Tampereen yliopisto

Elmeri Lumme

OHJELMOINNILLISEN AJATTELUN OPETTAMINEN DIGITAALISISSA OPPIMISYMPÄRISTÖISSÄ

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintutkielma
Joulukuu 2021

TIIVISTELMÄ

Elmeri Lumme: Ohjelmoinnillisen ajattelun opettaminen digitaalisissa oppimisympäristöissä
Kandidaatintutkielma
Tampereen yliopisto
Tietotekniikan tutkinto-ohjelma
Joulukuu 2021

Tutkijat määrittelevät ohjelmoinnillisen ajattelun tulevaisuuden taidoksi. Ohjelmoinnillinen ajattelu on ongelmanratkaisutaitoa ja perusta tietokoneella ratkaistavien ongelmien hahmottamiseen. Ohjelmoinnin opettaminen on kirjattu perusopetuksen opetussuunnitelmaan perusteisiin vuonna 2014 ja sen tavoitteet on kuvattu perusopetuksen opetussuunnitelmassa osana matematiikan ja käsityön oppiainekuvauksia. Perusopetuksen opetussuunnitelman perusteissa ohjelmoinnin tavoitteiden kuvaus ei ota riittävästi huomioon ohjelmoinnillista ajattelua, joka on osa ohjelmointiosaamista. Ohjelmoinnillisen ajattelun opettamisen tavoitteita on tarkennettu Opetus- ja kulttuuriministeriön koordinoimassa Uudet lukutaidot -kehittämishankkeen julkaisuissa. Ohjelmointiosaamisen tavoitteissa ilmenee ohjelmoinnillisen ajattelun komponentteja, mutta niiden konkreettinen opettaminen jää opettajan osaamisen ja pedagogiikan varaan. Tutkielman tavoitteena on selvittää, miten digitaalisissa oppimisympäristöissä ohjelmoinnillisen ajattelun komponenttien ilmeneminen tukee ohjelmoinnillisen ajattelun opettamista.

Kirjallisuuskatsauksessa perehdytään ensin ohjelmoinnillisen ajattelun määritelmiin ja esitellään tutkielman osaksi kehittämäni AHAA-malli, johon päädyin rajaamalla ohjelmoinnillisen ajattelun määritelmistä neljä komponenttia. Mallin komponentit ovat abstraktio, hajottaminen, algoritmi ja automatisointi. Mallin tarkoitus on helpottaa moniulotteisen ohjelmoinnillisen ajattelun määritelmän hahmottamista suhteessa käytettävien digitaalisten oppimisympäristöjen arviointiin. Seuraavaksi työssä avataan perusopetuksen opetussuunnitelman perusteiden ja Uudet lukutaidot -kehittämishankkeen tavoitteita, jotka antavat perusteet digitaalisten oppimisympäristöjen ja ohjelmointiosaamisen kehittämiseen. Tutkielman lopuksi AHAA-mallin pohjalta arvioidaan Scratchin, Code.orgin ja Khan Academyn digitaalisia oppimisympäristöjä, jotka ovat lasten ja nuorten ohjelmointiosaamisen kehittämiseen luotuja ohjelmointialustoja.

Kirjallisuuskatsauksessa havaitaan, että arvioitavista digitaalisista oppimisympäristöistä löytyy ohjelmoinnillista ajattelua tukevat AHAA-mallin komponentit ja ne tukevat ohjelmoinnillisen ajattelun opetusta erityisesti ongelmanratkaisun näkökulmasta. Oppimisympäristöjen tehtävät painottuvat ohjelmoinnin opetteluun, ja konkreettisia oppimistehtäviä ohjelmoinnillisesta ajattelusta on vaihtelevasti. Ohjelmoinnillisen ajattelun kehittymistä on vaikea arvioida luotettavasti graafisissa oppimisympäristöissä, jonka vuoksi oppilaille tulisi antaa tehtäviä, joissa ongelman ratkaisu velvoittaa ohjelmoinnillisen ajattelun komponenttien hahmottamista.

Avainsanat: ohjelmoinnillinen ajattelu, digitaaliset oppimisympäristöt, abstraktio, algoritmit, hajottaminen, automatisointi

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

SISÄLLYSLUETTELO

1. JOHDANTO 1
2. OHJELMOINNILLISEN AJATTELUN TAUSTA 3
2.1 Ohjelmoinnillisen ajattelun määrittely kirjallisuudessa 4
2.2 Ohjelmoinnillisen ajattelun komponenttien määrittely 5
3. OHJELMOINNILLISEN AJATTELUN OPETTAMINEN SUOMESSA 9
3.1 Opetussuunnitelmassa ohjelmointi osana laaja-alaisia taitoja 9
3.2 Uudet lukutaidot -kehittämishanke tarkentaa ohjelmoinnillisen ajattelun ja ohjelmoinnin opetusta 11
4. DIGITAALISET OPPIMISYMPÄRISTÖT JA OHJELMOINNILLINEN AJATTELU OPETUKSESSA 13
4.1 Graafiset ohjelmointisovellukset 13
4.1.1 Scratch 14
4.1.2 Code.org 16
4.2 Ohjelmointikielillä ohjelmoinnillista ajattelua 18
5. YHTEENVETO 20
6. LÄHDELUETTELO 23

1. JOHDANTO

Columbian yliopiston tietojenkäsittelytieteen professori Jeanette Wing on väittänyt osuvasti jo vuonna 2006, että ohjelmoinnillinen ajattelu kuuluu jokaisen tulevaisuuden taitoihin (Wing, 2006). Teknologiateollisuuden osaaja- ja osaamistarveselvityksen (Teknologiateollisuus, 2018) mukaan digiosaamisen tarve on ilmeinen ja osaaminen vaatii useimmiten ohjelmoinnin ja ohjelmoinnillisen ajattelun taitoja. Selvityksessä todetaan myös, että ohjelmoinnillisen ajattelun opettaminen jo perusopetuksesta lähtien vastaa osaltaan tietotekniikan työvoimapulaan Suomessa.

Suomessa Opetushallitus on julkaissut perusopetuksen opetussuunnittelman vuonna 2014, jossa määritellään ohjelmoinnillista ajattelua koskevat oppimistavoitteet. Suomessa ohjelmoinnillisen ajattelun taitoja tulisi opettaa jo varhaiskasvatuksesta lähtien siten, että toiselle asteelle siirtyvä oppilas kykenee suunnittelemaan ja ohjelmoimaan yksinkertaisia ohjelmia (Opetus– ja kulttuuriministeriö, 2021; Opetushallitus, 2014).

Ohjelmoinnillisen ajattelun opettaminen ohjelmoinnin (engl. *programming)* avulla on perusteltua, koska ohjelmoinnillinen ajattelu on osa ohjelmointia (Opetus – ja kulttuuriministeriö, 2021; Mannila et al., 2014). Opettamisessa tulisi keskittyä kuitenkin pelkän ohjelmoinnin lisäksi ohjelmoinnillisen ajattelun taitojen kehittämiseen (Buitrago Flórez et al., 2017). Tietoinen ohjelmoinnillisen ajattelun opettaminen kehittää oppilaan algoritmista ajattelua, ongelmanratkaisutaitoja ja (ohjelman)virheenkorjaustaitoja (Buitrago Flórez et al., 2017; Liukas & Mykkänen, 2014). Wing (2017) laajentaa ohjelmoinnillisen ajattelun hyötyjä ongelman ratkaisemisesta myös ongelman luomiseen. Tulevaisuuden haasteiden hahmottaminen ja rajaaminen ratkaistavaksi tietokoneohjelmien avulla on esimerkki ohjelmoinnillisen ajattelun hyödyntämisestä.

Työn tavoitteena on selvittää, miten digitaalissa oppimisympäristöissä ohjelmoinnillisen ajattelun komponenttien ilmeneminen tukee ohjelmoinnillisen ajattelun opettamista. Tutkimustulos osoittaa, että tutkittavista digitaalisista oppimisympäristöistä löytyy ohjelmoinnillista ajattelua tukevat komponentit. Lisäksi havaitaan, että rajoitteena oppimisympäristöjen hyödyntämiselle ohjelmoinnillisen ajattelun opettamiseen kuitenkin on: opettaja joutuu perehtymään oppimisympäristöön ja laatimaan ohjelmoinnillisen ajattelun hahmottamiseksi tukevia tehtäviä ja kysymyksiä. Ohjelmoinnillisen ajattelun

opettaminen tutkielmassa arvioitujen digitaalisten oppimisympäristöjen avulla vaatii opettajalta tehtävien valmistelua, jotta ohjelmoinnillisen ajattelun tavoitteet (Opetus – ja kulttuuriministeriö, 2021) voi saavuttaa.

