukemiseen ja sen opetteluun. Teknisen suorituksen hallitsemisen lisäksi tarvitaan siis kykyä ymmärtää, sisäistää ja soveltaa ohjelmoinnissa tarvittavia kognitiivisia taitoja, joita algoritmisen ajattelun näkökulma tarjoaa. Nykypäivän ohjelmointikoulutuksen ja -opetuksen merkitys korostuukin entisestään algoritmisen ajattelun näkökulmassa ja tarjoaa uudenlaisia ratkaisuja ohjelmointiopetukseen ja sen huomioimiseen. Vaikka ohjelmointiopetuksen merkitys korostuu erityisesti ohjelmointikoulutuksen kontekstissa, nousevat esille myös opiskelijoiden ja yksilöiden asenteet ohjelmoinnin opetusta kohtaan.

2 Työn tarkoitus, tavoitteet ja tutkimuskysymys

Tämän työn tarkoituksena oli tutkia algoritmisen ajattelun merkitystä ohjelmoinnin osaamisessa ja tehokkuudessa; miten algoritminen ajattelu edistää ohjelmointitaitoja. Lisäksi työssä käydään läpi, millaisia erilaisia näkökulmia algoritmisen ajattelun vaikutuksista ohjelmointiin on olemassa sekä miten eri näkökulmat tukevat tai eroavat aineistojen välillä. Edellä kuvatun pohjalta tavoitteena on lisätä algoritmisen ajattelun merkitystä ohjelmoinnissa ja siten tuoda esiin sen tärkeys ohjelmoin-

nin osaamisen ja toiminnallisuuden näkökulmasta; millaisia eri keinoja algoritmisen ajattelun näkökulma tuo ohjelmointitaitojen kehittämiseen. Lähtökohtana tämän työn toteuttamiseen oli oma mielenkiinto ja kiinnostus algoritmista ajattelua kohtaan sekä algoritmisen ajattelun merkitys ohjelmoinnin taitamisessa. Ohjelmointiopetuksessa on tapahtunut muutosta, johon algoritmisen ajattelun näkökulma tuo uudenlaisia ideoita ohjelmoinnin taitojen kehittämiseen. Aiemmin kuvattujen työn lähtökohtien, tavoitteiden ja tarkoituksen myötä esiin nousi seuraava algoritmisen ajattelun ja ohjelmoinnin hallitsemisen kannalta oleellinen tutkimuskysymys:

- Miten algoritminen ajattelu tukee yksilön ohjelmointitaitoja?

Algoritmisen ajattelun kokonaisvaltaisen käsitteen vuoksi tässä työssä algoritmista ajattelua on lähdetty tutkimaan sen keskeisimpien osa-alueiden kautta. Ohjelmointitaitojen merkittävyydessä huomioitiin taidot ja kyvyt, jotka edistävät ja tukevat ohjelmointiosaamista ohjelmoinnin perustaitojen lisäksi. Tutkimustyön luonteen ja aiheen pohjalta tutkimusmenetelmäksi valikoitui traditionaalinen eli kuvaileva kirjallisuuskatsaus. Lähtökohtana tutkimusmenetelmän valintaan oli lähestymistapa, joka tukisi tutkimuskysymyksen selvittämistä parhaalla mahdollisella tavalla. Tutkimusmenetelmän avulla oli mahdollista saada tarkempaa tietoa algoritmisen ajattelun merkityksestä ohjelmoinnissa ja siihen vaikuttavista tekijöistä. Lisäksi kuvaileva kirjallisuuskatsaus mahdollisti erilaisten näkökulmien tulkitsemisen ja johtopäätösten tekemisen kriittisesti tutkittavaa aihealuetta kohtaan.

Tämä työ lähtee liikkeelle teoreettisesta viitekehyksestä, jossa käsitellään ensin algoritmisen ajattelun määritelmää sekä siihen yleisimmin liitettyjä kognitiivisia osa-alueita. Ohjelmoinnin osalta esillä ovat yksilön ohjelmointiosaamista tukevat ominaisuudet. Teorian jälkeen siirrytään työn tutkimusosaan, jossa käsitellään tarkemmin työn toteutusta ja tutkimusprosessia. Esillä on myös tutkimustyön toteutuksessa huomioituja asioita niin tutkimuksen eettisyyden kuin luotettavuuden näkökulmasta. Teoriaan pohjautuen tarkastellaan tutkimustuloksia sekä niissä esiin nousseita havaintoja. Lopuksi esitellään työn johtopäätökset sekä analysoidaan tuloksia teoriaan pohjautuen. Edellä kuvatun lisäksi pohditaan tarkemmin algoritmisen ajattelun merkitystä tutkittavaa ilmiötä kohtaan.

3 Algoritminen ajattelu

Algoritmisen ajattelun taitoja voidaan hyödyntää monissa eri tilanteissa. Tässä työssä algoritmisen ajattelun määritelmää ja osa-alueita on lähdetty tutkimaan siihen yleisimmin liitettyjen kognitiivisten taitojen avulla. Lähdetään seuraavaksi katsomaan tarkemmin, miten algoritmisen ajattelun taidot tukevat ohjelmoinnissa tarvittavia ominaisuuksia ja mahdollistavat tehokkaan ja systemaattisen tavan ratkaista niitä.

3.1 Algoritmisen ajattelun määritelmä

Tietojenkäsittelytieteen merkitys yhteiskunnassamme on kasvanut merkittävästi viimeisten vuosikymmenien aikana. Teknologia, eri älylaitteet ja niiden kehitys ovat vaikuttaneet tapaan, miten toimimme. Tämä teknologian aikaansaama muutos on luonut tarpeen kokonaan uudenlaiselle osaamiselle. Tietojenkäsittelytiede (engl. Computer Science) on tutkimusala, joka tarjoaa perustan teknologian osaamiseen ja keskittyy yleisesti kehittämään ja tukemaan nyky-yhteiskunnan vaatimuksia eri tieteenaloilla. Sen perustana on ongelmanratkaisu ja looginen päättely erilaisissa tietojenkäsittelytieteellisissä konteksteissa. Yleisemmin tieteenalana se pyrkii ymmärtämään tietokoneiden toimintaa ja kehittämään menetelmiä, joiden avulla voidaan mahdollistaa tiedon tehokas käsittely. (Bourke 2018, 1–3.)

Teknologian ja tietojenkäsittelytieteen merkityksen myötä esiin on noussut algoritmisen ajattelun käsite. Sekä Selby ja muut (2013) että Wing (2006) kuvaavat algoritmisen ajattelun (engl. Computational Thinking, Algorithmic Thinking) olevan yksi nykypäivän tärkeimmistä taidoista. Se on keskeinen osa tietojenkäsittelytiedettä ja ohjelmointia, mutta sen periaatteita voidaan soveltaa myös laajemmin eri aloilla. (Selby ym. 2013; Wing 2006, 33.) Myös Cansu & Cansu (2019) korostavat algoritmisen ajattelun merkitystä nyky-yhteiskunnassa ja sen roolia yksilön kyvyssä ymmärtää sekä ratkaista erilaisia ongelmia digitalisoituneessa maailmassa. Algoritminen ajattelu nostaa siten esiin tavan ymmärtää ja toteuttaa tietojenkäsittelytieteellisisä ratkaisuja eri aloilla, vaikka taitoja voidaan Wingin (2006, 33–34) mukaan hyödyntää myös arkisissa toiminnoissa.

Ohjelmointi (engl. Programming) on taito, joka vaatii algoritmista ajattelua. Wing (2011) kuvaa algoritmisen ajattelun olevan joukko ajattelun taitoja, joiden avulla erilaisia ongelmia pystytään rat-

kaisemaan algoritmisesti. Sen avulla voidaan lähestyä monenlaisia ongelmia ja tehtäviä tehokkaasti ja selkeästi. Ajattelun taidot sisältävät niin analyyttista, loogista, matemaattista kuin ihmislähtöistä suunnittelua toiminnasta ja sen päämäärästä. Lisäksi ajattelutaidot auttavat yksilöitä hahmottamaan laajoja ja monimutkaisia ongelmia, pilkkomaan niitä pienempiin osaongelmiin, tunnistamaan ongelman kannalta olennaiset tiedot sekä analysoimaan ja kehittämään tehokkaita ratkaisuja ongelmien selvittämiseksi. Tärkeä osa algoritmista ajattelua on myös kyky arvioida toiminnan kannalta eri vaihtoehtoja. (Grover & Pea 2017, 34.) Algoritmisen ajattelun voidaan siten katsoa olevan moniulotteinen taito, joka kattaa erilaisia näkökulmia ja lähestymistapoja ongelmien ratkaisuun.

3.2 Algoritmisen ajattelun osa-alueet

Algoritminen ajattelu pitää sisällään useita toisiinsa vaikuttavia ominaisuuksia. Cansu ja muut (2019, 4–5) tuovat esille aikaisempiin tutkimuksiin ja julkaisuihin pohjautuen algoritmisen ajattelun koostuvan viidestä eri taidosta, jotka ovat:

- taito ajatella abstraktisti,
- ongelman pilkkominen pienempiin osiin,
- taito ajatella algoritmisesti,
- taito ajatella arvioivasti ja
- taito ajatella yleistäen.

Osa-alueiden myötä algoritminen ajattelu pohjautuu ongelmaratkaisutilanteen kokonaisvaltaiseen hahmottamiseen, suunnitteluun, toteutukseen ja analysointiin. Se mahdollistaa haastavien ongelmien selvittämisen ja ratkaisemisen yksilön omiin ajattelutaitoihin perustuen. Lisäksi on tärkeää huomata, että edellä kuvatut ajattelun taidot eivät ole täysin erillisiä, vaan ne liittyvät toisiinsa ja muodostavat laajemman lähestymistavan algoritmisen ajattelun käsitteeseen. (Wing 2006, 33–34; Selby ym. 2013.) Näin ollen algoritmisen ajattelun sisäistäminen auttaa yksilöä organisoimaan, analysoimaan ja hyödyntämään tietoa tehokkaasti, sekä ratkaisemaan monimutkaisia ja laaja-alaisia tehtäviä tai ongelmia algoritmeihin pohjautuen. Seuraavaksi käydään läpi tarkemmin edellä kuvattuja keskeisimpiä algoritmisen ajattelun taitoja.

3.2.1 Abstrakti ajattelu

Tietojenkäsittelytieteessä esiintyy lukuisia abstrakteja käsitteitä, mikä tekee abstraktista ajattelusta tärkeän taidon. Abstraktissa ajattelussa käsiteltävät kokonaisuudet eivät ole aina konkreettisesti havaittavissa, vaan yksilön on kyettävä sisäistämään ja kuvittelemaan asia mielessään (Kramer, 2007, 5–7). Yleisesti abstraktia ajattelua kuvataan yksilön kykynä hahmottaa suurempia asiakokonaisuuksia ja järjestää asioita helpommin ymmärrettäviksi tapahtumiksi. Tämä sisältää muun muassa tarpeettomien yksityiskohtien poistamisen kyseisestä tilanteesta. Lisäksi abstraktin ajattelun määritelmä pitää sisällään sopivien mallien tunnistamisen ja toimintojen yleistämisen. (Roberts 2009; Kramer 2007, 2.)

