Joona Laitinen

Ohjelmoinnillinen ajattelu ja sen haasteet opetuksessa

Tietotekniikan Kandidaatintutkielma

8. lokakuuta 2022

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Joona Laitinen

Yhteystiedot: joona.j.laitinen@student.jyu.fi

Ohjaaja: Ville Arvio

Työn nimi: Ohjelmoinnillinen ajattelu ja sen haasteet opetuksessa

Title in English: Computational thinking and its challenges within education

Työ: Kandidaatintutkielma

Opintosuunta: Kaikki opintosuunnat

Sivumäärä: 32+0

Tiivistelmä: Tämä kandidaatintutkielma käsittelee ohjelmoinnillista ajattelua sekä sen opetusta ja haasteita opetustyöhön liittyen. Ohjelmmoinnillisen ajattelun käsitettä avataan ja pyritään ymmärtämään, mistä ajattelussa on kyse. Opetusta ja oppimista tapahtuu monin eri tavoin, niin formaalissa koulumaailmassa kuin sen ulkopuolellakin. Opetuksen tämänhetkistä tilaa tullaan käymään läpi. Myös suomalaiseen opetuskenttään kurkistetaan. Ohjelmoinnillisen ajattelun opetus kohtaa myös haasteita, joita tutkielman loppupuolella pyritään erittelemään. Tutkielman tarkoitus on tuoda ohjelmoinnillisen ajattelun käsitettä kuuluvammaksi ja lähestyttävämmäksi. Digitalisoituvan maailman nuoret tarvitsevat uusia taitoja ja tavoitteena on tämän tutkielman kannalta valottaa näistä yhtä.

Avainsanat: Ohjelmoinnillinen ajattelu, opetus, haasteet

Abstract: This bachelor's thesis is about computational thinking, its teaching, and its challenges in education. History and the definition of the concept computational thinking will be clarified. What is it all about? Situation in the field of formal and informal learning will be reviewed, paying attention to the Finnish system especially. Issues and challenges towards computational thinking's education is being analyzed lastly. The aim of this thesis is to make concept of computational thinking commonplace. Youths in our digitalized world need new skills and this thesis examines one of those.

Keywords: Computational thinking, education, challenges

Esipuhe

Oi voi. Kirjoittaessani tätä esipuhetta, minulla on todella epätodellinen olo. En olisi koskaan uskonut saavani tätä projektia valmiiksi. Totta puhuakseni, en olisi koskaan uskonut tulevani, saatika pääseväni yliopistoon opiskelemaan. Lukion läpi kahlaaminen oli allekirjoittaneelle jo melko vaativa suoritus. Täällä sitä nyt ollaan ja vaikka aikataulusta on aika ajoin lipsuttu, niin aloitettu projekti on nyt maalissa. Aiheeksi valikoitui ohjelmoinnillinen ajattelu lähes itsestään, sillä siinä yhdistyi sekä kasvatuksellinen- että tietotekninen maailma. Tietokoneet saattavat välillä olla julmia ja kylmiä kapistuksia, mutta kasvatustieteellisestä näkökulmasta toimintaan saa huomattavan määrän humaania näkökulmaa. Vaikka kuinka hassulta saattaa kuulostaa, niin tällainen tekstimäärä on minunlaiselle, kasvatustieteelliselle tietoteknikolle oikeasti paljon. Keväällä puiden puhjetessa kukkaan aloitettu projekti on nyt lehtien tippuessa saatu maaliin. Aika aikaa kutakin ja lopussa kiitos seisoo.

Tahdon kiittää ensinnäkin omia vanhempiani siitä, että olen nyt tässä. Ystäviäni siitä, että olette puskeneet ja tsempanneet minua eteenpäin tämän projektin loppuunviemisen kanssa. Ohjaajaa Villeä siitä, että kärsivällisyytesi on kestänyt. Kiitokset myös kaikille vanhoille opettajilleni ja ohjaajilleni, jotka ovat muovanneet minusta tällaisen kun tänä päivänä olen.

Jyväskylässä 8. lokakuuta 2022

Joona Laitinen

Termiluettelo

Algoritmi	Yksinkertainen kuvaus ohjelman, tapahtuman tai asian toiminnasta. Järjestyksessä tapahtuvat vaiheet ovat algoritmin osia.
Tietojenkäsittely	Tietojenkäsittely (engl. computing) on tietokoneella tapahtuvaa toimintaa. Esimerkiksi laskentaa ja informaation prosessoimista, jolla pyritään saavuttamaan haluttu lopputulos.
K-12	K-12 opetuksella tarkoitetaan perusopetusta Yhdysvalloissa Kindergarten to the Twelfth.
OA	Ohjelmoinnillinen ajattelu, englanniksi CT. Tutkielmassa käytettävän toiston takia lyhennetty termi.
STEM	Science, Technology, Engineering, Mathematics

Kuviot

Sisällys

1 Johdanto

Maailman kehittyessä kokoajan teknologisempaan suuntaan, arkipäivissä tarvittavat taidot muuttuvat mukana. Oletko koskaan joutunut esimerkiksi miettimään tai laskemaan, kuinka monta kertaa sukset kannattaa hiihtolomalla vuokrata, ennen kuin niiden ostamisesta tulisi jo halvempaa? Olisiko legot kerättävissä lastenhuoneen lattialta järjestelmällisemmin, jotta vastaisuudessa oikean palan löytyminen kävisi helpommin? Oletko neuvonut joskus ystävääsi pääsemään perille? Kun tulet pääkatua, niin käännyt seuraavasta vasemmalle ja jatkat kunnes näet punaisen auton, jonka takaa käännyt pihalle. Nämä kaikki ovat esimerkkejä ohjelmoinnillisesta ajattelusta.

Mutta kuinka tällaista ajattelutapaa tulisi kehittää? Minkälaisia haasteita piilee ohjelmoinnillisen ajattelun opetustyössä? Tässä kandidaatintyössä tullaan selventämään lukijalle tarkemmin käsitettä ohjelmoinnillinen ajattelu ja ensimmäisessä luvussa perehdytään ainoastaan tähän. Käsite on verrattain uusi, eikä suomenkielistä tutkimustietoa vielä paljoa ole. Yleisesti asiasta on kuitenkin kyllä kirjoitettu ja ohjelmoinnillinen ajattelu on tällä hetkellä ajankohtainen aihe, mistä kielii esimerkiksi Scopus-tietokannasta kasvavaan tahtiin vuosi-vuodelta löytyvät julkaisut. Suomen kohdalla esimerkkinä viimeisin opetussuunnitelma, josta löytyy mainintaa algoritmisesta ajattelusta ja ohjelmoinnin sisällyttämisestä perusopetukseen. Käsite itsessään on monisyinen, joten lähdetään liikkelle siitä, että sitä selvennetään ensin. Toisessa luvussa siirrytään tutkimaan koulumaailmaa ja ohjelmoinnillisen ajattelun opetustyötä, tutkitaan myös miten käsite näkyy tämän hetkisessä opetussuunnitelmassa Suomessa. Kolmas luku käsitteleekin sitten opetukseen liittyviä haasteita. Tutkielman tarkoitus on kirjallisuuskatsauksen tavoin pyrkiä vastaamaan kysymyksiin:
1) mitä ohjelmoinnillinen ajattelu on?
2) miten sitä tällä hetkellä opetetaan? ja
3) millaisia haasteita opetustyöhön liittyy?

Kysymyksiin vastattua siirrytään yhteenvetoon ja summataan kaikki käsitelty teksti. Mietitään myös mihin jatkossa voisi suunnata tutkimusta ja minkälaisia kysymyksiä kirjoittaessa heräsi.

2 Ohjelmoinnillinen ajattelu

Ohjelmoinnillisen ajattelun englanninkieliselle käsitteelle ei ole tarkkaa suomenkielistä vastinetta, vaikka aihe on jo verrattain vanha. Denning (2017) kirjoittaa käsitteen juontuvan aina tietojenkäsittelytieteiden alkujuurille asti, 1950-luvulle. Suomeksi usein käytetään synonyymeina käsitteitä kuten algoritminen ajattelu, ohjelmoinnillinen ajattelu tai laskennallinen ajattelu. Esimerkiksi *Perusopetuksen opetussuunnitelman perusteet* ei mainitse käsitettä "ohjelmoinnillinen ajattelu" kertaakaan, kun taas "algoritminen ajattelu" tulee mainituksi (Opetushallitus 2014).