Tutkielman luvussa 2 perehdytään ensin ohjelmoinnillisen ajattelun määritelmiin ja komponentteihin, joiden avulla ohjelmoinnillisen ajattelun määritelmää rajataan. Sitten tarkastellaan neljää komponenttia: abstraktio, algoritmit, hajottaminen sekä automatisointi. Komponenttien alkukirjamisista muodostetaan AHAA-malli, jonka avulla voidaan arvioida oppimisympäristön soveltumista ohjelmoinnillisen ajattelun opettamisessa. Luvussa 3 avataan perusopetuksen opetussuunnitelman laaja-alaisen osaamisen tavoitteita ja Oikeus oppia -ohjelman Uudet lukutaidot -kehittämishankkeen ohjelmoinnillisen ajattelun tavoitteet liitetään AHAA-mallin komponentteihin. Luvussa 4 tutkitaan AHAA-mallin avulla digitaalisia oppimisympäristöjä ja analysoidaan niiden soveltumista ohjelmoinnillisen ajattelun opettamiseen. Lopuksi luvussa 5 kootaan työn havainnot ja johtopäätökset sekä esitetään jatkotutkimuksen aiheita.

Kirjallisuuskatsauksen lähteinä tässä työssä on käytetty artikkeleita, konferenssijulkaisuja, internet-sivustoja, raportteja ja Opetusministeriön julkaisuja. Lähteitä on etsitty muun muassa ACM- ja Google Scholar-tietokannoista hakusanalla *computational thinking* ja *computational thinking AND education*. Aiheen määritelmien käsittelyssä perehdytään erityisesti useasti viitattuihin artikkeleihin ja konferenssijulkaisuihin, joiden lisäksi aiheen käsittelyssä perehdytään opinnäytetöihin, oppaisiin ja blogi- ja standardisivustoihin. Digitaalisten oppimisympäristöjen valinta tutkimuksen kohteeksi perustuu niiden esiintymiseen käytetyissä artikkeleissa ja muissa lähteissä sekä kirjoittajan kokemukseen näiden oppimisympäristöjen kouluttamisesta opettajille.

2. OHJELMOINNILLISEN AJATTELUN TAUSTA

Ohjelmoinnillisen ajattelun (engl. *computational thinking)* käsitteen pioneerina pidetään Seyomour Papertia, joka esitteli käsitteen vuonna 1980 kirjassaan *Mindstorms: Children, Computers, and Powerful Ideas* (Papert 1980, p. 182, katso Lodi & Martini, 2021). Toinen merkittävä tutkija ohjelmoinnillisen ajattelun tutkimuksen historiassa on Wing, joka määritteli ohjelmoinnillisen ajattelun käsitteen seuraavasti: ”*Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science.*”(Wing, 2006, p.33) Wing ei rajaa mitään suoraan termistä pois, mutta sisällyttää määritelmään ongelmanratkaisun, järjestelmien suunnittelun ja ihmisen käyttäytymisen ymmärtämisen tietojenkäsittelytieteen käsitteitä hyödyntämällä. Wingin julkaisu on saavuttanut laajaa tieteellistä huomiota ja Wingin määritelmiin (Wing, 2006; Wing, 2008) viitataan laajasti tieteellisissä tutkimuksissa. Käsitteen yksiselitteistä määritelmää ei vielä ole, jonka vuoksi tässä tutkielmassa perehdytään myöhemmin komponentteihin. Komponenteilla ohjelmoinnillisen ajattelun käsitteen määrittely ja hahmottaminen on konkreettista.

Lodi ja Martini (2021) vertailevat tutkimuksessaan Wingin ja Papertin näkökulmia käsitteestä *computational thinking.* Mannila et al. (2014) toteavat, että Wingin ja Papertin määritelmistä löytyy samankaltaisuuksia: molemmat painottavat päättelyä. Papert esittelee kirjassaan tietokoneen sekä ohjelmoinnin roolia ja Wing puolestaan korostaa ajattelun taitoja. Lodi ja Martini toteavat, että näkökulmaerosta johtuen termit ovat väärinymmärrettyjä, mutta molemmat ovat avartaneet käsitteen tutkimusta ja määrittelyä. (Lodi & Martini, 2021)

Ohjelmoinnillinen ajattelu on laaja-alaista ajattelun taitoa (Lee et al., 2011; Barr et al., 2011; Selby & Woollard, 2013), joka sisältää muissakin oppiaineissa hyödynnettäviä ajattelutaitoja. Ohjelmoinnillinen ajattelu ei olekaan Wingin (2006) mukaan kokonaan uusi ajattelun taito, vaan se sisältää tuttuja ongelmanratkaisun menetelmiä ja loogista ajattelua (katso Barr et al,. 2011; Barr & Stephenson, 2011). Tarkastelemme tässä tutkielmassa ohjelmoinnillista ajattelua erityisesti ongelmanratkaisun näkökulmasta. Tutkimuksessa puhutaan myöhemmin ongelmasta, jolla viitataan tehtävään tai pulmaan, jolle etsitään ratkaisua.

Englanninkielinen *computational thinking* kääntyy kömpelösti suomen kielelle ja siksi suomenkielisessä tutkimuksessa esiintyy vaihtelevasti käsitteet algoritminen ajattelu ja ohjelmoinnillinen ajattelu. (Liukas & Mykkänen, 2014; Marttala, 2017; Koivisto, 2019;

Sormunen, 2019) Molemmat termit viittaavat samaan laajempaan ajattelutaitoon. Tässä tutkielmassa käsitteen suomennos pohjautuu Opetus- ja Kulttuuriministeriön *Oikeus Oppia* -ohjelman Uudet lukutaidot -kehittämishankkeeseen, jossa ohjelmointiosaamisen kuvauksessa käytetään otsikkoa *ohjelmoinnillinen ajattelu* (Opetus- ja kulttuuriministeriö, 2021).

2.1 Ohjelmoinnillisen ajattelun määrittely kirjallisuudessa

Laajan ja moniulotteisen käsitteen määrittelemisen helpottamiseksi ohjelmoinnillisen ajattelun käsitteen määrittelyssä hyödynnetään komponentteja tai avainsanoja, koska kirjallisuudessa ei ole yhtä vakiintunutta määritelmää käsitteelle (Barr et al., 2011). Kirjallisuuskatsauksessa tutkijat Cansu ja Cansu (2019) ovat koonneet eri tutkijoiden käyttämiä avainsanoja, joita esiintyy ohjelmoinnillisen ajattelun käsitteen määrittelyssä.

Wingin (2006) monitulkintaiseen määritelmään pohjautuen monet tutkijat ovat samaa mieltä ongelmanratkaisun sisällyttämisestä ohjelmoinnillisen ajattelun määritelmään. Barr et al. (2011) laajentavat käsitteen määritelmää konkreettisemmilla avainsanoilla: Ohjelmoinnillinen ajattelu on ongelmanratkaisuprosessi, joka sisältää ongelmien muotoilua tietokoneelle sopivaksi. Tämä tarkoittaa muun muassa loogista organisointia ja datan analysointia, datan esittämistä abstraktioiden, kuten mallien avulla, ratkaisujen automatisointia algoritmisen ajattelun kautta, ratkaisujen tunnistamista, analysointia ja toteuttamista tehokkaasti sekä prosessin yleistämistä laajasti ongelmiin (Barr et al., 2011).

Barr ja Stephenson (2011) vahvistavat näkemyksen ohjelmoinnillisen ajattelun määritelmästa ongelmanratkaisutaitona, jossa ongelman voi ratkaista tietokonetta hyödyntäen. Ongelmanratkaisun lisäksi tutkijat lisäävät osaksi ohjelmoinnillista ajattelua abstraktion, algoritmien luomisen, automaation, datan keräyksen, analysoinnin, testauksen ja debuggauksen. Myöhemmin Selby ja Woollard (2013) korostavat ajatteluprosessin, abstraktion ja hajottamisen sisällyttämistä ohjelmoinnillisen ajattelun käsitteen määritelmään. Tutkimustulostensa pohjalta tutkijat ehdottavat ohjelmoinnillisen ajattelun määritelmäksi ongelmanratkaisua ja ajatteluprosesseja, jotka hyödyntävät abstraktiota, hajottamista, algoritmista suunnittelua, arviointia ja yleistämistä (Selby & Woollard, 2013).

