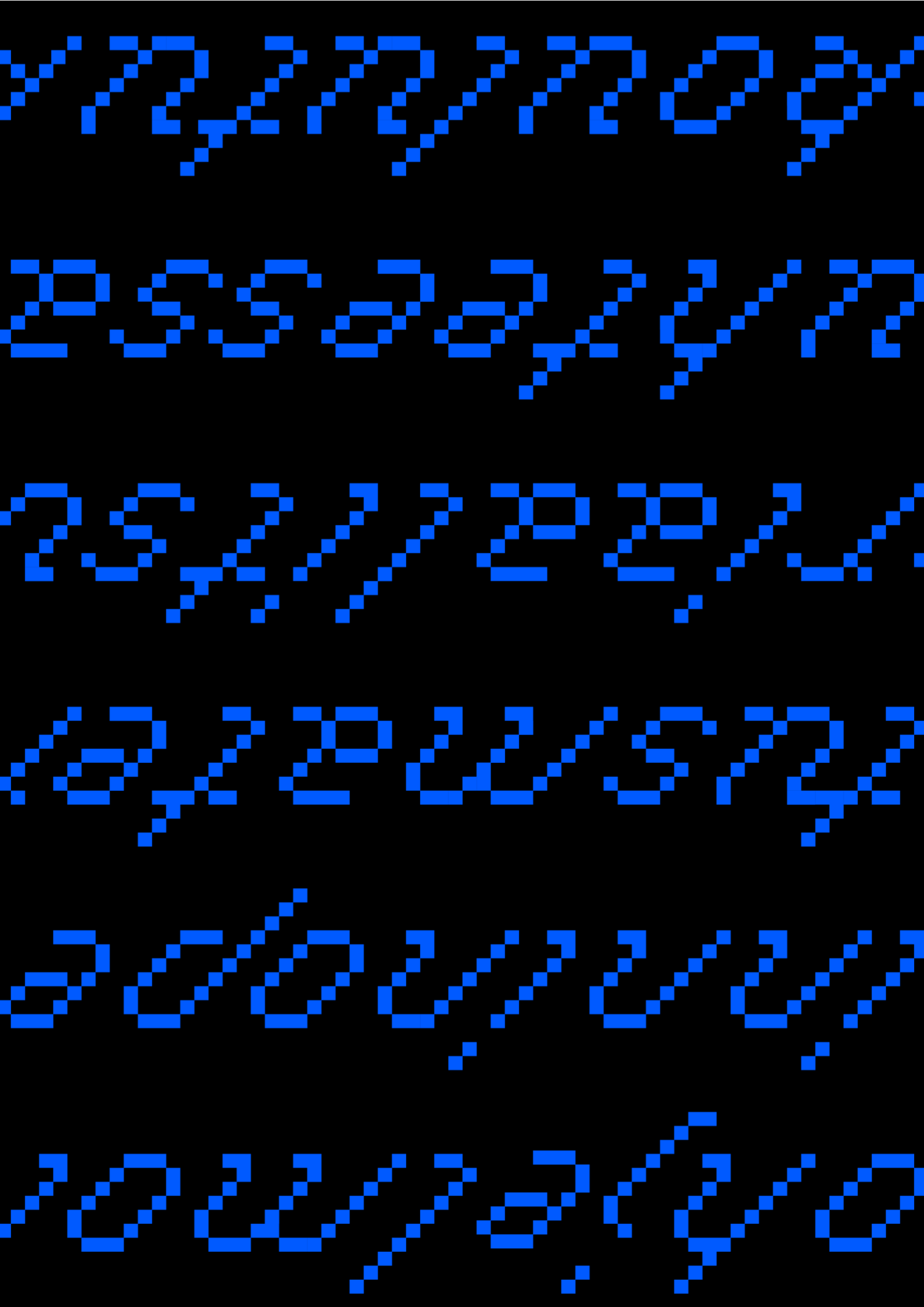


Ohjelmoinnin oppimateriaalit suhteessa koulutuksen tehtäviin ja koodimetaforiin

Hanna Hakala

Kasvatustieteen pro gradu –tutkielma
Kevätlukukausi 2023
Opettajankoulutuslaitos
Jyväskylän yliopisto

TIIVISTELMÄ

Hanna, Hakala. 2023. Ohjelmoinnin oppimateriaalit suhteessa koulutuksen tehtäviin ja koodimetaforiin. Kasvatustieteen pro gradu -tutkielma. Jyväskylän yliopisto. Opettajankoulutuslaitos. 66 sivua.

Tutkimuksen tarkoituksena oli tarkastella, miten ohjelmoinnin opetusta lähestytään peruskoulun opettajille suunnatuissa ohjelmoinnin oppimateriaaleissa. Oppimateriaaleilla on vaikutusta siihen, miten ohjelmoinnista puhutaan ja miten sitä tulisi opettaa, joten on tarpeellista tutkia, minkälaisia koodiin ja ohjelmointiin liittyviä arvoja ja asenteita oppimateriaalit välittävät.

Tarkastelin teoriaohjaavan sisällönanalyysin avulla kahdeksaa oppimateriaalia, ja vertasin niissä esiin nousevia teemoja Gert Biestan (2010) määrittelemiin koulutuksen tehtäviin sekä Tomi ja Mikko Dufvan (2016) koodimetaforiin. Biestan (2010) koulutuksen tehtävät mahdollistavat ohjelmoinnin opetuksen kasvatuksellisen puolen tarkastelun niin opittavien taitojen, yhteiskunnan vaatimusten kuin oppilaan näkökulmasta. Dufvan ja Dufvan (2016) koodimetaforat mahdollistavat oppimateriaalien tarkastelemisen erityisesti koodin näkökulmasta.

Tutkimuksen mukaan ohjelmoinnin opetus nähdään hyvin teknisestä näkökulmasta, ja lapsille ja nuorille pyritään tarjoamaan ennen kaikkea koodinkirjoitustaitoja tietoyhteiskunnassa toimimiseen. Biestan (2010) näkökulmasta kyse on kvalifikaation ja sosialisaation ylikorostumisesta. Koodimetaforissa (Dufva & Dufva, 2016) korostuvat funktionaalinen ja tulkitseva metafora sekä emansipatorisen näkökulman yhteiskunnallinen puoli.

Funktionaaliset taidot eivät kuitenkaan riitä koodin yhteiskunnallisten vaikutusten tarkasteluun, sillä näin ylläpidetään vallitsevia olosuhteita. Ohjelmoinnin arvovapaana matemaattis-loogisena päättelytaitona edesauttaa koodiutunutta ajattelua, jossa inhimilliset ongelmat pyritään ratkomaan riittävän datan ja oikeanlaisen algoritmin avulla. Opetukseen tulisi siis kuulua myös koodin eettisten kysymysten pohdintaa sekä erilaisia tiedon ja tietämisen muotoja.

Asiasanat: peruskoulu, oppimateriaalit, koodi, ohjelmointi, ohjelmoinnin opetus

SISÄLTÖ

1 JOHDANTO

Digitaalinen teknologia on yksi merkittävimmistä keksinnöistä kartan, kirjapainotaidon ja mekaanisen kellon ohella (Carr, 2010, s. 46–51, 86–87). Se yhdistää, viihdyttää ja toimii kykyjemme ja identiteettimme jatkeena. Samalla se muuttaa tapaamme ajatella, toimia ja hahmottaa maailmaa (Dufva, 2017a, s. 9). Tämä pätee myös formaalin kasvatuksen kenttään (varhaiskasvatus, perusopetus ja toinen aste), jossa käydään keskustelua digitaalisen teknologian hyödyntämisestä opetuksen ja oppimisen tukena.

Digitalisaatiolla tarkoitetaan yksinkertaisesti tietotekniikan yleistymistä arjessa (Kazmertsuk, 2019, s. 8). Koulutuksen digitalisaatiokehitys voidaan jakaa kahteen linjaan, joista toinen liittyy digitaalisiin laitteisiin ja sovelluksiin, kuten oppimisanalytiikkaan, ja toinen digitaalisiin ilmiöihin, kuten robotiikkaan, monialaiseen teknologiakasvatukseen (STEAM: science, technology, engineering, arts & mathematics) ja ohjelmointiin. Vaikka ohjelmoinnin opetuksessa hyödynnetään digitaalisia työkaluja, tämän tutkimuksen pääpaino on ohjelmoinnissa yhteiskunnallisena ilmiönä, joten laitteisiin liittyvän keskustelu rajautuu tarkasteluni ulkopuolelle.

Ohjelmoinnista on tullut 2010–2020-lukujen aikana kansainvälinen trendi (Albion, 2016, s. 1; Balanskat & Englehart, 2014, s. 8; Bocconi ym., 2018, iii) ja se otettiin osaksi suomalaisen perusopetuksen opetussuunnitelman perusteita (POPS) vuonna 2014. Opettajien koulutuksessa ohjelmoinnin opetusta ei ole kuitenkaan juuri ollut. Moni opettaja kokeekin ohjelmointiosaamisensa riittämättömäksi, ja vuoden 2016 jälkeen vain joka viides on opettanut ohjelmointia, vaikka täydennyskoulutus on toiminut hyvin ja tilanne on paranemaan päin (Tanhua-Piiroinen ym., 2019, s. 19, 21, 28, 29).

Ohjelmoinnin sisällyttäminen opetussuunnitelmaan on luonut markkinat ohjelmoinnin oppimateriaaleille koulutusalaan liittyvän valtavan kaupallisen potentiaalin vuoksi. Perusopetuksen ohjelmoinnin opetuksen puolestapuhujina on joukko hallituksen neuvonantajia, pääomasijoittajia, ajatushautomoita, järjestöjä ja teknologiayhtiötä (Williamson ym., 2018, s. 705–706). Toisin sanoen oppimateriaalit eivät ole arvovapaita (Rinne, 2019, s. 201–202; Miettunen, 2020, s. 29),

vaan niiden sisällöissä elävät tekijöidensä arvot ja asenteet (Ruuska, 2014, s. 24). Oppimateriaaleilla on näin ollen vaikutusta siihen, mitkä keskustelunaiheet luonnollistuvat ja mitä pidetään tärkeänä (ks. Säntti ym., 2021, s. 865), eli miten ohjelmoinnista puhutaan ja miten sitä pyritään opettamaan. Oppimateriaalien analysoiminen on näin ollen tärkeää, sillä ne vaikuttavat ohjelmoinnille annettuihin merkityksiin.

Nopeasti kehittyvän teknologian myötä ohjelmoinnin opetukseen liittyy lupaus nykyaikaisemmasta, tehokkaammasta ja yksilöllisemmästä opetuksesta (ks. Säntti, 2020). Ohjelmoinnin koetaan muun muassa aktivoivan oppilaita tietotekniikan passiivisen käyttämisen sijaan (Dufva, 2017a, s. 10; Mykkänen & Liukas, 2014, s. 73). Ohjelmoinnin tulo kouluihin liittyy myös niin sanotun koodiutuneen ajattelun lisääntymiseen, jonka myötä mitä erilaisimpia ongelmia pyritään ratkaisemaan algoritmien avulla (Koivisto ym., 2022b, s. 9). Ohjelmointi nähdään tärkeänä kansalaistaitona (Koivisto ym., 2022b, s. 4), jolla on vaikutus arjessa pärjäämiseen ja aktiivisena kansalaisena toimimiseen (Opetushallitus, 2014, s. 23). Ohjelmointia markkinoidaan myös yleisenä ajattelun taitona (Mykkänen & Liukas, 2014, s. 34) ja eräänlaisena digitalisaation tulkkina (Dufva, 2017a, s. 11). Dufva (2017a, s. 11) kuitenkin huomauttaa, että ohjelmointi myös peittää ja vaimentaa muita viestejä, sillä digitalisaation piilotettu luonne vaikeuttaa maailman hahmottamista ja suoraa kokemista: Näemme ohjelmissa vain sen pinnan, jonka ohjelmoija on koodiin eli konekieliseen käskysarjaan[1] luonut, joten opimme kuluttamaan, emme tekemään omia tulkintoja (Dufva, 2017a). Toisin sanoen koodi sekä mahdollistaa että rajoittaa toimintaamme (Rantala, 2018, s. 105, 107).

Ohjelmointi liittyy siis monin tavoin yleissivistykseen ja kriittiseen ajatteluun (Fagerlund, 2022). POPS:ssa esimerkiksi huomautetaan, että oppilaat saavat ohjelmoinnin kautta käsityksen siitä, että teknologian toiminta riippuu ihmisen tekemistä ratkaisuista (Opetushallitus, 2014, s. 157). Ohjelmoinnin opetus ja opetuksen tutkimus on kuitenkin painottunut koodin oikeinkirjoitukseen ja ehtolausekkeiden ja toistojen kaltaisten funktionaalisten taitojen opettamiseen (Fagerlund, 2022) ja se nähdään hyvin teknisestä näkökulmasta (Fagerlund, 2021, s. 88).

[1] Ohjelmointiin liittyvät käsitteet määritellään tarkemmin luvussa 2.1.

Ohjelmointia olisi kuitenkin hyvä tarkastella myös yhteiskunnan osana ja siihen vaikuttavana tekijänä, sillä oppilaita seurataan ja hallitaan koodin ja algoritmien avulla niin koulussa kuin sen ulkopuolella (Mertala ym., 2020, s. 21, 28). Tilanne on ristiriidassa opetussuunnitelman osallistumista ja demokratiaa painottavien arvojen kanssa (Opetushallitus, 2014, s. 16). Mertala ym. (2020, s. 23) kysyvätkin, miksi ohjelmointiin liittyviä taitoja opiskellaan – talouselämää vai inhimillistä elämää varten? He ehdottavat, että ohjelmointi nähtäisiin Kupiaisen (2017, s. 213–214) mukaisesti tekstitapahtumana, jossa tekstin merkitystä ei määrittele vain se, mitä teksti tarkoittaa vaan se, miten se toimii ja mitä se tuottaa (Mertala ym., 2020, s. 21, 28). Näin koodi näyttäytyisi perusopetuksessa niin teknisenä, sosiaalisena kuin symbolisena ilmiönä, jolla on abstraktiudestaan huolimatta materiaalinen perusta ja monimutkaiset yhteiskunnalliset vaikutukset (Rantala, 2018, s. 105, 107).

Kiinnostukseni aihetta kohtaan kumpuaa taustastani kuvataiteen aineenopettajana. Minua kiehtoo ohjelmoinnin mahdollisuudet visuaalisen ilmaisun työkaluna ja digitaalisen maailman tutkimusvälineenä. Tarkoituksenani on selvittää, minkälaisia merkityksiä ohjelmoinnille annetaan eri oppimateriaaleissa. Lähestyn aihetta tarkastelemalla kahdeksaa ohjelmoinnin oppimateriaalia ja vertaamalla niissä esiin nousevia teemoja Gert Biestan (2010) koulutuksen tehtäviin sekä Tomi ja Mikko Dufvan (2016) koodimetaforiin. Toivon tutkielman tuovan lisää sävyjä ohjelmoinnin opetuksesta käytävään keskusteluun sekä ohjelmoitujen ympäristöjen kriittiseen tarkasteluun.

2 NÄKÖKULMIA OHJELMOINTIIN JA KOODIIN

Ohjelmointia ja koodia voidaan lähestyä teknisestä ja yhteiskunnallisesta näkökulmasta. Tekninen näkökulma keskittyy ohjelmointiin funktionaalisena taitona, joka mahdollistaa ohjeiden antamisen tietokoneille. Yhteiskunnallinen näkökulma huomioi ihmisten elämän digitaalisten tietokoneiden ja ohjelmistojen kanssa ja niiden keskellä.

2.1 Tekninen ja yhteiskunnallinen näkökulma

Ohjelmointia lähestytään usein *dikotomisen* eli kahtia jakautuneen ajattelun avulla, sillä digitaalinen teknologia perustuu kahteen asiaan: koodiin ja elektroniikkaan (Rantala, 2018, s. 101). Koodia voidaan kuitenkin tarkastella niin teknisestä kuin yhteiskunnallisesta näkökulmasta. Voidaan sanoa, että koodi on digitaalisen teknologian rakennetta luova voima, jolle elektroniikka tarjoaa fyysisen alustan (Ruuskanen ym., 2015, s. 2). Lisäksi digitaalisuuden perustana on jako binääriseen järjestelmään eli bitteihin, ykkösiin ja nolliin, jossa ykköset edustavat sitä, että komento on päällä ja nollat sitä, että se ei ole päällä (Tujula & Ruuskanen, 2017). Emme kuitenkaan ohjelmoi ykkösillä ja nollilla, vaan käyttämämme ohjelmointikieli ja -ympäristö muuntavat koodin konekieleksi eli biteiksi (Tujula & Ruuskanen, 2017, s. 21).

Tekninen näkökulma. Ohjelmointiympäristö koostuu tietokoneesta, ohjelmistosta ja ohjelmointikielellä annetuista käskyistä, joiden perusteella tietokone käsittelee sille annetut tiedot ja tulostaa lopputuloksen (Mykkänen & Liukas, 2014). Ohjelmointikielet toimivat näin ihmisten ja tietokoneiden välisenä rajapintana. Ohjelmointikieliä on satoja ja niillä kaikilla oma *syntaksinsa* eli tiettyihin toimintoihin liittyvät kirjoitustapansa, joita ohjelma suorittaa rivi kerrallaan ohjelmoijan kirjoittaman koodin mukaisesti (Mykkänen & Liukas, 2014, s. 17–18). Yksinkertaisen ohjelman voi parhaimmillaan tehdä muutamassa minuutissa, mutta monimutkainen ohjelman kirjoittamiseen tarvitaan satoja ihmisiä ja kuukausien

työtä.[2] Onkin yleinen harhakäsitys, että ohjelmoinnin osaaminen olisi sama asia kuin jonkin ohjelmointikielen osaaminen (Heikkilä, 2014).