Englanniksi käsite on "computational thinking", jonka J. M. Wing (2006) artikkelissaan toi jälleen yleisön tietoisuuteen. Wingin (2006) mukaan ohjelmoinnillinen ajattelu antaa meille ymmärrystä kehittää ohjelmistoja ja ratkaista ongelmia tietokoneita hyödyntäen, joita olisi yksinään hankala ratkaista. OA antaa myös vastauksia siihen, mikä on ohjelmoitavissa. Missä tietokone on parempi, kuin ihminen ja toisinpäin. Vaikka Wing (2006) painottaa OA:n olevan perustavanlaatuinen tietotaito ihan jokaiselle, ei ainoastaan teoreettiselle tietojenkäsittelijälle, niin tarkkaa merkitystä käsitteelle ei esitetä. Käsitteillä tulee tieteellisessä maailmassa kuitenkin olla tarkat merkitykset. Aho (2012) kirjoittaa terminologian ymmärtämisen olevan oppimisen edellytys ja jotta voimme keskustella toistemme kanssa asioista selvemmin on termeillä oltava tarkka määritelmä. Tietojenkäsittelyn tiedemaailman viitekehyksessä määritelmillä on vielä entistäkin tärkeämpi rooli, kun tarkoituksena on rakentaa rajapintojen läpi sulavasti, käyttäjän ja toisten komponenttien kanssa, keskustelevia ohjelmistoja ja laitteistoja (Aho 2012). Vaikka jotkin aiheestakirjoittajat ja kantaaottavat tahot saattavat väittää, että tarkka määritelmä käsitteelle ei ole tarpeellinen, niin useammin on lähdekirjallisuudessa painotettu sen tarvetta (Selby ja Woollard 2013).

Artikkelissaan J. M. Wing (2006) ottaa kuitenkin kantaa hyvin kriittisesti sen hetkiseen tilanteeseen opetuksen tulevaisuuden suhteen ja kantava pääteesi artikkelissa onkin "kaikille ja kaikkialle". OA:n tulisi pyrkiä tavoittamaan kaikki, sillä ajattelutavasta on hyötyä kaikille. Ohjelmointiin ja tietojenkäsittelytieteeseen kokoajan enenevissä määrin nojautuva maailmamme on ymmärrettävissä ohjelmoinnillisen ajattelun kautta. Tämä herättikin ohjelmoinnillisen ajattelun käsitteen jälleen henkiin. Se keräsi paljon huomiota, sillä opettajat ja ohjaa-

jat ottivat tehtäväkseen auttaa lapsia tulemaan eteviksi laskennan ja tietojenkäsittelyn hyödyntäjiksi osana STEM-opetusta (Denning 2017). Myöhemmin Wing on myös artikkelissaan "Computational thinking, 10 years later" (2016) kommentoinut, ettei olisi voinut uskoakaan hänen elämänsä aikana K-12 opetuksen sisältävän tietojenkäsittelyn opetusta, ja on nykyisin onnellinen ollessaan väärässä tämän suhteen.

Wing aloitti siis jotain sellaista, mitä ei enää voinut pysäyttää. Hän on myös myöhemmin kirjoittanut asiasta uudestaan ja pyrkinyt selittämään käsitettä tarkemmin. J. Wing (2011) kirjoittaa ohjelmoinnillisen ajattelun olevan ajatusprosessi, johon kuuluu ongelmien muotoilu sellaiseen muotoon, että niiden ratkaisut ovat tehokasta ratkaista tietokonetta hyödyntäen. Myös Aho (2012) kuvaa OA:ta ajatusprosessiksi, johon kuuluu ongelmien muotoilu sellaiseksi, että ne voidaan esittää tietokoneellisin askelin ja algoritmein. Tärkeänä osana tätä tapahtumasarjaa on löytää yhteensopivat ja oikeat mallit, sekä tavat keskustella tietokoneelle.

Määritelmä on edelleen melko yleismaallinen ja herättää kysmyksen siitä, millaista tämä ongelmien muotoilu on? Millainen ajatusprosessi vaaditaan, että ongelmista tulee tietokoneystävällisiä? Vastauksena tähän Selby ja Woollard (2013) kirjoittivat aiheesta kirjallisuuskatsauksen, pyrkien käsittelemään kaikkea, mitä tapahtui vuoden 2006 jälkeen. Minkälainen olisi tarkempi kuvaus käsitteelle computational thinking - ohjelmoinnillinen ajattelu. Seuraavaksi tullaan käsittelemään aluksi läpi viisi termiä ja tutustutaan tarkemmin käsitteeseen näiden termien kautta. Luvun lopuksi käsite kiteytetään.

1. Abstraktio
2. Ongelman jakaminen pienempiin osiin (decomposition)
3. Algoritminen ajattelu
4. Evaluointi
5. Yleistäminen

2.1 Abstraktio

Ohjelmoinnillisen ajattelun perusolemus on abstraktio (J. M. Wing 2008). Kramer (2007) jakaa abstraktion kahteen erityisen olennaiseen näkökulmaan. Ensimmäisessä abstraktio tarkoittaa tapahtumaa tai prosessia, jossa poistetaan tai jätetään huomioimatta yksityiskohtia yksinkertaistaakseen asiaa. Myös Curzon ym. (2014) kirjoittavat abstraktion olevan tapa tehdä ongelmista ja järjestelmistä helpommin ymmärrettäviä piilottamalla tarpeetonta kompleksisuutta. Taito piilee siinä, että osaa tehdä päätöksen yksityiskohdan merkkittävyyden kannalta. Ongelmasta tulee tulla abstraktion kautta helpommin ymmärrettävä, ilman että mitään tärkeää menetetään (Curzon ym. 2014). Kramerin toinen näkökulma on se, että abstraktiolla voidaan tarkoittaa yleistämisprosessia, jonka avulla pyritään löytämään jotakin yhteistä asioiden välillä (Kramer 2007). Kohdassa 2.5 yleistämiseen paneudutaan tarkemmin.

Hyvä esimerkki abstraktiosta voidaan tehdä suoraan tietokoneen toimintalogiikasta. Ohjelmiston - (software) ja laitteiston - (hardware) välillä kulkee bittejä. Yhdellä bitillä voidaan ilmaista toista kahdesta tilasta “1 tai 0”, “kyllä tai ei”, “tosi tai epätosi”. Konekieli koostuu täysin biteistä, toisin sanoen tietokoneen toiminta perustuu vain ykkösiin ja nolliin. Vaativampien käskyjen antaminen tietokoneelle konekielellä olisi ihmiselle käytännössä mahdotonta, joten tätä varten tarvitsemme korkeampaa abstraktiotasoa, ohjelmointikieliä (Colburn ja Shute 2007). Tarpeeton kompleksisuus piilotetaan ja toiminnasta tulee helpommin ymmärrettävää.

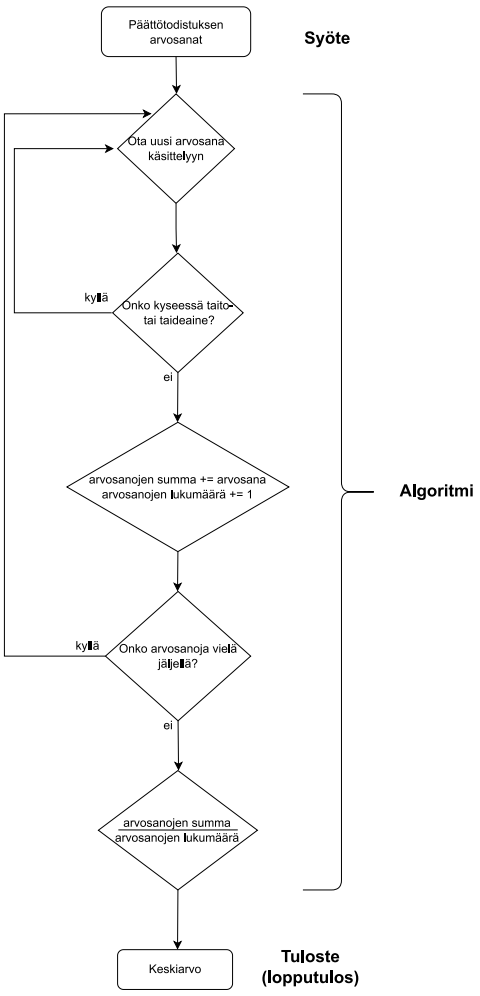
2.2 Ongelman jakaminen pienempiin osiin (decomposition)

Rich, Egan ja Ellsworth (2019) kirjoittavat pienempiin osiin jakamisen olevan hyvin yleinen tapa ja askel ongelmanratkaisussa eri aloilla. Tietojenkäsittelyn osa-alueella käsitteellä tarkoitetaan ongelman pilkkomista osiin joko rakenteen, funktioiden, järjestyksen tai riippuvuuden suhteen ja jokainen tavoista on tehokas tietyissä tilanteissa, tiettyjen olosuhteiden vallitessa (Rich, Egan ja Ellsworth 2019). Isot ongelmat voidaan siis purkaa pienempiin osiin. Näiden osa-ongelmien ratkaisu on helpompaa ja täten hankalaltakin tuntuvat suuret ongelmat saadaan ratkaistua, sekä laajempia kokonaisuuksia suunniteltua (Curzon ym. 2014).