Kramer (2007) korostaa abstraktin ajattelun yhteyttä yksilön matemaattisiin taitoihin ja sitä kautta kykyyn ratkaista haastaviakin ongelmia. Hyvät matemaattiset taidot edistävät abstraktin ajattelun kehitystä ja ovat siten suuressa roolissa tietojenkäsittelytieteen ja siinä esiintyvien erilaisten ongelmien näkökulmasta. (Kramer 2007, 5–7.) Sveitsiläinen kehityspsykologi Jean Piaget kuvaa omassa kognitiivisen kehityksen teoriassaan myös abstraktin ajattelun tärkeyttä sekä sen merkitystä yksilön kehitykseen. Hän jakaa yksilön kehityksen neljään eri vaiheeseen. Viimeisessä, formaalien operaatioiden vaiheessa, yksilö kykenee ajattelemaan asioita symbolisesti ja systemaattisesti, ilman konkreettisia havaintoja. (Roberts 2009; Kramer 2007, 6.) Abstraktin ajattelun kehittyminen on Piagetin mukaan keskeinen osa yksilön kognitiivista kehitystä, mahdollistaen abstraktin tiedon prosessoinnin ja ongelmanratkaisun. Lisäksi se mahdollistaa tieteellisen ajattelutavan kehittymisen ja hallitsemisen. (Kramer 2007, 6.)

3.2.2 Ongelman pilkkominen osiin

Sekä Rich, Egan & Ellsworth (2019) kuvaavat ongelman pilkkomisen olevan yleisesti hyödynnetty vaihe monissa tieteellisissä ja käytännön ongelmanratkaisutilanteissa. Tämä lähestymistapa mahdollistaa ongelman syvällisemmän ymmärtämisen sekä tehokkaampien ratkaisujen löytämisen. Keskeisenä tavoitteena on tunnistaa ongelmaan sisältyvät osatekijät tai -prosessit sekä niiden väliset suhteet ja riippuvuudet. Yhteyksien löytäminen ja ymmärtäminen ovat lähtökohta ongelman pilkkomisessa ja ratkaisujen löytämisessä. (Rich ym. 2019, 416–421.)

Sekä Selby ja muut (2013) että Boom ja muut (2022) kuvaavat ongelman pilkkomisen taidon olevan lähtökohta algoritmisen ajattelun muille osa-aluille. Ongelman pilkkominen osiin on tärkeä lähestymistapa ohjelmoinnissa ja yleisesti algoritmisen ajattelun osana, sillä se auttaa yksilöä hahmottamaan ongelman kokonaisuuden järkevämmin. Tämän lisäksi se auttaa käsittelemään monimutkaisia ongelmia helpommin hallittavissa olevina osina ja mahdollistaa tehokkaan ratkaisun kehittämisen yksi osaongelma kerrallaan. (Grover ym. 2017, 27–28; Boom ym. 2022, 8291; Selby ym. 2013.) Jakamalla haastava asia pienempiin osiin ja käsittelemällä osia erikseen, yksilö voi saada paremman ymmärryksen asiaan liittyvistä vaiheista. Lisäksi se mahdollistaa tehokkaan etenemisen kohti haluttua ratkaisua.

3.2.3 Kyky ajatella algoritmisesti

Algoritmit ovat tärkeä osa algoritmista ajattelua. Mitä algoritmeilla sitten oikein tarkoitetaan? Yleisesti algoritmilla kuvataan joukkoa sääntöjä tai toimintaohjeita, jotka mahdollistavat erilaisten tehtävien suorittamisen (Rusanen 2021). Algoritmit voivat olla esimerkiksi matemaattisia laskentasääntöjä, loogisia päättelymalleja tai myös arkipäivästä tuttuja toimintoja kuten ruuanlaitto tai leipominen. Ohjelmoinnin yhteydessä algoritmilla viitataan tietokoneohjelman tai ohjelmointikielen loogiseen rakenteeseen, joka kuvaa vaiheita, joita ohjelman tulee suorittaa tietyn tehtävän tai ongelman ratkaisemiseksi (Cormen ym. 2009, 5). Ukkonen (2003, 19) puolestaan kuvaa algoritmia tietojenkäsittelytieteen keskeisenä käsitteenä, sillä se mahdollistaa tehokkaan tietojen käsittelyn ja ongelmien ratkaisemisen. Jotta tiettyä toimintoa tai tehtävää voidaan kutsua algoritmiksi, tulee sen sisältää seuraavat ominaisuudet:

1) *Äärellisyys*: algoritmin pitää loppua aina tietyn äärellisen määrän jälkeen. Algoritmi ei siten saa jatkua loputtomiin.
2) *Rajaus*: algoritmin vaiheet tulee olla tarkasti rajattu.
3) *Syöte*: algoritmi sisältää aina vähintään nolla tai enemmän syötteitä.
4) *Lopputulos*: algoritmi sisältää aina lopputuloksen. Lopputulos voi sisältää yhden tai useamman ratkaisun.
5) *Tehokkuus: a*lgoritmin tulee olla mahdollisimman tehokas ja helposti toteutettavissa oleva. (Pietiläinen 2021, 42–43; Knuth 1997, 4–6.)

Edellä kuvatut kohdat varmistavat, että algoritmin tarkoitus toteutuu. Hyvä algoritminen suunnittelu vaatii kykyä ymmärtää algoritmisia ratkaisuja, sekä toteuttaa niitä osana käytännön ongelmanratkaisua (Laaksonen 2022, 1). Yksinkertaisesti aiemmin kuvattua ruuanlaittoa voidaan tarkastella algoritmisena prosessina. Ruoanlaitto on algoritminen toiminto, joka päättyy, kun ruoka on valmis. Lisäksi ruuanlaitto perustuu tarkkaan määriteltyyn reseptiin, joka ohjaa jokaisen valmistusvaiheen selkeästi kohti haluttua lopputulosta. Syötteenä toimivat ruuanlaittoon käytetyt raaka-aineet ja määrät, jossa lopputuloksena syntyy herkullinen annos. Samantapaisesti voidaan ajatella tietojenkäsittelytieteessä ja tarkemmin ohjelmoinnissa, jossa tietty ohjelma koostuu toiminnoista eli ”reseptistä”. Ohjelmoinnissa edellä kuvatut ruuanlaiton toiminnot määritellään ja järjestellään tiettyjen sääntöjen mukaisesti tavalla, jota tietokone pystyy ymmärtämään.

3.2.4 Yleistävä ja arvioiva ajattelu

Yleistävä ja arvioiva ajattelu ovat kaksi hyödyllistä ajattelun muotoa, joita käytetään algoritmisessa ajattelussa sekä yleisesti päätöksenteossa ja ongelmanratkaisussa. Sekä Cole (2023) että Buckley, Archibald, Hargraves & Trochim (2015) kuvaavat arvioivalla ajattelulla yksilön kykyä arvioida ja ajatella erilaisia vaihtoehtoja sekä päätöksiä, jotka perustuvat saatavilla olevaan tietoon. Se sisältää taidon arvioida niin etuja, haittoja, riskejä kuin seurauksia eri vaihtoehtojen välillä sekä tehdä perusteltuja päätöksiä edellä kuvattuihin taitoihin pohjautuen. Arvioivan ajattelun katsotaan sisältävän myös kriittistä ajattelua ja analyyttistä päättelykykyä. (Cole 2023; Buckley ym. 2015, 2–4.) Arvioiva ajattelu ei siten ole itse toiminnan toteuttamisesta vastaava ajattelun muoto, vaan se keskittyy oikean vaihtoehdon valintaan aikaisempaan tietoon perustuen (Cole 2023). Esimerkiksi päätöksenteossa arvioiva ajattelu auttaa yksilöä arvioimaan eri vaihtoehtojen tai tilanteiden hyötyjä ja riskejä, sekä tekemään päätöksen, joka perustuu yksilön tietoon kyseisestä asiasta.

Yleistävällä ajattelulla puolestaan tarkoitetaan kykyä ymmärtää ja hahmottaa ongelmaan liittyvät tekijät sekä hyödyntää aikaisempaa tietoa ratkaisuissa ja niiden tunnistamisessa. Ongelman hahmottamisen myötä aikaisemmat kokemukset ja niiden sisäistäminen sekä tilanteen yleistäminen mahdollistavat ratkaisujen kehittämisen. (Selby ym. 2013.) Ohjelmoinnissa voi esiintyä esimerkiksi tietynlainen rajattu toiminto tai tapahtuma, jota voidaan soveltaa aina tietyn tyyppisten ongelmien esiintyessä (Cansu ym. 2019, 5). Siten yleistävä ajattelu auttaa yksilöä tiivistämään ja ymmärtämään monimutkaista tietoa sekä tekemään johtopäätöksiä ja ennusteita aikaisempiin tilanteisiin pohjautuen. Yleistävän ajattelun avulla yksilö pystyy arvioimaan erilaisia vaihtoehtoja arvioivaan

ajatteluun pohjautuen. Yleistävä ajattelu mahdollistaa nimensä mukaisesti yleisten sääntöjen tai toimintamallien luomisen, joiden avulla yksilö pystyy arvioimaan erilaisia vaihtoehtoja arvioivaan ajatteluun pohjautuen. Edellä kuvattujen ajattelutapojen sisäistäminen auttaa siten yksilöä tekemään järkeviä päätöksiä sekä ratkaisemaan vastaantulevia ongelmia ja haasteita rationaalisin keinoin ja toimintatavoin.

4 Ohjelmointitaitoja tukevat tekijät

Ohjelmointitaitoja tukeviin tekijöihin ja osa-alueisiin panostaminen on tärkeää, sillä sen edistämisellä on merkitystä ohjelmoinnin hallitsemiseen. Vaikka algoritminen ajattelu tuo esiin monia ohjelmointitaitoja edistäviä ominaisuuksia, tarvitaan tueksi myös yleisesti oppimista tukevia toimintoja. Tässä työssä ohjelmointitaitoja tukevia tekijöitä käydään läpi yksilön motivaation, aikaisemman ohjelmointikokemuksen sekä opetuksen ja harjoittelun kontekstissa.

4.1 Motivaatio

Oppimisen kontekstissa motivaatiolla kuvataan yksilön halua ja intoa menestyä ja oppia. Erityisen tärkeänä näyttäytyvät juuri yksilöä motivoivat tekijät, jossa sisäiset motivaatiota ylläpitävät asenteet korostuvat. Berginin & Reillyn (2005) mukaan motivaatio voidaan jakaa kahteen eri osaan: sisäiseen ja ulkoiseen motivaatioon. Sisäinen motivaatio kytkeytyy yksilön omaa haluun ja intoon oppia asioita. Opiskelijan sisäinen motivaatio ylläpitää hänen myönteisiä tunteitansa ja luo paremmat mahdollisuudet opiskelussa pärjäämiseen ja jaksamiseen. Lisäksi se edistää henkilön oppimista, sillä sisäisesti motivoitunut opiskelija toimii omien arvojensa ja sisäisten kiinnostustensa mukaisesti. Ulkoisella motivaatiolla voidaan puolestaan kuvata muun muassa saatua palautetta, palkkioita tai arvosanaa. Ulkoinen motivaatio ei siten lähde opiskelijasta itsestään. Useat eri tutkimukset ovat osoittaneet, kuinka sisäistesti motivoituneet opiskelijat suoriutuvat paremmin opinnoissaan kuin ulkoisesti motivoituneet opiskelijat. (Bergin ym. 2005, 293.)