Ohjelmointina voidaan pitää tietokoneelle annettavien vaiheittaisten toimintaohjeiden eli *algoritmien* laatimista (Koivisto ym., 2022a, s. 8, 10). Algoritmin avulla jokin ongelma ratkeaa tai asia muuttaa muotoaan (Koivisto ym., 2022a, s. 12). Hyvin yleisesti käytetty esimerkki algoritmista on ruokaresepti, jonka avulla kuka tahansa pystyy leipomaan täysin samanlaisen kakun (Mykkänen & Liukas, 2014, s. 77–78). Vaikka koodausta ja ohjelmointia käytetään toistensa synonyymeinä, *koodaaminen* (koodin kirjoittaminen) on vain osa *ohjelmointia* (ohjelman suunnittelua) ja ohjelmoinnillista ajattelua (ongelman purkamista osiin, kaavojen tunnistamista ja toimintojen automatisoimista) (Manches & Plowman, 2017, s. 192; Petzold, 2000, s. 232; Albion, 2016, s. 1; Fagerlund, 2021, s. 18).

Ohjelmoitaessa syntyy helposti virheitä, joten kirjoitettua koodia on tärkeää oppia testaamaan (Petzold, 2000, s. 236). Ohjelmointi onkin usein virheiden etsimistä, mikä helpottuu taitojen karttuessa (Tujula & Ruuskanen, 2017; Järvenpää, 2018, s. 13–14). Vika voi olla puuttuvassa merkissä tai väärässä lauserakenteessa, jolloin koodi saattaa toteuttaa kaksi toisensa poissulkevaa komentoa (Tujula & Ruuskanen, 2017, s. 17).

Koodia luetaan niin laitteiston, ohjelmiston kuin verkon tasolla, ja se voidaan tallentaa eri medioihin (Rantala, 2018). Koodi on siis olemassa sekä materiaalisesti että ei-materiaalisesti, kuten puhe tai musiikki (Rantala, 2018, s. 102). Digitalisaation yhtenä ominaisuutena voidaankin pitää kaiken tulemista ohjelmoiduksi (Dufva, 2017a, s. 9). Tämä vaikeuttaa omalta osaltaan ohjelmoitujen ympäristöjen hahmottamista.

Yhteiskunnallinen näkökulma. Ohjelmointia voidaan tarkastella myös yhteiskuntaa rakentavana ja siihen vaikuttavana voimana. Koodi tehostaa palveluista, mutta ohjelmistot myös muokkaavat ajatteluamme ja todellisuuttamme (Carr, 2010, s. 14–15). Yhteiskunnan digitalisoituminen on aiheuttavat huolta mielipidevaikuttamisesta, työpaikkojen katoamisesta, epätasa-arvosta ja henkilötietojen seurannasta (Konle-Seidl & Danesi, 2021, s. 15, 20, 25). Digitaalisen teknologian toimintamallit eivät ole kuitenkaan muuttumattomia luonnonlakeja,

[2] El Kamel, S. (2018). Mitä koodaaminen oikeasti on? Helsingin Sanomat 14.7.2018, D14.

vaan ihmisen luomia, muokkautuvia malleja (Koivisto ym., 2022a, s. 39; Dufva, 2015, s. 2), joissa heijastuvat tekijöidensä tietoiset ja tiedostamattomat arvovalinnat (Dufva, 2015, s. 9; O´Neill, 2017, s. 198). Koodi ei ole näin ollen arvovapaa ajattelun väline, vaan sillä on aina sosiaalinen ulottuvuutensa (Rantala, 2018, s. 101).

Voidaan puhua *poliittisesta ohjelmoinnillisesta ajattelusta*, jossa tieteelliset, sosiaaliset, hallinnolliset ja inhimilliset ongelmat nähdään teknisinä ongelmina, jotka voidaan ratkoa riittävän datan ja oikeanlaisen algoritmin avulla (Kitchin, 2014).[3] Ohjelmoinnillinen ajattelu laajenee tällöin digitaalisista laitteista koko yhteiskuntaan. Vadén (2002) kutsuu tämän kaltaista teknologisia intressejä korostavaa suuntausta *koodiutumiseksi*. Koodiutuminen ei ole kuitenkaan vain pyrkimys etsiä biologisille, teknologisille ja sosiaalisille järjestelmille algoritminen selitys, vaan sen avulla koodi pyritään luonnollistamaan myös hallinnollisena ja oikeudellisena käsitteenä, johon kohdistuu voimakkaita taloudellisia pyrkimyksiä (Vadén, 2002, s. 10–11). Digitalisoituminen on esimerkiksi johtanut *alustoitumiseen* (platformization), jossa verkon infrastruktuurin päälle rakennettuja palveluita hallitsevat muutamat suuret teknologiayritykset. Alustoitunutta yhteiskuntaa pidetään monissa yhteyksissä jopa myönteisenä kehityksenä, sillä sen koetaan edustavan tasa-arvoista ja innovatiivista, ruohonjuuritasolta rakentuvaa organisaatiota (Ideland, 2021, s. 34–35).

2.2 Koodimetaforat yhdistävät teknisen ja yhteiskunnallisen näkökulman

Tekninen ja yhteiskunnallinen näkökulma koodiin eivät ole toisensa poissulkevia, vaan niitä molempia tarvitaan ylittämään dikotominen (kahtia jakautunut) ajattelu, eli täydentämään käsitystämme koodista ja ohjelmoiduista ympäristöistä ja tekemään ohjelmoinnista ilmiönä helpommin lähestyttävä (Dufva & Dufva, 2016; Mertala, 2021). Tällaista holistista ajattelua tukemaan Tomi ja Mikko

[3] Niin sanottuja *älykkäitä kaupunkeja* (smart cities) hallinnoidaan digitaalisella infrastruktuurilla, joka tuottaa valtavia määriä dataa (big data) ja mahdollistaa reaaliaikaisen tiedonkeruun (Kitchin, 2014). Älykkäisiin kaupunkeihin liittyy poliittisia ja hallinnollisia haasteita ja huomattavia tietoturvariskejä (Kitchin, 2014, s. 1–2, 11–12).

Dufva (2016) ovat laatineet yhdeksän metaforaa kuvaamaan koodin eri ulottuvuuksia (taulukko 1). Mikään metaforista ei riitä yksinään kuvaamaan koodin olemusta, mutta yhdessä ne mahdollistavat ohjelmoitujen ympäristöjen monipuolisen tarkastelun.

Funktionaaliset metaforat (kone, organismi, aivot, muutos) liittyvät ohjelmoinnin koodinkirjoitustaitoihin, kulttuuri ja poliittinen järjestelmä tulkitsevat yhteiskuntaa ja yhteisöjä koodin avulla, mielen vankila ja vallan väline liittyvät emansipatoriseen näkökulmia ja avaavat koodiin liittyvää vallan käyttöä. Viimeinen metafora (karnevaali) edustaa vapaampaa, postmodernia lähestymistapaa (Dufva & Dufva, 2016, s. 100–106), jossa koodi toimii itseilmaisun välineenä.

Taulukko 1

Yhdeksän koodimetaforaa (Dufva & Dufva, 2016, s. 100)

Metafora	*Koodinäkemys*	*Koodin tarkoitus*	*Esimerkki*
Kone	Koodi on lineaarinen käskyjono	Koneen ohjaaminen	Koodirivi, algoritmi
Organismi	Tietokoneohjelma muodostuu koodiryhmistä eli objekteista	Toiminnallisuuden ja vuorovaikutuksen lisääminen	Olio-ohjelmointi
Aivot	Koodi on ihmisen tekemien systeemien äly	Uuden tiedon luominen, oppiminen	Pilvilaskenta, AI
Muutos	Koodi on muutosta luova prosessi	Rakenteiden muodostaminen	Ohjelma muutoksena
Kulttuuri	Koodi on tapa ajatella ja ymmärtää maailmaa	Yhteisöjen luominen, ihmisten yhdistäminen	Vapaa koodi, hakkerointi
Poliittinen järjestelmä	Koodi on kannanotto ja keino muuttaa maailmaa	Uudenlaisen yhteiskunnan muodostaminen	Internet
Mielen vankila	Koodi on systeemi, johon pitää sopeutua	Ajattelun muokkaaminen	Suodatinkupla
Vallan väline	Koodi on vallan väline	Ihmisten kontrolloiminen	Data vallan lähteenä
Karnevaali	Koodi on luovan ilmaisun työkalu	Vallitsevan ajattelun haastaminen, keskustelun avaaminen	Luova ohjelmointi

2.2.1 Funktionaaliset metaforat: kone, organismi, aivot, muutos

Funktionaaliset metaforat edustavat vallitsevaa tapaa lähestyä ohjelmointia ja niissä koodi nähdään neutraalina, matemaattis-loogisena ongelmanratkaisuvälineenä, joten mahdolliset eettiset kysymykset jäävät metaforien ulkopuolelle (Dufva & Dufva, 2016, s. 100–102).

Kone-metaforassa koodi esiintyy lineaarisena sarjana koneelle syötettäviä komentoja, jotka käsitellään tietokoneessa ja tulostetaan lopuksi ratkaisuna käyttäjälle (Dufva & Dufva, 2016). Teknisessä mielessä kyse on digitaalisesta paradigmasta, jossa tietokone ymmärtää vain kahta numeroa: ykkösiä tai nollia. Kone-metaforassa digitaalisen teknologian koetaan etenevän vääjäämättömästi kohti yhä tehokkaampia tietokoneita ja monimutkaisempia ohjelmointikieliä. Metafora ei kuitenkaan huomioi ohjelmointiin liittyvien prosessien monimutkaisuutta (Dufva & Dufva, 2016, s. 101). Mertala ym. (2020, s. 25) esimerkiksi huomauttavat, ettei ulkoinen maailma ole vain fyysikaalisten objektien maailma, vaan se koostuu myös tunteista ja sosiaalisista käytänteistä, joiden kompleksisuuden muuntaminen ykkösiksi ja nolliksi on aina välitteistä ja sisältää vain viittauksia todellisen maailman ilmiöihin. Digitalisoinnin ulkopuolelle jää siis aina valtava määrä erilaisia tiedon ja tietämisen muotoja (Vadén, 2002, s. 13).

Organismi-metafora muistuttaa kone-metaforaa, mutta koodi nähdään lineaarista käskyjonoa laajemmin osana vuorovaikutussuhteiden verkostoa. Organismi-metaforaa voidaan pitää kone-metaforan jatkona, jossa koodi jaetaan ohjelmoijille paremmin hallittaviin osa-alueisiin (Petzold, 2000, s. 372–373). Teknisellä tasolla organismi-metafora vastaa olio-ohjelmointia, jossa koodi jaetaan erikseen käsiteltäviin objekteihin. Se tuo näin esille ohjelmoinnin moniammatillista luonnetta. (Dufva & Dufva, 2016, s. 100–101).

Aivot-metaforassa koodi rinnastuu ihmisen mieleen, joka käsittelee ja tuottaa tietoa (Tujula & Ruuskanen, 2017, s. 38). Kyse on ikään kuin ajattelevasta laitteesta. Näin metafora liittyy tekoälyyn ja koneoppimiseen, joka on nopeasti kasvava tekoälyn ala (O'Neill, 2016, s. 76). Tekoälyn yhteydessä älykkäällä käyttäytymisellä tarkoitetaan joustavaa, vuorovaikutuksellista ja tarkoituksenmukaista toimintaa monimutkaisessa ja muuttuvassa ympäristössä (Lappi ym., 2018, s. 43).

Tekoälyn huipentuma on *teknologinen singulaarisuus*, jossa tekoäly on ylittänyt ihmisälyn ja siitä tulee aistiva (Lanier, 2010, s. 35–37; Dufva & Dufva, 2016, s. 102).

Muutos-metafora siirtää painopisteen tehokkaasti toimivista tietokoneista koodin lupaukseen tehdä maailmasta parempi paikka. Kyse on koodiutuneesta ajattelusta, jossa ohjelmointi nähdään joustavana ja kätevänä ratkaisuna lähes kaikkiin ongelmiin, jotka voidaan pilkkoa osiin ja ratkaista koodin avulla. Muutoksen tarvetta tai suuntaa ei kuitenkaan pohdita (Dufva & Dufva, 2016, s. 102).

2.2.2 Tulkitsevat metaforat: kulttuuri, poliittinen järjestelmä

Tulkitsevissa metaforissa huomioidaan ohjelmoinnin yhteiskuntaa muokkaava vaikutus. Koodi nähdään kuitenkin ongelmattomana, ja mahdollisia eriäviä mielipiteitä pidetään varsin yleisluontoisina kysymyksinä koodin yhteiskunnallisista merkityksistä ja tavoitteista (Dufva & Dufva, 2016, s. 102–104).

Kulttuuri-metafora tuo esille sen, ettei koodi ole vain ohjeiden antamista tietokoneelle, vaan ohjelmointi on itsessään merkittävä kulttuuri-ilmiö ja vaikuttaa valtakulttuuriin aikaansaamalla erilaisia alakulttuureja, kuten avoimen lähdekoodin yhteisöjä, hakkerointia ja itse tekemisen kulttuuria (maker movement). (Dufva & Dufva, 2016, s. 102–103).

Koodi poliittisena järjestelmänä -metafora laajentaa kulttuuri-metaforaa ottamalla mukaan poliittisen ulottuvuuden ja kysymykset siitä, kenellä on oikeus datan keräämiseen ja jakamiseen. Koodi toimii näin ollen kannanottona yhteiskunnallisiin asioihin (Dufva & Dufva, 2016, s. 103–104).

2.2.3 Emansipatoriset metaforat: mielen vankila, vallan väline

Emansipatorisissa metaforissa on osittain samoja teemoja kuin tulkitsevissa metaforissa, mutta emansipatoriset näkökulmat painottavat erityisesti valtakysymyksiä ja sitä, miten vallan käyttö näkyy koodissa. Voidaan esimerkiksi tarkastella niin yksilön kuin yhteiskunnan tasolla, edistääkö koodi emansipaatiota, eli voimaannuttaako koodi ihmisiä ja yhteisöjä toimimaan itselleen edullisella tavalla (Dufva & Dufva, 2016, s. 105).

Mielen vankila -metafora korostaa yksilön ja koodin välistä ongelmallista valtasuhdetta. Sen avulla voidaan tarkastella esimerkiksi sitä, miten koodi ohjaa,

mahdollistaa ja muokkaa ihmisten ajattelua ja käyttäytymistä. Toisin sanoen koodi on systeemi, johon pitää sopeutua (Dufva & Dufva, 2016, s. 105). Pariser (2012) puhuu *suodatinkuplasta* (filter bubble), jokaisen käyttäjän ainutlaatuisesta tiedon universumista, jonka algoritmit luovat valitessaan räätälöityjä sisältöjä. Käyttäjä ei kuitenkaan näe valintojen perusteluja, vaan hyödyntää niitä tiedostamattaan (Pariser, 2012, s. 2–3). Jos seuraamme vain samanmielisiä sisältöjä, rajaamme itsemme ulos osasta yhteiskunnallista keskustelua. Saatamme myös tiedostamattamme muuttaa käyttäytymistämme koodin vaikutuksesta. Sosiaalisesta mediasta on esimerkiksi tullut keino ilmaista tunteita (Lanier, 2010, s. 64–67) ja jakaa henkilökohtaisia kokemuksia (Soronen & Koivunen, 2022, s. 1346). Ihmiset tuntuvatkin haluavan ristiriitaisia asioita: vaikka kuluttajat vastustavat tunkeilevia seuranta-algoritmeja, he kritisoivat mainontaa, joka ei ole tarpeeksi henkilökohtaista ja merkityksellistä (Ruckenstein & Granroth, 2019, s. 10).

Vallan väline -metaforassa koodi nähdään keinona hallita yhteisöjä (Dufva & Dufva, 2016). Dataa muun muassa kerätään yhä laajemmin ja näkymättömämmin, mikä mahdollistaa sen väärinkäytön. Toisin sanoen valta on niillä, joilla on pääsy koodiin. Koodi mahdollistaa myös kapinoinnin valtaapitäviä vastaan: hakkerit voivat esimerkiksi ohittaa hallituksen rajoituksia julkaisemalla viestejä vaihtoehtoisilla kanavilla. Koodi voidaan siis nähdä sekä vallan välineenä että sen ilmentymänä (Dufva & Dufva, 2016, s. 105).

2.2.4 Postmoderni metafora: karnevaali

Postmoderni **karnevaali-metafora** keskittyy niihin mekanismeihin, joiden avulla luomme uusia merkityksiä ja tuo esille sen, että koodi myös inspiroi ja synnyttää tunteita (Dufva & Dufva, 2016, s. 106). Metaforan avulla voidaan näin tutkia ihmisten reaktioita ohjelmointia kohtaan ja haastaa vallitsevia ajattelutapoja. Yksi konkreettinen esimerkki karnevaali-metaforasta on *luova koodaus*, jonka avulla taiteilijat voivat kyseenalaistaa koodia ja ilmaista itseään: koodia voidaan esimerkiksi käyttää väärässä paikassa sekä korostaa epäkohtia ja itsestään selvinä pi-

dettyjä asioita (Dufva & Dufva, 2016, s. 106). Taiteilijakollektiivi Random International[4] esimerkiksi tutkii ihmisen ja teknologian välistä suhdetta vuorovaikutteisten installaatioiden avulla: *Swarm Study* havainnollistaa lintujen käyttäytymistä LED-valojen ja messinkiputkien avulla, ja kun katsoja lähestyy teosta, ”parvi” reagoi ääneen ja muuttaa liikkeensä suuntaa.[5]

[4] Random-international.com

[5] Random International (2015, Februry 17). *Swarm Study VII.* [Video]. YouTube. https://www.youtube.com/watch?v=ajV8A5Y2_dE

3 OHJELMOINTI PERUSKOULUSSA

3.1 Koulutuksen kolme tehtävää avaavat ohjelmoinnin opetuksen merkityksiä

Gert Biestan (2020) on nimennyt formaalille kasvatukselle kolme tehtävää: *kvalifikaatio*, *sosialisaatio* ja *subjektifikaatio*. Yhden osa-alueen painottaminen vaikuttaa aina muihin osa-alueisiin, joten niitä ei voi erottaa toisistaan. Kun kasvatusprosesseja tarkastellaan näistä kolmesta eri näkökulmasta, niitä voidaan lähestyä aiempaa kokonaisvaltaisemmin (Biesta, 2016, s. 36).