2.3 Algoritmit

Algoritmi on rakennuspalikka tietojenkäsittelytieteessä (Hill 2016). J. Wing (2011) määrittelee algoritmin käsitteen siten, että algoritmi on abstraktio prosessista, jossa käsitellään syötettä seuraamalla tiettyjä askeleita peräjälkeen. Prosessin lopputulos muovautuu askeleita seuraten ja päämäärä saavutetaan. Algoritminen ajattelu on tapa saavuttaa haluttu lopputulos seuraamalla tarkasti määriteltyjä vaiheita (Curzon ym. 2014).

Turingin kone tarjoaa käsitteelle "algoritmi" myös tarkan määritelmän (Aho 2012). Yksinkertaisuudessaan algoritmi funktiolle f on toinen turingin kone, joka laskee f:n. (Aho 2012). Tässä tutkielmassa ei turingin konetta avata käsitteenä tarkemmin, vaan pysytään algoritmisen ajattelun ja algoritmien osalta askel-askeleelta tapahtuvassa määrittelyssä (kts. Kuvio 1).



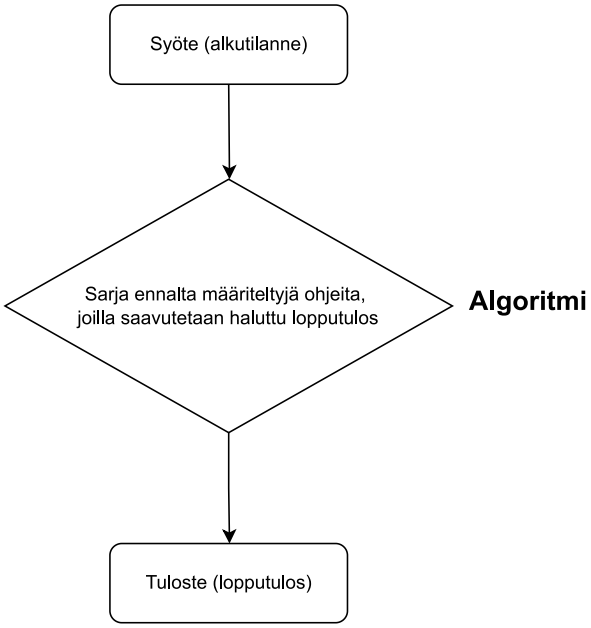
Kuvio 1. Havainnollistava vuokaavio siitä, mikä on algoritmi.

2.4 Evaluointi

Evaluoinnilla tarkoitetaan prosessia, jossa arvioidaan ratkaisun hyvyyttä ja punnitaan tehtyjä valintoja. Onko ratkaisu esimerkiksi tarpeeksi nopea, kustannustehokas tai käyttäjäystävällinen ja mitä halutaan painottaa. Usein on tehtävä myös kompromisseja. (Curzon ym. 2014). Esimerkiksi terveydenhoitoalalla ei voida joustaa tietoturvallisuudesta potilastietojen salassapidon takia, mikä voi näkyä kustannustehokkuudessa. J. M. Wing (2006) mainitsee myös vaihtokauppojen ja valintojen olevan tärkeä osa ohjelmoinnillista ajattelua. Ohjelmoinnillinen ajattelija tekee valintoja esimerkiksi ajan ja tilan välillä, tai suorituskyvyn ja tallennuskapasiteetin välillä (J. M. Wing 2006). Evaluate kääntyy englannista suomeksi - “arvioida” (*MOT Englanti,* n.d.).

2.5 Yleistäminen

Yleistäminen mainittiin jo aiemmin abstraktion yhteydessä ja J. Wing (2011) kirjoittaakin yleistämisen olevan osa abstraktiota. Yleistämistä voidaan käyttää myös uusien ongelmien ratkaisemisessa (Curzon ym. 2014). Yleistetään ongelma ja hyödynnetään jo aiemmin ratkaistuja ongelmia, sekä niihin käytettyjä tapoja (Curzon ym. 2014). Käsite “yleistäminen” ei ole kirjallisuudessa kuitenkaan usein esiintynyt (Selby ja Woollard 2013). Huomionarvoista voisikin olla, että se jää useimmiten abstraktio-käsitteen takia piiloon, kuten tässäkin tutkielmassa on juuri huomattu. Huolimatta termin käytön vähäisyydestä kirjallisuudessa, on näytteitä vastaavista käsitteistä, joita tutkimuskirjallisuudessa käytetään (Selby ja Woollard 2013). Termin kuitenkin ollessa helposti määriteltävissä ja tarkka, on se lisätty täsmentämään OA:n käsitettä (Selby ja Woollard 2013). Kuviossa 1 havainnollistettiin keskiarvon laskemiseen sopivaa algoritmia. Yleistämällä voidaan hyödyntää algoritmin käsitettä ja algoritmista ajattelua muuallakin, katso Kuvio 2.



Kuvio 2. Algoritmin käsite yleistettynä.

2.6 Ohjelmoinnillisen ajattelun tarkka määritelmä

Ohjelmoinnillisen ajattelun tarkka määritelmä, jota Selby ja Woollard (2013) ehdottavat, koostuu edellä mainituista alaluvuista. Tarkemman määritelmän kautta pystymme ainakin paremmin jäsentämään ongelmia, ajatustyötä ja työkaluja, joita ohjelmoinnillisessa ajattelussa kohdataan ja tarvitaan. Kuitenkaan tämä Selbyn ja Woollardin (2013) ehdottama näkemys ei ole täysin jaettu ympäri maailman ja Shute, Sun ja Asbell-Clarke (2017) kirjoittavat käsityksen ohjelmoinnillisen ajattelun määritelmästä olevan edelleen todella häilyvä. Ohjelmoinnillinen ajattelu esimerkiksi sekoittuu edelleen usein käsitteiden “tietojenkäsittelytiede” ja “ohjelmointi” kanssa (Czerkawski ja Lyman 2015). Tässä tutkielmassa pitäydytään ajatuksessa, että ohjelmoinnillinen ajattelu on paljon muutakin kuin taito käyttää tietokonetta tai ohjelmoida tietokoneohjelmia (J. M. Wing 2006). Ohjelmoinnillinen ajattelu on looginen tapa ajatella ja jäsentää omaa ongelmanratkaisua (Shute, Sun ja Asbell-Clarke 2017). Kun käsitteen saa edellämainitun tavoin jaettua pienempiin osa-joukkoihin ja palasiin voidaan kehittää arviointityökaluja, joilla ohjelmoinnillisen ajattelun taitoja voidaan mitata (Selby ja Woollard 2013).

3 Ohjelmoinnillisen ajattelun opetus

J. M. Wing (2006) kirjoitti artikkelissaan, että ohjelmoinnillisen ajattelun tulee tavoittaa muitakin kuin tietojenkäsittelytieteiden opiskelijoita. Aivan kuten kirjoitus- ja lukutaito sekä aritmetiikka ovat jokaisen nuoren perusosaamista, tulisi ohjelmoinnillinen ajattelu lisätä osaksi tätä osaamista. Nuori olisi täten kykenevämpi toimimaan tulevaisuuden haasteita vastaan. OA on täysin verrattavissa muihin kognitiivisiin kykyihin, joita tavallisen henkilön kuvitellaan omaavan tässä yhteiskunnassa (Council 2010). Asia tarkentui, kun J. M. Wing (2008) kirjoitti artikkelissaan OA:n tulevan koskemaan kaikkia alasta riippumatta ja luomaan paljon uusia aloja. Jos haluamme taata ohjelmoinnillisen ajattelun perustaitoja jokaiselle kansalaiselle on oppimisen hyvä tapahtua varhaisessa lapsuudessa.

Tullaan keskittymään opetuksen suhteen tässä kandidaatintutkielmassa perusopetukseen, koska se on yhdistävä ja tärkeä tukipilari kaikille yhteiskunnan jäsenille. Esimerkiksi Suomessa perusopetus tarjoaa jokaiselle samanlaiset valmiudet jatko-opintoja varten ja tukee oppilaan kasvua ihmisenä ja yhteiskunnan jäsenenä. Kaikille tarpeelliset tietotaidot kartutetaan perusopetuksen piirissä POPS (2014). Englanninkielisessä tiedekirjallisuudessa peruskoulutukselle synonyymeina toimivat amerikkalaisesta koulujärjestelmässä K-9- ja K-12-opetus. K-9 eli kindergarten to 9th on lähempänä suomalaista järjestelmää, jossa perusopetus kestää yhdeksänteen vuosiluokkaan asti. K-12-opetus on taas erilainen konsepti, jossa koulu aloitetaan jo “kindergartenissa” eli esikoulussa jatkuen aina 12:nteen vuosiluokkaan asti. Molemmat ovat toimivia käsitteitä kuvaamaan ennen korkeakoulua tapahtuvaa opetusta. K-12-opetus kattaa esimerkiksi Suomen kontekstissa myös lukiokoulutuksen. K-9-opetus toimii tarkempana käsitteenä perusopetukselle Pohjoismaisessa järjestelmässä, ja sitä termiä käytetään myös akateemisessa kirjallisuudessa (Heintz ym. 2017; Fagerlund ym. 2021).