Ilman motivaatiota ja halua ymmärtää ohjelmointiin tarvittavia taitoja ja kykyjä oppiminen on erittäin vaikeaa (Jenkins 2001, 1–2). Mitkä tekijät sitten vaikuttavat yksilön motivaatioon sekä miten yksilön motivaatiota voidaan ylläpitää ja edistää? Kori ja muut (2016) toivat esiin sen, että aikaisempi kokemus ohjelmoinnissa tukee opiskelijan motivaatiota ja mahdollistaa myönteisempien

tunteiden kokemisen. (Kori ym. 2016, 332.) Myös Cheah (2020) toi esiin, miten myönteiset asenteet ohjelmointiopetusta kohtaan edistävät opiskelijan motivaatiota niin itse ohjelmointia kuin sen oppimista kohtaan. Negatiiviset tunteet ja asenteet toimivat siten esteenä ohjelmoinnin opetuksessa ja harjoittelussa. Motivaatioon vaikuttavia tekijöitä tulisi huomioida ohjelmointiopetuksessa ja harjoittelussa. Keskeiseksi tekijäksi nousee motivoiva ja tukeva oppimisympäristö, joka ylläpitää ja mahdollistaa myönteisten tunteiden kokemisen.

4.2 Aikaisempi ohjelmointikokemus

Ohjelmointikokemuksella on paljon merkitystä ohjelmoinnin eri toimintojen sisäistämisessä (Siegmund, Kästner, liebig, Apel & Hanenberg 2013, 1301). Kuten myös edellisessä kappaleessa juuri todettiin, aikaisempi kokemus vaikuttaa myönteisesti yksilön motivaatioon ja sitä kautta myös ohjelmointitaitojen kehittymiseen. Alexandron, Armori, Grodon & Harel (2012) kuvaavat, miten oppiminen rakentuu aina vanhan tiedon päälle. He tuovat esiin skeemateorian, joka kuvaa tiedon rakentuvan aina tiettyjen rakenteiden ja toimintojen ympärille. Kun tietyntyyppinen ongelma ilmenee, voidaan ongelma ratkaista aikaisemmin opitun pohjalta. Siten oppimisen pohjalta käytetyt skeemat sisältävät abstraktin ratkaisun aina ongelmaan. (Alexandron ym. 2012.) Aikaisempi ohjelmointikokemus auttaa yksilöä rakentamaan parempia ja monimutkaisempia ratkaisuja kokemuksen ja aikaisemman harjoittelun myötä. Lisäksi se auttaa yksilöä yhdistämään aikaisempaa tietoa ja löytämään ratkaisuja erityyppisiin ohjelmointiongelmiin.

4.3 Opetus ja harjoittelu

Käytännön ohjelmointiharjoittelu on tärkeä osa ohjelmointitaitojen kehittymistä. Sekä Bergin ja muut (2005, 293), Jenkins (2001, 1) että Kori, Pedaste, Leijen & Tonisson (2016, 332) kuvaavat miten ohjelmointi nähdään usein haastavana taitona oppia, jossa opiskelijoiden tulee ymmärtää ja sisäistää monia erilaisia taitoja. Tämän pohjalta esiin on noussut erilaisia näkökulmia, miten ohjelmointia tulisi opettaa sekä mitkä tekijät vaikuttavat ohjelmoinnin oppimiseen. Jenkins muun muassa (2001, 43) kuvaa, miten perinteiset oppimistavat ehkäisevät ohjelmointiopetusta ja voivat vaikuttaa haitallisesti yksilön kykyyn omaksua ohjelmoinnissa tarvittavia taitoja ja prosesseja. Esille nousevat etenkin opiskelijan ongelmanratkaisutaidot, abstrakti ajattelu sekä ohjelmakoodin suunnittelu ja toteutus. (Cheah 2020). Perinteinen oppiminen tarvitsee tuekseen harjoittelua, jossa opiskelija pääsee soveltamaan opittua tietoa ja siten kehittämään taitojaan entisestään (Jenkins,

2001, 22). Tämän pohjalta on alettu kiinnittämään huomiota kolmeen ohjelmointitaitoja tukevaan tekijään:

- käytännönläheinen harjoittelu,
- tietomateriaalien käyttö sekä
- asiantuntijan opastus ja neuvonta.

Käytännönläheinen harjoittelu mahdollistaa oppilaiden taitojen soveltamisen ja ongelmien ratkaisemisen osana opetusta. (Jenkins 2001, 44.) Tietomateriaalien, kuten oppikirjojen ja erilaisten verkkomateriaalien käyttö tarjoavat tukea ja ohjeita ohjelmoinnin toimintojen ymmärtämiseen ja toteuttamiseen. Koulutuksen ja opetuksen kontekstissa asiantuntijuudella voidaan puolestaan viitata opettajaan, jonka tehtävänä on luoda oikeanlaiset puitteet opiskelijoiden ohjelmoinnin oppimiseen ja harjoitteluun. Asiantuntijaopastus ja neuvonta, joissa opettaja avustaa oppilaita ohjelmoinnin perusteiden ja käsitteiden oppimisessa, luovat opiskelijoille hyvät lähtökohdat menestyä ohjelmoinnissa.

5 Toteutus

Tutkimustyö toteutettiin kuvailevana kirjallisuuskatsauksena. Työ ei siten perustunut erikseen toteutettuun tutkimukseen tai haastatteluun, vaan työssä perehdyttiin verkosta löytyviin tieteellisiin artikkeleihin algoritmisesta ajattelusta, ohjelmoinnin tehokkuudesta ja ohjelmointitaidoista. Kirjallisuuskatsauksen tarkoituksena oli kartoittaa, mitä tietoa ja tieteellistä tutkimusta aiheeseen liittyen oli tutkimushetkellä olemassa ja miten tulokset tukevat tai eroavat toisistaan. Seuraavaksi kuvataan tarkemmin työn toteutusta: miten tiedonkeruumenetelmä, tiedonhaku ja analysointimenetelmä valittiin sekä mitä asioita tutkimuksen toteutuksessa otettiin kokonaisuudessaan huomioon.

5.1 Kuvaileva kirjallisuuskatsaus tutkimusmenetelmänä

Ohjelmoinnin ja algoritmisen ajattelun kokonaisvaltaisen luonteen kannalta työn tiedonkeruumenetelmäksi valikoitui kuvaileva eli traditionaalinen kirjallisuuskatsaus. Kuvailevan kirjallisuuskatsauksen tarkoituksena on kuvailla tutkittavaa aihealuetta teoreettiseen viitekehykseen pohjautuen ja peilata sitä aiheeseen liittyvään erikseen valitun tieteellisen kirjallisuuden kautta (Vilkka 2023, luku 1). Se toimii tiedonkeruumenetelmänä hyvin tutkimuksissa, joissa tarkoituksena on tuoda

esiin laajempi yleiskuva tutkittavasta aiheesta sekä tarvittaessa luokitella tarkastavan aiheen ominaisuuksia tai piirteitä (Salminen 2011, 6). Kuitenkin kuvailun tulee olla rajattua, jäsennettyä ja perusteltua. Tämä lisäksi menetelmän tarkoituksena on saada aihealueeseen liittyvää aikaisempaa tutkimustietoa, jonka avulla tutkija pystyy tekemään johtopäätöksiä tutkimusaiheeseen ja -kysymykseen liittyen. (Vilkka 2023, luku 1.) Tässä työssä menetelmä mahdollisti algoritmisen ajattelun kokonaisvaltaisen käsityksen muodostamisen ohjelmoinnin tehokkuuteen ja siinä ilmeneviin tekijöihin liittyen. Tämän lisäksi kuvailevan kirjallisuuskatsauksen avulla pystyttiin selvittämään tarkemmin, miten eri tutkimukset tukevat algoritmisen ajattelun yleistä merkitystä ohjelmoinnissa sekä mitä eroavaisuuksia niiden välillä voi mahdollisesti olla.

Kuvailevan kirjallisuuskatsauksen voidaan katsoa koostuvan kolmesta eri vaiheesta. Ensimmäinen vaihe pitää sisällään aineiston aihepiirin ja tutkimuskysymysten tai -kysymysten muodostamisen valitusta aiheesta sekä tiedonhaun suunnitelman. Seuraavana vaiheena on aineiston tunnistaminen ja laadunarviointi, jonka tarkoituksena on löytää aineisto, joka vastaa tutkimuksen alussa määritettyyn tutkimuskysymykseen. Lisäksi samaiseen aiheeseen kuuluvat valitun aineiston analyysi ja tulokset. Viimeisenä, kolmantena vaiheena on tutkimusaiheesta saadun raportin kirjoittaminen ja sen julkaiseminen. (Vilkka 2023, luku 1.) Lähdetään seuraavaksi katsomaan edellä kuvattuihin vaiheisiin liittyen työn prosessia käytännössä.

5.2 Tiedonhaku ja aineiston valinta

Tämän työn tutkimusaineiston tiedonhaku suoritettiin 15.7. – 31.7.2023 välisenä aikana. Aineiston keruussa käytettiin hyväksi kolmea eri tietokantaa, jotka olivat Janet Finna, SpringerLink ja ScienceDirect. Näiden edellä valittujen tietokantojen valintaperusteena toimi niiden laajuus, sillä siten tarvittavaa tietoa oli helpompi löytää aihealueeseen liittyen. Työn tietojenkäsittelytieteellisen näkökulman myötä kohderyhmänä oli tietojenkäsittelytieteeseen liittyvät artikkelit, jolloin muiden alojen aineisto jätettiin huomiotta. Tämän pohjalta tutkimustyön keskeisimmiksi käsitteiksi muodostuivat ohjelmoinnillinen ajattelu, algoritminen ajattelu, ohjelmointiosaaminen ja ohjelmoinnin tehokkuus.

Kuvailevaan kirjallisuuskatsaukseen valittaviin artikkeleihin asetettiin sisäänotto- ja poissulkukriteereitä, jotka on kuvattu tarkemmin taulukossa 1. Aineistoon valittiin tieteellisiä tutkimusartikkeleita, jotka oli julkaistu vuosien 2013–2023 välillä. Työn teoriaa kirjoittaessa oli selvää, että suurin

osa aiheeseen liittyvästä kirjallisuudesta olisi englanninkielisiä. Tämän pohjalta aineistoon hyväksyttiin vain englanninkielisiä artikkeleja. Lisäksi työhön valittiin vain artikkelit, joissa koko artikkeli oli saatavilla. Aineistoon valittujen artikkelien tuli myös käsitellä algoritmista tai ohjelmoinnillista ajattelua ja ohjelmointiosaamista tai tehokasta ohjelmointia. Muita opinnäytetöitä tai Pro Gradu -tutkielmia ei hyväksytty tutkimusaineistoksi, sillä tutkimustyön tarkoituksena oli selvittää olemassa olevaa akateemisempaa tutkimusta valitusta aiheesta.

Taulukko 1. Työn sisäänotto- ja poissulkukriteerit

Sisäänottokriteerit	**Poissulkukriteerit**
Englanninkielinen artikkeli	Muut kuin englanninkieleinen artikkeli
Julkaisuvuosi on välillä 2013-2023	Julkaisuvuosi on 2012 tai aikaisemmin
Koko artikkeli on saatavilla	Koko artikkeli ei ole saatavilla
Tieteellinen artikkeli	Ei ole tieteellinen artikkeli
Artikkeli käsittelee algoritmista tai ohjelmoinnillista ajattelua	
Artikkeli käsittelee ohjelmointitaitoja tai tehokasta ohjelmointia	

Edellä kuvattuihin työn keskeisiin käsitteisiin pohjautuen kehitettiin hakusanat, varsinaista tiedonhakua varten. Hakusanat määräytyivät teoreettisen viitekehyksen ja työn alussa laaditun tutkimuskysymyksen perusteella. Hakusanat muodostuivat sanoista algorithmic thinking, computational thinking, effective programming ja programming skills. Aineiston varsinaista etsintää varten hakusanat yhdisteltiin edellä kuvatuissa tietokannoissa käyttämällä AND ja OR sanoja. Tietokantojen lisäksi tutkimusaineistoon etsittiin artikkeleja Google Scholarin avulla manuaalisesti. Tiedonhaun hakusanat ja tulokset esitelty tarkemmin taulukossa 2.