Kvalifikaation kautta koulu tarjoaa lapsille ja nuorille tarvittavat tiedot ja taidot yhteiskunnassa toimimiseen. Kvalifikaatio liittyy näin kiinteästi koulutuksen institutionaaliseen puoleen ja yhteiskunnalliseen kontrolliin. Ohjelmoinnin opetuksella pyritään esimerkiksi varmistamaan tulevien ammattikoodarien saatavuus (Mertala ym., 2020, s. 23). Ohjelmoinnin markkinoiminen tulevaisuuden taitona (Binkley ym., 2012) ja varmana uravalintana kuitenkin idealisoi koodaustyön monotonisuutta (Williamson, 2016, s. 49–50) ja ohjelmointialan työllistävää vaikutusta, joka saattaa jäädä lupauksia pienemmäksi (Mertala ym., 2020, s. 24). Ohjelmoijat joutuvat opettelemaan jatkuvasti uusia ohjelmointikieliä eivätkä ehdi tarkastelemaan alaansa kriittisesti, joten he tasapainoilevat jatkuvan riittämättömyyden kanssa eivätkä muodosta yhtenäistä asiantuntijaryhmää (Kitchin & Dodge, 2011, s. 35). Toisin sanoen työelämän ohjelmointikäytännöt eivät ole yhtä systemaattisia, objektiivisia tai korkealaatuisia kuin ohjelmoinnin puolestapuhujat antavat ymmärtää (Williamson, 2016, s. 50).

Sosialisaation kautta lapset ja nuoret kasvavat tietyn sosiaalisen, kulttuurisen ja poliittisen yhteisön jäseniksi sisäistämällä sen tärkeinä pidettyjä arvoja ja tapoja (Biesta, 2020, s. 92). Koulu on siten opettamisen ja oppimisen, vuorovaikutussuhteiden ja tilan lisäksi osa yhteiskunnallista uusintamista ja uudistamista (Mertala, 2022c, s. 144–150). Ohjelmoinnin välttämättömyyttä perustellaankin usein sillä, että lasten ja nuorten osallisuus ja aktiivinen toimijuus digitaalisessa yhteiskunnassa edellyttävät *digitaalisia kansalaistaitoja*, jotka toimivat sekä pääomana että digitalisaation haittavaikutuksilta suojaavana tekijänä (Koivisto ym.,

2022a, s. 45; Román-González, Pérez-González ja Jiménez-Fernández, 2017; Kaarakainen, 2019). Toisin sanoen ohjelmoinnin opetus tarjoaa lapsille ja nuorille keinoja tarkastella, tutkia ja rakentaa ohjelmoituja laitteita ja ympäristöjä (kvalifikaatio), jotta heillä on riittävät tietotekniset taidot työelämään ja yhteiskunnalliseen toimintaan osallistumiseen (sosialisaatio).

Kasvatuksen tehtävä on myös tukea lasten ja nuorten itsemääräytymistä eli omana itsenään olemista (Mertala, 2022b, s. 27). Tässä formaalin kasvatuksen kolmannessa tehtävässä eli **subjektifikaatiossa** on kyse itsenäisesti ajattelemista ja ympäröivän maailman rajojen kohtaamisesta (reality check), ei joksikin kehittymisestä (Biesta, 2020, s. 99–100). Subjektifikaatio on riippumattomuutta sosiaalisista normeista ja niihin liittyvistä hyvän elämän ideaaleista, eli sen voi ymmärtää myös sosialisaation vastakohdaksi (Mertala, 2022b). Sen sijaan että oppilas olisi ulkoisten määrittelyjen kohde, subjektifikaatioon liittyy vapaus toimia oman elämän subjektina, jonka olemusta ja tulevaisuutta muut ihmiset eivät määrittele (Mertala, 2022b, s. 30).

Ohjelmoinnin perustellaankin mahdollistavan lasten ja nuorten aktiivisen toimijuuden (subjektifikaatio) passiivisen kuluttamisen sijaan. Sosiaaliseen mediaan, kuten Facebookiin, Twitteriin tai YouTubeen, tuotetut sisällöt kuitenkin hämärtävät kuluttamisen ja tuottamisen välistä rajaa (Williamson, 2016). Tämä ohjelmoinnin opetuksen piilo-opetussuunnitelma sosiaalistaa lapset ja nuoret digitaalisten sisältöjen luomiseen, ja koulii heistä vapaaehtoisia sisällöntuottajia kasvavan digitaalisen hallinnon ylläpitämiseen (Williamson, 2016, s. 41, 53–55). Brunila (2011, s. 421) pitää tätä vallan hajauttamista yhä väliaikaisempiin ja epävirallisempiin rakenteisiin yhtenä viime vuosikymmenten merkittävimmistä hallinnollista muutoksista.

Kun ohjelmoinnin opetusta tarkastellaan subjektifikaation näkökulmasta, voidaan kysyä, miten opetus voisi parhaiten tukea lasten ja nuorten mahdollisuuksia ilmaista itseään ja tehdä itselleen merkityksellisiä asioita. Kun ohjelmoinnin opetuksen lähtökohdaksi otetaan todellisia ilmiöitä, keskustelu laajenee teknologiasta tekoälyyn ja tietosuojaan liittyviin eettisiin kysymyksiin (Fagerlund, 2022; Mertala ym., 2020). Näin oppilaat huomaavat, että heillä on mahdollisuus

vaikuttaa omilla valinnoillaan digitaalinen yhteiskunnan kehitykseen (Koivisto ym., 2022a, s. 45; Koivisto ym., 2022b, s. 42).

Ohjelmointiosaamiseen (kvalifikaatio) tulisi näin ollen kuulua funktionaalisen ja lineaarisen koodinkirjoitustaidon lisäksi laajempi, kriittinen ymmärrys ohjelmoiduista ympäristöistä (sosialisaatio), sillä koululla on rooli eräänlaisena välitilana, jossa oppilailla on tilaa kasvaa, kokeilla ja epäonnistua (subjektifikaatio) ilman ulkopuolisia odotuksia (Biesta, 2016, s. 39). Koska koulu mahdollistaa kouluttautumisen omaa sosioekonomista taustaa pidemmälle, se tarjoaa mahdollisuuden kyseenalaistaa yhteiskuntaa ja kunkin omaa asemaa siinä (Saari, 2020). Tiedon tulee siis olla vapaa välitöntä hyötyä koskevista vaatimuksista (Ruuska, 2014, s. 44), sillä opiskeltavat asiat ovat arvokkaita itsessään (Saari, 2020). Tämä ei poissulje ohjelmointitaitojen työelämähyötyjä, mutta ohjelmointia ei tulisi opiskella pelkästään työllistymisen näkökulmasta.

3.2 Ohjelmoinnin opetus perusopetuksen opetussuunnitelman perusteissa

Kunkin aikakauden kasvatuskäsitys formaalista koulutuksesta materialisoituu opetussuunnitelmassa, johon kirjataan kasvatuksen tavoitteet ja ihanteet (Antikainen ym., 2021, s. 196, 199) ja jonka laatimiseen osallistuu niin asiantuntijoita kuin työelämän edustajia (Kärki, 2015). Opetussuunnitelma on oman aikansa peili (Säntti, 2020, s. 63), ja tarjoaa aina sen hetkisen näkemyksen siitä, mitä jokaisen kansalaisen tulisi osata ja minkälaisten opetusmenetelmien avulla näitä tietoja ja taitoja parhaiten omaksutaan (Antikainen ym., 2021).

Ohjelmointi esiintyy POPS:ssa sekä laaja-alaisen osaamisen tavoitteissa että osana matematiikan ja käsityön sisältötavoitteita, joten sen opetuksesta vastaavat alakoulussa luokanopettajat ja yläkoulussa matematiikan ja käsityön aineenopettajat. Laaja-alaisessa osaamisessa ohjelmointiosaaminen on sidottu etenkin tietoja viestintäteknologiseen osaamiseen (L5), mutta se liittyy ohjelmoinnillisen ajattelun myötä muihin laaja-alaisen osaamisen alueisiin, kuten ajattelun ja oppimaan oppimisen tavoitteisiin (L1), kulttuuriseen osaamiseen, vuorovaikutukseen ja ilmaisuun (L2), itsestä huolehtimiseen ja arjen taitoihin (L3), monilukutai-

toon (L4), työelämä- ja yrittäjyystaitoihin (L6) sekä osallistumiseen, vaikuttamiseen ja kestävään elämäntapaan (L7) (Opetushallitus, 2014, s. 20–24, 99–101, 155–158, 281–285).

Vaikka opetussuunnitelmassa määritellään ohjelmoinnin oppimistavoitteet (Opetushallitus, 2014, s. 101, 129, 157, 235, 284, 375, 379), opetuksen toteutusta ei kuvailla kovinkaan tarkasti. Opetus vaihteleekin koulujen ja kuntien välillä ilman yhtenäistä linjaa (Leino ym., 2019, s. 10–11; Mertala, 2021, s. 2227).

Vuosiluokilla 1–2 ohjelmointi on leikillistä ja toiminnallista (Opetushallitus, 2014, s.130) ja oppilaat harjoittelevat kaavojen tunnistamista ja toimintaohjeiden antamista erilaisten verkkotehtävien, robottien ja ohjattavien laitteiden avulla (Parviainen ym., 2016, s. 4; Koivisto ym., 2022a, s. 7). Opetuksessa harjoitellaan yhtäläisyyksien, erojen ja säännönmukaisuuksien löytämistä, vertaillaan ja luokitellaan ja järjestellään asioita ja pyritään löytämään niiden välisiä syy-seuraussuhteita (Opetushallitus, 2014, s. 129) sekä pohditaan tieto- ja viestintäteknologian käyttöä ja merkitystä omassa arjessa (Opetushallitus, 2014, s. 101).

Vuosiluokilla 3–6 ohjataan oppilaita ymmärtämään, että teknologian toiminta on seurausta ihmisen tekemistä ratkaisuista (Opetushallitus, 2014, s. 157). Kyse ei ole vain koneelle annettavista ohjeista, vaan teknologian yhteiskunnallisten vaikutusten pohtimisesta. Oppilaat tutustuvat helppokäyttöisiin ohjelmointiympäristöihin ja ohjelmoinnin peruskäsitteisiin aluksi konkreettisesti, jonka jälkeen abstraktien käsitteiden määrää lisätään vähitellen (Opetushallitus, 2014, s. 235; Järvenpää, 2018, s. 18). Robotiikka on yksi keino tuoda ohjelmointia konkretian tasolle (Anundi ym., 2018). Oppilaat tutustuvat matematiikan tunneilla graafisiin ohjelmointiympäristöihin (Opetushallitus, 2014, s. 235) ja käsityön tunneilla robotiikkaan ja automaatioon (Opetushallitus, 2014, s. 271).

Graafisessa ohjelmointiympäristössä koodi muodostetaan valmiita käskypalikoita yhdistelemällä, mikä luo konkreettisen elektronisen rakentelun tuntua ja ohjaa ajattelua kohti korkeamman tason konsepteja (Fagerlund, 2018, s. 20–21). Yksi suosituimmista peruskouluissa käytetyistä graafisista ohjelmointikielistä on ilmainen ja verkkoselaimella toimiva Scratch (Järvenpää, 2018, s. 20–21; Maloney ym., 2010, s. 3; Resnick ym., 2009).

Vuosiluokilla 7–9 oppilaiden käsitykset digitaalisten laitteiden, ohjelmistojen ja palvelujen käytöstä ja toiminnasta syvenevät, ja he pääsevät tekemään ohjelmoituja projekteja sekä itsenäisesti että yhdessä (Opetushallitus, 2014, s. 284). Yläkoulussa kehitetään oppilaiden algoritmista ajattelua, ohjelmointikäytäntöjä ja hyödynnetään ohjelmointia erilaisissa ongelmanratkaisutilanteissa (Opetushallitus, 2014, s. 375). Vuosiluokilla 7–9 aloitetaan myös perehtyminen johonkin varsinaiseen ohjelmointikieleen (Mykkänen & Liukas, 2014, s. 47). Hyviä aloituskieliä ovat esimerkiksi Processing, Python, Ruby ja JavaScript (Meyer, 2018, s. xix; Mykkänen & Liukas, 2014, s. 35, 100). Oppilaat tekevät oppiainerajat ylittäviä ohjelmointiprojekteja ja tutustuvat monimutkaisempiin ohjelmoinnin rakenteisiin, kuten aliohjelmiin ja funktioihin (Järvenpää, 2018, s. 18). Toisten tuottamat koodit ja valmiit ohjelmat ovat tällöin keskeisessä roolissa, eli koodia opetellaan sekä lukemaan että tulkitsemaan (Koivisto ym., 2022b, s. 7).

Matematiikan hyvän osaamisen (arvosana kahdeksan) mukaisesti oppilas osaa kuudennen luokan päättyessä toteuttaa tietokoneohjelman graafisessa ohjelmointiympäristössä (Opetushallitus, 2014, s. 239) ja yhdeksännen luokan päättyessä ohjelmoida yksinkertaisia tietokoneohjelmia ja soveltaa algoritmisen ajattelun periaatteita (Opetushallitus, 2014, s. 379).[6]

[6] *Ohjelmoinnillisen ajattelun* (computational thinking) (Papert, 1980, s. 182; Wing, 2006, s. 33–34) merkitys vaihtelee tutkimuskirjallisuudessa konkreettisemmista ohjelmointitaidoista yleisiin ajattelun- ja ongelmanratkaisutaitoihin (Denning, 2017; Fagerlund, 2021, s. 32–33). Mykkäsen ja Liukkaan (2014, s. 77–78) mukaan ohjelmoinnillinen ajattelu on kykyä purkaa ongelma osiin, tunnistaa toistuvia syy-seuraussuhteita, luoda algoritmeja ja automatisoida toistuvia ratkaisutapoja. POPS:ssa ei kuitenkaan puhuta ohjelmoinnillisesta ajattelusta vaan *algoritmisesta ajattelusta* (Opetushallitus, 2014, s. 375, 379), joka voidaan nähdä yhtenä ohjelmoinnillisen ajattelun osa-alueena (Labusch ym., 2019, s. 14, 69): algoritmi on eräänlainen suunnitelma ennen varsinaisen koodin kirjoittamista ja ohjelmointi tämän aikomuksen kirjaamista toimintaohjeiksi (ks. Mykkänen & Liukas, 2014, s. 16–17, 77–78). Algoritminen ajattelu on näin ollen taito, jota oppilaat kehittävät opetellessaan ohjelmointia (Labusch ym., 2019, s. 72).

4 TUTKIMUSTEHTÄVÄ JA -KYSYMYKSET

Tämän tutkielman tarkoituksena on tarkastella, miten ohjelmoinnin opetusta lähestytään peruskoulun opettajille suunnatuissa ohjelmoinnin oppimateriaaleissa. Opettajille suunnatut materiaalit toimivat ohjelmoinnin opetuksen tukena, mutta vaikuttavat samalla heidän käsityksiinsä siitä, minkälaista ohjelmoinnin opetuksen tulisi olla. On siis tarpeen tutkia, minkälaisia koodiin ja ohjelmointiin liittyviä arvoja ja asenteita ohjelmoinnin oppimateriaalit välittävät.

Vertailen laadullisen teorialähtöisen sisällönanalyysin avulla kahdeksan oppimateriaalin sisältöjä Gert Biestan (2010) koulutuksen tehtäviin (kvalifikaatio, sosialisaatio, subjektifikaatio) sekä Tomi ja Mikko Dufvan (2016) koodimetaforiin eli vaihtoehtoisiin näkökulmiin koodista. Tutkimuskysymykseni ovat:

1. Minkälaisia merkityksiä ohjelmoinnille annetaan eri oppimateriaaleissa suhteessa kvalifikaatioon, sosialisaatioon ja subjektifikaatioon?

2. Miten ohjelmoinnin oppimateriaalien käsitys ohjelmoinnista suhteutuu koodimetaforiin?

5 TUTKIMUSAINEISTO JA MENETELMÄT

5.1 Aineistona ohjelmoinnin oppimateriaalit

Laadullisen tutkimuksen aineisto (otos) on usein valittu niin, että se tarjoaa mahdollisimman laajan näkökulman tutkittavaan ilmiöön (Palinkas ym., 2015, s. 542). Monipuolisen otoksen idea on saada käsitys aineiston vaihtelusta (maximum variation), ei niinkään aineistoja yhdistävistä tekijöistä, vaikka tutkimus saattaakin tuottaa yleiskäsityksen tutkimusaineistosta analyysin seurauksena (Patton, 2002, s. 240).

Valitsin tällä periaatteella kahdeksan ohjelmoinnin oppimateriaalia vuosilta 2014–2022, joka kattaa ajanjakson, jolloin ohjelmointi on ollut osa perusopetusta. Lisäksi aineistossa on mukana eri tahojen tuottamia materiaaleja: Polkuja 1–6 ja Polkuja 7–9 edustavat valtiollisen tason, eli Opetus- ja kulttuuriministeriön koordinoiman Uudet lukutaidot -kehittämishankkeen (2020–2023) laatimia oppimateriaaleja, Vantaan kaupungin tukimateriaali (2018) edustaa kuntatasolla tuotettua oppimateriaalia ja ViLLE (2016) on Turun yliopiston tuottama ohjelmointiopas. Robo1 (2015) ja Robo 2 (2017) ovat kuvataidekasvattajien kirjoittamia oppaita ja Koodi2016 (2014) edustaa kasvatus- ja koulutussektorin ulkopuolisia tahoja. Kaupallista Raapaisu-opasta lukuun ottamatta kaikki materiaalit ovat avoimia julkaisuja ja vapaasti internetistä saatavilla. Suurin osa ohjelmoinnin oppimateriaaleista ei ole sidottu mihinkään tiettyyn ohjelmointiympäristöön, mutta Raapaisu-opas (2018) on tehty nimenomaan Scratch-ohjelmointiympäristöä ajatellen ja ViLLE-opas (2016) Turun yliopiston ViLLE-oppimisympäristöä varten.