Tässä luvussa tullaan tarkastelemaan ohjelmoinnillisen ajattelun nykytilaa opetuksessa ja sitä, miten ohjelmoinnillisen ajattelun ideaa pyritään peruskoulutasolla opettamaan. Formaalin opetuksen lisäksi myös koulun ulkopuolella tapahtuva oppiminen on merkittävässä osassa ohjelmoinnillisen ajattelun kehityksen suhteen.

3.1 Opetuksen nykytila

J. Wing (2011) kirjoitti innostuksen ja kiinnostuksen OA:n ympärillä nousseen ja aiheen sisällyttäminen peruskouluopetuksen alkoi olla nähtävissä. Myös OA:n käsite koulutuksen suhteen alkoi tällöin levitä ympäri maailman. Tästä on kulunut jo vuosikymmen ja onkin aika tutkia, miltä tilanne vaikuttaa nyt.

3.1.1 Ohjelmointi

Ohjelmoinnista on tullut nykyisin osa peruskoulun opetussuunnitelmaa (Fagerlund ym. 2021). Onkin siis hyvä ymmärtää myös ohjelmoinnin merkitys ohjelmoinnillisen ajattelun suhteen. Ohjelmoinnin osaaminen voi edesauttaa OA-taitojen syntymistä (Mannila ym. 2014). Ohjelmointia harjoittaessaan oppilaat altistuvat ajattelemaan ohjelmoinnillisesti ja tämä on innostanut tutkijoita sekä opettajia tutkimaan, kuinka ohjelmoinnin avulla voidaan parantaa OA:n opetusta luokkahuoneympäristöissä (Lye ja Koh 2014). Ohjelmointi onkin koulutuksen näkökulmasta paljon laajempi käsite kuin aiemmin. Se ei ole enää ainoastaan opittavissa oleva yleistaito, vaan ohjelmoinnilla voidaan nähdä olevan myös muita ulottuvuuksia (Mannila ym. 2014). Kafai ja Burke (2013) kirjoittavat sen olevan esimerkiksi sosiaalista ja yhteisöllistä toimintaa, kun vaikka kirjoitamme ohjelmia toisillemme jaettavaksi ja näytettäväksi tai sekoitamme yhteistä lähdekoodia. Ohjelmointi voidaan nähdä nykyisin myös tapana ilmaista itseään ja kehittää omaa luovuttaan (Resnick ym. 2009).

Ohjelmointi on paljon muutakin, kuin koodaamista (Lye ja Koh 2014). Koodaus on vain murto-osa ohjelmointiurakasta, tietokoneelle käskyjen antamista kaiken ajatustyön - ohjelmoinnillisen ajattelun - jälkeen (Mannila ym. 2014). Jotta Wingin (2011) määrittelemä ajatusprosessi, ohjelmoinnillinen ajattelu, pääsisi työskentelyn ja opetuksen keskipisteeksi, on nämä kaksi käsitettä, “koodaus” ja “ohjelmointi”, pystyttävä erottamaan toisistaan.

Toisaalta OA:n ja ohjelmoinnin välinen suhde ei kuitenkaan ole suoraviivainen. Ohjelmointi osaltaan tarjoaa työkalut kehittää ohjelmoinnillista ajattelua, mutta ohjelmoinnillinen ajattelu taas parantaa ja antaa uusia mahdollisuuksia ohjelmointiin (Tikva ja Tambouris 2021). Kuten huomataan ohjelmoinnissa ja sen opetuksessa, etenkin OA:n näkökulmasta, piilee myös haasteensa. Käsitellään näitä tarkemmin luvussa neljä.

3.1.2 Formaali opetus

Formaalilla opetuksella tarkoitetaan muodollista, tarkoituksen mukaista ja tavoitteellista oppimista (Hautamäki 2009). Esimerkiksi opetussuunnitelmiin sisältyvä kouluopetus on formaalia oppimista. Formaali oppiminen on sinällään erittäin merkittävä oppimisen muoto, sillä opetus koskee valtaosaa yhteiskunnan jäsenistä. Jo nyt monet maat ovat ottaneet ohjelmoinnillisen ajattelun opetuksen osaksi kansallista opetussuunnitelmaa ja OA:n opetuksesta sekä oppimisesta on tullut formaalin koululaitoksen kautta kaikille välttämätöntä (So, Jong ja Liu 2020). Koululaitoksen toiminnan keskiössä on kehittää oppijoiden peruskäsitystä maailmasta ja elämiseen liittyviä taitoja. Fagerlund (2022) kirjoittaa, että koulu ja formaali opetus ei voi jättää yhteiskunnallisesti näinkin tärkeitä asioita, kuten ohjelmoinnillisia ilmiöitä huomioimatta.

Eri maissa on maailmalla erilainen tilanne, sillä koulujärjestelmät ympäri maailman toimivat eritavoin. Vitikka ja Hurmerinta (2011) kirjoittavat esimerkiksi selvityksessään opetussuunnitelman olevan aina kulttuuri- ja yhteiskuntasidonnainen tuote ja keskeinen osa koulutusjärjestelmää jokaisessa maassa. Koulutusjärjestelmä kokonaisuudessaan kertoo paljon yhteiskunnasta ja sen tarpeista kasvattaa, sekä kouluttaa kansalaisensa. Järjestelmän kautta välittyvät yhteiskunnalliset arvot, perinteet ja käytännöt, eikä koulutusjärjestelmää suoranaisesti voi kopioida valtiosta ja kulttuurista toiseen. Rahoitus, lainsäädäntö, koulutuksen ohjaus, opettajankoulutus ja koulutuksen arviointi ovat näkökulmia, joilla järjestelmiä usein vertaillaan (Vitikka ja Hurmerinta 2011). Voi siis sanoa, että koulumaailma on erilainen ympäri maailman.

Jopa Suomen sisällä opetus vaihtelee koulukohtaisesti. Ainoastaan opetussuunitelmaan määritellyt velvoitteet tulee toteutua opetuksessa. Perusopetuksen opetussuunnitelman perusteet (2014) antavat ainoastaan yhtenäisen pohjan paikalliselle opetussuunnitelmalle, joka on koulukohtaisesti laadittavissa (Bocconi, Chioccariello ja Earp 2018). Opetuksen järjestäjä päättää myös, mitä valinnaisia aineita tarjotaan. Valinnaisia aineita on vähintään yhdeksän viikkotuntia, vuosiluokilla 1-9 (Opetushallitus 2014). Tietotekniikka voi olla yksi valinnaisista aineista, mitkä koulukohtaisesti järjestetään. Kaikista yleisin tapa onkin tuoda ohjelmoinnillista ajattelua opetukseen juuri tietotekniikan oppiaineen kautta. Tähän voidaan laskea mukaan erilliset kurssit, tunnit tai työpajat, missä aihetta painotetaan (Lockwood ja Mooney

2017).

Mannila ym. (2014) kartoittivat laaja-alaisessa katsauksessaan perusteellisesti K-9-tasoista opetusta OA:n suhteen. K-9-käsitteellä viitataan taas suomalaiseen peruskouluopetukseen, päiväkodista yhdeksänteen vuosiluokkaan. Formaalin opetuksen tilanne tulee ilmi katsauksessa muutamaan maahan kohdistuneen maakohtaisen raportin kautta. Mukana olleet maat ovat Liettua, Suomi, Ruotsi, Hollanti, Italia ja Yhdysvallat. Raportit antavat ymmärtää, että OA:n sisällyttäminen opetussunnitelmiin jokaisessa katsauksen maassa tuli mainituksi jollain tavalla. Selkeästi ohjelmointiin liittyviä mainintoja opetussuunnitelmissa oli vain harvoissa.

Kyseiset - edellä mainitut - maat edustavat laajalti länsimaista lähestymistapaa, mutta myös esimerkiksi Aasiassa on toteutettu samankaltainen katsaus. So, Jong ja Liu (2020) kirjoittavat katsauksessaan laajasti ohjelmoinnillisen ajattelun opetuksen tilasta Aasiassa. Heidän mukaansa Aasian asema ict-markkinoilla on herättänyt maat huomioimaan ohjelmoinnillisen ajattelun opetussuunnitelmien laatimisessa. Ottaen myös huomioon, että alueella asuu huomattava osa maailman väestöstä, on tämä alue mielenkiintoinen ja merkittävä kohde, kun käsitellään ohjelmoinnillisen ajattelun opetustyötä maailmanlaajuisesti.