The Computer Science Teachers Association (CSTA) ja The International Society for Technology in Education (ISTE) ovat määritelleet ohjelmoinnillisen ajattelun standardit opettamisen tueksi K-12 opetukseen (Barr et al., 2011). K-12 on kansainvälinen lyhenne ja tarkoittaa luokkia esikoulusta toiselle asteelle. Määritelmän mukaan ohjelmoinnillinen ajattelu on ongelmanratkaisuprosessi, joka sisältää (mutta ei ole rajoitettu seuraaviin) ongelman muotoilemista ja luomista ratkaistavaksi tietokoneen avulla, datan loogista järjestelemistä ja analysointia sekä kuvaamista abstraktioilla. Standardiin on sisällytetty myös ratkaisujen automatisointia sarjalla järjestettyjä vaiheita algoritmisen ajattelun tavoin, niiden tunnistamista, analysoimista ja toteuttamista tavoitteena saavuttaa tehokkain(engl. *efficient*) ja vaikuttavin(engl. *effective*) vaiheiden ja resurssien yhdistelmä sekä ajatteluprosessin yleistämistä ja siirtämistä erilaisiin ongelmiin. (ISTE & CSTA, 2011, katso Koivisto, 2019)

Sivuillaan ISTE määrittelee opettamisen tueksi ohjelmoinnillisen ajattelun ydinkomponentit: hajottaminen, datan keräys ja analysointi, abstraktio, algoritmien suunnittelu ja vaikutus yhteisöön (ISTE, 2011). ISTE:n tavoitteena on tukea kouluttajia sisällyttämään ohjelmoinnillista ajattelua eri oppiaineissa kaikkien oppilaiden kanssa (ISTE). Suomalaista tukea ohjelmoinnillisen ajattelun opettamiseen ovat tuottaneet Linda Liukas ja Juhani Mykkänen. Kirjoittamassaan oppaassa *KOODI2016 – ensiapua ohjelmoinnin opettamiseen peruskoulussa* Liukas ja Mykkänen (2014) korostavat myös ajattelutaitoja, joita ohjelmoinnillinen ajattelu sisältää: ongelman hajottamista, kaavojen tunnistamista, algoritmien luomista havaituille kaavoille ja algoritmien esittämistä yleisessä muodossa.

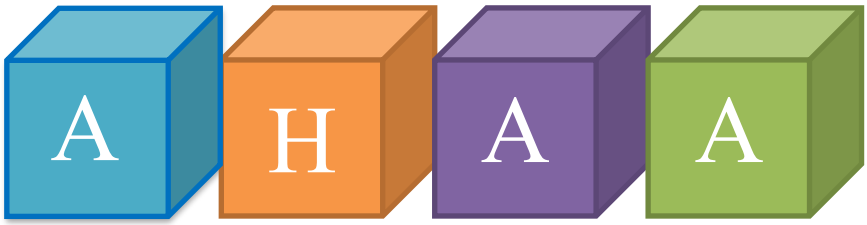
2.2 Ohjelmoinnillisen ajattelun komponenttien määrittely

Tässä tutkielmassa ohjelmoinnillisen ajattelun käsitteen määritelmä rajataan neljään ydinkomponenttiin, jotka esitellään taulukossa 1. Rajauksen tarkoituksena on helpottaa moniulotteisen ohjelmoinnillisen ajattelun käsitteen määritelmän hahmottamista suhteessa käytettävien digitaalisten oppimisympäristöjen arviointiin. Tutkielmasta rajataan pois komponentteja, joita käytetään myös ohjelmoinnillisen ajattelun käsitteen määrittelyyn. Rajattuja komponentteja ovat muun muassa arviointi, analysointi, debuggaus, rinnakkaistaminen, soveltaminen ja suunnittelu. Julkaisussaan Cansu ja Cansu (2019) rajaavat ohjelmoinnillisen ajattelun käsitteen määritelmäksi em. viisi komponenttia, joista automatisointi ja yleistys ovat yhdistetty tässä tutkielmassa.

Taulukko 1 AHAA-mallin komponentit ja selitykset

Komponentti	**Selitys**	**Lähteitä**
Abstraktio	Ongelman yksityiskohtien karsimista, jotta asia on riittävän informatiivinen. Yleistysten esittämistä ongelmasta.	Wing (2008), ISTE & CSTA (2011), Lee (2011), Selby & Woollard (2013)
Hajottaminen	Ongelman jakaminen pienempiin osiin, jolloin suunnittelu ja toteuttaminen ovat helpompaa.	ISTE & CSTA (2011), Barr et al. (2011), Selby ja Woollard (2013)
Algoritmi	Toimintaohjeet, sarja vaiheita, joilla ongelma ratkaistaan. Nämä tietokone lukee ohjelman suorittamisessa.	ISTE & CSTA (2011), Barr et al, (2011)
Automatisointi ja yleistys	Ratkaisun muotoilu yleiseen muotoon, jotta sitä voi hyödyntää samankaltaisten ongelmien ratkaisemiseen tehokkaasti.	Wing (2008), Lee (2011), ISTE & CSTA (2011)

Komponenttien alkukirjaimista muodostuu kuvan 1 AHAA-malli. Mallia käytetään luvussa 4 arvioinnin perustana digitaalisten oppimisympäristöjen soveltumiseen ohjelmoinnillisen ajattelun opetuksessa. Mallin avulla moniuloitteisen ohjelmoinnillisen ajattelun sisältöä ja oppimistavoitteita on helpompi rajata oppilaille ja opettajalla on mahdollisuus arvioida oppimisympäristön soveltumista ohjelmoinnillisen ajattelun opettamiseen.



Kuva 1 AHAA tulee sanoista abstraktio, hajottaminen, algoritmi ja automatisointi

Wing (2008) määrittelee ohjelmoinnillisen ajattelun ytimeksi abstraktion. Abstraktio tarkoittaa ongelman pelkistämistä yksityiskohtia karsimalla siten, että asia on riittävän informatiivinen (Wing, 2008, Csizmadia et al., 2015). Esimerkiksi tien neuvominen asemalle: Ohjeesta karsitaan yksityiskohdat, kuten tien jyrkkyys tai leveys, jotka eivät ole oleellisia perille löytämisessä. Wingin (2008) abstraktiota tapahtuu ainakin kahdessa tasossa, yleensä useammassa. Hän esittää, että abstraktion tasojen rajapintojen määritteleminen hyvin antaa mahdollisuuden ratkoa laajoja kompleksisia ongelmia – esimerkiksi vuorovaikutuksessa ohjelmointirajapinnan kanssa käyttäjän ei tarvitse tietää yksityiskohtia. Barrin ja Stephensonin (2011) mukaan tietojenkäsittelytieteen käsitteet ehtolause, silmukka ja rekursio ovat osa abstraktiota. Näillä yleistetään toistuvien algoritmien suorittaminen.

Hajottaminen helpottaa abstraktoidun ongelman muotoilemista ja esittämistä. ISTE (ISTE) määrittelee hajottamisen seuraavasti: Se on ongelman purkamista pienempiin osiin ohjelman suunnittelun, toteutuksen ja tarkistamisen helpottamiseksi. Laajan tehtävän purkaminen pienempiin osiin on osa ohjelmoinnillisen ajattelun taitoa ja se toteutuukin usein ohjelmointiohjelman käyttöönotossa (Mannila et al., 2014). Ongelman hajottaminen helpottaa hahmottamaan vuorovaikutussuhteita. Hajottamisen taito on edellytys ohjelmoinnillisen ajattelun osa-alueille, kuten abstraktion ja algoritmien suunnittelulle (Selby, 2014; katso Koivisto, 2019). Ongelman jakaminen hierarkiaan auttaa hahmottamaan ongelmaa osina, joka rakentuu tasojen välisestä vuorovaikutuksesta. Esimerkiksi ruoanlaitto voidaan hajottaa osiin, jotka suoritetaan tiettyä resptiä noudattaen - vaiheittain algoritmin omaisesti.