Oppimateriaalien kaltainen *luonnollisesti esiintyvä data* on olemassa tutkijasta riippumatta (Eskola & Suoranta, 2000, s. 15), mutta se pitää kerätä ja analysoida. Luonnollisesti esiintyvä data tarjoaa mahdollisuuden ymmärtää tutkimusaihetta ilman, että tutkimusasetelma lisää tutkimuksen kompleksisuutta: tutkimuskohde ei ole tietoinen tutkimusasetelmasta eikä vaikuta aineiston keruuseen (Drewett & O'Reilly, 2021, s. 1, 4). On kuitenkin huomioitava, ettei luonnollinen data ole neutraalia, itsenäistä tai ajatonta, vaan kontekstisidonnaista ja aina jonkun tekemää (Potter, 2002, s. 539–541).

Ohjelmoinnin oppimateriaalien perustiedot ja niistä jatkossa käytettävät lyhenteet on tiivistetty taulukkoon 2:

Taulukko 2

Tutkimuksessa käytettyjen oppimateriaalien lyhenteet

Oppimateriaalin nimi	*Lyhenne*	*Vuosi*	*Kirjoittaja*
Koodi20116 – Ensiapua ohjelmoinnin opettamiseen peruskoulussa	Koodi2016-opas ja Koodi2016	2014	Juhani Mykkänen ja Linda Liukas
Robo1 – Elektroniikkaa ja askartelua	Robo1-opas ja Robo1	2015	Roi Ruuskanen, Harri Vähänissi ja Iiro Tujula
ViLLE-opintopolku – Alakoulun ohjelmointiopas	ViLLE-opas ja ViLLE	2016	Marika Parviainen, Essi Tamminen, Einari Kurvinen,
Robo2 – Elektroniikkaa ja ohjelmointia	Robo2-opas ja Robo2	2017	Petra Enges-Pyykönen
Raapaisu – Ohjelmoinnin opas	Raapaisu-opas ja Raapaisu	2018	Iiro Tujula ja Roi Ruuskanen
Vantaan kaupungin ohjelmoinnin, robotiikan, tekoälyn ja esineiden internetin opetussuunnitelman toteuttamisen tukimateriaali	Vantaan kaupungin tukimateriaali	2018	Miika Anundi, Panu-Pekka Kuitunen, Hannu Olin, Timo Järvenpää ja Timo Hurskainen
Polkuja ohjelmointiosaamiseen – Opas vuosiluokille 1–6	Polkuja 1–6 -opas ja Polkuja 1–6	2022	Lauri Palsa, Jussi Koivisto ja Tarmo Toikkanen
Polkuja ohjelmointiosaamiseen – Opas vuosiluokille 7–9	Polkuja 7–9 -opas ja Polkuja 7–9	2022	Lauri Palsa, Jussi Koivisto ja Tarmo Toikkanen

5.2 Metodina teoriaohjaava sisällönanalyysi

Laadullisen tutkimuksen lähtökohtana on jokin todellisen ilmiön kuvaaminen mahdollisimman kattavasti (Hirsjärvi ym., 2007, s. 157). Tutkimuksessa on usein käsiteltävänä varsin pieni joukko tapauksia, ja aineisto voi olla pelkistetyimmillään tekstiä (Eskola & Suoranta 2000, s. 15, 18), kuten omassa tutkimuksessani.

Biestan (2010) määrittämät kvalifikaatio, sosialisaatio ja subjektifikaatio mahdollistavat ohjelmoinnin opetuksen kasvatuksellisen puolen tarkastelun niin opittavien taitojen, yhteiskunnan vaatimusten kuin oppilaan itsensä näkökulmasta. Nämä koulutuksen kolme tehtävää muodostavat löyhän, *abduktiivisen teoriaohjaavan analyysin*, jossa teoria antaa tilaa aineistosta esiin nouseville käsitteille, teemoille ja tulkinnoille (Tuomi & Sarajärvi, 2018, s. 107–113).

Dufvan ja Dufvan (2016) koodimetaforat muodostavat puolestaan tiiviin tutkimusasetelman, joten aineiston analyysi on tältä osin *teorialähtöistä* eli tukeutuu valmiiseen teoriakehikkoon ja sen määrittämiin teemoihin (Tuomi & Sarajärvi 2002). Tätä voidaan myös kutsua *deduktiiviseksi* analyysiksi (yleisestä yksittäiseen) (Tuomi & Sarajärvi 2002, 95-99). Koodimetaforat mahdollistavat aineiston tarkastelemisen nimenomaan koodin ja ohjelmoinnin näkökulmasta.

5.3 Sisällönanalyysin suorittaminen käytännössä

Aineistosta on hyvä saada aluksi yleiskuva, jotta siitä voidaan tehdä alustavia tulkintoja (Heisakala, 1990). Aloitinkin aineiston analysoinnin lukemalla kaikki oppimateriaalit läpi ja kirjaamalla ensimmäisiä havaintoja ylös.

Yksi laadullisen aineiston tiivistämisen keinoista on aineiston koodaus, missä yhdistetään samaa tarkoittavat asiat ja pyritään saamaan aineisto käsiteltävään muotoon (Kananen, 2014). Koodaus ei ole vielä aineiston analyysi, vaan välivaihe, joka mahdollistaa analyysin, joten se ei saa vähentää aineiston laadullista sisältöä. Koodausjärjestelmä on jokaisen tutkijan oma luomus, ja se toimii eräänlaisena kehikkona, joka lasketaan aineiston päälle. Liian yleisluontoinen koodaus kadottaa osan tiedosta ja liian tiheä koodaus tuottaa vaikeasti tulkittavan rakenteen. (Kananen, 2014, s. 103–104).

Jotta pystyin luettelemaan aineistosta ne yksiköt, joihin tulkinta perustuu, minun piti määritellä *analyysiyksikkö*, eli pienin merkityssisältö – ei sen vuoksi että jokainen lausuma laskettaisiin, vaan jotta ne olisi identifioitu (Mäkelä, 1990, 47–59). Tutkimukseni analyysiyksikkönä toimi lause tai kappale.

Flick (2002, s. 177) jakaa koodauksen kolmeen vaiheeseen: *avoimeen koodaukseen*, *erittelevään koodaukseen* ja *valikoivaan koodaukseen*. Etsin avoimen koodauksen

vaiheessa aineistosta tutkimukseen liittyviä asiasisältöjä ja käytin luokittelun apuna peruskysymyksiä, kuten kuka teki mitä, miten, milloin ja miksi (Flick, 2002, s. 178–181). Kopioin työn edetessä koodimetaforiin, kvalifikaatioon, sosialisaatioon ja subjektifikaatioon liittyviä lauseita ja kappaleita omiin taulukkoihinsa. Erittelevä koodaus kiteytti ja yhtenäisti avoimen koodauksen avulla syntyneitä kategorioita ja valikoi jatkokäsittelyyn tutkimuskysymysten kannalta mielenkiintoisimmat jaottelut (Kananen, 2014, s. 107). Tutkija käyttää erittelevän koodauksen vaiheessa induktiivista ajattelua asiayhteyksien muodostamiseen ja deduktiivista ajattelua muodostettujen kategorioiden testaamiseen (Flick, 2002, s. 181).

Tarkastelin valikoivan koodauksen vaiheessa aineistoa yhä abstraktimmalla tasolla (Kananen, 2014, s. 107). Tavoitteenani oli muodostaa pääkategoriat, joihin kaikki muut alakategoriat sulautuvat (Flick, 2002). Lopulta muodostui teoria, jota voitiin testata tutkittavaan aineistoon: tiettyjen ehtojen täyttyessä tapahtuu tiettyjä asioita. Osa teemoista yhdistyi kirjoitustyön edetessä huomatessani, että kyse on yhdestä ja samasta aihekokonaisuudesta. Tutkimus oli valmis, kun aineiston kyllääntyminen saavuttiin, eli koodauksen jatkaminen tai oppimateriaalien määrän kasvattaminen eivät olisi tuottaneet uutta tietoa (Flick, 2002, s. 182–183).

Dufvan ja Dufvan (2016) yhdeksän koodimetaforaa muodostivat valmiiksi määritellyt teemat, joten kiinnitin huomiota siihen, missä suhteessa eri metaforat aineistossa esiintyvät. Tämä tapahtui sisällön erittelyn ja kvantifioinnin keinoin, eli laskin jokaisesta oppimateriaalista havaintoyksiköt (lause tai kappale) ja luokittelin ne johonkin koodimetaforaan kuuluvaksi.[7] Näin oli mahdollista tarkastella eri näkökulmien dominanssia ja keskinäisiä voimasuhteita.

Tarkastellessani aineistoa Biestan (2010) koulutuksen tehtävien näkökulmasta, oppimateriaaleista löytämiäni havaintoyksiköitä (lause tai kappale) oli ajoittain vaikea luokitella yksiselitteisesti joko kvalifikaatioon, sosialisaatioon tai subjektifikaatioon kuuluviksi, koska nämä esiintyvät aina yhdessä (Biesta, 2016,

[7] Vastaavaa kvantifiointia ei voitu suorittaa Biestan (2010) määrittämistä koulutuksen tehtävistä, koska niiden luokittelu ei ollut yhtä selkeärajaista.

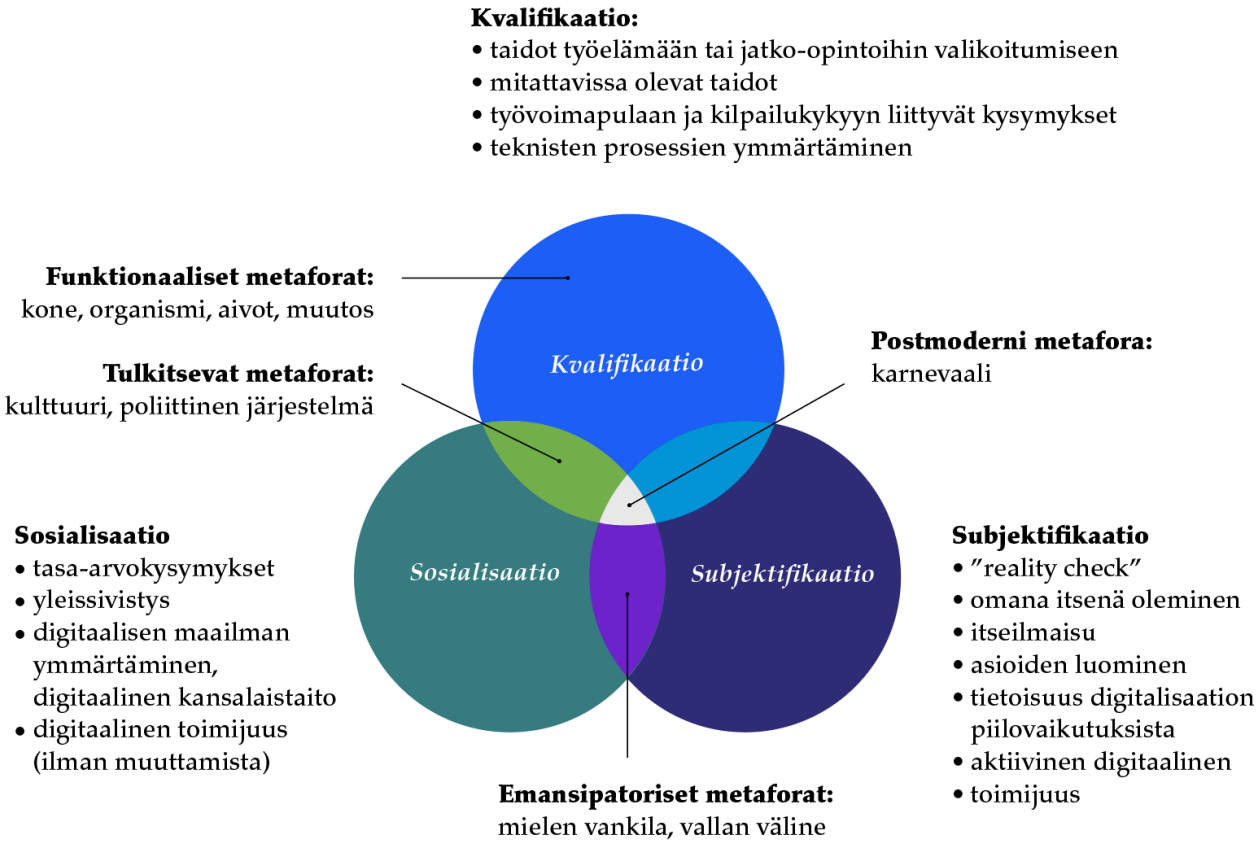
s. 35–36) (kuvio 1). Luokittelin tällöin havaintoyksikön kumpaankin vaihtoehtoon kuuluvaksi. Esimerkiksi lasten ja nuorten digitaalinen toimijuus liittyy sosialisaatioon, jos se ymmärretään arjessa pärjäämisen näkökulmasta: ”Kaikilla oppilailla on oikeus saada yhdenvertaisesti opetusta ohjelmointiin sekä hyvään elämään ja aktiiviseen toimintaan ohjelmoiduissa ympäristöissä” (Polkuja 7–9, s. 4). Aktiivinen digitaalinen toimijuus puolestaan kallistuu subjektifikaation puolelle ja vahvistaa lasten ja nuorten mahdollisuuksia ilmaista itseään ja tehdä itselleen merkityksellisiä asioita: ”Ohjelmointiosaamisen kehittämisessä tärkeitä ovat oppilaiden havainnot ympäröivästä maailmasta ja niiden kautta syntyvä mielekäs tekeminen ja tutkiminen sekä oppilaiden oman elämän piiriin liittyvät kokeilut ja innovaatiot” (Polkuja 1–6, s. 4).

Kuvio 1. Kvalifikaatioon, sosialisaatioon ja subjektifikaatioon liittyvien havaintoyksiköiden (lause tai kappale) esiintyminen ohjelmoinnin oppimateriaaleissa.

6 TULOKSET

Avaan tässä luvussa ohjelmoinnin oppimateriaaleissa esiintyviä koulutuksen tehtäviä (Biesta, 2010) ja koodimetaforia (Dufva & Dufva, 2016). Koulutuksen kolme tehtävää (ks. kappale 3.1) esiintyvät aina yhdessä, joten eri tehtävät leikkaavat osittain toisiaan (kuvio 2). Osa koodimetaforista sijoittuu leikkauskohtiin muodostuviin osa-alueisiin, mikä havainnollistaa käyttämieni taustateorioiden toisiaan täydentävää vaikutusta. Tulosten mukaan ohjelmoinnin oppimateriaaleissa painottuu ohjelmointi kvalifikaationa ja sosialisaationa, samoin funktionaalinen koodimetafora eli tekninen näkökulma.

Kvalifikaatioon, sosialisaatioon ja subjektifikaatioon liittyviä teemoja kertyi koodauksen tuloksena yhteensä yhdeksän: ohjelmointi kvalifikaationa (algoritminen ajattelu ja työskentelyn taidot, laitteiden suunnittelu ja rakentaminen, työvoiman saatavuuden varmistaminen), ohjelmointi sosialisaationa (ohjelmointi on osa yleissivistystä ja tasa-arvokysymys, koodi yhdistää ihmisiä) ja ohjelmointi



Kuvio 2. Kvalifikaatio, sosialisaatio ja subjektifikaatio suhteessa koodimetaforiin.

subjektifikaationa (ohjelmointi konkretisoi digitaalista maailmaa, kuluttajasta tekijäksi, ohjelmointi motivoi, miltä digitalisaatio tuntuu?)

6.1 Ohjelmoinnin oppimateriaalit suhteessa koulutuksen tehtäviin

Tarkastelen seuraavaksi ohjelmoinnin oppimateriaaleja ja niissä esiintyviä havaintoyksiköitä (lause tai kappale) suhteessa Biestan (2010) määrittämiin koulutuksen tehtäviin: kvalifikaatio, sosialisaatio ja subjektifikaatio. Niissä kohden, kun tekstissä on pitkä lista lähdemateriaaleja mutta ei suoraa nostoa jostakin oppimateriaalista, on viittaus laitettu alaviitteeseen luettavuuden helpottamiseksi.

6.1.1 Ohjelmointi kvalifikaationa

Kvalifikaationa voidaan pitää työelämään ja jatko-opintoihin valikoitumiseen liittyviä taitoja, joihin kuuluu perustason koodaustaidot (koodin kirjoitus), ohjelmointitaidot (ohjelman suunnittelu) sekä ohjelmoinnillisen ajattelun taidot. Tähän liittyy myös ymmärrys ohjelmointiin liittyvistä laitteista ja sovelluksista, sähkön perusominaisuuksista, itse rakennettavista ja valmiista roboteista sekä internetin ja tekoälyn kaltaisten digitaalisten systeemien toiminnasta. Kyse on siis selkeästi mitattavista taidoista ja erilaisten teknisten prosessien ymmärtämisestä. Teknologian nopea kehitys edellyttää jatkuvaa taitojen uudelleenarviointia ja täydennyskoulutusta. Kvalifikaatioon liittyvät taidot liittyvätkin tiiviisti keskusteluun Suomen kilpailukyvystä ja ne nähdään keinona paikata tulevaisuuden työntekijävajetta**.** Tähän kategoriaan sisältyy kolme teemaa (algoritminen ajattelu ja työskentelyn taidot, laitteiden suunnitteleminen ja rakentaminen, työvoiman saatavuuden varmistaminen).