Teknologiajätteinä Aasian maat kuten Kiina, Korea, Taiwan sekä erillisalue Hong Kong ovat ilmoittaneet opetussuunnitelman uudistuksista ohjelmoinnilliseen ajatteluun liittyen (So, Jong ja Liu 2020). Hong Kongissa OA on merkittävä osa ohjelmoinnin opetusta ja esimerkiksi Taiwanissa uusi opetussuunnitelma OA:n osalta julkistettiin vuonna 2019 (So, Jong ja Liu 2020). Opetussuunnitelma linjaa erityisesti OA:lle olennaisia osia, joita tulee koulutukseen sisällyttää. Luokilla 7-9 käytetään OA-konseptien opetukseen vähintään yksi viikkotunti, kun taas luokat 10-12 paneutuvat aiheeseen kaksi tuntia viikossa.

Vaikka työtä OA:n opetuksen suhteen on tehty, on opetus edelleen vielä täysin lähtötelineissään (Lockwood ja Mooney 2017). Uusien pedagogisten ajatusten tuominen osaksi formaalia koulumaailmaa on ensinnäkin usein hyvin hankalaa (Fagerlund 2022). Tarkkaa määritelmää ei ohjelmoinnilliselle ajattelulle ole saavutettu ja niimpä oppilaiden OA-taitoja on hankala mitata ja arvioida (Lockwood ja Mooney 2017). Monet maat eivät myöskään ole ottaneet OA:ta vielä osaksi opetussuunnitelmaansa kaikilla asteilla (Lockwood ja Mooney 2017).

Tietotekniikasta ei myöskään ole tullut valtavirtainen opetettava aine monissakaan maissa, vaikka tämän juuri voisi nähdä ilmeisenä paikkana opettaa OA-taitoja, kehitystä on kuitenkin nähtävissä (Lockwood ja Mooney 2017).

3.1.3 Informaali ja non-formaali oppiminen

Opettaja ja formaali koulutuslaitos eivät enää nykypäivänä ole kaiken oppimisen yksinoikeutettu valtias. Oppimisympäristön käsite on laajentunut koulumaailman ulkopuolelle ja nykyisin puhutaan oppimisen kaikkiallisuudesta. Oppimista tapahtuu kaikkialla ja kokoajan (Lonka 2015). Informaalilla ja non-formaalilla oppimisella viitataan muodollisen opetuksen ulkopuolella tapahtuvaan oppimiseen. Käsitteet ovat kuitenkin määritelmiltään eriävät ja nekin tulisi erottaa toisistaan. Non-formaali oppiminen voi olla esimerkiksi harrastuksissa tapahtuvaa tavoitteellista oppimista, kuten kirjaston atk-kerhossa ohjelmoinnin harrastamista. Informaalilla oppimisella taas tarkoitetaan suoranaisesti vapaa-ajalla, harrastusten ja koulumaailman ulkopuolella tapahtuvaa oppimista ja tilanteissa harjaantumista (Hautamäki 2009). Tästä esimerkkinä voisi toimia arjessa tapahtuva mahdollinen ajatusten yhteenloksahtaminen. Esimerkiksi kotona perheen kanssa vuorovaikuttaen tai vapaavalintaisia videoita katsellen. Informaali oppiminen eroaa edellisistä kahdesta, non-formaalista ja formaalista oppimisesta niin, että oppimistilanteessa ei ole ketään auktoriteettiasemassa olevaa välikättä (Eshach 2007). Informaali oppiminen saattaa myös usein olla tahatonta (Rogers 2014).

Käsitteet informaali ja non-formaali saattavat siis usein mennä sekaisin (Rogers 2014) ja tässäkin kandidaatintutkielmassa tullaan puhumaan yleisesti kaikesta koululaitoksen ulkopuolella tapahtuvasta oppimisesta. Selvyyden vuoksi on hyvä mainita nyansseista, jotka saattavat akateemisessakin tutkimuksessa joskus hälventyä.

Tulee siis ottaa myös huomioon OA:n opetuksen suhteen kaikki informaali ja non-formaali oppiminen ja opetus, sillä monille lapsille ja nuorille ensikohtaaminen OA:n kanssa tapahtuu formaalin koulumaailman ulkopuolella (Mannila ym. 2014). Tällaisiksi oppimistilanteiksi voidaan lukea Mannilan ym. (2014) mukaan esimerkiksi erinäiset kerhot, kilpailut, sekä erilaisten organisaatioiden järjestämät tapahtumat ja toimintapajat, joissa aiheeseen otetaan kantaa. Kerhot ovat useimmiten vapaaehtoisten ylläpitämiä, ohjelmointiin orientoituneita se-

kä ilmaisia ja ne voivat olla joko kansainvälisiä tai kansallisia (Mannila ym. 2014). Kerhot saattavat myös keskittyä erityisesti vähemmistöryhmien opettamiseen, kuten *Geek Girl Mini* tai *Made with code*. Ylipäätään kehittyvät informaalit OA-oppimisympäristöt kaventavat sukupuolijakaumaa tietotekniikan alalla (Grover ja Pea 2013). OA-opetuksen tukemiseksi järjestetään myös paljon kilpailuita ympäri maailman, joissa eri ikäiset ja taitotasoiset nuoret kilpailevat (Mannila ym. 2014). Osa kilpailuista on suunnattu selkeästi lahjakkaille nuorille, kuten *International Olympiad in Informatics*. Toisten tarkoituksena on taas pääosin esitellä tietojenkäsittelyä ja ohjelmoinnillista ajattelua nuorille kilpailun kautta syntyvän motivaation keinoin. Jälkimmäisestä voidaan mainita esimerkkinä *Bebras*, mikä Liettuasta lähtöisin oleva kansainvälinen kilpailu kaiken ikäisille lapsille. Bebras toimii tietojenkäsittelyn sekä ohjelmoinnillisen ajattelun puolestapuhujana ja sen tarkoitus on herättää eri ikäisille suunnitelluin tehtävin nuoret ajattelemaan ohjelmoinnillisesti (Mannila ym. 2014).

Myös nämä erinäiset organisaatioiden järjestämät tapahtumat ovat tärkeä osa non-formaalia oppimista. Tällaisten tapahtumien selkeä päämäärä on tuoda OA lähelle lapsia ja nuoria. Esimerkkinä tällaisesta *Hour of Code*, jonka idea on alunperin Yhdysvalloista. “Koodaustunnin” tarkoituksena on tarjota materiaaleja siihen, että kuka tahansa voi opettaa perusasioita ohjelmoinnillisesta ajattelusta. Tapahtumaa ovat tukeneet suuret yritykset, kuten Amazon, Microsoft ja Apple.

Tällaisissa formaalin opetuksen ulkopuolella tapahtuvissa tapahtumissa ja tilanteissa on paljon positiivista. Non-formaalin oppimisen järjestämisestä vastaavat ihmiset ovat usein intohimolla mukana levittämässä tietojenkäsittelyn ilosanomaa ja he ovat koulutettuja, tietävät tarkasti, mitä puhuvat ja opettavat. Usein tapahtumat herättävät myös paljon huomiota mediassa ja lasten vanhemmissa, mikä ylläpitää aiheen ajankohtaisuutta (Mannila ym. 2014). Haasteitakin tästä toiminnasta löytyy. Haasteita käsitellään kattavammin seuraavassa kappaleessa.

3.2 Opetuksen tilanne Suomessa

Suomalaisessa koulumaailmassa tietojenkäsittely oli osa lukion opetussuunnitelmaa vielä 2000-luvun alussa, mutta sittemmin se on poistettu (Mannila ym. 2014). Tietotekniikkaa pyrittiin tämän jälkeen tuomaan jokaiseen oppiaineeseen integroimalla, mikä aiheutti tarkastelun kohdistuvan enenevissä määrin pikemminkin tietotekniseen laitteistoon, kuin itse OA-taitoihin (Mannila ym. 2014). Vuonna 2016 perusopetus sai Suomessa uuden opetussuunnitelman, jossa iso painoarvo on tulevaisuuden taitojen kehittymisellä (Mannila ym. 2014). Opetussuunnitelman perusteissa otetaan kantaa ohjelmoinnilliseen ajatteluun (Opetushallitus 2014), nimikkeellä "algoritminen ajattelu", mikä voidaan ymmärtää käsitteen "ohjelmoinnillinen ajattelu" synonyymina (Bocconi, Chioccariello ja Earp 2018). Itse ohjelmoinnista tuli myös pakollinen osa perusopetusta uuden opetussuunnitelman myötä (Fagerlund ym. 2021). Ohjelmointi onkin tärkeä osa OA:ta, sillä monet tutkijat näkevät sen, eivät ainoastaan opittavana taitona itsessään, vaan perustavanlaatuisena tapana edistää ohjelmoinnillista ajattelua (Fagerlund ym. 2021). Koulujen opetus Suomessa pitää siis sisällään ohjelmoinnillista ajattelua.