Taulukko 2. Tiedonhaun tulokset eri tietokannoissa

Hakulauseke	**Tuloksia yht.**	**Otsikon mukaan valitut**	**Tiivistelmän mukaan valitut**	**Koko artikkelin perusteella valitut**
Janet Finna				
("algorithmic thinking" OR "computational thinking") AND ("effective programming" OR "programming skills")	497	13	6	3
ScienceDirect				
("algorithmic thinking" OR "computational thinking") AND ("programming skills" OR "effective programming")	44	3	0	0
("algorithmic thinking" OR "computational thinking") AND ("computing skills")	4	0	0	0
SpringerLink				
("algorithmic thinking" OR "computational thinking") AND ("programming skills" OR "effective programming")	98	3	1	1

Kirjallisuuskatsauksen analysointiin valitun aineiston keskeinen valintaperuste oli artikkelien osuvuus tutkimuskysymyksen kannalta. Tämän pohjalta tiedonhaun tulokset käytiin läpi systemaattisesti niin artikkelien otsikoiden, tiivistelmän kuin koko tekstin osalta. Tiedonhaun tulokset aloitettiin käymällä läpi otsikot, jotka liittyivät tutkimuskysymykseen. Mikäli otsikko kuvasi tutkittavaa aihetta siirryttiin tutkimusartikkelin tiivistelmään. Tiivistelmän pohjalta taas luettiin artikkeli. Jos aineisto sisälsi algoritmisen ajattelun tai ohjelmoinnillisen ajattelun ja tehokkaan ohjelmoinnin tai ohjelmointiosaamisen näkökulmat, sisällytettiin artikkeli tutkimusaineistoon. Lopullinen aineiston analysointi toteutettiin kuudesta artikkelista, joista kaksi oli Google Scholarin artikkeleja. Kaikki aineiston analysointiin valitut artikkelit ovat kuvattuna tarkemmin taulukossa 3.

Taulukko 3. Kirjallisuuskatsauksen aineiston kuvaus

Kirjoittaja(t) Julkaisuvuosi	**Julkaisun nimi**	**Mitä julkaisussa tutkittiin**	**Keskeiset tulokset**
Boom, K. ym. 2022	Relationships between computational thinking and the quality of computer programs	Selvitettiin algoritmisen ajattelun ja ohjelmoinnin laadun välistä suhdetta.	Algoritminen ajattelu edisti ohjelmoinnin laatua.
Türker, P.ym. 2020	A Study on Students' Computational Thinking Skills and Self-Efficacy of Block-Based Programming	Tutkittiin algoritmisen ajattelun taitojen merkitystä ohjelmoinnissa.	Algoritminen ajattelu tuki yksilön ohjelmointitaitoja.
Kılıç, S. ym. 2020	A Valid and Reliable Scale for Developing Programming-Oriented Computational Thinking	Tutkittiin algoritmista ajattelua ja ohjelmoinnin taitojen välistä merkitystä.	Ohjelmointikokemuksella on merkittävä vaikutus opiskelijoiden algoritmisen ajattelun taitamiseen.
Agbo F. ym. 2019	A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions	Tutkittiin miten algoritmista ajattelua on käytetty ohjelmoinnin opetuksessa.	Algoritmisen ajattelun käyttö voi tukea ohjelmoinnin oppimista monipuolisesti.
Chen, G. 2017	Programming Language Teaching Model Based on Computational Thinking and Problem-based Learning	Tutkittiin ohjelmointikielen oppimisen tarkoitusta ja tavoitetta keskittyen algoritmisen ajattelun kehittämiseen osana ohjelmointiopetusta.	Ohjelmointia tulisi opettaa algoritmisen ajattelun näkökulmaa hyödyntäen. Esillä on erityisesti ongelmalähtöisen oppimisen hyödyntäminen osana ohjelmoinnin opetusta.
Li, Y. 2016	Teaching Programming Based on Computational Thinking	Tutkittiin, miten algoritmisen ajattelun keinot auttavat yksilöä hahmottamaan, ymmärtämään ja soveltamaan ohjelmoinnin ongelmia.	Algoritmisen ajattelun sisällyttäminen opetukseen auttaa kehittämään opiskelijoiden ajattelu- ja ongelmanratkaisua.

Kuvailevan kirjallisuuskatsauksen hyötynä on se, että aineisto saa olla laaja. Lisäksi tutkimusaineistoon ei tarvitse käyttää kovinkaan tarkkaa seulaa, mikä mahdollistaa suuremman otoksen keräämisen. Tämä mahdollistaa myös sen, että aihetta voidaan tarkastella monipuolisesti eri näkökulmista. (Vilkka 2023, luku 2.) Eri näkökulmien mahdollistaminen ja vertailu ja analysointi ovatkin kuvailevan kirjallisuuskatsauksen yksi hyödyistä. Lisäksi tämän työn kannalta se mahdollisti tutkimustiedon kokonaisvaltaisemman tarkastelun ja analyysin aihealueeseen liittyen. Kuvataan seuraavaksi tarkemmin työhön valittua analyysimenetelmää ja siihen liittyviä toimintoja.

5.3 Aineiston kuvaaminen ja analysointi

Tutkimustyön tutkimusaineiston analysointi on tärkeä vaihe kuvailevassa kirjallisuuskatsauksessa. Kuvailevassa kirjallisuuskatsauksessa analysointivaiheessa aineistosta etsitään tutkimuksen kannalta olennaisia asioita (Kangasniemi, Utriainen, Ahonen, Pietilä, Jääskeläinen & Liikanen 2013, 297). Analysoinnin avulla vastataan tutkimuksen alussa määritettyyn tutkimuskysymykseen valitun tutkimusaineiston pohjalta. Lisäksi tutkimusaineistosta voidaan etsiä eri tutkimuksia yhdistäviä ja eriäviä näkökulmia sekä pohtia syitä, mistä tutkijoiden näkökulmat tutkimuskysymykseen johtuvat. Analysointi voidaan tuloksien pohjalta toteuttaa monella eri tavalla. Tärkeää on, että aineistoista löytyvät olennaiset asiat jaotellaan tarkemmin sisällön mukaisesti eri kokonaisuuksiksi. (Vilkka 2023, luku 2.)

Tutkimuksen analyysissä tarkoituksena ei ole raportoida tai referoida tutkimusaineistoa, vaan vertailla, analysoida ja tehdä päätelmiä aineiston eri artikkeleiden välillä. Tutkimusaineistoiden tietoja yhdistetään ja kuvaillaan sekä sen pohjalta valittua aineistoa analysoidaan kriittisesti. (Kangasniemi ym. 2013, 295–297.) Tässä työssä tutkimusaineiston analyysi toteutettiin laadullisesti teemojen avulla. Artikkelit luettiin läpi tarkasti ja niistä esiin nousseet aiheet kirjattiin erikseen ylös. Tämän jälkeen aiheita oli helpompi yhdistää suuremmiksi kokonaisuuksiksi. Apuna käytettiin taulukkoa, sillä sen pohjalta suuremmat teemat oli helpompi jakaa erilaisiin ryhmiin. Lisäksi taulukointi mahdollisti aineistossa esiin nousseiden yhtäläisyyksien ja eroavaisuuksien paremman tunnistamisen ja vertailun. Tutkimuksen aineiston analysointi toteutettiin syys- ja lokakuun 2023 aikana.

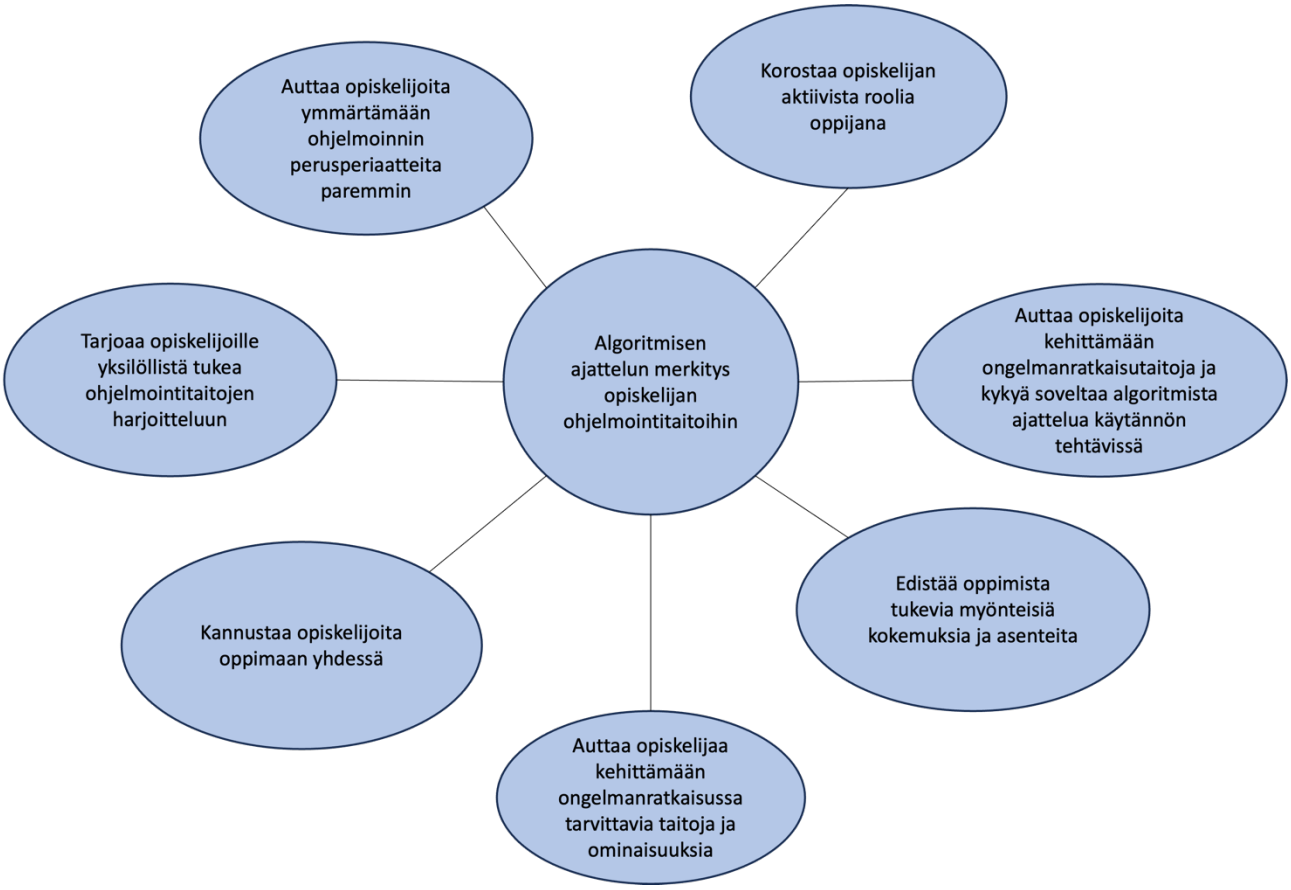
5.4 Työn eettisyyden periaatteet

Työn jokaisessa vaiheessa noudatettiin hyviä eettisyyden ja luotettavuuden periaatteita. Tutkimusetiikalla kuvataan tutkimuksen tieteellisen käytännön oikeanlaista noudattamista, joka pitää sisällään säännöt, miten jokaisen tutkimuksen tekijän tulee toimia tutkimusprosessin eri vaiheissa. Säännöt kattavat koko tutkimuksen prosessin ja koskevat niin tutkimusmenetelmän valintaa, tiedonhankintaa ja tutkimustulosten hallintaa kuin analysointia sekä raportointia. Edellä mainittujen tekijöiden lisäksi tutkimuksen tekijän tulee toimia rehellisesti ja huolellisesti tutkimuksen jokaisessa vaiheessa. (Vilkka 2023, luku 3.) Tutkimusetiikan voidaan katsoa vaikuttavan tutkimuksen luotettavuuteen, sillä hyvin toteutettu työ on laadukasta ja luotettavaa. Luotettavuuteen vaikuttavat siten koko tutkimuksen prosessi, alusta loppuun asti (Karjalainen 2015, 16). Työn eettisyys vaikuttaakin työn laatuun ja luotettavuuteen.