Algoritmi on Wingin (2008) mukaan abstraktio vaiheittaisesta käsittelyjärjestyksestä syötteen vastaanottamiseksi ja halutun tulosteen tuottamiseksi. Algoritmi voi olla laaja kokonaisuus tai ongelman pienempi osa-alue: algoritmin hajottaminen pienempiin osiin helpottaa suunnittelua ja toteutusta. Mannilan et al. (2014) tutkimuksessa havaittiin, että algoritmeja käytetään ja suunnitellaan opettamisessa. Esimerkiksi tutkimuksessa haastatellut opettajat olivat havainneet, että resepti on algoritmi, jossa ohjeet tulee toteuttaa järjestyksessä.

Automatisointi liittyy ohjelmoinnilliseen ajatteluun Wingin (2008) mukaan abstraktioiden kautta. Hän yhdistää abstraktion ja automatisoinnin tietojenkäsittelyn määritelmään seuraavasti: ”*Computing is the automation of our abstractions*” (Wing, 2008, s. 3718). Tietojenkäsittely tapahtuu siis automatisoimalla luotuja abstraktioita. Lee et al. (2011) täsmentää Wingin määritelmää automatisoinnista: Automatisointi on prosessi, jossa tietokoneelle annetaan joukko tehtäviä suoritettavaksi, joka säästää työtä tietokoneen

ollessa laskentatehokkuudessa ihmistä nopeampi ja tehokkaampi. Ratkaisun yhtäläisyyksien tunnistaminen ja uudelleen käyttäminen ovat tutkijoiden mukaan osa ohjelmoinnilista ajattelua (Selby & Woollard, 2013). Algoritmin yleistäminen johtaa tehokkuuteen ja automatisoi ongelmanratkaisua. Esimerkiksi suorakulmion piirtävä algoritmi kannattaa yleistää, jolloin algoritmille kelpaa parametreiksi kaikki sivun pituudet.

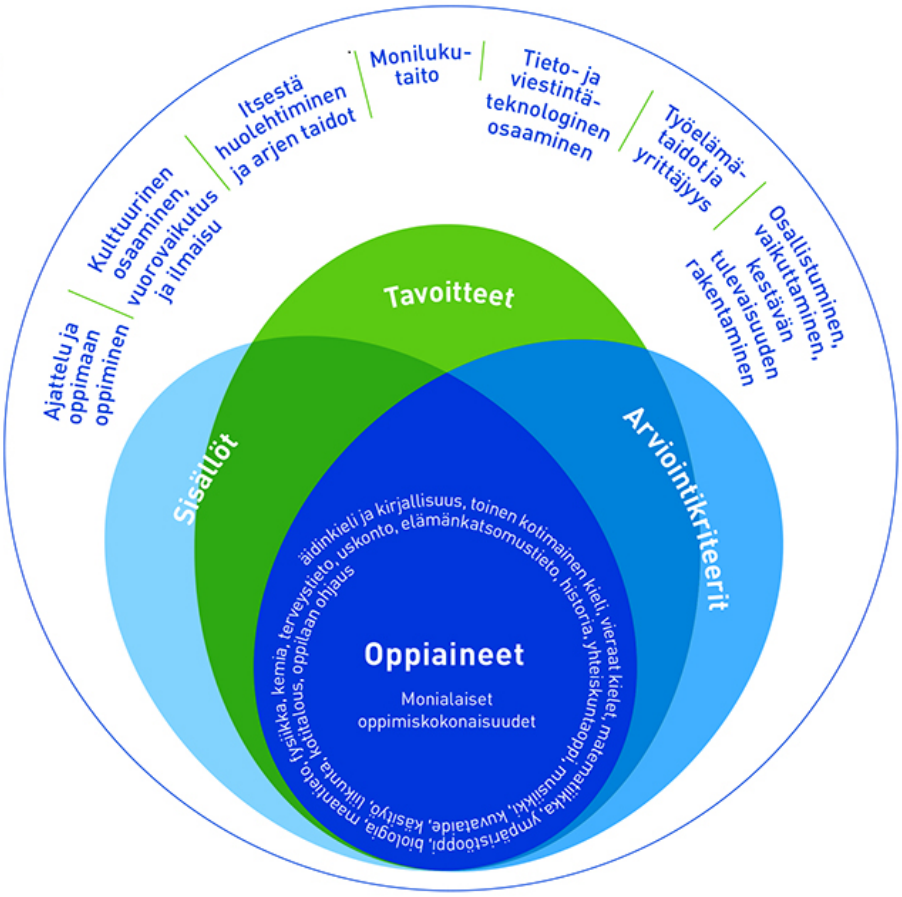
3. OHJELMOINNILLISEN AJATTELUN OPETTAMINEN SUOMESSA

Tässä luvussa esitellään Opetushallituksen laatiman perusopetuksen opetussuunnitelman perusteiden (2014) laaja-alaisen osaamisen osa-alueista monilukutaito ja tieto- ja viestintäteknologinen osaaminen, jotka sisältävät muun muassa digitaalisten oppimisympäristöjen käyttämiseen osana ohjelmoinnin opetusta. Vaikka useat laaja-alaisen osaamisen osa-alueiden tavoitteet linkittyvät toisiinsa, tarkastelusta on rajattu ajattelu ja oppimaan oppimisen osa-alue, joka voisi myös soveltua ohjelmoinnillisen ajattelun tavoitteiden tarkasteluun.

Toisessa alaluvussa tarkastelemme Oikeus Oppia -ohjelmaan liittyvän Uudet lukutaidot -kehittämishankkeen (Opetus- ja kulttuuriministeriö, 2021) tarkempia ohjelmointiosaamisen tavoitteita. Tavoitteista etsitään AHAA-mallin komponentteja kuvaavat määritelmät, jotka on koottu taulukkoon luvun 4 digitaalisten oppimisympäristöjen arviointia varten.

3.1 Opetussuunnitelmassa ohjelmointi osana laaja-alaisia taitoja

Suomen peruskouluissa opetusta ja tavoitteita ohjaa opetussuunnitelman perusteet (OPS) (Opetushallitus, 2014). Uudistetussa opetussuunnitelmassa on huomioitu myös digitaalisten oppimisympäristöjen kasvava merkitys tulevaisuuden koulun arjessa. Kuva 2 esittelee laaja-alaisen osaamisen osa-alueet. Tavoitteet, oppiaineet, sisällöt ja arviointikriteerit ohjaavat eri tiedon- ja taidonalueiden hallitsemiseen, jota opiskelu, työelämä ja aktiivinen kansalaisuus edellyttävät. (Opetushallitus, 2021)



Kuva 2 Laaja-alaisen osaamisen osa-alueet (Opetushallitus 2021)

OPS:n (Opetushallitus, 2014, s. 22-23) luvussa ”Tavoitteena laaja-alainen osaaminen” monilukutaidon tavoitteita määritellään seuraavasti:

> *”Monilukutaitoon sisältyy monia erilaisia lukutaitoja, joita kehitetään kaikessa opetuksessa. Oppilaiden tulee voida harjoittaa taitojaan sekä perinteisissä että monimediaisissa, teknologiaa eri tavoin hyödyntävissä oppimisympäristöissä.”*

ja tieto- ja viestintäteknologian (tvt) osaamisen tavoitteita määritellään seuraavasti (Opetushallitus, 2014, s. 22-23):

> *”Oppilaita ohjataan ymmärtämään tieto- ja viestintäteknologian käyttö- ja toimintaperiaatteita ja keskeisiä käsitteitä sekä kehittämään käytännön tvt-taitojaan omien tuotosten laadinnassa.”*

Ohjelmointia ei mainita kummassakaan laaja-alaisen osaamisen osa-alueessa, mutta ilmaisut ”*teknologiaa eri tavoin hyödyntävissä oppimisympäristöissä*” ja ”*kehittämään*

käytännön tvt-taitojaan omien tuotosten laadinnassa" velvoittavat digitaalisten oppimisympäristöjen hyödyntämistä ohjelmoinnin ja ohjelmoinnillisen ajattelun opetuksessa.