Informaaleja, opetussuunnitelman ulkopuolisia hankkeita löytyy myös Suomesta. Monet yliopistot, kuten Helsingin yliopisto ja Jyväskylän yliopisto tarjoavat peruskoulutasoisille oppijoille kursseja ja materiaaleja. Helsingin yliopiston *Linkki* on yliopiston tietojenkäsittelytieteiden laitoksella operoiva tiedeluokka, joka järjestää 9-16 -vuotiaille lapsille harrastusmahdollisuuksia tietojenkäsittelytieteen, ohjelmoinnin ja tietokoneiden parissa. Jyväskylän yliopiston järjestämä *Nuorten peliohjelmointikurssi* on myös hyvä esimerkki peruskoulumaailman ulkopuolisesta toiminnasta, jossa yläkouluikäiset nuoret pääsevät opiskelemaan ohjelmoinnin alkeita ja suunnittelemaan oman tietokonepelin.

Harrastus- ja kerhotoiminnan lisäksi Suomestakin löytyy kilpailuja, sekä organisaatioiden järjestämiä ohjelmia. Esimerkkeinä mainittakoon *Innokas*-verkosto, joka haluaa tuoda peruskoulumaailmaan luovuutta ja innovatiivisuutta teknologian avulla. *Innokas*-verkosto järjestää myös vuosittain kilpailullisen turnaustapahtuman, jossa pääpaino on ohjelmoinnilla ja robotiikalla. Aiemmin mainitun Code.org:in järjestämä *Hour of Code* tarjoaa myös materiaalejaan suomen kielellä.

Voisi siis olettaa, että tilanne on Suomessa verrattain hyvä. Tämä ei kuitenkaan heijastu tutkimuksissa, sillä Kaarakainen ym. (2017) kirjoittavat ohjelmointiosaamisen olevan oppilailla heikkoa. Myöskään hyvälle tasolle ei ylletä vielä millään ict-taitotestin osa-alueella, jossa pyritään laaja-alaisesti mittaamaan osaamisen tasoa opetussuunnitelmissa mainituissa taidoissa (Kaarakainen ym. 2017). Vaikka Suomi on Ruotsin ohella todella omaksunut vahvasti OA:n osaksi koulutusjärjestelmää, painottuu ohjelmointiopetus suuremmilta osin matematiikan oppiaineeseen (Bocconi, Chioccariello ja Earp 2018) ja ohjelmoinnilliset oppimäärät jäävät peruskoulutasolla melko suppeiksi (Fagerlund 2022; Opetushallitus 2014). Käsite “algoritminen ajattelu” ei tule mainituksi laaja-alaisen osaamisen tavoitteissa kertaakaan POPS(2014). Suunta on kuitenkin oikea (Kaarakainen ym. 2017) ja seuraavassa kappaleessa keskitytään tarkemmin ohjelmoinnillisen ajattelun opetustyön haasteisiin, jotta suunta pysyisi oikeana.

4 Haasteet ohjelmoinnillisen ajattelun opetustyössä

Kun ohjelmoinnillisen ajattelun käsite on avattu ja tutkittu nykyisen opetuksen tasoa, on oiva aika keskittyä opetustyöhön liittyviin haasteisiin. Tässä kappaleessa keskitytään käsittelemään haasteita, joita ohjelmoinnillisen ajattelun opetustyö voi kohdata ja aiheuttaa. Aiemmin tekstissä on jo sivuttu mahdollisia haasteita ja nyt tässä luvussa ne avataan. Kaikkien opettaminen ajattelemaan ohjelmoinnillisesti on jalo ajatus, mutta pitää sisällään pedagogisia haasteita (Lu ja Fletcher 2009). Näitä sanoja mukaillen käsitellään seuraavaksi, minkälaisista haasteista on kyse.

4.1 Käsitteen määrittely

Yksi ongelmakohta ohjelmoinnillisen ajattelun opettamisessa on käsitteen määrittely (Denning 2017). Käsite “ohjelmoinnillinen ajattelu” on edelleen monelle hyvin vaikeasti ymmärrettävissä ja yhtenäistä, selkeää määrittelyä ei ole (Grover ja Pea 2013). Käsitteen epäselvä määrittely on laittanut opettajat ja kasvatustietelijät miettimään, mitä ohjelmoinnillinen ajattelu on? Onko siitä hyötyä kaikille? Miten sitä tulisi edes arvioida? (Denning 2017). So, Jong ja Liu (2020) mainitsevat myös, että käsitteen kompleksisuudella voi mennä pieleen, kun OA:ta tuodaan uusiin kulttuureihin. Jokaisen maan kulttuuri on uniikki ja jos implementointi kulttuurien välillä ei toimi, voi jalolla ajatuksella olla negatiivisia seurauksia (So, Jong ja Liu 2020). Aiemmin mainitut opettajien esittämät kysymykset saavat siis eri kulttuureissa varmasti erilaisia vastauksia.

Niin kauan, kuin selkeää määritelmää OA:lle ei ole, on vaikea hahmottaa opetettavaa sisältöä. OA:n opetuksen suhteen tarve täsmälliselle määritelmälle on tärkeä. Määritelmän avulla voidaan kehittää arviointityökaluja OA:lle ja varmistaa, ettei OA:n opetus ole ainoastaan kasa erilaisia apukeinoja, joita opettajan mielenkiinnon mukaan esitellään (Selby ja Woollard 2013). Angeli ja Giannakos (2020) kirjoittavatkin, että jos OA halutaan omaksua tehokkaasti osaksi koulujärjestelmää, on lisätutkimusta käsitteen määrittelyn suhteen tehtävä. Pitää pystyä kehittämään ja vahvistamaan yhteinen, vahva ja teoreettinen käsitys siitä, mitä on ohjelmoinnillinen ajattelu (Angeli ja Giannakos 2020).

4.2 Opettajat

Opettajien taidot opetuksen laatuun nähden ovat hyvin merkittävässä asemassa. Päätekijä vaihteluun oppilaiden koulumenestyksessä on opettajien pätevyys (Barber, Mourshed ja Company 2007). Jotta oppijat voivat kehittää ja oppia OA-taitoja, on opettajilla oltava riittävät taidot OA:sta ja sen sisällyttämisestä opetukseen (Yadav ym. 2014). On kohtuutonta olettaa, että 2000-luvun tietotaitoja tarvitsevat lapset ja nuoret saavat koskaan tarvitsemaansa opetusta opettajilta joiden ohjenuorana toimii 1900-luvun mallit (Mannila ym. 2014). Uudistumiseen on siis pyrittävä. Opettajien pitää olla valmiita opettamaan ohjelmoinnillista ajattelua ja kohtaamaan mahdolliset haasteet (Garcıa Peñalvo ym. 2016). Täydennyskoulutusta on kehitettävä virassa oleville opettajille ja samaan aikaan opiskelevien, tulevien opettajien koulutukseen tulee löytää tapoja sisällyttää OA:ta (Angeli ja Giannakos 2020).

Opettajien määrä on suuri ja haaste on iso. Euroopan 28 maassa on noin kaksi miljoona peruskoulutason opettajaa ja kaksi ja puoli miljoonaa toisen asteen opettajaa. Ohjelmoinnillisen ajattelun sisällyttäminen opetussuunnitelmaan kaikilla tasoilla tulee siis varmasti vaatimaan suurta määrää täydennyskoulutusta virassa oleville (Bocconi ym. 2016). Opettajankoulutuksessakin on vielä epäselvää, kuinka OA:ta tulisi tuoda osaksi koulutusta (Yadav ym. 2017).

4.3 Formaalin opetuksen sivuhaara

Formaalin opetuksen ulkopuolella tapahtuvaan non-formaaliin ja informaaliin opetukseen sisältyy myös haasteita. Vaikka formaalin opetuksen ulkopuolisella opetuksella on tietysti hyvät aikeet, on opetuksella kääntöpuolensa (Mannila ym. 2014). Usein opetuksella ei ole juurikaan yhtymäkohtia formaaliin, opetussuunnitelmaa mukailevaan opetukseen. Lapsen tai nuoren voi olla hankala yhdistellä omia ajatuksiaan koulumaailman ja ulkopuolisen opetuksen välillä (Mannila ym. 2014). Tämän lisäksi koulun ulkopuolella tapahtuva opetus on todella usein ohjelmointipainotteista (Mannila ym. 2014). Shute, Sun ja Asbell-Clarke (2017) kirjoittavat monien non-formaalien oppimiskokonaisuuksien väittävän opettavansa ohjelmointitaitoja, mutta harvemmin ne tarkastelevat syvemmin tapoja ajatella ohjelmoinnillisesti. Näitä haasteita ohjelmointiin liittyen käsitellään tarkemmin seuraavassa osiossa.