Ohjelmoinnin opetus varmistaa työvoiman saatavuuden. Ohjelmoinnin opetuksella on tarkastelemissani oppimateriaaleissa voimakas kvalifioiva rooli, sillä opetus pyrkii niiden mukaan tarjoamaan lapsille ja nuorille digitaidot arjessa selviämiseen. Aineistossa korostuu myös tarve ohjelmointitaitoisen työvoiman saatavuudesta, vaikka ohjelmoinnin opetusta perustellaankin lapsille ja nuorille

yleissivistävänä taitona (ks. ohjelmointi sosialisaationa). Oppimateriaaleissa esimerkiksi vakuutellaan, ettei kaikista tarvitse tulla koodareita, mutta kaikkien tulisi saada käsitys siitä, mitä ohjelmointi on.[8] Huomio on kuitenkin työelämässä: ViLLE-oppaassa (s. 5) esimerkiksi puhutaan ”tulevaisuuden aikuisista”, jotka ratkaisevat ongelmia ohjelmistojen avulla. Puhe osaamisvajeesta ei liity niinkään ohjelmointitaitoihin oppilaiden arjessa selviämisen, digitaalisen yhteiskunnan hahmottamisen tai monilukutaidon näkökulmasta, vaan huoli kohdistetaan työvoiman saatavuuteen ja Suomen kilpailukyvyn parantamiseen. Yritysmaailmasta kohdistuu näin kouluun voimakas muutospaine:

> Peruskoulun opetussisältöjä ei voi lähteä muuttamaan yksittäisten yritysten osaamistarpeiden perustella. Silti myös alan puolueettomat selvitykset kertovat, että pula it-alan osaajista on kansainvälinen trendi. (Koodi2016, s. 53)

Teknologia-alan yritysyhteistyötä pidetään tärkeänä, sillä ohjelmointitaidosta arvellaan tulevan jopa työn saannin edellytys (Koodi2016, s. 9, 56). Koska ohjelmoinnin opetuksessa on kyse Suomen maineesta ja kilpailukyvystä, ohjelmoinnin opiskelu nähdään jopa eräänlaisena kansalaisvelvollisuutena:

> Voidaan myös ajatella niin, että halutaan pitää huolta Suomen ja suomalaisen koulusysteemin maineesta. – – Jos Suomi ei ryhdy välittömiin toimiin lasten ja nuorten ohjelmointitaitojen kehittämisessä, muut maat menevät tulevaisuuden kannalta keskeisissä taidoissa ohi. – – tulevaisuuden Suomea rakentavat ohjelmoinnin jo lapsena aloittaneet kooditaiturit. (Koodi2016, s. 9)

Ohjelmointi opettaa algoritmista ajattelua ja työskentelyn taitoja. Ohjelmoinnin opetuksessa on aineiston mukaan keskeistä sen ymmärtäminen, minkälaisia ongelmia ohjelmoimalla voidaan ratkaista ja miten nämä ongelmat voidaan muotoilla koneelle ymmärrettävään muotoon. Ohjelmoinnin opetus rohkaiseekin kokeilemaan erilaisia päättely- ja ongelmanratkaisumalleja.[9] Ohjelmoinnin opiskelu ei siis ole sovellus- tai laitesidonnaista, vaan siinä on kyse erilaisten ajattelun taitojen opettelemisesta.[10] Ajattelun taidoilla viitataan joko yleisiin älyllisiin tai-

[8] (Koodi2016, s. 8, 37, 57, 59, 127; Raapaisu, s. 6)

[9] (Raapaisu, s. 6; ViLLE, s. 4)

[10] (Polkuja 1–6, s. 4; Raapaisu, s. 6)

toihin (Robo2, s. 10–11), tai tarkemmin erojen ja yhtäläisyyksien etsimiseen, luokittelemiseen, syy-seuraussuhteiden luomiseen ja ongelmanratkaisuun.[11] Myös muissa materiaaleissa korostui ohjelmointi toimintaohjeiden kirjoittamisena:

> Ohjelmointi on käskyjen kirjoittamista tietokoneen muistiin niin, että se saadaan toimimaan halutulla tavalla. (Vantaan kaupungin tukimateriaali)

> Ohjelmoinnin opetus lähtee alakoulussa päättely- ja ongelmanratkaisutaitojen kehittämisestä. (ViLLE, s. 4)

Ohjelmointitaitojen koetaan auttavan myös arkisten ongelmien ratkomisessa:

> Ohjelmoinnin avulla oppii myös yleisesti hyödyllisiä kognitiivisia taitoja. Looginen – – ajattelu, tarkka työskentely, kyky hahmottaa ongelma ja muodostaa sille erilaisia ratkaisuvaihtoehtoja sekä visualisoida ja käsitteellistää ne ovat hyödyllisiä kaikissa aineissa ja elämässä itsessään. (Koodi2016, s. 57)

Aineistossa korostuvat algoritmiset taidot, eli vaiheittaisesti etenevien ohjeiden antaminen. Oppimateriaaleissa painotetaan, että käskyjen on oltava yksiselitteisiä, sillä tietokone ei ymmärrä epämääräisiä ohjeita.[12] Tämän vuoksi oppilaiden on hyvä tarkastella valmista koodia ja oppia hyviä koodinkirjoituskäytäntöjä:

> Koodia opetellaan lukemaan ja tulkitsemaan. Tutkimalla ja hyödyntämällä olemassa olevia koodeja omien tuotosten tekemisessä, opitaan myös ymmärtämään, miten koodi toimii. (Polkuja 7–9, s. 7)

Koneelle annettuja ohjeita on tärkeää oppia testaamaan, jotta niissä ilmeneviä virheitä voidaan korjata[13] Oppilaiden onkin hyvä tutustua innovoinnille tyypilliseen iteratiiviseen (useaan kertaan toistuvaa) kehitysprosessiin, johon kuulu ideointia, palautteen saamista, uusien versioiden testaamista ja pyrkimistä kohti yhteisiä tavoitteita.[14]

> Ohjelmoinnissa ja loogisessa ongelmanratkaisussa parhaat tulokset saadaan iteratiivisesti kokeilemalla, havaintoja tekemällä ja virheitä korjaamalla. (Polkuja 1–6, s. 29)

---

[11] (Polkuja 1–6, s. 4; Polkuja 7–9, s. 4; Vantaan kaupungin tukimateriaali)

[12] (Koodi2016, s. 57, Robo1, s. 2; Robo2, s. 40; ViLLE, s. 5; Raapaisu, s. 6; Vantaan kaupungin tukimateriaali)

[13] (Koodi2016, s. 17; Polkuja 1–6, s. 7)

[14] (Polkuja 1–6, s. 17, 37; Polkuja 7–9, s. 23)

Kyse on tarkan työskentelyn taidoista, joissa on olennaista jatkuva prosessin ja oman työskentelyn itsearviointi (Vantaan kaupungin tukimateriaali).

Ohjelmointi mahdollistaa asioiden suunnittelun ja rakentamisen. Oppimateriaaleissa korostettiin myös elektroniikkataitoja, sillä digitaalinen teknologia toimii sähköllä. Ohjelmointi toimii näin tuotannollisena taitona, eli työkaluna asioiden suunnittelemisessa ja valmistamisessa.[15] Tarkastelemalla ja rakentamalla ohjelmoituja laitteita syvennetään ymmärrystä ohjelmoinnista, ohjelmistojen toiminnasta ja asioiden keskinäisistä vuorovaikutussuhteista (Robo2, s. 4). Sama ajatus nousi esiin muissakin oppimateriaaleissa:

> Ympärillämme on päivittäin satoja digitaalisia laitteita, jotka hyödyntävät ohjelmointia. Puhelimet, liikennevalot, pesukone. Tavoitteena on, että pystyisimme paremmin ymmärtämään näiden laitteiden toimintaa, ohjelmoimaan niitä ja keksimään uusia. (Vantaan kaupungin tukimateriaali)

> Ohjelmointi – – vaatii taakseen elektroniikkaa, jolla koodi pyörii, joten elektroniikan perustoiminnan ymmärtäminen voidaan nähdä tärkeänä alustana itse digitaalisuuden ja ohjelmoinnin ymmärtämiselle. (Robo1, s. 2)

6.1.2 Ohjelmointi sosialisaationa

Lapset ja nuoret kasvavat sosialisaation kautta tietyn sosiaalisen, kulttuurisen ja poliittisen yhteisön jäseniksi, joten formaalin kasvatuksen tehtävänä on tarjota heille yhteiskunnan tärkeinä pitämiä taitoja ja arvoja (Biesta, 2020, s. 92). Sosialisaatiolla on siten merkittävä rooli kulttuuria eteenpäin vievänä voimana (Mertala, 2022c, s. 144–150). Ohjelmoinnin opetuksen kautta oppilaat kiinnittyvät digitaalisen yhteiskunnan arvoihin, asenteisiin ja käytäntöihin, joten ohjelmoinnin opetusta voidaan pitää jopa tasa-arvon edellytyksenä. Kvalifikaatio ja sosialisaatio linkittyvät näin toinen toisiinsa. Tähän kategoriaan sisältyy kaksi teemaa (ohjelmointi on tasa-arvokysymys, koodi yhdistää ihmisiä).

Ohjelmointi on tasa-arvokysymys. Aineistossa pidettiin tärkeänä digitaalisen ympäristön hahmottamista niin henkilökohtaisen tietosuojan kuin yhteiskunnalliseen toimintaan osallistumisen näkökulmasta. Raapaisu-oppaan (s. 6) mukaan ohjelmoinnista on tullut välttämätön osa yleissivistävää kasvatusta

[15] (Robo2, s. 10; Koodi2016, s. 45)

koska erilaiset ohjelmat ja ohjelmoitavat laitteet ohjaavat ihmisten toimia yhä enemmän, eli koodi sekä mahdollistaa että rajoittaa ihmisten toimintaa. Polkuja 7–9 -oppaassa (s. 33) tuotiin esille, että algoritmeihin pohjautuva profilointi vaikuttaa lasten ja nuorten toimintaan valikoimalla tietoa, luomalla yhteyksiä ja suuntaamalla toimintamahdollisuuksia: niin julkishallinto, yritykset kuin erilaiset yhteisöt keräävät ihmisistä tietoa, jota monet julkaisevat myös itse sosiaalisessa mediassa. Lapset ja nuoret tarvitsevat näin ollen ymmärrystä sekä tietotekniikasta, internetistä että ohjelmoiduista ympäristöistä, jotta eivät ”jäisi väliinputoajan asemaan”.[16] Tämä voi tarkoittaa syrjäytymistä niin sosiaalisesta kuin yhteiskunnallisesta toiminnasta. Ohjelmoinnin opetukseen liittyvät taidot liittyvät näin arjessa pärjäämiseen (Polkuja 1–6, s. 4) ja auttavat ymmärtämään digitaalisten järjestelmien toimintaa, huolehtimaan henkilökohtaisista tiedoista ja kehittämään tietoturvaa (Polkuja 7–9, s. 33). Voidaan puhua *digitaalisista kansalaistaidoista*, jotka ovat tasa-arvoisen yhteiskunnan edellytys (Robo2, s. 10–11). Aineistossa korostuikin tarve digitalisoituneen yhteiskunnan hahmottamiseen:

> Ohjelmointiosaamisessa ja digitaalisten rakenteiden ymmärtämisessä on kyse myös tasa-arvosta, demokratiasta ja valtasuhteista: Taitojen ja ymmärryksen epätasapaino voi johtaa osallisuuden eriytymiseen. (Polkuja 7–9, s. 33)

> – – pitäisi saada läpi ajatus siitä, että ohjelmointi ei ole erityistaitoa jollekin tietylle pienelle porukalle. (Koodi2016, s. 37)

Koodi yhdistää ihmisiä. Oppimateriaaleissa tuotiin myös esille digitaalisen teknologian ajasta ja paikasta riippumaton luonne, mikä on luonut kokonaan uudenlaisia sosiaalisia käytänteitä:

> Etsimme tietoa internetistä, kommunikoimme Whatsappissa, lataamme kuvia Instagramiin. Digitaalinen teknologia yhdistää useat mediat – – linkittäen niitä toisiinsa ja luoden uudenlaisia käytäntöjä. (Robo2, s. 9)

Scratchin kaltaiset selainpohjaisten ohjelmointiympäristöjen kuvattiin yhdistävän harrastajia kielirajoista riippumatta (Raapaisu, s. 20–21). Samalla kun laitteet ja ohjelmistot korvaavat ihmisten välistä välitöntä vuorovaikutusta, olemme

[16] (Koodi2016, s. 52; Robo2, s. 9, 11)

myös yhä riippuvaisempia teknologiasta. Robo2-oppaassa (s. 4) esimerkiksi huomautetaan, että digitalisaation edellytyksenä on rinnakkaiselo erilaisten laitteiden ja ohjelmoitujen ympäristöjen kanssa. Oppimateriaaleissa korostettiin myös yhdessä oivaltamisen riemua. Esimerkiksi Polkuja 1–6 -oppaassa (s. 18) kannustetaan pariohjelmointiin vaikka kullakin oppilaalla olisi oma tietokone, sillä parin kanssa on mielekästä jakaa ideoita ja suunnitella projekteja. Polkuja 7–9 -oppaassa huomautetaankin, ettei ohjelmointiosaamisessa ole kyse pelkistä koodinkirjoitustaidoista:

> Kyse ei ole pelkästään koodaamisesta, vaan osaamiseen kuuluvat olennaisesti myös monipuoliset ajattelun ja yhteistyön taidot sekä ymmärrys digitaalisesta, ohjelmoidusta maailmasta ja siinä toimimisesta. (Polkuja 7–9, s. 4)

6.1.3 Ohjelmointi subjektifikaationa

Subjektifikaatiossa on kyse elämisestä ”maailmassa ja maailman kanssa” (Biesta, 2010, s. 25), eli omien mahdollisuuksien ja rajoitteiden hahmottamisesta. Siihen kuuluu omana itsenä oleminen, itseilmaisu, aktiivinen digitaalinen toimijuus ja pyrkimys kehittää asioita. Tähän kategoriaan sisältyy neljä teemaa (ohjelmointi konkretisoi digitaalista maailmaa, kuluttajasta tekijäksi, ohjelmointi motivoi, miltä digitalisaatio tuntuu), jotka liittyvät siihen, mitä ohjelmoinnin oppiminen merkitsee oppilaille itselleen. Ohjelmointi on taito, joka mahdollistaa itseilmaisun sekä omien ohjelmien luomisen ja projektien toteuttamisen. Tämän lisäksi erilaiset digitaaliset alustat ja niihin liittyvät yhteisöt mahdollistavat eri alakulttuurien muodostamisen ja harrastajaryhmien välisen yhteydenpidon.

Ohjelmointi konkretisoi digitaalista maailmaa. Tarkastelemissani materiaaleissa tuotiin esille ohjelmointi eräänlaisena nykyajan kädentaitona joka tuo näkyväksi teknologian piilotettua luonnetta. Projekteissa voidaan esimerkiksi hyödyntää käytöstä poistettujen lelujen elektronisia komponentteja, jolloin käsillä tekeminen tekee teknologiasta helpommin lähestyttävää ja kehittää kriittistä ajattelua (Robo1, s. 3). Ohjelmointi voi näin ollen konkretisoida digitaalista maailmaa, toimia fyysisen ja abstraktin välisenä tulkkina ja tarjota digitaaliseen todellisuuteen liittyvää kehollista ymmärrystä (Robo2, s. 10–12):

> Kun ohjelmointi ymmärretään käsillä tekemisenä, se ei ole vain tuotannollinen taito, vaan keino luoda yhteys digitalisoituneeseen yhteiskuntaan. – – Tämä edellyttää kuitenkin ohjelmoinnin käsittämistä laajemmin kuin vain osana matematiikan opetusta. Älyllisen ajattelun ohella ohjelmointi on työkalu itsensä ilmaisuun sekä keino ymmärtää ja muokata ympäröivää maailmaa. (Robo2, s. 10–12)

> Usein elektroniikasta ja ohjelmoinnista luodaan mystistä kuvaa: on taianomaista kun valot syttyvät äänen voimasta tai leikkirobotti osaa puhua. Mielikuvaa tukee digitaalisen teknologian suljettu luonne. Ei haluta jakaa omia keksintöjä, vaan piilotetaan ja estetään rakenteiden ymmärtäminen. Rakenne on peräisin yritysmaailmasta, jossa luonnollisesti halutaan piilotella omia keksintöjä, jotteivat toiset varastaisi niitä. Samalla piilottelu kuitenkin estää meitä ymmärtämästä koko teknologiaa. (Robo1, s. 3)

Passiivisesta kuluttajasta aktiiviseksi tekijäksi. Koska emme useinkaan ymmärrä, miten käyttämämme teknologia toimii, oppimateriaaleissa pidettiin tärkeänä omakohtaista kokemusta digitaalisten laitteiden ja ohjelmien suunnittelusta ja toteuttamisesta. Koodi2016- (s. 73) mainitaan, että oppilaan ajattelutapa mullistuu, kun hänestä tulee digitaalisten sisältöjen kehittäjä pelkän kuluttamisen sijaan. Samantyyppinen ajatus nousi esiin Robo2-oppaassa:

> Digitaalinen maailma esittäytyy meille usein valmiina. – – toimimme siinä käyttäjinä ja asiakkaina mutta emme tekijöinä. (Robo2, s. 10)

Sekä Koodi2016- että Raapaisu-oppaassa huomautetaan, ettei opettajakaan aina tiedä oikeaa vastausta, jolloin hän voi tarkastella ongelmaa oppilaan kanssa yhdessä (Koodi2016, s. 121; Raapaisu, s. 6). Tämä antaa tilaa oppilaan omalle asiantuntijuudelle:

> Ohjelmoinnin oppimisen polku voi olla yhteinen opettajalle ja oppilaalle. (Raapaisu, s. 6)