6 Tulokset

6.1 Keskeiset tulokset

Tutkimusaineiston analysoinnin tuloksena ilmeni useita keskeisiä teemoja ja havaintoja, jotka auttoivat ymmärtämään paremmin algoritmisen ajattelun merkitystä ohjelmoinnin taitamisessa. Valitut aineistoon sisällytetyt artikkelit keskittyivät ohjelmointikoulutuksen ja oppimisen konteksteihin, jonka myötä tuloksissa keskityttiin tarkastelemaan tätä yhteyttä. Artikkelien pohjalta tutkimustulokset lajiteltiin kolmeen eri teemaan: opiskelijakeskeinen oppiminen, älykkäät oppimisympäristöt sekä ongelmakeskeinen oppiminen. Teemojen tarkastelun myötä oli mahdollista saada parempi ymmärrys siitä, mitkä tekijät edistävät yksilön ohjelmoinnissa tarvittavia taitoja ja työskentelytapoja. Teemojen avulla pystyttiin myös hahmottamaan paremmin, miten algoritminen ajattelu on merkityksellistä ohjelmoinnin sujuvuuden ja tehokkuuden kannalta. Tutkimusaineiston keskeisimmät tulokset ovat kuvattuna tarkemmin kuviossa 1.



Kuvio 1. Algoritmisen ajattelun merkitys opiskelijan ohjelmointitaitoihin

6.2 Opiskelijakeskeinen oppiminen

Algoritmisen ajattelun näkökulman pohjalta oppilaat pääsivät kehittämään ajattelutaitojaan ohjelmoinnin opetuksessa. Sekä Agbo, Oyelere, Suhonen ja Adewumi (2019), Chen (2019, 130), Li (2016) että Türker ja Pala (2019, 28) saivat selville tutkimuksissaan, että algoritmisen ajattelun integroiminen käytännön ohjelmointiharjoitteluun ja ohjelmointitaitojen kehittämiseen on keskeistä, jotta opiskelijat voivat kehittää syvällisempää ymmärrystä ohjelmoinnin toiminnasta. Algoritminen ajattelu tarjoaa ohjelmoinnin opetukseen uudenlaisia työkaluja ja kannustaa luovaan ongelmanratkaisuun. Lisäksi se antaa oppilaille valmiuksia soveltaa oppimiaan taitoja erilaisiin käytännön tilanteisiin. Myös Boom ja muut (2022) toivat tuloksissaan esiin, että algoritmisen ajattelun lisääminen ohjelmointiopetukseen on yhteydessä ohjelmoinnin laatuun. Opiskelijakeskeisen näkökulman sisällyttäminen ohjelmointiopiskeluun tuki tulosten mukaan ohjelmointitaitojen parempaa hallitsemista. (Boom ym. 2022, 8305–8306.) Tulosten perusteella voidaan todeta, että opiskelijakeskeinen oppiminen tarjoaa tehokkaan tavan tukea ja edistää yksilön ohjelmointiprosessin sisäistämistä erilaisin harjoittelun keinoin. Lisäksi se mahdollistaa erilaisten oppimistyylien sisäistämisen

ohjelmointiopetukseen ja sen harjoittamiseen. Ei pelkästään riitä, että tietää, mitä algoritminen ajattelu pitää yleisesti sisällään, yksilön on kyettävä soveltamaan algoritmista ajattelua käytännön harjoittelussa.

Ohjelmointikoulutuksen merkitys korostuu paljon opiskelijakeskeisessä oppimisessa. Sekä Li (2016) että Chen (2017, 130) toivat esiin sen, että ohjelmointiopetuksen tavoitteena ei ole pelkästään ymmärtää ohjelmointikieltä, vaan opettaa ohjelmointia algoritmisen ajattelun näkökulmasta. Algoritmisen ajattelun opettaminen auttaa opiskelijoita hahmottamaan, miten monimutkaiset ongelmat pystytään hajottaa yksinkertaisempiin vaiheisiin sekä miten ne voitaisiin ratkaista loogisesti vaihe vaiheelta. Sen sijaan, että opiskelijat nähdään passiivisina, heitä kannustetaan osallistumaan aktiivisesti oppimisprosessiin: kysymään kysymyksiä, tutkimaan sekä soveltamaan oppimaansa. Tämä muutos opetusmenetelmissä antaa opiskelijoille mahdollisuuden kehittää kriittistä ajatteluaan, luovia ongelmanratkaisutaitojaan sekä sisäistämään algoritmisen ajatteluun liitettyjä ominaisuuksia. (Li 2016.) Myös Chen (2017) toi esiin opiskelijoiden aktiivisen roolin ohjelmoinnissa. Tulokset osoittivat, että ohjelmointikoulutuksen kehittäminen algoritmisen ajattelun näkökulmasta edistää opiskelijoiden oppimiskykyä ja ohjelmoinnin taitoja. (Chen 2017, 30.) Algoritmisen ajattelun vahvistaminen varhaisessa vaiheessa koulutusta valmistaa opiskelijoita menestymään paremmin tulevaisuuden ohjelmointihaasteissa.

6.3 Älykkäät oppimisympäristöt

Algoritminen ajattelu on avannut uudenlaisia mahdollisuuksia ohjelmoinnin opetukseen. Esiin ovat nousseet älykkäät oppimisympäristöt, joiden avulla opiskelijat voivat harjoitella ja kehittää ohjelmointitaitojaan interaktiivisella ja tehokkaalla tavalla. Agbo, Oyelere, Suhonen ja Adewumi (2019) toivat tutkimuksessaan esiin älykkäiden oppimisympäristöjen merkityksen ohjelmointitaitoja tukevana voimavarana. Älykkäiden oppimisympäristöjen tarkoituksena on tehostaa opiskelijan oppimista ja edistää yksilöllisiä ratkaisuja ohjelmoinnin oppimisen haasteissa. Lisäksi ne tarjoavat mahdollisuuden hahmottaa ohjelmointitaitojen oleelliset osa-alueet personoidun, mukautuvan ja motivoivan lähestymistavan kautta. Tällainen yksilöllinen tuki voi tulosten mukaan johtaa parempiin oppimistuloksiin ja auttaa ohjelmoinnin opetuksen tehokkaammassa toteuttamisessa. Lisäksi älykkäiden oppimisympäristöjen katsottiin tasoittavan ohjelmointitaitojen eroa opiskelijoiden välillä, joilla on aiempaa ohjelmointitaustaa ja niillä, joilla ei ollut. (Agbo ym. 2019.) Älykkäät oppimis-

ympäristöt tarjoavat siten opiskelijalle yksilöllisiä mahdollisuuksia oppia ja harjoitella ohjelmoinnissa tarvittavia taitoja. Lisäksi ohjelmointitaitojen käytännönharjoittelu auttaa opiskelijoita sisäistämään, soveltamaan ja kehittämään algoritmisen ajattelun ominaisuuksia. Tutkimusaineistossa mainittuina oppimisympäristöinä olivat muun muassa Scratch, mBot, Arduino ja MATLAB. Myös muita oppimisympäristöjä nousi esiin aineiston analysoinnin yhteydessä.

Tulokset osoittivat myös, että älykkäät oppimisympäristöt tukevat ohjelmoinnissa tarvittavia taitoja ja kykyjä sekä edistävät myönteisiä tunteita, joita tarvitaan ohjelmoinnin harjoittelussa. Sekä Türker ja muut (2019, 26–27) että Agbo ja muut (2019) saivat selville tutkimuksessaan, että älykkäiden oppimisympäristöjen sisällyttäminen opiskelijoiden ohjelmointiopetukseen tukee opiskelijoiden ohjelmoinnissa tarvittavia kognitiivisia kykyjä sekä lisäävät myönteisesti heidän ongelmanratkaisutaitojen kehittymistä. Tutkimukset ja havainnot osoittivat myös, että älykkäät oppimisympäristöt voivat vaikuttaa merkittävästi opiskelijan ohjelmointitaitojen kehittymiseen sekä lisätä myönteisiä tunteita itse harjoittelua kohtaan.

Türkerin ja muiden (2019, 27) sekä Agbon ja muiden (2019) tutkimustulokset osoittivat, että algoritmisen ajattelun harjoittaminen älykkäiden oppimisympäristöjen avulla ohjelmointikoulutuksessa on ollut kiinnostavaa ja hyödyllistä. Osa opiskelijoista kuitenkin koki harjoittelun tylsänä ja ahdistavana (Agbo ym. 2019). Myös Kilicin ja muiden (2021, 27) tutkimus osoitti eroja käytännön ohjelmointiharjoittelussa, vaikka yleisesti erot katsottiin liittyvän opiskelijan haluttomuuteen tai heikompaan itseluottamukseen omia ohjelmointitaitoja kohtaan. Ohjelmoinnin eri haasteet voivat vaikuttaa yksilön motivaatioon oppia ohjelmointia, mikä puolestaan vaikuttaa harjoittelun sujumiseen. Myönteiset tunteet ja motivoituminen ovat tärkeitä tekijöitä oppimisprosessissa, sillä ne voivat vaikuttaa suuresti opiskelijoiden sitoutumiseen sekä kykyyn sisäistää opittavaa tietoa. Opetuksessa olisi hyvä huomioida opiskelijoiden aikaisempi kokemus ja tarvittaessa tarjota ohjelmointia tukevia algoritmisen ajattelun keinoja ohjelmointitaitojen vahvistamiseen. Tämä korostuu ehkä enemmän juuri opiskelijoilla, joilla on vähemmän ohjelmointikokemusta takana. On tärkeää, että jokainen kokee ohjelmoinnin harjoittelun mielekkäänä, vaikka yksilöllisiä eroavaisuuksia esiintyisikin.

6.4 Ongelmakeskeinen oppiminen

Algoritmisen ajattelun pohjalta esiin on noussut erilaisia tapoja parantaa opiskelijoiden käytännön ongelmanratkaisutaitoja osana ohjelmoinnin opetusta. Algoritmisen ajattelun integroiminen käytännön ohjelmointiharjoitteluun ja ohjelmointitaitojen kehittämiseen auttoi opiskelijoita hahmottamaan paremmin ohjelmoinnin perusperiaatteita ja syventämään ymmärrystään tietokoneohjelmoinnin logiikasta. Sekä Chen (2017, 128–131), Li (2016) että Kilic ja muut (2021) korostivat tutkimuksissaan käytännön ongelmaratkaisutaitojen kehittämistä osana ohjelmointiopetusta. Chen (2017) toi esille ongelmalähtöisen oppimisen merkityksen. Tämä lähestymistapa ei keskity vain tiettyjen ongelmien ratkaisemiseen, vaan korostaa taitoja ja ominaisuuksia, joita yksilö tarvitsee ongelmanratkaisussa. Ongelmalähtöinen oppiminen kannustaa oppimaan yhteisöllisessä ympäristössä, joka edistää tiedon jakamista, yhteistyötä sekä eri näkökulmien huomioon ottamista. (Chen 2017, 128–131.) Tutkimustulokset osoittivat, että ohjelmoinnin oppiminen ei rajoitu vain syntaksin hallintaan tai valmiiden ratkaisujen opetteluun. Lähestymistapa kannustaa opiskelijaa syventämään ymmärrystään ongelmien taustalla olevista käsitteistä ja kehittämään joustavia ajattelutapoja. Tämän pohjalta voidaan todeta, että ongelmalähtöinen oppiminen korostaa opiskelijoiden aktiivista roolia ongelmien ratkaisemisessa. Yhteisöllisyys ja mahdollisuus kehittää ohjelmoinnin toimintatapoja yhdessä tarjoaa erilaisia näkökulmia ja tapoja ratkaista ongelmia.