Koulun ulkopuolella tapahtuva oppiminen on tärkeää, mutta tämän lisäksi tarvitaan julkista

koulutusjärjestelmää, eikä sitä saa unohtaa (Pollak ja Ebner 2019). Tietojenkäsittelyn oppiminen pitäisi nähdä ennemmin erilaisten oppimismuotojen summana (Dagiene ja Stupuriene 2016). Formaalit luennot ja oppitunnit eivät ole riittäviä pitämään oppilaiden motivaatiota yllä ja syvemmän ymmärryksen luomiseksi pitäisi oppilaiden pystyä pelaamaan tietoteknisten käsitteiden, sekä ajatusten kanssa päivittäin (Dagiene ja Stupuriene 2016).

4.4 Ohjelmointi ja "koodaus", sekä ohjelmointiorienteituneisuus

Aluksi on hyvä muistuttaa siitä, että käsitettä ohjelmointi ja “koodaaminen” ei tule sekoittaa, kuten jo aikaisemmin kappaleessa Ohjelmoinnillisen ajattelun opetus huomattiin. Ohjelmointi on paljon muutakin, kuin koodaamista.

Itse ohjelmointia ei kuitenkaan tulisi myöskään sekoittaa ohjelmoinnilliseen ajatteluun. Ohjelmoinnillisen ajattelun rinnastaminen ohjelmointitaitoihin voi olla liian poissulkevaa (Shute, Sun ja Asbell-Clarke 2017). Ohjelmoinnillinen ajattelu on paljon muutakin kuin tietokoneohjelman kirjoittamista ja ohjelmoimista, OA vaatii ajattelun taitoja monella eri abstraktiotasolla (J. M. Wing 2006). Yleisesti ajateltuna tietotekniikka on laajempi käsite, kuin ohjelmointi ja OA taas laajempi kuin tietotekniikka (Shute, Sun ja Asbell-Clarke 2017).

Lu ja Fletcher (2009) ehdottaa jopa, että ohjelmoinnillisen ajattelun käsitteet ja tavat tulisi omaksua ennen, kuin oppilas edes näkeekään ohjelmakoodia. Ohjelmoinnin tulisi olla tapa kehittyä tietotekniikan saralla vain paremmaksi, kun aluksi käsitteet ja toimintalogiikka on sisäistetty. Ohjelmointi on tietojenkäsittelylle sitä, mitä todistaminen on matematiikalle(Lu ja Fletcher 2009). Toisin sanoen, asiat tulisi pystyä ymmärtämään ennen, kuin niitä kirjoitetaan auki koodilla. Tätä väitettä puoltaa myös Pérez-Marín ym. (2020) tekemä tutkimus, jossa tutkittiin, voiko ohjelmoinnillinen ajattelu edistyä paremmin hyödyntämällä kielikuvia ja lohko-ohjelmointia ohjelmoinnin opetuksessa. Tutkimustulokset vahvistivat käsitystä siitä, että OA-taidot paranevat.

Ohjelmoinnin osaaminen ei tarkoita, että osaa ajatella ohjelmoinnillisesti, mutta ohjelmointitaidot voidaan nähdä karttuneiden OA-taitojen hyötynä (Shute, Sun ja Asbell-Clarke 2017). Jos taas ohjelmointia painotetaan ja sen kautta tuodaan uusia näkökulmia ohjelmoinnillisen ajattelun kehittymiseen, niin sen ei tulisi jäädä ainoastaan yhden oppiaineen - matematiikan

alle, vaan sitä tulisi integroida muihinkin oppiaineisiin (Mannila ym. 2014).

5 Yhteenveto

Tällä hetkellä maailman digitalisoituessa jatkuvasti, uusia taitoja tarvitaan. Ohjelmoinnillisen ajattelun taito on todennäköisesti tulevaisuuden kansalaiselle yksi näistä. Ohjelmoinnillinen ajattelu on kuitenkin laaja ja monimutkainen käsite, jota voi olla hankala sisäistää. Tarkkaa ja yhtenäistä määritelmä käsitteelle ei ole selvillä vielä edes alan asiantuntijoiden kesken. Pystymme helposti listaamaan mitä ohjelmoinnillinen ajattelu pitää sisällään ja mitä se ei taas pidä. Selkeä ja yhtenäinen määritelmä siltä vielä puuttuu. Jotta ohjelmoinnillisen ajattelun opetustyö voisi kukoistaa, tulee ensiksi pystyä yhdessä ymmärtämään, mistä puhutaan. Tarjoan tutkielmassani suomen kielellä tietoa siitä, mitä OA pitää sisällään ja mitä se voisi tarkoittaa.

Ohjelmoinnillista ajattelua pyritään tuomaan osaksi nuorten elämää monin tavoin. Sekä formaalilla koululaitoksella että koululaitoksen ulkopuolisilla toimijoilla on omat intressinsä ja tapansa. Kaikkien opettaminen ajattelemaan ohjelmoinnillisesti edellyttää toimia kuitenkin enenevissä määrin formaalilta opetukselta. Kuten huomattiin, näihin molempiin oppimistilanteisiin liittyy haasteita. Haasteiksi listattiin muun muassa käsitteen epäselvyys, tämänhetkinen opettajien taso, ohjelmointiorientoituneisuus ja formaalin koululaitoksen ulkopuolinen opetustyö. Nämä on saatava ratkaistua, jotta taitoja voidaan tulevaisuudessa opettaa.

Herääkin kysymys siitä, että miten ohjelmoinnillista ajattelua tulisi opettaa? Kuinka ohjelmoinnillinen ajattelu saadaan ohjaamaan kaikkia koulussa opetettavia aineita? Kuinka saamme opettajista päteviä opettamaan ohjelmoinnillista ajattelua? Kysymys ei ole tietoteknisestä osaamisesta, vaan tavasta ajatella.

Päättäjien tulisi nyt herätä kunnolla aikaan jota elämme. Opettajille tulisi puhua näistä asioista ja heidän tulisi olla valmiita kokeilemaan rohkeasti uusia metodeja luokassa. Kaiken kaikkiaan on aika ymmärtää todenteolla se, että tietoyhteiskunnassa eläminen tulee vaatimaan todella erilaisia taitoja, kuin ne mihin olemme toistaiseksi tottuneet.

Lähteet

Aho, Alfred V. 2012. “Computation and computational thinking”. *The computer journal* 55 (7): 832–835.

Angeli, Charoula, ja Michail Giannakos. 2020. “Computational thinking education: Issues and challenges”. *Computers in Human Behavior* 105:106185. ISSN: 0747-5632. https://doi.org/https://doi.org/10.1016/j.chb.2019.106185. https://www.sciencedirect.com/science/article/pii/S0747563219303978.

Barber, Michael, Mona Mourshed ja McKinsey Company. 2007. “How the World’s Best-Performing School Systems Come Out on Top”. *http://lst-iiep.iiep-unesco.org/cgi-bin/wwwi32.exe/[in=e* (tammikuu).

Bebras. https://www.bebras.org/. Accessed: 2022-06-13.

Bocconi, Stefania, Augusto Chioccariello, Giuliana Dettori, Anusca Ferrari, Katja Engelhardt, Panagiotis Kampylis ja Yves Punie. 2016. “Developing computational thinking in compulsory education”. *European Commission, JRC Science for Policy Report* 68.

Bocconi, Stefania, Augusto Chioccariello ja Jeffrey Earp. 2018. “THE NORDIC APPROACH TO INTRODUCING COMPUTATIONAL THINKING AND PROGRAMMING IN COMPULSORY EDUCATION” (tammikuu). https://doi.org/10.17471/54007.

Colburn, Timothy, ja Gary Shute. 2007. “Abstraction in computer science”. *Minds and Machines* 17 (2): 169–184.

“Computational thinking, 10 years later”. 2016, viitattu 8. maaliskuuta 2022. https://www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later/.

Council, National Research. 2010. *Report of a Workshop on the Scope and Nature of Computational Thinking.* Washington, DC: The National Academies Press. ISBN: 978-0-309-14957-0. https://doi.org/10.17226/12840. https://www.nap.edu/catalog/12840/report-of-a-workshop-on-the-scope-and-nature-of-computational-thinking.

Curzon, Paul, Mark Dorling, Thomas Ng, Cynthia Selby ja John Woollard. 2014. “Developing computational thinking in the classroom: a framework”.