Vaikka ohjelmoinnin tai ohjelmoinnillisen ajattelun tavoitteita ei ole erikseen määritelty tvt:n yleisessä kuvauksessa, on tavoitteet kuitenkin kuvattu osana vuosiluokkien tieto- ja viestintäteknologisen osaamisen, matematiikan ja käsityön kokonaisuuksia (Opetushallitus, 2014). Vaikka ohjelmoinnillisessa ajattelussa onkin paljon yhtymäkohtia matemaattisen ajattelun (Wing, 2008) kanssa, matematiikan lisäksi ohjelmoinnillista ajattelua tulisi hyödyntää laaja-alaisesti yli oppiainerajojen (ISTE). Näin laaja-alaiset tavoitteet toteutuvat paremmin ja oppilas voi hyödyntää ohjelmoinnillista ajattelua laajempien ja kompleksisten ongelmien ratkaisuissa yli oppiainerajojen.

3.2 Uudet lukutaidot -kehittämishanke tarkentaa ohjelmoinnillisen ajattelun ja ohjelmoinnin opetusta

Uudet lukutaidot -kehittämishankkeen sisältää kolme osaamisen kuvausta: tieto- ja viestintäteknologinen osaaminen, medialukutaito ja ohjelmointiosaaminen. Ohjelmointiosaamisen kuvauksen mukaan ohjelmointi on osa TVT- ja monilukutaidon laaja-alaista osaamista (Opetus- ja kulttuuriministeriö, 2021). Ohjelmoinnin osuus nykyisessä perusopetuksen opetussuunnitelmassa on vähäinen ja tässä ohjelmassa ohjelmointi on nostettu keskiöön medialukutaidon ja tieto- ja viestintätaitojen rinnalle. Uudet lukutaidot - nettisivuilla ohjelmointiosaaminen on jaettu kolmeen päälukuun: 1) ohjelmoinnillinen ajattelu, 2) tutkiva työskentely ja tuottaminen sekä 3) ohjelmoidut ympäristöt ja niissä toimiminen. Ohjelmoinnillisen ajattelun osa-alueet ovat jaettu a) looginen ajattelu ja tiedon käsittely, b) ongelmien ratkaiseminen ja mallintaminen, c) ohjelmoinnin käsitteet ja perusrakenteet ja d) käytännön taidot. (Opetus- ja kulttuuriministeriö, 2021)

Ohjelmoinnillisen ajattelun osa-alueiden kuvauksissa ilmenee yhtäläisyyksiä AHAA-mallin komponentteihin. Taulukossa 2 osaamistavoitteet (Opetus- ja kulttuuriministeriö, 2021) on liitetty mallin komponentteihin.

Taulukko 2 Ohjelmoinnillisen ajattelun osaamistavoitteet liitettynä AHAA-malliin

KOMPONENTTI	OSAAMISTAVOITTEET (Opetus- ja kulttuuriministeriö, 2021)
Abstraktio (A)	”käsittelee erilaisiin yleistyksiin sisältyviä tietoja”.
Hajottaminen (H)	”…tekee havaintoja syy-seuraussuhteista.” ”purkaa tuttuun arjen ilmiöön liittyvän ongelman osiin sekä etsii ja kokeilee sen ratkaisuun erilaisia vaihtoehtoja.”
Algoritmi (A)	”laatii täsmällisiä ja yksityiskohtaisia toimintaohjeita käyttäen toisto- ja valintarakenteita.”
Automatisointi ja yleistys (A)	”Oppilas arvioi ratkaisuja hyödyntäen jotakin kriteeriä, kuten toimivuus, luettavuus tai tehokkuus”

4. DIGITAALISET OPPIMISYMPÄRISTÖT JA OHJELMOINNILLINEN AJATTELU OPETUKSESSA

Digitaalisia oppimisympäristöjä (DO) hyödynnetään yhä laajemmin ja oppilaiden ohjelmointikokemukset ovat lisääntyneet Suomen peruskouluissa (Tanhua-Piiroinen et al., 2018, 2020; Valtioneuvosto, 2020). Opetustavoitteista riippuen niiden soveltuminen ohjelmoinnillisen ajattelun opetukseen vaihtelee. Opettajan olisi täten hyvä perehtyä käytettävään digitaaliseen oppimisympäristöön ja tarkastella sitä oppimistavoitteiden kannalta (Liukas ja Mykkänen, 2014). Lahtinen et al. (2005) painottavat myös huolellista materiaalien ja tehtävien valmistelua, jolla on edistävä vaikutus ohjelmointia aloittelevan oppilaan tietojen ja taitojen kehittymiseen.

Digitaalinen oppimisympäristö on mobiililaitteella tai tietokoneella käytettävä oppimisympäristö, jota voidaan hyödyntää ohjelmoinnillisen ajattelun opetukseen. Suomalaisten opettajien tueksi Liukkaan ja Mykkäsen (2014) kirjoittamassa oppaassa esitellään tunnetuimpia digitaalisia oppimisympäristöjä. Esiteltyjä digitaalisia oppimisympäristöjä voi hyödyntää soveltuvin osin myös varhaiskasvatuksen ja toisen asteen opetuksessa. Tässä luvussa arvioidaan AHAA-mallin avulla Scratchin, Code.org:n ja Khan Academyn soveltumista ensisijaisesti ohjelmoinnillisen ajattelun opetukseen. Luvussa tutkitaan AHAA-mallin komponenttien avulla miten komponentit ilmenevät digitaalisissa oppimisympäristöissä ja tukevatko digitaaliset oppimisympäristöt ohjelmoinnillisen ajattelun opetusta.

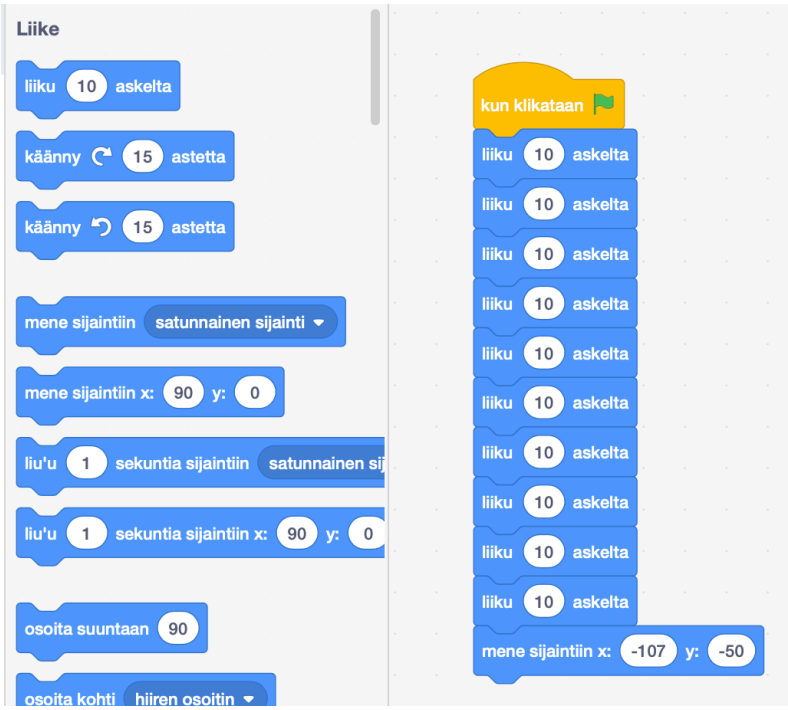
4.1 Graafiset ohjelmointisovellukset

Ohjelmoinnillisen ajattelun opettamisessa on luonnollista hyödyntää graafisia ohjelmointisovelluksia niiden helppokäyttöisyyden takia. Ohjelmien tekeminen ja ongelman ratkaiseminen eivät vaadi tietyn ohjelmointikielen tai syntaksin opettelua, jota tulisi tutkijoiden (Buitrago Florez et al., 2017) mukaan välttää aloittelevien oppilaiden kanssa. Tämä säästää aikaa ajattelutaidon opettamiseen ohjelmointikielen syntaksin sijaan.

4.1.1 Scratch

Scratch on graafinen ohjelmointialusta, joka on suunniteltu käyttäjän tausta, ikä ja kiinnostus huomioiden käyttäjäystävälliseksi (Buitrago Florez et al., 2017). Scratch edistää ohjelmoinnillista ajattelua ja ongelmanratkaisutaitoja sekä edesauttaa oppilaiden omien tuotosten laadinnassa käytännön tvt-taitoja. Scratchillä voi luoda pelejä, animaatioita tai tarinoita. Sitä voi käyttää tietokoneen selaimessa suoraan ilman ohjelman erillistä asentamista, jolloin ohjelman käyttöönottoon ei tuhlaannu opetuksesta aikaa. (Scratch)

Scratchissä olemassaolevien tehtävien, opettajille saatavilla olevien opetuskokonaisuuksien tai vapaan luomisen avulla on mahdollista avata oppilaille ohjelmoinnillisen ajattelun käsitteitä ja opettaa hahmottamaan ratkaisemaan ongelmia ohjelmoinnillista ajattelua hyödyntäen. Kuvassa 3 on esimerkkiohjelma, jossa on luotu algoritmi hahmon liikuttamiseen. Yksinkertaisen algoritmin luomisessa voidaan opettaa ohjelmoinnillista ajattelua.