Agbo ja muut (2019) sekä Li (2016) toivat tutkimuksessaan myös esiin ongelmakeskeisen oppimisen osana ohjelmointitaitojen kehittämistä. Aikaisemmin ohjelmointikurssit keskittyivät ohjelmoinnin teknisiin taitoihin, eikä opetuksessa kiinnitetty huomiota ajattelutaitojen kehittämiseen ja edistämiseen (Li 2016). Algoritmisen ajattelun sisällyttäminen opetukseen tarjoaa mahdollisuuden harjoittaa eri ajattelumalleja, jotka edistävät käytännön ongelmanratkaisua sekä ohjelmointitaitojen kehittymistä (Li 2016; Agbo ym. 2019). Algoritmisen ajattelun taitoja tulisi siten soveltaa ja harjoitella aktiivisesti ohjelmoinnin kursseilla (Li 2016). Harjoitukset voivat Agbon ja muiden (2019) mukaan pitää sisällään erilaisia pulmapelejä, jotka tukevat opiskelijan abstraktia ajattelua ja loogista päättelykykyä. Myös käytännön ohjelmointiharjoittelu nousi esiin monissa tutkimuksissa. Yleisesti ongelmanratkaisutaitojen kehittäminen ja harjoittaminen opettaa opiskelijoita soveltamaan algoritmisen ajattelun taitoja käytännön tehtävissä, sekä kehittämään kykyä löytää innovatiivisia ratkaisuja monimutkaisiin haasteisiin.

Toisaalta, vaikka ongelmakeskeinen oppiminen on tärkeää, tulee siinä huomioida yksilölliset haasteet ja oppimisprosessia tukevat ominaisuudet. Chen (2017, 130) toi esiin, että ongelmien ei tule olla liian haastavia eikä myöskään liian yksinkertaisia. Liian yksinkertaiset ongelmat eivät tue opiskelijoiden kehitystä tarpeeksi, kun taas liian haastavat ongelmat voivat johtaa muun muassa turhautumisen sekä epäonnistumisen tunteisiin. Tämän myötä, ihanteellinen oppimistilanne syntyy, kun opiskelijat saavat ratkaistavakseen juuri sopivan tasoisia haasteita, jotka edistävät oppimista ja tarjoavat mahdollisuuden onnistumisen tunteisiin sekä omien taitojen kehittämiseen. Tällaiset tehtävät rohkaisevat opiskelijoita kehittämään ongelmanratkaisutaitojaan ja luovuuttaan, samalla ylläpitäen heidän motivaatiotansa ja oppimisen iloa ohjelmoinnin eri haasteita kohtaan. Boom ja muut (2022) toivat tutkimuksessaan esiin kuitenkin sen, että ongelmanratkaisun ei välttämättä tarvinnut olla oikea. Riittää, että opiskelija käyttää algoritmisen ajattelun osa-alueita ja pyrkii löytämään oikean ratkaisun ongelmaan. Ongelman syvällinen pohdinta ja virheellisten ratkaisujen tutkimisen katsottiin edistävän opiskelijan oppimisprosessia. (Boom ym. 2022, 8304.) Tulokset korostavat siten opiskelijoiden halua ja intoa kokeilla ja harjoitella käytännön ohjelmointia. Itse harjoittelu näyttäytyy opiskelijan algoritmista ajattelua kehittävänä toimintona.

6.5 Johtopäätökset

Ohjelmointitaitojen sisäistäminen ei ole synnynnäinen kyky, vaan opittavissa ja kehitettävissä oleva taito. Analysoidun tutkimusaineiston tulokset osoittivat, että algoritmisen ajattelun sisällyttäminen ohjelmointiopetukseen ja -koulutukseen auttaa opiskelijoita kehittämään ohjelmointia tukevia ajattelutaitoja, joiden avulla yksilöt pärjäävät paremmin ohjelmointiin liittyvissä haasteissa. Ohjelmointitaitojen oppiminen ja harjoittelu voi tapahtua erilaisin algoritmisen ajattelun menetelmin, kuten käytännön ohjelmointiharjoitusten, ongelmakeskeisen oppimisen tai erilaisten älykkäiden oppimisympäristöjen avulla. Edellä kuvattujen algoritmisen ajattelun menetelmien pohjalta esiin nousi ennen kaikkea opiskelijakeskeinen näkökulma, joka korosti yksilön aktiivista roolia oppijana. Kiinnittämällä huomiota opettajan rooliin, oppimista tukeviin työkaluihin, opiskelijat kykenevät hahmottaman algoritmisessa ajattelussa tarvittavat eri menetelmät. Lisäksi he saavat tukea ja apua oman osaamisensa mukaisesti.

Käytännön harjoittelu ja aktiivinen itsensä kehittäminen näkyi vahvasti tutkimustuloksissa. Älykkäät oppimisympäristöt tarjosivat innovatiivisen tavan kehittää ohjelmoinnissa tarvittavia taitoja.

Tällainen yksilöllinen tuki näyttää auttavan opiskelijoita saavuttamaan vahvemman ohjelmointiosaamisen, erityisesti niiden välillä, joilla on vähemmän taustaa ohjelmoinnissa. Lisäksi oppimismenetelmä tuki opiskelijoiden kykyä kehittää luovuutta, ongelmanratkaisutaitoja sekä ymmärrystä ohjelmoinnin eri osa-alueita kohtaan. Vaikka yleisesti tutkimukset tukivat älykkäiden oppimisympäristöjen myönteisiä vaikutuksia opiskelijan asenteisiin ohjelmointia kohtaan, nousi esiin myös kielteisiä kokemuksia oppimisympäristöjä ja yleisesti ohjelmoinnin harjoittelua kohtaan. Ongelmakeskeinen harjoittelu puolestaan korosti ohjelmoinnin eri toimintojen ymmärtämisen tärkeyttä käytännön toteutuksessa. Tämän pohjalta muun muassa käytännön ongelmanratkaisutaitojen harjoittaminen algoritmisen ajattelun pohjalta näytti olevan merkittävää ohjelmointikoulutuksen näkökulmasta. Taito pilkkoa ongelma pienempiin, helpommin hahmotettaviin osaongelmiin osana harjoittelua loi oppilaille paremmat lähtökohdat ratkaista abstrakteja sekä haastaviakin ohjelmointiongelmia. Lisäksi mahdollisuus kehittää omia ongelmanratkaisutaitoja ja ajattelutaitoja osana ohjelmoinninopetusta tuki oppimisprosessia ja vaikutti myönteisesti yksilön motivaatioon sekä yksilön asenteisiin varsinaista ohjelmointiharjoittelua kohtaan.

Vielä lopuksi, työn tavoitteiden kannalta oli mielenkiintoista huomata, miten paljon algoritmisen ajattelun taidot ja niiden kehittäminen vaikuttavat yksilön ohjelmoinnin taitamiseen ja tehokkuuteen. Tutkimustulokset tarjosivat arvokasta tietoa ohjelmointiopetuksen kehittämiseen sekä opiskelijoiden ammatilliseen ja akateemiseen kehitykseen. Usein ohjelmoinnin opettamisen yhteydessä lähdetään liikkeelle ohjelmointikielen syntaksin ja rakenteiden näkökulmasta. Tämä on tärkeää, mutta ohjelmoinnin syvällisemmän ymmärtämisen pohjana on algoritmisen ajattelun osa-alueiden sisäistäminen. Algoritminen ajattelu vaatiikin siten opettelua ja erilaisten taitojen sisäistämistä ohjelmoinnin teknisten taitojen lisäksi. Kokonaisuudessaan tutkimustulokset tukivat ajatusta siitä, että algoritmisen ajattelun taito edistää ja tukee yksilön ohjelmointitaitoja ja niiden kehittymistä. Tutkimustulosten pohjalta voidaan myös todeta, että algoritminen ajattelu kiinnitti huomiota ennen kaikkea opiskelijan ajattelu- ja ongelmanratkaisutaitojen kehittämiseen teknisten ohjelmointitaitojen sijasta sekä tarjosi useita yksilöllisiä ja innovatiivisia ratkaisuja ohjelmoinnin eri haasteisiin.

7 Pohdinta

7.1 Keskeiset huomiot työn toteutuksesta

Tämän työn tarkoituksena oli selvittää, miten algoritminen ajattelu edistää ohjelmoinnin sujuvuutta ja tehokkuutta. Tavoitteena oli tuoda esiin algoritmisen ajattelun näkökulmia ja niiden vaikutuksia ohjelmointiin liittyen: miten eri näkökulmat tukevat tai eroavat aineistojen välillä. Aihetta lähdettiin tutkimaan tarkemmin kuvailevan kirjallisuuskatsauksen avulla. Tutkimusmenetelmän avulla oli mahdollista saada tarkempaa tietoa algoritmisen ajattelun merkityksestä ohjelmointiin ja sitä edistäviin ominaisuuksiin.

Kuvailevan kirjallisuuskatsauksen avulla löytyi kuusi keskeistä artikkelia, jotka tarjosivat arvokasta tietoa tutkimusaiheen ymmärtämiseen ja sitä koskevan tutkimuskysymyksen selvittämiseen. Tutkimusaineiston keräämisessä käytettiin sekä tietokantahakuja että manuaalista hakua. Jo työn alkuvaiheessa tuli esiin, että suomenkielinen kirjallisuus algoritmisen ajattelun ja ohjelmointitaitojen yhteydestä on hyvin vähäistä. Tästä syystä tietokantahakujen kohderyhmää päätettiin rajoittaa englanninkielisiin artikkeleihin. Aineiston analyysin perusteella löytyi kolme pääteemaa, jotka keskittyivät algoritmisen ajattelun merkitykseen ohjelmoinnin taitamisessa. Seuraavaksi tarkastellaan tarkemmin tutkimustuloksia ja niihin vaikuttaneita tekijöitä sekä työn merkittävyyttä niin tutkimuksen eettisyyden kuin luotettavuuden näkökulmasta. Lisäksi esitetään työn kannalta mielenkiintoisia jatkotutkimusaiheita.

7.2 Algoritminen ajattelu kehittämässä ohjelmointiopetuksen toimintatapoja

Tutkimustulokset toivat esiin hyvän kuvan siitä, miten algoritmisen ajattelun taitojen hallitseminen ja kehittäminen vaikuttavat yksilön ohjelmointitaitojen muodostumiseen. Kuten työssä tuotiin aikaisemmin esiin, algoritminen ajattelu auttaa yksilöä hahmottamaan laajoja ja monimutkaisia algoritmisia ongelmia, pilkkomaan niitä pienempiin osaongelmiin, tunnistamaan ongelman kannalta olennaiset tiedot sekä analysoimaan ja kehittämään tehokkaita ratkaisuja ongelmien selvittämiseksi (Grover 2017, 34). Algoritminen ajattelu käsittää siten monia ohjelmoinnin sisäistämisen kannalta tärkeitä taitoja, jotka mahdollistavat tehokkaan ongelmanratkaisun osana käytännön toteutusta. Aineistossa ilmenneet tulokset tukivat hyvin edellä kuvattua algoritmisen ajattelun ja ohjelmointitaitojen välistä suhdetta. Teorian ja aineiston analysoinnin pohjalta voidaan todeta, että

hyvät algoritmisen ajattelun taidot ovat keskeinen osa ohjelmoinnin perusperiaatteiden ymmärtämistä. Tässä työssä ohjelmointia tukevia algoritmisen ajattelun menetelminä näyttäytyivät älykkäät oppimisympäristöt sekä ongelmakeskeinen oppiminen ja opiskelijakeskeinen oppiminen. Jokaisessa menetelmässä korostui käytännön harjoittelu.