Czerkawski, Betul C, ja Eugene W Lyman. 2015. “Exploring issues about computational thinking in higher education”. *TechTrends* 59 (2): 57–65.

Dagiene, Valentina, ja Gabriele Stupuriene. 2016. “Informatics concepts and computational thinking in K-12 education: A Lithuanian perspective”. *Journal of Information Processing* 24 (4): 732–739.

Denning, Peter J. 2017. “Computational thinking in science”. *American Scientist* 105 (1): 13–17.

Eshach, Haim. 2007. “Bridging In-school and Out-of-school Learning: Formal, Non-Formal, and Informal Education”. *Journal of Science Education and Technology* 16 (2).

Fagerlund, Janne. 2022. “Tietokonevallankumous ja ohjelmoinnillinen ajattelu peruskoulussa – Havaintoja mikro- ja makrotasoilta: Lectio praecursoria”. *Kasvatus amp; Aika* 16, numero 1 (maalismaalis): 121–127. https://doi.org/10.33350/ka.111888. https://journal.fi/kasvatusjaaika/article/view/111888.

Fagerlund, Janne, Päivi Häkkinen, Mikko Vesisenaho ja Jouni Viiri. 2021. “Computational thinking in programming with Scratch in primary schools: A systematic review”. *Computer Applications in Engineering Education* 29 (1): 12–28.

Garcıa Peñalvo, Francisco José, Daniela Reimann, Maire Tuul, Angela Rees, Ilkka Jormanainen ym. 2016. “An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers”.

Geek Girl Mini. http://geekgirlmini.se/varfor-geek-girl-mini/. Accessed: 2022-06-13.

Grover, Shuchi, ja Roy Pea. 2013. “Computational thinking in K–12: A review of the state of the field”. *Educational researcher* 42 (1): 38–43.

Hautamäki, Antti. 2009. “Oppimisen muuttuva maasto”, viitattu 11. kesäkuuta 2022. https://www.sitra.fi/julkaisut/oppimisen-muuttuva-maasto/.

Heintz, Fredrik, Linda Mannila, Lars-Åke Nordén, Peter Parnes ja Björn Regnell. 2017. “Introducing programming and digital competence in Swedish K-9 education”. Teoksessa *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives,* 117–128. Springer.

Hill, Robin K. 2016. “What an algorithm is”. *Philosophy & Technology* 29 (1): 35–59.

Hour of Code. https://hourofcode.com/us/fi/. Accessed: 2022-06-13.

Innokas. https://www.innokas.fi/. Accessed: 2022-06-14.

International Olympiad in Informatics. https://ioinformatics.org/. Accessed: 2022-06-13.

Kaarakainen, Meri-Tuulia, Suvi-Sadetta Kaarakainen, Erika Tanhua-Piironen, Jarmo Viteli, Antti Syvänen ja Antero Kivinen. 2017. “Digiajan peruskoulu 2017-Tilannearvio ja toimenpidesuositukset”.

Kafai, Yasmin B, ja Quinn Burke. 2013. “The social turn in K-12 programming: moving from computational thinking to computational participation”. Teoksessa *Proceeding of the 44th ACM technical symposium on computer science education,* 603–608.

Kramer, Jeff. 2007. “Is abstraction the key to computing?” *Communications of the ACM* 50 (4): 36–42.

Linkki. https://linkki.cs.helsinki.fi/fi/. Accessed: 2022-06-14.

Lockwood, James, ja Aidan Mooney. 2017. “Computational Thinking in Education: Where does it fit? A systematic literary review”. *arXiv preprint arXiv:1703.07659.*

Lonka, kirjoittaja, Kirsti. 2015. *Oivaltava oppiminen.* 1. painos. Helsingissä: Otava.

Lu, James J, ja George HL Fletcher. 2009. “Thinking about computational thinking”. Teoksessa *Proceedings of the 40th ACM technical symposium on Computer science education,* 260–264.

Lye, Sze Yee, ja Joyce Hwee Ling Koh. 2014. “Review on teaching and learning of computational thinking through programming: What is next for K-12?” *Computers in Human Behavior* 41:51–61.

Made with code. https://edu.google.com/code-with-google/. Accessed: 2022-06-13.

Mannila, Linda, Valentina Dagiene, Barbara Demo, Natasa Grgurina, Claudio Mirolo, Lennart Rolandsson ja Amber Settle. 2014. “Computational Thinking in K-9 Education”. Teoksessa *Proceedings of the Working Group Reports of the 2014 on Innovation amp; Technology in Computer Science Education Conference,* 1–29. ITiCSE-WGR ’14. Uppsala, Sweden: Association for Computing Machinery. ISBN: 9781450334068. https://doi.org/10.1145/2713609.2713610. https://doi.org/10.1145/2713609.2713610.

MOT Englanti. n.d. Evaluate. Kielikone Oy. Viitattu 10. maaliskuuta 2022. https://www.sanakirja.fi/english-finnish/evaluate.

Nuorten peliohjelmointikurssi. https://trac.cc.jyu.fi/projects/npo. Accessed: 2022-06-14.

Opetushallitus. 2014. *Perusopetuksen opetussuunnitelman perusteet.* Viitattu 24. helmikuuta 2022. https://www.oph.fi/fi/koulutus-ja-tutkinnot/perusopetuksen-opetussuunnitelman-perusteet.

Pérez-Marín, Diana, Raquel Hijón-Neira, Adrián Bacelo ja Celeste Pizarro. 2020. “Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children?” *Computers in Human Behavior* 105:105849. ISSN: 0747-5632. https://doi.org/https://doi.org/10.1016/j.chb.2018.12.027. https://www.sciencedirect.com/science/article/pii/S0747563218306137.

Pollak, Michael, ja Martin Ebner. 2019. “The missing link to computational thinking”. *Future Internet* 11 (12): 263.

Resnick, Mitchel, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman ym. 2009. “Scratch: programming for all”. *Communications of the ACM* 52 (11): 60–67.

Rich, Peter J, Garrett Egan ja Jordan Ellsworth. 2019. “A framework for decomposition in computational thinking”. Teoksessa *Proceedings of the 2019 ACM conference on innovation and technology in computer science education,* 416–421.

Rogers, Alan. 2014. *The base of the iceberg: Informal learning and its impact on formal and non-formal learning.* Verlag Barbara Budrich.

Selby, Cynthia, ja John Woollard. 2013. “Computational thinking: the developing definition”, viitattu 13. kesäkuuta 2022. https://eprints.soton.ac.uk/356481/.

Shute, Valerie J., Chen Sun ja Jodi Asbell-Clarke. 2017. “Demystifying computational thinking”. *Educational Research Review* 22:142–158. ISSN: 1747-938X. https://doi.org/https://doi.org/10.1016/j.edurev.2017.09.003. https://www.sciencedirect.com/science/article/pii/S1747938X17300350.

So, Hyo-Jeong, Morris Siu-Yung Jong ja Chen-Chung Liu. 2020. *Computational thinking education in the Asian Pacific region,* 1.

Tikva, Christina, ja Efthimios Tambouris. 2021. “Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review”. *Computers Education* 162:104083. ISSN: 0360-1315. https://doi.org/https://doi.org/10.1016/j.compedu.2020.104083. https://www.sciencedirect.com/science/article/pii/S0360131520302815.

Wing, Jeanette. 2011. “Research notebook: Computational thinking—What and why”. *The link magazine* 6:20–23.

Wing, Jeannette M. 2006. “Computational thinking”. *Communications of the ACM* 49 (3): 33–35.

———. 2008. “Computational thinking and thinking about computing”. Cited by: 699; All Open Access, Green Open Access, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366 (1881): 3717–3725. https://doi.org/10.1098/rsta.2008.0118. https://www.scopus.com/inward/record.uri?eid=2-s2.0-51849116469&doi=10.1098%5C%2frsta.2008.0118&partnerID=40&md5=07bb54d1cfa3122a96902282848837a6.

Vitikka, Erja, ja Elisa Hurmerinta. 2011. “Erja Vitikka & Elisa Hurmerinta”, viitattu 12. kesäkuuta 2022. https://www.oph.fi/sites/default/files/documents/vitikka-e.-ja-hurmerinta-e.-kansainvaliset-opetussuunnitelmasuuntaukset.-2011.pdf.

Yadav, Aman, Sarah Gretter, Jon Good ja Tamika McLean. 2017. “Computational thinking in teacher education”. Teoksessa *Emerging research, practice, and policy on computational thinking,* 205–220. Springer.

Yadav, Aman, Chris Mayfield, Ninger Zhou, Susanne Hambrusch ja John T Korb. 2014. “Computational thinking in elementary and secondary teacher education”. *ACM Transactions on Computing Education (TOCE)* 14 (1): 1–16.