Ohjelmointi motivoi. Oppimateriaaleissa haluttiin korostaa ohjelmoinnin motivoivaa puolta. Vaikka ohjelmoinnin opettelu on alkuvaiheessa haastavaa (Koodi2016, s. 90), graafisten ohjelmointiympäristöjen valmiit käskypalikat vähentävät kirjoitusvirheiden mahdollisuutta ja madaltavat yrittämisen kynnystä. Kun ohjelmointiprojektit liittyvät lisäksi lasten ja nuorten omaan kokemusmaailmaan ja kiinnostuksen kohteisiin, niistä tulee merkityksellisiä ja innostavia (Polkuja 1–6, s. 4). Raapaisu-opas (s. 6) painottaakin, että ohjelmoinnin perusteiden

opettamisessa on tärkeintä oppilaiden innostaminen ja keksimisen ja oivaltamisen riemu. Monialaisten tieto- ja viestintäteknologisten projektien koettiin lisäävän opiskelumotivaatiota (Polkuja 7–9, s. 14), sillä ohjelmoinnin oppimistavoitteet ovat yksittäisiä oppiaineita laajemmat (Vantaan kaupungin tukimateriaali). Aineistosta nousi esille, että ohjelmoinnin opiskelu on innostaa monella tavalla:

> Robotiikka on – – ilmiö ja monialainen oppimiskokonaisuus, jonka motivaatiovoima on vertaansa vailla. – – robotiikka on aina riippuvainen ohjelmoinnista. (Vantaan kaupungin tukimateriaali)

> Pääasia on halu tuoda esille, mistä ohjelmoinnissa on kysymys. Ja innostaa nuoria huomaamaan, että ohjelmointi on jännittävää. (Koodi2016, s. 48)

Scratchin kaltaiset ohjelmointialustat mahdollistavat erilaisten oppimisyhteisöjen muodostamisen (Raapaisu, s. 20). Omien projektien jakaminen ja palautteen saaminen tekee yhteiskehittäjyyden prosessia näkyväksi (Polkuja 1–6, s. 37), mikä tukee subjektifikaatioon kuuluvaa itsenäistä ajattelua sekä omien mahdollisuuksien ja rajojen kohtaamista (reality check; ks. kappale 3.1):

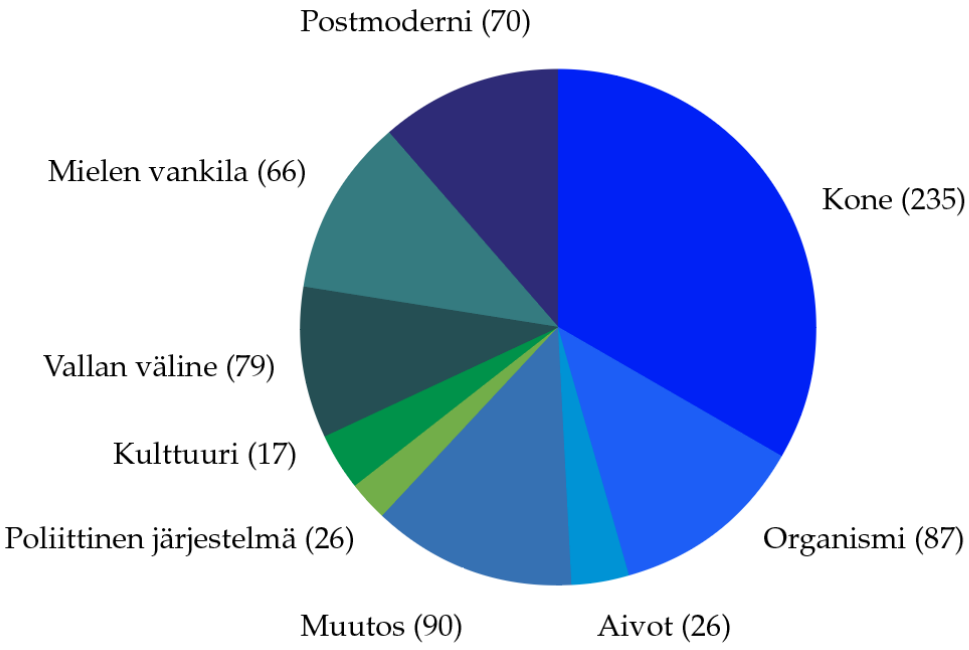
> Ohjelmakoodien jakaminen on yleinen ja tärkeä keino edistää ohjelmoinnin kehittymistä. Älä siis turhaan omi ohjelmaasi vaan jaa se myös muiden muokattavaksi. Vastavuoroisesti opit paljon toisilta. (Robo2, s. 61)

Miltä digitalisaatio tuntuu? Aineistossa jäi huomattavasti vähemmälle huomiolle sen tarkasteleminen, miltä maailma näyttää teknologian välityksellä. Koodin avulla voidaan kuitenkin pohtia yhä keskeisemmäksi muodostunutta ihmisen ja teknologian välistä suhdetta (Robo2, s. 4–6). Erilaiset pelit, videopalvelut ja viestintäpalvelut voivat esimerkiksi ylittää aikaan ja paikkaan liittyviä rajoitteita, sillä fysiikan lait eivät rajoita ohjelmistojen ja palveluiden toimintaa (Polkuja 1–6, s. 39). Digitaalisen teknologian käyttö luo näin mielikuvan vapaudesta, mutta vapauden mukana tulee vastuu: käyttäjän pitää pystyä rajoittamaan toimintaansa, sillä ohjelmistojen algoritmit tarjoavat jatkuvasti uutta sisältöä ja suuntaavat toimintamahdollisuuksiamme (Polkuja 7–9, s. 33). Tunneilla voidaan esimerkiksi tarkastella ohjelmoidun ja ei-ohjelmoidun maailman eroavaisuuksia, ja ottaa lähtökohdiksi kysymykset siitä, miltä teknologia tuntuu, ja miten se vaikuttaa siihen, miten havainnoimme ympäristöämme (Robo2, s. 4–6):

> Teknologia on tapamme olla olemassa ja jopa ajatella. Teknologian avulla lisäämme itseemme kykyjä, joita meillä ei ole. – – Teknologia on kuin ulkoistettu muistimme, joka auttaa kehitystämme, mutta myös edesauttaa unohtamista. (Robo2, s. 9)

6.2 Ohjelmoinnin oppimateriaalit suhteessa koodimetaforiin

Tarkastelen seuraavaksi ohjelmoinnin oppimateriaaleja ja niissä esiintyviä havaintoyksiköitä (lause tai kappale) suhteessa Tomi ja Mikko Dufvan (2016) koodimetaforiin. Funktionaalisia metaforia esiintyi tutkimusaineistossa eniten (kuvio 3), ja niitä kertyi yhteensä 438 kappaletta (kone 235, organismi 87, aivot 26, muutos 90). Tulkitsevia metaforia esiintyi tutkimusaineistossa yhteensä 43 kappaletta (kulttuuri 17, poliittinen järjestelmä 26), emansipatorisia metaforia (mielen vankila 66, vallan väline 79) löytyi yhteensä 145 kappaletta ja postmoderniin metaforaan (karnevaali) liittyviä havaintoja 70 kappaletta.



Kuvio 3. Koodimetaforien esiintyvyys ohjelmoinnin oppimateriaaleissa.

6.2.1 Funktionaaliset metaforat: koodi koneena, organismina, aivoina ja muutoksena

Funktionaaliset metaforat ja näistä erityisesti kone- ja organismi-metafora edustavat vallitsevaa tapaa lähestyä ohjelmointia (Dufva & Dufva, 2016, s. 106). Funktionaaliset metaforat liittyvät opetuksen kvalifioivaan puoleen, eli työelämässä hyödynnettäviin taitoihin ja mitattavissa olevaan osaamiseen. Kaikissa metaforissa ohjelmointi nähdään arvovapaana, matemaattis-loogisena ongelmanratkaisuvälineenä ilman kysymyksiä yhteiskunnallisista tai eettisistä epäkohdista (Dufva & Dufva, 2016, s. 100–101).

Kone-metaforassa koodi nähdään lineaarisena sarjana koneelle tietyssä järjestyksessä syötettäviä komentoja, jotka käsitellään tietokoneessa ja tulostetaan lopuksi ratkaisuna käyttäjälle (Dufva & Dufva, 2016, s. 100). Raapaisu-oppaassa esimerkiksi (s. 3) mainitaan, että ”ohjelmointi eli koodaaminen on yksityiskohtaisten ohjeiden antamista tietokoneelle tai muulle mikroprosessoripohjaiselle laitteelle”. Koodi myös perustuu joko-tai-ajatteluun, jossa jokin asia joko tapahtuu tai ei tapahdu, sillä tietokone ymmärtää vain ykkösiä tai nollia (Robo2, s. 17). Tietokone ei siis kyseenalaista mitään, vaan tekee täsmälleen sitä, mitä koodaaja on koodiin kirjoittanut (ViLLE, s. 27). Koodi on näin ollen digitaalisen teknologian rakennetta luova voima, jonka avulla saamme elektroniikan toimimaan haluamallamme tavalla (Robo1, s. 3). Koodiin ei liity arvokysymyksiä, joten keskustelu liittyy lähinnä sopivan ohjelmointikielen valitsemiseen niin kyseistä ongelmaa kuin myöhempää työllistymistä ajatellen (Dufva & Dufva, 2016, s. 101).

> Ohjelmoinnin opettaminen peruskoulutasolla on ennen kaikkea ohjelmoinnillisen ajattelun perusteiden oppimista – –. Ohjelmoinnillisen ajattelun perusteita ovat esimerkiksi kyvyt pilkkoa ongelma osiin, antaa yksikäsitteisiä komentoja tietokoneelle ja pohtia, mitkä komennot missä järjestyksessä ratkaisevat ongelman. (Koodi2016, s. 9)

> Musiikkinuottien lukeminen ja kirjoittaminen on hyvin samankaltaista ohjelmoinnin kanssa. Molemmissa asiat tapahtuvat järjestyksessä, toistuvasti ja sääntöjen mukaan. (Polkuja 1–6, s. 19)

Organismi-metaforassa koodi muodostuu keskenään vuorovaikutuksessa olevista koodiryhmistä, mikä parantaa koodin toiminnallisuutta (Dufva & Dufva,

2016, s. 101). Esimerkkejä organismi-metaforasta löytyi muun muassa Raapaisu-oppaasta:

> Algoritmien yleiskäyttöisyyttä ja luotettavuutta voi parantaa laatimalla modulaarisia ohjelmia. Modulaarisessa ohjelmoinnissa ohjelma jaetaan pienempiin osiin, joiden avulla ratkaistaan koko tehtävä. Näin ohjelmasta saadaan helpommin luettava ja virheiden korjaaminen sekä ohjelman ylläpito helpottuvat. (Raapaisu, s. 12)

Aivot-metaforassa koodi nähdään keskusyksikkönä, joka luo uutta tietoa. Metafora liittyy muun muassa tekoälyyn, koneoppimiseen ja pilvilaskentaan, mutta niihin liittyvät eettiset kysymykset jäävät metaforan ulkopuolelle (Dufva & Dufva, 2016, s. 101–102). Vantaan kaupungin tukimateriaalissa esimerkiksi mainitaan, että tekoäly on tietokoneen ohjelmoimista jäljittelemään ihmisen ajattelua eli oppimaan uusia asioita ja soveltamaan opittua tietoa älykkäällä tavalla. Nykyalgoritmit tunnistavatkin jo erinomaisesti kieliä niin tekstuaalisessa kuin puhutussa muodossa, joten omaa ohjelmaa tehtäessä voidaan havaita, että ohjelmalle syötetyn tekstin määrä parantaa kielen tunnistusta (Polkuja 7–9, s. 26).

> Ohjelmoinnista voi ajatella myös, että luotu koodi on kuin koneen ajatukset muun laitteiston eli elektroniikan ollessa sen keho. Digitaalisessa maailmassa ohjelmoidut ”ajatukset” ovat aina vain lopulta joko kyllä tai ei – binäärisesti ilmaistuna joko 1 tai 0. Tietokoneen aivot voivat esimerkiksi lähettää komennon johtoja pitkin käskien moottorin mennä päälle ja pois päältä samalla tavalla kuin voimme ajatella itsekin liikuttavamme jalkaa. Kuitenkaan ei tule erehtyä ajattelemaan, että ihminen olisi kuin ohjelmoitu tietokone. (Robo2, s. 38)

Muutos-metafora laajentaa ajattelua tietokoneista niiden ympäristöön ja kiteytyy koodin lupaukseen parantaa maailmaa. Ohjelmointi nähdään yleispätevänä ratkaisuna mihin tahansa ongelmaan, joka voidaan pilkkoa osiin ja ratkaista koodin avulla. Muutoksen tarvetta tai suuntaa ei kuitenkaan pohdita. (Dufva & Dufva, 2016, s. 102). Tämä näkyi myös tarkastelemissani oppimateriaaleissa, joissa ohjelmointi esitettiin yleishyödyllisenä ongelmanratkaisutyökaluna. Monissa oppimateriaaleissa korostettiin ohjelmoinnin kognitiivisia hyötyjä, kuten algoritmisen ajattelun hyödyntämistä tiedon käsittelyssä ja ongelmanratkaisussa eri oppiaineissa.[17] Ohjelmoinnin koettiin auttavan ihmisen luovien ideoiden ja

[17] (Koodi2016, s. 57; Polkuja 1–6, s. 4; Polkuja 7–9, s. 4, 7; Vantaan kaupungin tukimateriaali)

tietokoneen voimavarojen yhdistämisessä: ohjelmointi mahdollistaa uusien lääkkeiden kehittämisen, tietokonepelien suunnittelemisen, robottien rakentamisen ja planeettojen tutkimisen (Koodi2016, s. 121). Esimerkiksi ViLLE-oppaassa näkyy voimakas usko ohjelmointitaitojen mahdollistamaan muutokseen:

> Tämän päivän peruskoululaisesta kasvaa hydrobiologi joka kehittää tekniikkaa jolla puhdistetaan valtameriä, sairaanhoitaja joka pystyy käyttämään enemmän aikaa potilaan kanssa ja opettaja joka ottaa rohkeasti käyttöön uusia opetusta parantavia työkaluja. (ViLLE, s. 5)

6.2.2 Tulkitsevat metaforat: koodi kulttuurina ja poliittisena järjestelmänä

Tulkitsevissa metaforissa huomioidaan ohjelmoinnin yhteiskuntaa muokkaava vaikutus, ja niissä on sekä kvalifikaation että sosialisaation piirteitä. **Kulttuurimetafora** laajentaa koodia ohjeiden antamisesta merkittäväksi kulttuuri-ilmiöksi (Dufva & Dufva, 2016, s. 102–103). Ohjelmointiin liittyy esimerkiksi erilaisia alakulttuureja, kuten avoimen lähdekoodin yhteisöt, hakkerikulttuuri ja itse tekemistä korostava *maker movement* (Hatch, 2014, s. 11−31). Oppimateriaaleissa ei juurikaan avata koodia kulttuuria muokkaavana ilmiönä, mutta Robo2-oppaassa (s. 61) kannustetaan avoimeen ohjelmointikulttuuriin ja koodin jakamiseen. Siinä painotetaan lisäksi tasa-arvoisen ja yhteistoimintaa korostavan yhteiskunnan ylläpitämiseen tarvittavia digitaalisia kansalaistaitoja: nämä perustuvat koodin loogisen luonteen ymmärtämisen lisäksi konkreettiseen käsillä tekemiseen, mikä on jäänyt vähemmälle huomiolle (Robo2, s. 10–11). Elektroniikkarakentelu ja luova, kokeellinen ohjelmointi (Dufva, 2021, s. 272–273; Dufva, 2018) ovat keinoja digitaalisen todellisuuden hahmottamiseen. Elektroniikkarakentelu konkretisoi ohjelmoinnissa tarvittavia taitoja ja voimaannuttaa lapsia ja nuoria tarkastelemaan teknologiaa heidän omilla ehdoillaan, valmiiden kaupallisten tarjoumien ulkopuolelta (Dufva, 2017b, s. 138). Ohjelmoinnin toimintatavoista kehittyy omia alakulttuureitaan ja käytänteitään, jotka luovat yhteenkuuluvuuden tunnetta:

> Ohjelmointipiireissä on 1970-luvulta lähtien yleistynyt sympaattinen tapa: kun uutta kieltä lähdetään opettamaan, perinteisesti ensimmäinen asia on opetella, miten kielen avulla tulostetaan tietokoneen näytölle sanat ”Hello World!” (Koodi2016, s. 41)

Poliittinen järjestelmä -metafora tuo mukaan poliittisen ulottuvuuden ja kysymykset siitä, kenellä on oikeus datan keräämiseen ja jakamiseen (Dufva & Dufva, 2016). Kun koodi nähdään kulttuuri-metaforassa yksilön näkökulmasta, poliittinen systeemi -metaforassa tarkastellaan, miten koodi luo arkeemme vaikuttavia hierarkkisia rakenteita. Esimerkiksi verkosta ladattava tieto oli aluksi kaikkien saatavilla, mutta on sittemmin suojattu tekijänoikeuksin tai erilaisin rajoituksin. Kysymykset siitä, miten ja miksi koodia tulisi kontrolloida ovat yhä keskeisempiä koodin nivoutuessa yhä tiiviimmin jokapäiväiseen elämäämme verkkopohjaisten sovellusten ja älypuhelinten avulla (Dufva & Dufva, 2016, s. 103–104). Digitaalisessa maailmassa korostuukin ymmärrys tiedon luotettavuudesta ja tietosuojasta (Polkuja 7–9, s. 33, 35). Polkuja 1–6 -oppaan avulla oppilaat voivat esimerkiksi pohtia, minkälaista tietoa hänen toiminnastaan kerätään. Tavoitteena on, että he pystyvät nimeämään vähintään yhden tahon, joka hyödyntää heistä kerättyä tietoa (Polkuja 1–6, s. 39–40).

> Toimiessamme verkkoympäristöissä, meistä kerätään tietoa, jonka avulla järjestelmät profiloivat meitä ikään kuin laatikoihin. Profiloinnin perusteella meille tarjotaan muun muassa yksilöllisiä videoehdotuksia ja mainoksia. Kyseessä on sarja lajittelualgoritmeja, jotka muodostavat suosittelujärjestelmän. (Polkuja 1–6, s. 15)

6.2.3 Emansipatoriset metaforat: koodi mielen vankilana ja vallan välineenä

Emansipatorisissa metaforissa otetaan huomioon koodiin liittyvät valtakysymykset niin yksilön kuin yhteiskunnan tasolla (Dufva & Dufva, 2016, s. 105). Metaforassa tuodaan esille, että koodi voi sekä rajoittaa että voimaannuttaa, eli ne liittyvät etenkin koodin sosialisoivaan puoleen.