Älykkäät oppimisympäristöt toivat hyvin esiin ohjelmointitaitoja tukevia huomioita. Tutkimustulosten osalta voitiin huomata, että älykkäät oppimisympäristöt eivät ainoastaan tue opiskelijoiden myönteisiä asenteita ohjelmointia kohtaan, vaan myös innostavat ja tekevät oppimisesta mielenkiintoista. Tulosten kannalta oli merkittävää, että suurin osa oppilaista koki oppimisympäristöjen käytön kiinnostavana ja hyödyllisenä. Vain pieni osa niiden käytön tylsänä ja tehottomana. Älykkäiden oppimisympäristöjen yhtenä hyötynä voidaankin katsoa olevan opiskelijan mahdollisuus harjoitella ohjelmoinnissa tarvittavia ominaisuuksia oman taitotasonsa mukaisesti. Ehkä juuri älykkäiden oppimisympäristöjen kyky mukautua helposti kaiken tasoisille käyttäjille tukee tuloksissa esiin nousseita myönteisiä asenteita. Esiin nousi myös oppimisympäristöjen mahdollistama reaaliaikainen palaute. Välittömän palautteen avulla käyttäjä tiedostaa virheensä ja pystyy aktiivisesti korjaamaan niitä. Nopea palaute motivoi siten opiskelijaa myös kehittämään taitojaan. Voisiko yksinkertaisempi lähtökohta ohjelmointitaitojen kehittämiseen näyttäytyä älykkäiden oppimisympäristöjen hyötynä? On todennäköistä, että älykkäät oppimisympäristöt mahdollistivat oppilaille kestävän tavan oppia ohjelmoinnin perusperiaatteita ja prosesseja tehokkaasti, jolloin myös myönteisyys oppimista kohtaan kasvoi. Tulosten pohjalta esiin nousi Cheahin (2020) tutkimuksessa ilmennyt huomio; myönteiset tunteet edistivät motivaatiota niin ohjelmointia kuin ohjelmoinnin harjoittelua kohtaan. Positiivista tällöin on se, että älykkäiden oppimisympäristöjen merkitys ei näyttäydy vain ohjelmointitaitojen kehittymisessä, vaan kasvattaa yksilön halua oppia ja kehittää ohjelmoinnin taitoja entisestään. Seurauksien voidaan siten katsoa olevan kauaskantoisempia.

Ongelmakeskeinen oppiminen tuli laajasti esiin tutkimusaineistossa. Algoritmisen ajattelun avulla oppilaat pystyivät syventämään ymmärrystään ohjelmoinnin toiminnasta ja kehittämään omia ajattelu- ja ongelmanratkaisutaitojaan joko itsenäisesti tai yhdessä eri kokoisissa ryhmissä. Tuloksissa korostuikin juuri muiden tuki ja yhdessä tekeminen; oppiminen ja harjoittelu nähtiin sosiaalisena prosessina, jossa oppilaat saivat tukea ja apua toisiltaan. Siten muun muassa ongelmien pohtiminen yhdessä, ajatusten jakaminen ja neuvojen kysyminen auttavat oppilaita kehittämään

ohjelmoinnissa tarvittavia taitoja ja ominaisuuksia sekä keskustelemaan vastaantulevista haasteista. Tulosten pohjalta näyttäisi siltä, että ongelmakeskeinen lähestymistapa kannustaa opiskelijaa syventämään ymmärrystään ongelmien taustalla olevista käsitteistä sekä kehittämään joustavia ajattelutapoja ja ratkaisuja osana harjoittelua. Harjoittelun merkitys korostuu myös Jenkinsin (2001, 22) tutkimuksessa, jossa hän toi esiin, miten käytännön ohjelmointiharjoittelu tukee ohjelmoinnissa tarvittavia taitoja. Tällöin juuri algoritmisen ajattelun sisällyttäminen osaksi harjoittelua, voidaan katsoa tukevan ohjelmointitaitoja vielä entisestään; oppilaat saavat rakennuspalikoita ja ideoita ongelmanratkaisuun.

Opiskelijakeskeinen oppiminen puolestaan korosti nimensä mukaisesti opiskelijan oppimista tukevia työkaluja ja toimintoja. Tulosten pohjalta opettajan rooli erityisesti korostui. Tämä tuli ilmi myös teorian ja aineiston analysoinnin yhteydessä; ohjelmoinnin prosessien ymmärtämisen periaatteena näkyy asiantuntijuuden rooli ja ohjaus. Opettajan tehtävänä on tukea opiskelijan oppimista ja luoda paremmat olosuhteet ongelmanratkaisuun sekä ohjelmointia tukevien taitojen kehittämiseen (Jenkins 2001, 44). Tulokset toivatkin hyvin esiin, miten algoritminen ajattelu tukee opiskelijan aktiivista roolia opetuksessa. Opettajan tehtävänä oli edistää opiskelijan oppimisprosessia sekä kannustaa häntä ajattelemaan kriittisesti ja itsenäisesti. Tulosten pohjalta nousi myös esiin hyviä huomioita ohjelmointiopetuksen tärkeyteen liittyen; opettamalla miten ongelmien ratkaisussa tulee lähteä liikkeelle sekä miten ongelma saadaan ratkaistuksi algoritmisen ajattelun keinoin antaa opiskelijoille keinoja pärjätä paremmin ohjelmoinnin eri haasteissa. Lisäksi se auttaa heitä ymmärtämään ohjelmoinnin taustalla olevia periaatteita.

Työ toi esiin paljon Jenkinsin (2001, 41) tutkimuksessa esiin nousseita huomioita; ohjelmointiopetuksen katse on kiinnittynyt ohjelmointiprosessin ja sen eri vaiheiden sisäistämiseen teknisten taitojen sijasta. Nykypäivän oppimisympäristö on muuttunut paljon viime vuosikymmenien takaisesta tavasta oppia. Tueksi tarvitaan siten uudenlaisia tapoja opettaa ja oppia ohjelmointia. Tutkimusaineiston tulosten pohjalta algoritmista ajattelua tukevat opetusmenetelmät edistävät opiskelijan ohjelmointiosaamisen syvällisempää hahmottamista. Tämän työn tulokset tarjoavat niin opiskelijoille kuin opettajille paremman ymmärryksen algoritmisen ajattelun vaikutuksista ohjelmointitaitojen kehittymiseen sekä työkaluja sen toteutuksen suunnitteluun. Tutkimustulokset viittaavat siihen, että monipuolinen algoritmisen ajattelun lähestymistapa ohjelmointiopetukseen voi luoda

parhaat tulokset opiskelijoiden ohjelmointitaitojen kehittämiseen. Edellä kuvatun perusteella voidaan myös todeta, että koulutuksessa tulisi panostaa enemmän taitojen opettamiseen, jossa opiskelijoilla on mahdollisuus hyödyntää älykkäitä oppimisympäristöjä sekä ongelmakeskeistä ja opiskelijakeskeistä oppimista. Myös tietoisuus algoritmisen ajattelun merkityksestä sekä sen ominaisuuksista lisää opiskelijoiden omien ajattelutaitojen kehittämistä tukevaa harjoittelua, sillä sen avulla yksilö voi tietoisesti toimia ohjelmointia tukevien toimintojen ja mallien mukaisesti. Tärkeää, on että opiskelijat pääsevät aktiivisesti toteuttamaan algoritmisen ajattelun menetelmiä.

7.3 Tutkimuksen eettisyys ja luotettavuus

Tämän tutkimustyön eettisyyttä ja luotettavuutta edistettiin selkeän ja vaiheittain etenevän tutkimuksen avulla. Tutkimuksen prosessi tuotiin esiin ja siinä esiintyvät vaiheet kuvailtiin tarkasti. Tämän lisäksi tiedonhaku toteutettiin suunnitelmallisesti ja avainsanat oli määritelty etukäteen. Myös aineiston valinta toteutettiin systemaattisesti ja tarkasti; aineistoon perehdyttiin syvällisesti ja siihen valitut artikkelit vastasivat tutkimuskysymykseen. Hyvin suunniteltu tutkimus edisti siten koko tutkimusprosessin luotettavuutta ja sitä kautta myös hyviä eettisyyden periaatteita. Yksi luotettavuutta vähentävä tekijä liittyi sähköisten artikkeleiden käytettävyyteen; osa artikkeleista ei ollut saatavilla kokonaan, jonka myötä niitä ei sisällytetty tutkimukseen. Siten tutkimustulosten kannalta hyödyllistä, yllättävää tai tärkeää tietoa saattoi jäädä huomioimatta.

Toinen luotettavuuteen keskittyvä huomio liittyi teoreettiseen viitekehykseen ja aineiston analysointiin. Algoritmisen ajattelun ja ohjelmoinnin kannalta keskityttiin tehokkaaseen ohjelmointiin ja ohjelmointiosaamiseen, jonka myötä tutkimusaineistoksi valittiin aihealueeseen liittyvät artikkelit. Lisäksi tutkimus vastasi työn teoreettista viitekehystä, mikä tuki siten työn tutkimusosaa, ja siten koko työn eettisyyttä ja luotettavuutta. Tämän lisäksi tutkimusaineisto koostui pelkästään englanninkielisistä aineistosta, minkä myötä huolellinen lukeminen korostui; analysoitava aineisto luettiin moneen otteeseen läpi siinä ilmenevän tiedon luotettavuuden varmistamiseksi. Edellä kuvatun lisäksi aineiston luotettavuutta lisättiin myös artikkelien valinnan avulla. Muutaman artikkelin analysointi ei anna luotettavaa kuvaa tutkimustuloksista. Työn alusta lähtien olikin selvää, että aineiston valinnassa hyödynnetään useampaa tietokantaa luotettavampien tutkimustulosten varmistamiseksi.

Edellä mainittujen tekijöiden lisäksi työn luotettavuutta lisättiin ulkoisten tekijöiden avulla. Ulkoisissa luotettavuuteen sekä eettisyyteen vaikuttavissa tekijöissä huomioidaan aineiston soveltavuutta tutkimukseen (Vilkka 2023, luku 3). Työn prosessin näkökulmasta aineiston valinnassa huomioitiin oikeiden tietokantojen käyttö. Siten voitiin varmistaa, että tutkimusaineisto koostuu relevantista ja asiantuntevasta tietomateriaalista, joka tukee tutkimuskysymyksen ja tavoitteiden tarkastelua aiheen ja tutkimuskysymyksen näkökulmasta. Tämän lisäksi sekä työn teoreettisen viitekehyksen että tutkimusaineiston analysoinnissa teoria ja tulokset esitettiin oikeassa valossa, eikä tieto vääristelty.