Kuva 3 Lohkot raahataan oikealle puolelle ja yhdistetään algoritmiksi (Scratch)

Scratchissä toteutuu Wingin (2006) abstraktion määritelmä: yksityiskohdat on karsittu ydinasioiden tieltä, mutta riittävä informaatio ohjelman rakentamiseen on esillä. Ongelman ratkaiseminen valmiilla lohkoilla ei kohdista käyttäjän huomiota yksityiskohtiin, kuten ohjelmointikielen syntaksiin, funktioiden luomiseen tai muuttujien

tyypin määrittelyyn. Ohjelman kompleksisuutta on abstraktoitu palikoiden väreillä, muodoilla sekä yleistämällä mahdollisia toimintoja (kuva 4). Esimerkiksi liike-lohko on yleistetty vain horisontaaliseen suuntaan ja silmukka-lohko abstraktoi toistuvan ”liiku 10 askelta” -komennon suorittamisen. Scratch tarjoaa automatisointiin mahdollisuuden monistaa valmiita algoritmeja. Esimerkiksi silmukka-rakenne on valmis osa, jota voi monistaa ja käyttää useiden ongelmien ratkaisuissa. Automatisointiin kuuluvaa luettavuutta ja tehokkuutta voi mitata tässä kontekstissa (vrt. kuva 3 ja 4) algoritmin luomiseen käytetyllä ajalla tai lohkojen lukumäärällä.

Kuva 4 Silmukka-rakenne abstraktoi toistuvien komentojen suorittamisen (Scratch)

Ohjelman suunnittelu helpottuu hajottamalla halutut toiminnallisuudet pienempiin osiin. Kuvasta 5 havaitaan nuolinäppäimille omat toiminnallisuudet, jotka ovat selkeästi eroteltuna kahdeksi omaksi osaksi. Ongelman ratkaisun hajottaminen pienempiin osiin on luontevaa, kunhan opettaja tietää miten osien välinen vuorovaikutus toimii. Scratch tarjoaa myös jokaiselle taustan ja hahmon toiminnallisuudelle oman koodialustan, johon lohkoja voi raahata (Scratch).

Algoritmien luominen tapahtuu raahaamalla lohkoja listasta koodialustalle. Lohkot ovat liitettävissä toisiinsa hunajakennon tavoin (kuva 5). (Scratch) Algoritmin rakentamisessa ei tarvitse tietää niiden keskinäisiä riippuvuuksia tai miten algoritmit toteutetaan yksityiskohtaisesti, vaan riittää tietää mitä toiminnallisuutta tarvitaan. Ohjelman ei tarvitse olla valmis toimiakseen, vaan ratkaisuja on mahdollisuus testata ja laajentaa myöhemmin, kun laajempien ja kompleksisten algoritmien hahmottaminen on luontevaa

oppilaalle. Algoritmin luomisessa toiminnallisuuksien lisääminen tai poistaminen antaa mahdollisuuden muokata algoritmia yksilöllisen taitotason mukaan.

Kuva 5 Ongelman hajottaminen pienempiin osiin on visuaalisesti selkeä (Scratch)

4.1.2 Code.org

Code.org on Scratchin tavoin graafinen ohjelmointisovellus, jossa lohkoja raahataan koodialustalle (kuva 6). Code.org:ssa voi valita joko täysin oman ohjelman tai valmiiksi rakennettuja oppimistehtäviä. Code.org on rakennettu oppimisympäristöksi, jolla on tarkoitus mahdollistaa tietojenkäsittelytiedettä kaikille oppilaille kaikissa kouluissa. Opettamisen tueksi organisaatio on rakentanut opetussuunnitelman, jossa opetetaan tietojenkäsittelyn ja ohjelmoinnin käsitteitä rajattujen tehtävien avulla. (Code.org)

Code.orgin ulkoasu ja tehtävien toimintalogiikka mukailee Scratchiä. Abstraktio on hahmotettavissa samoin kuin Scratchissä. Kuvassa 6 on esimerkkitehtävä, jossa harjoitellaan silmukoiden käyttämistä.

Kuva 6 Tehtävä antaa ongelmaan osittain valmiin algoritmin (Code.org)

Hajottaminen on huomioitu ohjelmassa luomalla tehtävät ja tehtävänannot siten, että oppilas harjoittelee tehtävässä rajattua osaa kokonaisuudesta. Tehtävän suorittamiseen vaadittavat lohkot on annettu, eikä mahdollisia muita toteutuksia ole mahdollistettu. Kuvan 6 tehtävässä oppilas on velvoitettu käyttämään silmukka-rakennetta tehtävän toteuttamiseen. Ohjelma suorittaa algoritmin ja vasemmassa reunassa näkyvä suoritus tarjoaa vihjeitä ja visuaalisen tavoitteen, johon tulee päästä. Algoritmin luomisessa keskitytään logiikan rakentamiseen, eikä oppilaan tarvitse huomioida yksityiskohtia. Tehtävän valmis algoritmi näkyy kuvassa 7.

Kuva 7 Algoritmin komennot ovat yksityiskohtaisia (Code.org)

Code.orgin tehtävissä automatisointi ja yleistys tulevat esille käyttämällä annettua lohkoa tehtävän suorittamiseen. Luettavuus ja tehokkuus ovat tässäkin ratkaisussa parempia, kuin kuvan 3 ratkaisussa, jossa algoritmi on tehty ilman silmukkaa.

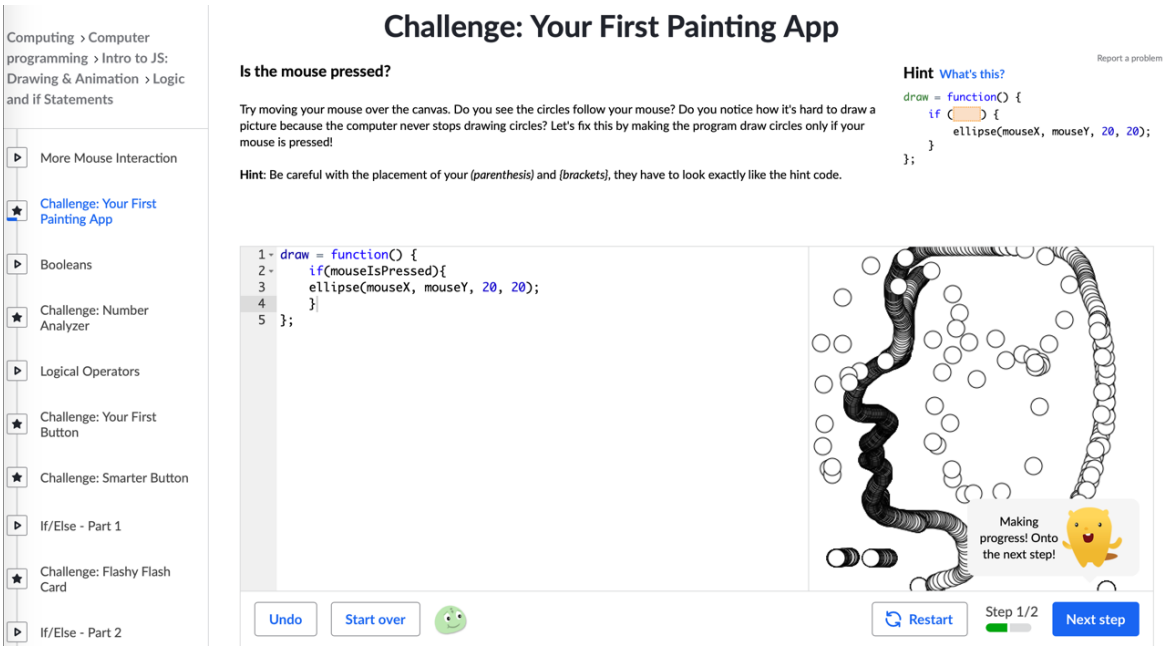
Code.orgin graafinen ohjelmointikieli sekä lohkojen muodot ja värit sisältävät samankaltaisuuksia Scratchiin verrattuna. Myös Scratchistä löytyvät AHAA-mallin komponentit toistuvat lähes samoin Code.org:ssa. Näiden digitaalisten oppimisympäristöjen suurin eroavaisuus on hajottamisessa, koska valmiin tehtävän pilkkominen pienemmiksi osiksi ei ole mahdollista osassa Code.orgin tehtävistä.