Mielen vankila -metafora tuo esille ihmisen ja koodin välisiä haasteita, kuten sen, että koodi ohjaa ihmisten ajattelua ja käyttäytymistä. Polkuja 1–6 -oppaan avulla oppilas voi muun muassa tarkastella mekanismeja, joiden avulla erilaiset sisältöpalvelut seuraavat käyttäjiä ja tarjoavat heille jatkuvasti kiinnostavaa sisältöä. Oppilaat voivat esimerkiksi keskustella Yle Areenan ja YouTuben kaltaisten sisällönsuosittelujärjestelmien hyödyistä ja haitoista (Polkuja 1–6, s. 44). Polkuja 7–9 -opas ohjaa lapsia ja nuoria myös tarkastelemaan vapaa-ajalla ja koulussa

käyttämiään palveluita. Oppilaiden on esimerkiksi hyvä pohtia, minkälaisia motiiveja sovellusten taakse kätkeytyy tai miten erottaa kaupallinen ja yleishyödyllinen palvelu toisistaan. Lisäksi heidän on hyvä pohtia sovellusten ansaintalogiikkaan liittyviä eettisiä kysymyksiä (Polkuja 7–9, s. 36).

> Toimiessamme verkkoympäristöissä, meistä kerätään tietoa, jonka avulla järjestelmät profiloivat meitä ikään kuin laatikoihin. Profiloinnin perusteella meille tarjotaan muun muassa yksilöllisiä videoehdotuksia ja mainoksia. Kyseessä on sarja lajittelualgoritmeja, jotka muodostavat suosittelujärjestelmän. (Polkuja 1–6, s. 15)

Robo1-oppaassa mainitaan, että digitaalisuus nojaa aina valintaan ja asioiden yksinkertaistamiseen. Koska koodi perustuu ykkösiin ja nolliin, eli kyllä- tai ei-vastauksiin, siitä kumpuava ajattelu johtaa asioiden yksinkertaistamiseen. On kuitenkin tärkeää tiedostaa, etteivät reaalimaailman ilmiöt ole koskaan näin selkeärajaisia (Robo1, s. 2–3). Tämä on myös hyvä tuoda mukaan koodista käytävään keskusteluun:

> Koodi – – heijastaa laajalti niin tiedostettuja kuin tiedostamattomiakin arvovalintoja, oli kyseessä sitten ohjelmoijan, ohjelmointifirman tai laajemman kulttuurisen taustan käsitys hyvästä koodista. (Robo2, s. 9)

Vallan väline -metaforassa keskittyy koodin mahdollisuuksiin hallita yhteisöjä käyttäjistä kerätyn personoidun datan avulla. Ohjelmoinnillista ajattelua voidaankin pitää taitona, jota ilman emme pysty osallistumaan yhteiskunnalliseen keskusteluun (Robo2, s. 9). Ymmärrys erilaisten ohjelmoitujen ympäristöjen toiminnasta auttaa lisäksi kehittämään tietoturvaa ja huolehtimaan henkilökohtaisista tiedoista (Polkuja 1–6, s. 33). Koodi mahdollistaa myös kapinoinnin valtaapitäviä vastaan esimerkiksi tietojärjestelmiä hakkeroimalla, joten koodi voidaan nähdä sekä vallan välineenä että sen ilmentymänä (Dufva & Dufva, 2016, s. 105).

Polkuja 7–9 -opas (s. 41) ohjaakin oppilaita pohtimaan, mitä keinoja heidän käyttämillään sovelluksilla ja verkkosivuilla on kerätä käyttäjätietoja. Opas laajentaa ohjelmoinnin näkökenttää automaatioon ja robotiikkaan liittyviin yhteiskunnallisen tason eettisiin kysymyksiin (Polkuja 7–9, s. 39). Tulevaisuutta ajatellen onkin hyvä pohtia menneisyyttä, kuten sitä, ketkä osallistuivat ensimmäisten tietokoneiden tai internetin kehittämiseen ja mihin tarkoitukseen ne alun perin kehitettiin (Koodi2016, s. 125). Näin he oppivat huomaamaan, että teknologia on

aina jonkun ihmisen tiettyä kayttötarkoitusta varten suunnittelemaa. Robo2-oppaassa pyritäänkin voimaannuttamaan lapsia ja nuoria muodostamaan omia mielipiteitään arjessa kohtaamastaan teknologiasta:

> Parhaimmillaan opetuksen keskiöön nousee uusia toiminta- ja ajattelutapoja, joissa lapsi ymmärtää, ettei teknologian käytössä tarvitse alistua vain muiden määrittelemiin toimintatapoihin vaan voi alkaa luomaan aktiivisesti omia ratkaisuja. (Robo2, s. 6)

6.2.4 Postmoderni metafora: koodi karnevaalina

Sekä emansipatoriset metaforat (mielen vankila ja vallan väline) että postmoderni metafora liittyvät subjektifikaatioon, eli siihen, miten ohjelmoinnin opetus voi tukea lasten ja nuorten kasvua autonomisiksi, itsenäisesti ajatteleviksi yksilöksi. Postmoderni **karnevaali-metafora** keskittyy niihin mekanismeihin, joiden avulla luomme uusia merkityksiä ja korostaa koodin roolia inspiraation lähteenä (Dufva & Dufva, 2016, s. 106). Ohjelmointia voidaan esimerkiksi lähestyä leikillisesti, jolloin huomio kiinnittyy rakenteen sijaan kokemukseen ja sellaisiin merkitysrakenteisiin, joihin ei muuten kiinnitettäisi huomiota (Dufva, 2017a, s. 11). Robo2-opas esimerkiksi painottuu luovaan mediakasvatukseen, jossa kannustetaan omaehtoisen ilmaisun löytämiseen. Tällöin ohjelmointi laajenee matemaattis-luonnontieteellisestä kontekstista henkilökohtaiseen kokemukseen ja oppilaalle merkitykselliseen ilmaisuun (Robo2, s. 4, 6), jossa ei tarvitse aina pohtia, onko jonkin asian toteuttaminen mahdollista vai ei (Robo2, s. 75).

> On hyvä muistaa, että ohjelmoitaessa on monia tapoja tehdä sama asia, eikä yhtä oikeaa ratkaisua ole. – – joskus myös väärin toimiva koodi saattaa osoittautua mielenkiintoiseksi mahdollisuudeksi kehittää ohjelmaa aivan uuteen suuntaan. Älä siis pelkää epäonnistumista. (Robo2, s. 38)

Oppilaita kannustetaankin useammassa oppimateriaalissa[18] hyödyntämään ohjelmointia omassa itseilmaisussaan ja luovassa työskentelyssään. Lopputulos voi olla taiteellisen työskentelyn tavoin odottamaton ja yllättävä:

> Ohjelmoinnissa ei koskaan ole yhtä oikeaa ratkaisua. Erilaisten ratkaisutapojen oivaltaminen avaa mahdollisuuksia kaikelle vielä keksimättömälle. (Raapaisu, s. 6)

[18] (Koodi2016, s. 126; Robo2, s. 4, 6; Polkuja 7–9, s. 4)

7 POHDINTA

Tutkimuksen tavoitteena oli selvittää, minkälaisia merkityksiä ohjelmoinnille oppimateriaalissa annetaan. Lähestyin aihetta tarkastelemalla kahdeksaa oppimateriaalia ja vertaamalla niissä esiin nousevia teemoja Gert Biestan (2010) koulutuksen tehtäviin sekä Tomi ja Mikko Dufvan (2016) koodimetaforiin.

7.1 Kvalifikaatio, sosialisaatio ja funktionaalinen metafora ylikorostuvat ohjelmoinnin oppimateriaaleissa

Tarkastelemassani aineistossa korostui funktionaalinen koodimetafora sekä ohjelmointi kvalifikaationa ja sosialisaationa. Kvalifioiva fokus näkyy muun muassa siinä, että lapsille ja nuorille pyritään tarjoamaan ennen kaikkea monipuolisia koodinkirjoitustaitoja. Ohjelmoinnin opetus lähtee ikään kuin siitä oletuksesta, että ohjelmoinnissa tarvittavat osat ovat jo olemassa (kuten numerot matematiikassa) ja niitä on vain opittava hyödyntämään. Tutkimukseni vahvisti näin aiempien tutkimusten (Dufva & Dufva, 2016; Mertala ym., 2020) tuloksia siitä, että ohjelmoinnin opetuksessa korostuu tekninen näkökulma koodiin. Sosialisaation korostuminen näkyi siinä, että ohjelmointi nähdään digiajan kansalaistaitona ja tulevaisuuden työelämätaitona. Tämä on yhteneväinen Mertalan ym. (2020, s. 22–23) havainnoille, joiden mukaan ohjelmointiin liittyvä keskustelu on tulevaisuuspainotteista ja työelämäkeskeistä, vaikka julkisessa keskustelussa toistuukin ajatus ohjelmoinnista nykyajan kansalaistaitona. Subjektifikaatioon ja postmoderniin koodimetaforaan sisältyviä kriittisiä näkökulmia huomioitiin oppimateriaaleissa huomattavasti vähemmän, ja ohjelmoinnin katsottiin lisäävän lasten ja nuorten digitaalista osaamista kuin itsestään.

Tarkastelemieni oppimateriaalien näkemys ohjelmoinnista oli siis kahtia jakautunut ja se liitettiin joko uhkiin tai mahdollisuuksiin, joilla kummallakin perusteltiin ohjelmoinnin välttämättömyyttä. Funktionaalisiin metaforiin (kone, organismi, aivot, muutos) liittyviä teemoja oli eniten, eli ohjelmoinnin opetuksessa painottuu ohjelmoitujen ympäristöjen ja laitteiden toimintalogiikan ymmärtäminen. Toiseksi suurin ryhmä olivat emansipatoriset metaforat (mielen vankila,

vallan väline), jotka liittyivät ensisijaisesti ohjelmoinnin välttämättömyyteen osana perusopetusta (kvalifikaatio ja sosialisaatio) digitaalisten kansalaistaitojen ja tulevan digitalisaatiokehityksen mukana pysymisen kannalta, eivätkä liittyneet niinkään oppilaan voimaannuttamiseen ja digitaaliseen osallistamiseen tässä hetkessä (subjektifikaatio). Mertala ym. (2020) huomauttavatkin, että huomion ollessa työelämän ja yhteiskunnan vaatimuksissa, unohdetaan että myös nykyisyys on merkityksellistä. Tulevaisuudessa eläminen opettaa piiloisesti täyttämään ulkoisia, valmiiksi määriteltyjä odotuksia, eikä jätä tilaa oppilaan henkilökohtaiselle kasvulle (Mertala ym., 2020, s. 30). Vaikka postmoderni, eri näkökulmia yhdistävä metafora oli oppimateriaaleissa hyvin esillä, sitä lähestyttiin innostavasta ja oppiaineita yhdistävästä hyötynäkökulmasta, ei niinkään yhteiskuntakriittisestä tulokulmasta. Ohjelmointi poliittisena järjestelmänä ja kulttuurina olikin tarkastelemissani oppimateriaaleissa kaikkein vähiten esillä, vaikka ohjelmoinnin alakulttuurien tarkasteleminen voisi olla oppilaille hyvinkin motivoiva tapa lähestyä ohjelmointia.

Koska ohjelmoinnin oppimateriaalit ja täydennyskoulutus ovat pitkälti ulkopuolisten tahojen tuottamia (Mertala ym., 2020, s. 41), ohjelmointi kytkeytyy tiiviisti koulun ulkopuolisiin toimijoihin, jotka vaikuttavat kasvatuksen arvoihin ja päämääriin (Simons ym., 2013, s. 419). Näiden tahojen intresseissä ei ole välttämättä niinkään kriittinen keskustelu ohjelmoinnin yhteiskunnallisista vaikutuksista, vaan ohjelmointitaitoisen työvoiman tuottaminen. Vaikka koulun tulee mediassa usein ”avautua maailmalle” ja ”seurata aikaansa” (Saari, 2020), yritykset eivät lähesty asioita näin suoraan, vaan käyttävät innostavia ja optimistisia sanavalintoja (Ideland ym., 2021, s. 93–94, 97). Ohjelmoinnin opetusta perustellaan koulutusalan päättäjille usein helppona ja vaivattomana ajattelun taitona ja eräänlaisena opetuksen lisäosana,[19] vaikka algoritmiset käytännöt kietoutuvat koulun arkeen niin ohjelmistojen, laitteiden kuin koneoppimisen keinoin (Williamson & Hogan, 2020). Konkreettisia haasteita, kuten oppilaan tietoturvakysy-

[19] ”Kun ohjelmointi *laitetaan* peruskouluun, se antaa kaikille samat mahdollisuudet – –.” (Koodi2016, s. 45).

myksiä (Williamson, 2017, s. 3–4) ei kuitenkaan käsitellä, vaan keskustelu on tulevaisuuspainotteista ja ratkaisukeskeistä. On kuin ongelmia ei olisikaan, vain mahdollisuuksia (Ideland ym., 2021, s. 95–96). Tämän kaltainen puhe esittää oppimisen yksinkertaisena tapahtumana, jota on helppo ohjailla myyntipuhein ja paketoida oppimistuotteiksi (Jokisaari, 2017, s. 109, 74–75).

Tutkimukseni antaa myös varovaisia viitteitä siitä, että Koodi2016-oppaan kaltaisilla vanhemmilla ohjelmoinnin oppimateriaaleilla on eräänlainen auktoriteettiasema. Koodi2016 sai jo ilmestyessään paljon huomiota ja siinä esitettyjä näkökulmia nostetaan yhä uudestaan keskusteluun mukaan (esim. Paananen, ei pvm; Koodikoulu, 2017–2023): kun nämä työelämän vaatimuksia korostavat näkökulmat saavat jatkuvasti näkyvyyttä, niiden luoma kuva koodista arvovapaana ongelmanratkaisuvälineenä luonnollistuu eikä sitä enää kyseenalaisteta. Koodi2016 oli tarkastelemistani oppimateriaaleista ainoa, joka ei ole opettajien tai muiden kasvatustieteiden ammattilaisten kirjoittama. [20]

Vaikka ohjelmoinnin opetukseen liittyvä tutkimus (esim. Williamson, 2016; Fagerlund, 2021; Mertala, 2021; Dufva & Dufva, 2016) on tuonut esiin koodin yhteiskunnallisia vaikutussuhteita, asenteet muuttuvat hitaasti. Kuvataidekasvattajien kirjoittamat Robo1 (2015) ja Robo2 vuodelta (2017) olivat ilmestyessään ovenavauksia ohjelmoinnin hyödyntämisessä osana luovaa ja kriittisestä itseilmaisua (subjektifikaatio ja postmoderni metafora). Kuitenkin vasta vuonna 2022 ilmestyneissä, moniammatillisen työryhmän kirjoittamissa Polkuja 1–6- ja Polkuja 7–9-oppaissa on koodin yhteiskunnallinen ulottuvuus otettu systemaattisesti huomioon luokittelemalla ohjelmoinnin opetus ohjelmoinnilliseen ajatteluun, tutkivaan työskentelyyn ja tuottamiseen sekä ohjelmoiduissa ympäristöissä toimimiseen. Näin ohjelmointia voidaan lähestyä huomattavasti aiempaa suunnitelmallisemmin ja jäsennellymmin. Toisaalta myös vanhemmista oppimateriaaleista löytyi kriittisiä ääniä. Koodi2016-oppaassa (s. 124–125) kehotetaan esimerkiksi pohtimaan internetin kaltaisten keksintöjen taustoja ja tietokoneiden

[20] Linda Liukas on koodauksen puolestapuhuja ja kansainvälisesti palkittu lasten tietokirjailija (lindaliukas.com) ja Juhani Mykkänen sisällöntuottaja ja muun muassa sovelluspohjaisen Wolt-ruokalähettipalvelun perustaja (juhani.mykkanen.com).

työelämään mukanaan tuomaa muutosta. Oppimateriaaleja ei voida näin ollen luokitella selkeästi omiin kategorioihinsa, mutta niistä on tunnistettavissa erilaisia painotuksia.

7.2 Tulosten merkitys ohjelmoinnin opetukselle

Oppimateriaaleissa havaitsemani kvalifikaation ja sosialisaation ylikorostuminen formaalissa kasvatuksessa on osa kansainvälistä trendiä, joka liittyy Biestan (2020) mukaan rajoittuneeseen käsitykseen demokratiasta. Hän puhuu *aristokraattisesta demokratiakäsityksestä*, joka tarjoaa valmiiksi hyväosaisille kulttuuriset resurssit omien taitojen hiomiseen (Biesta, 2020, s. 93). Kasvatus ei ole kuitenkaan tarkka, vaan hidas prosessi, jolla ei ole takuuvarmaa lopputulosta.[21] Kasvatukseen sisältyy siis aina riski, etteivät oppilaat omaksu asioita suunnitellulla tavalla, sillä täysin ennakoitava kasvatus objektivoi oppilaan ja muuttuu *indoktrinaatioksi*, eli yksilö pakotetaan omaksumaan vallitseva näkökulma (Biesta, 2020, s. 91, 102–103).

Biesta (2020, s. 91–92) huomauttaa, etteivät oppilaat ”vain opi”, vaan he oppivat ”jotakin”, ”jostakin syystä” ja ”joltakulta”. Oppilaat eivät siis vain opettele ohjelmointia, vaan he oppivat sitä ja siitä eri näkökulmista opettajan lähestymistavasta riippuen. Oppiminen ei ole myöskään selkeä ja nopea tapahtuma, vaan oppilaille on annettava aikaa kohdata maailma ja itsensä suhteessa maailmaan (Biesta, 2020, s. 98). Raapaisu-opas (s. 7) oli kuitenkin ainoa tarkastelemani oppimateriaali, jossa mainittiin, että käsitteellisten asioiden hahmottaminen ja algoritmisen ajattelun ymmärtäminen vievät aikaa, joten ohjelmoinnin etenemisessä ei tule kiirehtiä.