7.4 Jatkotutkimustarpeet

Tulokset tarjosivat yleisen näkökulman algoritmisen ajattelun hyötyihin, mutta aihetta olisi myös mielenkiintoista tutkia syvemmin. Tämän pohjalta tuloksia tarkasteltaessa on hyvä huomioida, että negatiiviset asenteet älykkäitä oppimisympäristöjä kohtaan eivät välttämättä kohdistu juuri kokemukseen oppimisympäristöstä. Tuloksiin voivat vaikuttaa monet eri tekijät aikaisemmasta kokemuksesta ja motivaatiosta erilaisiin oppimisen haasteisiin. Tämän pohjalta olisi mielenkiintoista selvittää, mitkä eri tekijät vaikuttavat juuri älykkäiden oppimisympäristöjen negatiivisempiin kokemuksiin. Tämä ei tullut ilmi tuloksissa. Siten oppimisympäristöjä voitaisiin paremmin kehittää tulevaisuutta ajatellen. Tulosten pohjalta huomasin myös Cheahin (2020) tutkimuksessa ilmenneen huomion; myönteiset asenteet ohjelmointiopetusta kohtaan lisäsivät opiskelijan motivaatiota ohjelmointiharjoitteluun. Tämän pohjalta jatkossa olisi kiinnostavaa tutkia tarkemmin myönteisten ohjelmointikokemusten syitä ja seurauksia vielä tarkemmin.

Työn tulosten mukaan tutkimuksessa olisi hyvä kiinnittää lisähuomiota myös itsearvioinnin ja palautteen merkitykseen. Kuinka opiskelijat arvioivat omaa edistymistään algoritmisen ajattelun taitojen kehittyessä? Tällaisen tutkimuksen tulisi ehkä ulottua enemmän yksilön pidempiaikaisempiin seurauksiin. Lopuksi, vaikka tulokset antavat käsityksen siitä, miten opiskelijat kokevat algoritmisen ajattelun opetuksen tietyssä hetkessä, olisi mielenkiintoisa selvittää, miten kyseiset kokemukset vaikuttavat opiskelijoiden taitoihin ja arvioihin pidemmällä aikavälillä. Tämän pohjalta voidaan paremmin arvioida, mitkä erilaiset algoritmisen ajattelun opetusmenetelmät ovat parempia ja tarjoavat pidempiaikaisempaa tukea ohjelmointitaitojen kehittämiseen.

Lähteet

Agbo, F., Oyelere, S., Suhonen, J. & Adewumi, S. 2019. A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions. Viitattu 1.9.2023. https://scholar.google.com, ACM Digital Library.

Alexandron, G., Amori, M., Gordon, M. & Harel, D. 2012. The Effect of Previous Programming Experience on the Learning of Scenario-based Programming. The 12th Koli Calling International Conference on Computing Education Research. Viitattu 14.11.2023. https://www.researchgate.net/publication/258678504_The_Effect_of_Previous_Programming_Experience_on_the_Learning_of_Scenario-based_Programming

Bergin, S. & Reilly, R. 2005. The influence of motivation and comfort-level on learning to program. Sussex University. Viitattu 1.10.2023. http://mural.maynoothuniversity.ie/8685/1/SB-Influence-2005.pdf

Boom, K-D., Bower, M., Siemon, J. & Arguel, A. 2022. Relationships between computational thinking and the quality of computer programs. Education and Information Technologies, 27, 6, 8289-8310. Viitattu 22.7.2023. https://link.springer.com/article/10.1007/s10639-022-10921-z

Bourke, C. 2018. Computer Science 1. University of Nebraska–Lincoln. Viitattu 10.6.2023. https://cse.unl.edu/~cbourke/ComputerScienceOne.pdf

Buckley, J., Archibald, T., Hargraves, M. & Trochim, W. M. 2015. Defining and Teaching Evaluative Thinking: Insights from Research on Critical Thinking. American Journal of Evaluation, 36, 3, 1-14. Viitattu 17.7.2023. https://janet.finna.fi, SAGE Journals Premier.

Cansu, S. K. & Cansu, F. K. 2019. An Overview of Computational Thinking. International Journal of Computer Science Education in Schools 3, 1, 1-11. Viitattu 13.6.2023 https://www.researchgate.net/publication/332724947_An_Overview_of_Computational_Thinking

Cheah, C. S. 2020. Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review. Contemporary Educational Technology, 12, 2. Viitattu 18.10.2023. https://www.cedtech.net/download/factors-contributing-to-the-difficulties-in-teaching-and-learning-of-computer-programming-a-8247.pdf

Chen, C. 2017. Teaching Programming Based on Computational Thinking. Viitattu 1.9.2023. https://scholar.google.com , IEEE Xplore.

Cole, M. J. 2023. Evaluative thinking. Sage Journals, 23, 2, 70–90. Viitattu 22.7.2023. https://janet.finna.fi, SAGE Journals Premier.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. 2022. Introduction to algorithms. 4. uudistettu painos. Cambridge, Massachusetts : The MIT Press. Viitattu 10.6.2023. https://dl.ebooksworld.ir/books/Introduction.to.Algorithms.4th.Leiserson.Stein.Rivest.Cormen.MIT.Press.9780262046305.EBooksWorld.ir.pdf

Grover, s. & Pea, R. D. 2017. Computational Thinking: S Competency Whose Time Has Come. Computer Science Education. London: Bloomsbury Academic 19-38. Viitattu 10.8.2023. https://www.researchgate.net/publication/322104135_Computational_Thinking_A_Competency_Whose_Time_Has_Come

Jenkins, T. 2001. The Motivation of Students of Programming. The University of Kent. Viitattu 26.9.2023. https://kar.kent.ac.uk/13559/1/the_motivation_of_students_jenkins.pdf

Kangasniemi, M., Utriainen, K., Ahonen, S.-M., Pietilä, A.-M., Jääskeläinen, P. & Liikanen, E. 2013. Kuvaileva kirjallisuuskatsaus: eteneminen tutkimuskysymyksestä jäsennettyyn tietoon. Hoitotiede 25, 4, 291–301. Viitattu 12.7.2023. https://janet.finna.fi, Journal.

Karjalainen, L. 2015. Tilastotieteen perusteet. 2. uudistettu painos. Keuruu: Otavan Kirjapaino Oy.

Kilic, S., Gökoglu, S. & Öztürk, M. 2021. A Valid and Reliable Scale for Developing Programming-Oriented Computational Thinking 59, 2, 257-286. Viitattu 1.9.2023. https://janet.finna.fi, SAGE Journals Premier.

Kingsley-Hughes, A. & Kingsley-Hughes, K. 2005. Beginning Programming. Indianapolis, Indiana: Wiley Publishing, Inc. Viitattu 21.9.2023. https://janet.finna.fi, Ebook Central.

Knuth, E. 1997. The Art of Computer Programming. 3. painos. Addison Wesley Longman. Viitattu 15.6.2023. http://broiler.astrometry.net/~kilian/The_Art_of_Computer_Programming%20-%20Vol%201.pdf

Kori, K., Pedaste, M., Leijen, Ä. & Tonisson, E. 2016. The Role of Programming Experience in ICT Students' Learning Motivation and Academic Achievement. International Journal of Information and Education Technology, 6,5, 331-337. Viitattu 2.10.2023. https://www.researchgate.net/profile/Eno-Tonisson/publication/276485030_The_Role_of_Programming_Experience_in_ICT_Students'_Learning_Motivation_and_Academic_Achievement/links/55e15a8708aede0b5730416c/The-Role-of-Programming-Experience-in-ICT-Students-Learning-Motivation-and-Academic-Achievement.pdf

Kramer, J. 2007. Is abstraction the key to computing? Communications of the ACM, 50, 4, 36-42. Viitattu 18.7.2023. https://janet.finna.fi, EBSCOhost Business Source Elite.

Laaksonen, A. 2022. Tietorakenteet ja algoritmit. Helsingin yliopisto. Viitattu 8.6.2023. https://www.cs.helsinki.fi/u/ahslaaks/tirakirja/

Li, Y. 2016. Teaching Programming Based on Computational Thinking. Viitattu 1.9.2023. https://janet.finna.fi, IEEE Xplore.

Lokkila, E., Rjala, T., Veersamy, A., Enges-Pyykkönen, P., Laakso, M.-J. & Salakoski, T. 2016. How Students' Programming Process Differs from Experts – A Case Study with a Robot ProgRAMMING Exercise. Turun yliopisto. Viitattu 28.9.2023. https://www.utupub.fi/bitstream/handle/10024/170869/Robot_paper.pdf?sequence=1

Meyer, J. 2018. Programming 101: The How and Why of Programming Revealed Using the Processinf Programming Language. New York: Apress. Viitattu 23.6.2023. https://janet.finna.fi, Skillport platform.

Pietiläinen, K. 2021. Algoritmit. Helsinki: Terra Cognita.

Rich, P. J., Egan, G. & Ellsworth, J. 2019. A Framework for Decomposition in Computational Thinking. Association for Computing Machinery 416-421. Viitattu 31.8.2023. https://www.researchgate.net/profile/Peter-Rich/publication/334579725_A_Framework_for_Decomposition_in_Computational_Thinking/links/5d52b9ef92851c93b62c3075/A-Framework-for-Decomposition-in-Computational-Thinking.pdf

Roberts, P. 2009. Abstract thinking: a predictor of modelling ability? International Conference on Model Driven Engineering Languages and Systems: Educators' Symposium. Denver, Colorado, USA. Viitattu 23.7.2023. https://www.cs.colostate.edu/models09/eduPapers/1_RobertsFinal.pdf

Rusanen, A.-M. 2021. Algoritmit aakkoset. Älykäs huominen: Miten tekoäly ja digitalisaatio muuttavat maailmaa? Toim. P. Myllymäki, A.-M. Rusanen, H. Toivonen, L. Ruotsalainen, S. Tarkoma, H. Niemi, H. Niemi, S. Martikainen, K. Saarkivi & M. Huotilainen. Helsinki: Gaudeamus 33–39. Viitattu 10.6.2023. https://janet.finna.fi, Ellibs.

Salminen, A. 2011. Mikä kirjallisuuskatsaus. Johdatus kirjallisuuskatsauksen tyyppeihin ja hallintotieteellisiin sovelluksiin. Vaasan yliopisto. Viitattu 27.6.2023. https://www.uwasa.fi/materiaali/pdf/isbn_978-952-476-349-3.pdf.

Selby, C. & Woollard, J. 2013. Computational thinking: the developing definition. Special Interest Group on Computer Science Education (SIGCSE) 2014. Viitattu 12.6.2023. https://www.researchgate.net/publication/299450690_Computational_thinking_the_developing_definition

Siegmund, J., Kästner, C., Liebig, J., Apel, S. & Hanenberg, S. 2013. Measuring and modeling programming experience, 19, 5, 1299-1334. Viitattu 14.11.2023. https://janet.finna.fi, ProQuest Central.

Türker, P. & Pala, F. 2020. A Study on Students' Computational Thinking Skills and Self-Efficacy of Block-Based Programming, 15, 3, 18. Viitattu 1.9.2023. https://janet.finna.fi, ProQuest Central.

Ukkonen, E. 2003. Mihin algoritmeja tarvitaan? Tieteessä tapahtuu 21,7. Viitattu 12.7.2023. https://janet.finna.fi, Journal.

Vilkka, H. 2023. Kirjallisuuskatsaus metodina, opinnäytetyön osana ja tekstilajina. Helsinki: Art House Oy. Viitattu 16.7.2023. https://janet.finna.fi, Ellibs.

Wing, J. M. 2006. Computational Thinking. Communications of the ACM 49,3, 33-35. Viitattu 13.6.2023. https://janet.finna.fi, EBSCOhost Business Source Elite.

Wing, J. M. 2011. Research Notebook: Computational Thinking--What and Why? The magazine of the Carnegie Mellon University School of Computer Science 6.3.2011. Viitattu 27.6.2023. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why