4.2 Ohjelmointikielillä ohjelmoinnillista ajattelua

Oppilaat, joiden taitotaso riittää siirtymään graafisista ohjelmointisovelluksista ohjelmointikielien avulla toteutettuihin ohjelmiin, voivat käyttää Khan Academyn oppimisympäristöä. Ohjelmointikielen syntaksin opettelun lisäksi vaaditaan englannin kielen taitoa, koska oppimisympäristö ei ole saatavissa suomeksi.

Khan Academy on oppimisympäristö, jossa opettaja voi rakentaa kursseja oppilaille tai oppilaat voivat suorittaa kursseja itsenäisesti ilman ohjelmistojen asentamista. Khan Academy on voittoa tavoittelematon organisaatio, jonka missiona on ”…*to provide a free, world-class education to anyone, anywhere*.” (Khan Academy) Suomessakin on mahdollista käyttää Khan Academyn digitaalista oppimisympäristöä mission mukaisesti. Kursseja löytyykin muun muassa matematiikasta, luonnontieteistä, tietojenkäsittelytieteistä, historiasta, taiteesta ja ekonomiasta. (Khan Academy)

Khan Academyn ero Code.orgiin ja Scratchiin verrattuna on käytetty ohjelmointikieli. Ohjelmoinnin kursseissa käytettävät ohjelmointikielet ovat JavaScript, HTML ja CSS. Khan Academyn ohjelmointikurssien pääpaino on tehtävien suorittamisessa, mutta tehtävät ohjaavat oppilaita käyttämään ohjelmoinnin käsitteitä ja opettaa ratkaisemaan ongelmia ohjelmoinnillista ajattelua hyödyntäen. Kuvan 8 esimerkkitehtävässä luodaan algoritmi, jossa käytetään ehtolausetta. Tehtävänannoltaan Khan Academyn tehtävä on samallainen kuin Code.orgin tehtävä. Ongelma on suppea ja valmiiksi annettu algoritmi auttaa alkuun tehtävässä. Tämä painottaa kuitenkin enemmän ohjelmointiosaamista ja ongelmanratkaisua, kuin ohjelmoinnillisen ajattelun komponenttien oppimista.



Kuva 8 Yksittäinen ehtolause muodostaa oman osaratkaisun (Khan Academy)

Kuten Code.org:n tehtävissä, Khan Academyn tehtävät ovat rajattuja ja oppilas harjoittelee ohjelmoinnin käsitteitä ja rakenteita osa kerrallaan. Hajottaminen tarkoittaa tässä tehtävässä ongelman pilkkomista yksittäiseksi ehtolauseeksi. Tehtävää tekemällä ei kuitenkaan voi varmistua, että oppilas on oppinut hajottamaan isomman ongelman pienempiin osiin. Avaamalla oppilaalle hajottamisen merkitystä laajemman ongelman ratkaisussa, tehtävää voidaan kuitenkin käyttää esimerkkinä ohjelmoinnillisen ajattelun opetuksessa. Hyviin ohjelmointikäytäntöihin kuuluu ohjelman jakaminen funktioihin ja pienempiin osiin (Foord 2017).

5. YHTEENVETO

Kirjallisuudesta ei löydy konsensusta ohjelmoinnillisen ajattelun määritelmälle. Koska kyseessä on ongelmanratkaisun ajattelutaito, on määritelmälle vaikeaa löytää reunaehtoja. Ohjelmoinnillisessa ajattelussa on ominaista ongelman yksityiskohtien karsiminen eli abstraktointi sekä hajottaminen pienempiin osaongelmiin. Nämä ratkaistaan algoritmeilla, joita voidaan automatisoida silmukkarakenteella tai laajentamalla muihin osaongelmiin. Osaongelmat kasataan sitten kokonaiseksi ratkaisuksi. Näiden komponenttien ollessa oleellinen osa ohjelmoinnillista ajattelua, voidaan ratkaista yhä laajempia ja kompleksisempia ongelmia (Wing, 2008; Csizmadia et al., 2015). Ohjelmoinnillinen ajattelu on tulevaisuuden taito, joka kasvattaa oppilasta hahmottamaan ongelmia abstraktilla tasolla ja siten muotoilemaan ongelmia tietokoneen ratkaistavaksi. Ilman ohjelmoinnillista ajattelua emme pysty luomaan ongelmiin algoritmeja ja hyödyntämään tietokoneen laskentakapasiteettia laajojen ja kompleksisten ongelmien ratkaisemiseen.

Luvussa 2 esitelty AHAA-malli pitää sisällään neljä ohjelmoinnillisen ajattelun komponenttia, jotka ovat abstraktio, hajottaminen, algoritmi sekä automatisointi. AHAA-mallin komponentteihin voidaan liittää ohjelmointiosaamisen tavoitteiden kuvaukset Oikeus oppia -ohjelman Uudet lukutaidot -kehittämishankkeen julkaisuissa. Mallia käytetään digitaalisten oppimisympäristöjen arvioinnin lähtökohtana luvussa 4. Taulukkoon 3 on koottu digitaalisista oppimisympäirstöistä havaitut AHAA-mallin komponenttien esiintyminen. Havaitaan, että tutkielmassa esitellyt ilmaiset digitaaliset oppimisympäristöt Scratch, Code.org ja Khan Academy tukevat ohjelmoinnillisen ajattelun opettamista ongelmanratkaisun näkökulmasta. Digitaalisten oppimisympäristöjen arviointi AHAA-mallilla tarjoaa opettajalle lähtökohdan ohjelmoinnillisen ajattelun opettamiseen.

Lisäksi oppimisympäristöjä vertailtaessa havaitaan, että komponentit eivät esiinny samalla tavalla, koska Scratch ja Code.org ovat graafisia ohjelmointikieliä toisin kuin Khan Academyn käyttämä JavaScript.

Taulukko 3 Valitut oppimisympäristöt sisältävät AHAA-mallin komponentit.

Komponentti	Scratch	Code.org	Khan Academy
Abstraktio	Lohkojen ulkoasu, ohjelmoinnin käsitteet ja rakenteet (esim. silmukat)	Lohkojen ulkoasu, ohjelmoinnin käsitteet ja rakenteet (esim. silmukat)	Ohjelmoinnin käsitteet ja rakenteet (esim. silmukat)
Hajottaminen	Algoritmit eroteltavissa	Rajattuja osatehtäviä	Rajattuja osatehtäviä
Algoritmi	Valmiit lohkot	Valmiit lohkot, valmiita algoritmin runkoja	Valmiita algoritmin runkoja
Automatisointi	Tehokkuus lohkojen määränä, monistaminen	Yleinen ratkaisumuoto, jota voi käyttää useiden ongelmien ratkaisuissa	Yleinen ratkaisumuoto, jota voi käyttää useiden ongelmien ratkaisuissa

AHAA-malli on digitaalisten oppimisympäristöjen arviointityökaluna muistettava, mutta komponenttien esiintymisen arviointi ei riitä ohjelmoinnillisen ajattelun opettamiseen. Ohjelmoinnillisen ajattelun komponenttien ilmeneminen oppimisympäristössä kehittää ongelmanratkaisua, mutta ohjelmoinnillisen ajattelun hahmottamiseen ja opettamiseen opettajan tulee luoda tehtäviä ja kysymyksiä oppilaille. Tämä rajoittaa digitaalisten oppimisympäristöjen hyödyntämistä ohjelmoinnillisen ajattelun opetuksessa.

Pelkän ohjelmoinnin lisäksi peruskouluissa tulisi opettaa ajattelutaitoja, jotta oppilas pystyy saavuttamaan opetussuunnitelman mukaiset laaja-alaisen osaamisen tavoitteet (Opetushallitus, 2014). Opettamisen ja arvioinnin helpottamiseksi tulisi luoda tarkemmat kriteerit ei pelkästään ohjelmoinnin vaan myös ohjelmoinnillisen ajattelun tavoitteisiin. Nyt yksittäisen opettajan vastuulle jää arvioida oppilaan osaamista. Tämä on osittain ongelmallista, koska oppilaiden ohjelmoinnillisen ajattelun kehittymistä graafista ohjelmointisovellusta käyttämällä on vaikea arvoida luotettavasti (Brennan & Resnick, 2012). Näin ollen ohjelmoinnillisen ajattelun opettaminen Scratch ja Code.org oppimisympäristöissä on rajoitettua.

Jotta tvt-osaaminen ja monilukutaito täyttyisivät laaja-alaisina taitoina, ohjelmoinnillista ajattelua olisi hyvä opettaa myös muissa oppiaineissa kuin matematiikassa ja käsitöissä. Ajattelutaitona ohjelmoinnillinen ajattelu ei koske pelkästään matematiikkaa, vaikka yhtäläisyyksiä ongelmanratkaisusta ja loogisesta päättelystä löytyykin. Aiheen tutkimusta voisi jatkaa tarkempaan empiiriseen analyysiin ohjelmoinnillisen ajattelun opetuksen tilasta Suomen peruskouluissa. Myös nykyisten ohjelmoinnin ja ohjelmoinnillisen ajattelun tavoitteiden tutkiminen ja tarkempien sisältöjen laatiminen ohjelmoinnillisen ajattelun opettamiseen vastaisivat paremmin tulevaisuuden tietotekniikan työvoimapulaan (Teknologiateollisuus, 2018).

6. LÄHDELUETTELO

Barr, D., Harrison, J. & Conery, L. (2011). Computational thinking: A digital age skill for everyone Learning & Leading with Technology, (38), pp. 20-23 Saatavissa: https://files.eric.ed.gov/fulltext/EJ918910.pdf

Barr, V. & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? ACM Inroads 2, 1 48–54. Saatavissa: https://doi.org/10.1145/1929887.1929905

Brennan, K. & Resnick, M. (2012) New frameworks for studying and assessing the development of computational thinking. Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada Saatavissa: http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S. & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. Review of Educational Research, 87(4), 834–860. Saatavissa: https://doi.org/10.3102/0034654317710096

Cansu, F. K. & Cansu, S. K. (2019). An Overview of Computational Thinking. International Journal of Computer Science Education in Schools, 3(1), 17–30. Saatavissa: https://doi.org/10.21585/ijcses.v3i1.53

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. & Woollard, J. (2015). Computational thinking - a guide for teachers. Saatavissa: https://eprints.soton.ac.uk/424545/1/150818_Computational_Thinking_1_.pdf

Code.org, Oppimis- ja ohjelmointiympäristö, Internet-sivu. Saatavissa: https://code.org Viitattu 12.12.2021

Foord, M (2017). 30 best practises for software development and testing. Saatavissa: https://opensource.com/article/17/5/30-best-practices-software-development-and-testing

ISTE, Standards for computational thinking. Saatavissa: https://www.iste.org/standards/iste-standards-for-computational-thinking Viitattu 12.12.2021

ISTE & CSTA. (2011). Operational definition of computational thinking for K-12 education. Saatavissa: https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf

Khan Academy, Oppimis- ja ohjelmointiympäristö, Internet-sivu. Saatavissa: https://www.khanacademy.org/coach/dashboard Viitattu 12.12.2021

Koivisto, J. (2019). Algoritmisen ajattelun kehittäminen ohjelmoinnin ja kielentämisen avulla matematiikan opetuksessa. Pro gradu, Tampereen yliopisto. Saatavissa: https://urn.fi/URN:NBN:fi:tuni-201907162616

Lahtinen, E. Ala-Mutka, K. & Järvinen, H-M. (2005). A study of the difficulties of novice programmers. SIGCSE Bull. 37, 3, 14–18. Saatavissa: https://doi.org/10.1145/1151954.1067453

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. & Werner, L. (2011). Computational thinking for youth in practice. ACM Inroads. 2. 32-37. Saatavissa: https://doi.org/10.1145/1929887.1929902

Lodi, M. & Martini, S. (2021). Computational Thinking, Between Papert and Wing. Science & Education. 30. Saatavissa: 10.1007/s11191-021-00202-5.

Liukas, L. & Mykkänen, J. (2014). KOODI2016; Ensiapua ohjelmoinnin opettamiseen peruskoulussa. Saatavissa: https://s3-eu-west-1.amazonaws.com/koodi2016/Koodi2016_LR.pdf

Mannila, L., Settle, A., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., & Rolandsson, Lennart. (2014). Computational Thinking in K-9 Education. ITiCSE-WGR 2014 - Working Group Reports of the 2014 Innovation and Technology in Computer Science Education Conference. Saatavissa: https://www.researchgate.net/publication/273772180_Computational_Thinking_in_K-9_Education

Marttala, L. (2017). Tietotekniikan valtakunnallisten oppisisältöjen toteutuminen Keski-Suomen peruskoulujen opetuskäytänteissä. Pro gradu, Jyväskylän yliopisto. Saatavissa: https://jyx.jyu.fi/handle/123456789/55992

Opetus- ja kulttuuriministeriö (2021). Uudet lukutaidot -kehittämishankke. Saatavissa: https://uudetlukutaidot.fi/osaamisen-kuvaukset/ohjelmointiosaaminen/

Opetushallitus (2014). Perusopetuksen opetussuunnitelma. Saatavissa: https://www.oph.fi/sites/default/files/documents/perusopetuksen_opetussuunnitelman_perusteet_2014.pdf

Opetushallitus (2021). Perusopetuksen opetussuunnitelman ydinasiat. Saatavissa: https://www.oph.fi/fi/koulutus-ja-tutkinnot/perusopetuksen-opetussuunnitelman-ydinasiat

Scratch, Oppimis- ja ohjelmointiympäristö, Internet-sivu. Saatavissa: https://scratch.mit.edu Viitattu 12.12.2021

Selby, C. & Woollard, J. (2013). Computational thinking: the developing definition. University of Southampton. Saatavissa: https://eprints.soton.ac.uk/356481/

Sormunen, K. (2019). 6.-luokkalaisten ohjelmoinnin osaaminen alakoulun päättyessä. Pro gradu, Jyväskylän yliopisto. Saatavissa: https://jyx.jyu.fi/bitstream/handle/123456789/64698/1/URN%3ANBN%3Afi%3Ajyu-201906193288.pdf

Tanhua-Piiroinen, E., Kaarakainen, S., Kaarakainen, M., Viteli, J., Syvänen, A. & Kivinen, A. (2018). Digiajan peruskoulu. Opetus- ja kulttuuriministeriö, Saatavissa: http://urn.fi/URN:ISBN:978-952-287-634-8

Tanhua-Piiroinen, E., Kaarakainen, S., Kaarakainen, M. & Viteli, J. (2020). Digiajan peruskoulu II. Opetus- ja kulttuuriministeriö, Saatavissa: http://urn.fi/URN:ISBN:978-952-263-823-6

Teknologiateollisuus (2018). 9 ratkaisua Suomelle - Teknologiateollisuuden koulutus ja osaaminen -linjaus 2018, Helsinki, Saatavissa: https://teknologiateollisuus.fi/sites/default/files/file_attachments/teknologiateollisuus_koulutus_ja_osaaminen_linjaus_2018.pdf

Valtioneuvosto (2020). Digiajan peruskoulu II, (2020). Vuoden 2020 loppuraportin päätulokset ja kokonaiskuva perusopetuksen digitalisaatiosta, Opetus- ja kulttuuriministeriö, Saatavissa: http://urn.fi/URN:ISBN:978-952-263-835-9

Wing, J. (2006). Computational thinking. Commun. ACM 49, 3 (March 2006), 33–35. Saatavissa: https://doi.org/10.1145/1118178.1118215

Wing, J. (2008). Computational thinking and thinking about computing. Philosophical transactions. Series A, Mathematical, physical, and engineering sciences. 366. 3717-25. Saatavissa: 10.1098/rsta.2008.0118

Wing, J. (2017). Computational thinking's influence on research and education for all. Italian Journal of Educational Technology, 25(2), 7-14. Saatavissa: doi: 10.17471/2499-4324/922