Vaikka nykyinen perusopetuksen opetussuunnitelma pyrkii lisäämään oppimisen motivaatiota, elämyksellisyyttä ja onnistumisen kokemuksia (Opetushallitus, 2014, s. 30), subjektifikaatio ei ole jotakin, joka on ratkaistavissa ennakoinnilla ja tehostamisella, vaan se on itsessään yksi koulutuksen päämääristä.

[21] Esimerkiksi eliittikoulujen vaikutus korkeapalkkaiselle alalle päätymiseen on luultua vähäisempi (Tervonen ym., 2018, s. 384).

Jos kasvatuksen kahta osa-aluetta ylikorostetaan, oppilas ainutlaatuisena yksilönä jää vähemmälle huomiolle. Kvalifikaation voimakas painottaminen luo kilpailullisen ilmapiirin (Biesta, 2016, s. 35–36), mikä ei ole toivottavaa, jos oppilailta toivotaan ohjelmointitaitojen luovaa soveltamista. Itse asiassa yksi syy joidenkin opettajien vastahakoisuuteen ohjelmoinnin opettamista kohtaan on ohjelmoinnin opetuksen tuotteistuminen ja markkinoituminen, mikä on vastakohta oppilaiden sosiaalista hyvinvointia painottavalle opettajan työlle (Mertala, 2021, s. 2228–2229).

Voidaankin kysyä, tarjoaako koulun ohjelmoinnin opetus ja siihen liittyvä elektroniikkarakentelu oppilaille valmiita elämyksiä, vai antaako se tilaa oppilaiden omille projekteille ja kokeiluille. Tällöin opettajien on siedettävä kaaosta ja epäjärjestystä, joita syntyy osana omaperäistä työskentelyä. Luovalla työskentelyllä on kuitenkin perinteisissä luokkatiloissa omat haasteensa, sillä tekemiseltä loppuu usein sekä tila että aika (Kupiainen, 2011, s. 106). Koodinlukutaito ei kuitenkaan kerry kokemuksen kautta vaan sitä tulee tietoisesti harjoitella (Dufva & Dufva, 2016, s. 98). Koska kyseessä on merkittävä demokratiaan ja tasa-arvoon liittyvä taito (Knochel ja Patton, 2015, s. 33–34), ohjelmoinnin opetukseen tulisi saada lisää sävyjä ja vaihtoehtoisia lähestymistapoja.

Kun ohjelmoitujen ympäristöjen subjektivoiva ulottuvuus, eli eettiset kysymykset ja ”maailmassa oleminen” (Biesta, 2010, s. 25) otetaan opetukseen mukaan, voidaan pohtia koodiin liittyviä tasa-arvon ja oikeudenmukaisuuden kysymyksiä. Näin ohjelmointi ei näyttäydy oppilaille (ja opettajille) vain arvovapaana ongelmanratkaisutaitona ja elämyksellisenä puuhasteluna, sillä mikään toiminta ei itsessään kehitä kriittistä ajattelua, vaikka innostavat ohjelmointiympäristöt ja käytännön elektroniikkarakentelu edistävätkin oppilaiden ohjelmoinnillista ajattelua.

Puolimatka (1996, s. 273–274) huomauttaakin, että kasvatus kriittisyyteen on aina arvokasvatusta: jos kriittisyys ymmärretään moraalisesti neutraaliksi taidoksi, se tukee epäsuorasti yhteiskunnallisten epäkohtien säilymistä. Itsekriittisyys, jolla tavoitellaan arvioinnin puolueettomuutta ja valmiutta luopua aiemmista käsityksistä, on vaativimpia ajattelun pyrkimyksiä, sillä ilman sitä taitava

ajattelu voi välineellistyä tarjoamaan vain keinoja muiden manipuloimiseen ja omien ennakkoluulojen puolustamiseen (Tomperi, 2017, s. 97).

Mutta mitä asialle voidaan tehdä? Fagerlundilla (2022) on viisi ehdotusta ohjelmoinnin opetuksen kehittämiseksi. Hän kannattaa ensinnäkin *todellisten ilmiöiden* ottamista ohjelmoinnin opetuksen lähtökohdaksi. Tällöin opetus laajentuu laitteista ja ohjelmointiympäristöistä kohti ohjelmoinnin lukutaitoa (Mertala ym., 2020), joka ei edes edellytä ohjelmointiosaamista. Hän peräänkuuluttaa *tutkivan oppimisen materiaaleja*, joissa tarkastellaan esimerkiksi robotiikkaa, verkkopalveluiden ohjausalgoritmeja ja pelien toimintaperiaatteita, ja pohditaan tekoälyn etiikkaa oppilaiden ikä- ja kehitystason mukaisesti. Tällaisia ovat muun muassa tarkastelemani Uudet lukutaidot -kehittämishankkeen Polkuja 7–9 ja Polkuja 1–6 -oppaat, jossa ohjelmointi nähdään funktionaalista näkökulmaa laajemmin. Oppaiden tavoitteena on muun muassa digitaalisen yhdenvertaisuuden vahvistaminen (Polkuja 1–6, s. 45; Polkuja 7–9, s. 42) ja oppilaiden ohjaaminen tarkastelemaan omia mahdollisuuksiaan digitaalisessa yhteiskunnassa. Myös Helsingin yliopiston, Itä-Suomen yliopiston ja Oulun yliopiston vuonna 2023 käynnistämässä Generation AI -hankkeessa[22] on huomioitu tarve subjektifikaationäkökulmalle ohjelmoituja ympäristöjä tarkasteltaessa. Hankkeessa kehitetään muun muassa perusopetuksen oppimateriaaleja tekoälyyn pohjautuvien teknologioiden yleistyttyä nopeasti yhteiskunnan eri osa-alueilla.

Yksi mahdollisuus lähteä laajentamaan funktionaalista, konkreettisiin koodinkirjoitustaitoihin perustuvaa ohjelmointikäsitystä on Mertalan ym. (2020, s. 27–28, 31–32) kehittämä, monilukutaidon pedagogiikkaa hyödyntävä laaja-alaisen ohjelmoinnin pedagogiikka, joka tarjoaa mahdollisuuden koodin yhteiskunnallisten ja yksilöllisen tason vaikutusten tarkastelemiseen. Pienempien lasten kanssa pysytään mikrotasolla, eli omakohtaisten kokemusten analysoinnissa, isompien lasten kanssa voidaan puolestaan keskustella todellisten ilmiöiden digitaalisen mallintamisen haasteista (Mertala, 2021, s. 2235). Tämänkaltainen, erilaisia lukutaitoja hyödyntävä lähestymistapa saattaakin olla monille opettajille

[22] https://www.generation-ai-stn.fi/

luonteva tapa lähestyä ohjelmointia (Fagerlund, 2022), sillä se ei poissulje funktionaalisia ohjelmointitaitoja, joita tarvitaan koodin tuottamiseen ja arvioimiseen (Opetushallitus, 2014, s. 22, 100, 156, 283).

Fagerlundin (2022) mukaan ohjelmointia voitaisiin myös *hyödyntää monipuolisemmin* eri oppiaineissa esimerkiksi tarinallisin keinoin. Tämä edellyttää *ohjelmoinnillisen ajattelun* tarkemman kirjaamisen opetussuunnitelmaan. Perinteinen ajattelu ohjelmoinnista vaiheittaisina käskysarjoina ei riitä kuvaamaan tekoälyn ja koneoppimisen kaltaisia monimutkaisia ilmiöitä, joten *käsitystä ohjelmoinnillisesta ajattelusta* tulisi laajentaa (Fagerlund, 2022).

Vaikka tekoäly ja koneoppiminen pohjautuvat ihmisen käyttäytymistutkimukseen (Lappi ym., 2018, s. 46), tekoälyn ja ihmisen älykkyys ovat osoittautuneet täysin erilaisiksi. Koneoppiminen ei ole edes kovin tehokasta ihmisen aivoihin verrattuna, mutta tiedon saatavuuden ja koneen väsymättömyyden vuoksi tietokoneet ovat pystyneet oppimaan luonnollisia kieliä, joiden ohjelmoiminen lukuisine poikkeuksineen on varsinainen ohjelmoijan painajainen (O´Neill, 2017, s. 76–77). Tekoälyn tutkiminen luonnossa esiintyvien ekologisten verkostojen ja ryhmä-älyn näkökulmasta tarjoaisi mahdollisuuden ymmärtää ja arvostaa älykkyyden eri muotoja planeetallamme (Bridle, 2022, s. 27–29, 31, 57). Mielenkiintoinen esimerkki on japanilaisten tutkijoiden (Tero ym., 2010, s. 439–342) projekti, jossa he hyödynsivät limasientä (*Physarum polycephalum*) metroreittien suunnittelussa. Koska vaihtoehtoisten reittien laskeminen on haastavaa niin ihmisille kuin algoritmeillekin, tutkijat asettivat petrimaljaan limasienten ravintoa eli kauraryynejä Tokion asutuskeskusten mukaisesti. Sienirihmasto loi tehokkaimman verkoston vain muutamassa tunnissa (Bridle, 2022, s. 191–194).

7.3 Tutkimuksen luotettavuuden arviointi

Laadullisen tutkimuksen luotettavuuden arviointiin ei ole yksiselitteisiä ohjeita (Eskola & Suoranta 2000, 211), eivätkä määrällisen tutkimuksen piirissä käytetyt *reliabiliteetti* ja *validiteetti* välttämättä sovellu sellaisenaan laadullisen tutkimuksen luotettavuuden mittareiksi (Mäkelä, 1990, s. 47). Reliabiliteetti tarkoittaa tu-

losten pysyvyyttä eli tutkimusmenetelmien toistettavuutta ja validiteetti tutkimuksen pätevyyttä, eli sitä, miten hyvin käytetyt tutkimusmenetelmät soveltuvat tutkittavan ilmiön tarkasteluun (Kananen, 2014, s. 147). Validiteetti voidaan jakaa edelleen ulkoiseen ja sisäiseen validiteettiin: tutkimuksen sisäinen validiteetti liittyy tulkintojen syy-seuraussuhteisiin ja ulkoinen validiteetti tulosten yleistettävyyteen (Kananen, 2014, s. 146–148). Laadullisessa tutkimuksessa ei kuitenkaan pyritä niinkään yleistyksiin vaan ilmiön ymmärtämiseen (Kananen, 2014, s. 353).

Laadullisen tutkimuksen luotettavuustarkastelu sivuaakin Kanasen (2014) mukaan objektiivisutta, sillä tutkija päättää, mitä tutkitaan ja miten kerätty aineisto analysoidaan ja tulkitaan. Laadullisen tutkimuksen luotettavuutta voidaan kuitenkin arvioida muun muassa tulkinnan vahvistettavuuden, tutkimuksen arvioitavuuden (riittävän dokumentaation), tulkinnan ristiriidattomuuden sekä kyllääntymisen avulla (Kananen, 2014, s. 151). Tarkastelen näiden kriteerien avulla oman tutkimukseni luotettavuutta.

Tulkinnan vahvistettavuus. Tutkielman luotettavuutta vahvistavana tekijänä voidaan pitää *teoriatriangulaatiota*, jolla tarkoitetaan useamman teorian hyödyntämistä aineiston analysoinnissa (Carter, 2014, s. 545). Triangulaatio ei ole kuitenkaan päämäärä itsessään, vaan tutkijan tulee punnita, palveleeko moninäkökulmainen lähestymistapa tutkimuksen tarkoitusta ja onko sen käyttö perusteltua (Tuomi & Sarajärvi 2018, 166, 168). Biestan (2010) kolmijako ja Dufvan ja Dufvan (2016) koodimetaforat mahdollistivat aineiston monipuolisen tarkastelemisen: Biestan teoria olisi ollut liian yleisluontoinen ja koodimetaforat liian yksityiskohtaisia tutkimukseni ainoiksi taustateoriaksi, mutta yhdessä ne muodostivat tiiviin ja toisiaan täydentävän teoreettisen verkoston ohjelmoinnin oppimateriaalien tarkasteluun.

Tutkimuksen arvioitavuus. Kananen (2014, s. 153) pitää riittävää dokumentaatiota yhtenä tärkeimmistä asioista, joka lisää työn luotettavuutta, jotta lukija voi seurata tutkijan päättelyä. Eskola ja Suoranta (2000, s. 16) myös huomauttavat, etteivät tutkimustulokset ole ajattomia ja paikattomia, vaan aikaan ja paikkaan sidonnaisia. Pyrin tämän vuoksi avaamaan käyttämäni tiedonkeruu- ja analysointimenetelmät sekä erittelemään käyttämäni tulkintasäännöt vaiheisiin,

jotta tulkinnan kulku olisi selvää niin lukijalle kuin tutkijalle itselleen (Mäkelä, 1990, s. 59; Tuomi & Sarajärvi 2009, 141–142).

Tulkinnan ristiriidattomuus eli sisäinen validiteetti. Pyrin lisäämään tulkintojeni sisäistä validiteettia käyttämällä aiempaa tutkimusta omien tulkintojeni tukena. Kanasen (2014, s. 153) mukaan tutkimuksen sisäistä validiteettia voi kuitenkin heikentää esimerkiksi tutkijan tekemät väärät johtopäätökset. Työskentelyn systemaattisuus ja analyysivaiheiden auki kirjoittaminen kuitenkin parantavat tutkimuksen arvioitavuutta ja avaavat lukijalle tutkijan ajatuksenjuoksua (Mäkelä, 1990, s. 57). Vaikka en voi täysin luottaa tehneeni joka teeman kohdalla oikeita tulkintoja, luotan johdonmukaisen dokumentointini vähentävän tulosten epäuskottavuutta (Tutkimuseettinen Neuvottelukunta, 2023, s. 13).

Aineiston kyllääntyminen vaikutti siihen, kuinka monta oppimateriaalia valikoitui mukaan tutkimukseen. Kun aineisto on riittävän laaja, voidaan puhua kyllääntymisestä, eli uudet tapaukset eivät tuota tutkimuksen kannalta uutta tietoa (Kananen, 2014, s. 95). Koin, ettei näkökulmani olisi muuttunut merkittävästi lisämateriaalia tarkastelemalla (Eskola & Suoranta 2000, s. 62–63) eli valitsemani oppimateriaalit auttoivat muodostamaan kokonaiskuvan ohjelmoinnin opetuksesta perusopetuksessa.

Tutkimukseen käytetty aika on yksi laadullisen tutkimuksen luotettavuuden mittareista (Tuomi & Sarajärvi, 2009, s. 139). Perusvaatimuksena on, että tutkijalla on ollut riittävästi aikaa tutkimuksen tekemiseen (Tuomi & Sarajärvi 2009, 141–142). Tutkimukseni tekemiseen meni reilu vuosi, joten aihe sai kehittyä riittävän pitkään. Aloitin aiheen ideoinnin keväällä 2022, mutta kokopäiväinen työ viivästytti kirjoitustyön aloittamista syksyyn 2022. Rauhallinen eteneminen mahdollisti analyysin ja tulkintojen kypsyttelemisen, mutta vaikeutti hetkittäin kokonaiskuvan hahmottamista.

Noudatin kaiken kaikkiaan hyvää tieteellistä käytäntöä (Tutkimuseettinen neuvottelukunta, 2023, s. 11–12), eli *luotettavuutta* tutkimuksen suunnittelussa, toteutuksessa ja analysoinnissa, *rehellisyyttä* (puolueettomuutta ja avoimuutta) yksityiskohtia salaamatta, tutkijayhteisön työpanoksen *arvostamista* asianmukaisilla viittauksilla ja yleistä laadullista *vastuunkantoa* koko tutkimusprosessin ajan.

Tarkastelin esimerkiksi kaikkia oppimateriaaleja tasavertaisesti useasta eri näkökulmasta omat ennakkoluuloni tiedostaen.

7.4 Jatkotutkimusmahdollisuudet

Vaikka tutkimuskysymykseni hahmottuivat teoreettista viitekehystä kirjoitettaessa, aineiston analysointi ei edennyt täysin ongelmattomasti vaiheesta toiseen. Minun oli jatkuvasti palattava koodattuihin materiaaleihini, jotta en olisi ajautunut sivupolulle, ja tulin lukeneeksi paljon ylimääräistä, aiheen reunamille sijoittuvaa kirjallisuutta, jota oli paikoin haastavaa ja haikeaakin rajata tutkimuksen ulkopuolelle. Tutkielmaa kirjoittaessani aloin esimerkiksi pohtia abstraktin koodin ja (uus)materiaalisen maailman välisiä kytköksiä (ks. Lehtonen, 2008, s. 22), vaikka laitteisiin liittyvä keskustelu rajautuikin heti alkuvaiheessa tarkasteluni ulkopuolelle.

Ohjelmoitujen ympäristöjen tutkiminen sosiomateriaalisesta ja kestävän kehityksen näkökulmasta olisi mielenkiintoinen jatkotutkimuksen kohde, samoin ohjelmoitujen ympäristöjen peilaaminen postkolonialistisiin valtasuhteisiin. Vaikka koodi on abstrakti ilmiö, digitaalisuus rakentuu hyvin aineelliselle pohjalle: datakeskukset kuluttavat valtavia määriä energiaa (Dayarathna ym., 2016, s. 732) ja meren pohjassa kulkevat internetkaapelit uusintavat kolonialista perintöä (Gray, 2016). Näiden asioiden esiin nostaminen loisi mahdollisuuden toisin ajattelulle ja uudenlaisten lähestymistapojen löytämiselle.

Tutkimustulokseni antoivat tietoa siitä, missä määrin Biestan (2010) koulutuksen tehtävät ja Dufvan ja Dufvan (2016) koodimetaforat aineistossa esiintyvät. Tulosten perusteella ei voida kuitenkaan sanoa, johtuuko kvalifikaation ja sosialisaation painottuminen oppimateriaaleissa siitä, että funktionaalinen koodinäkökulma olisi yhteiskunnassa vallitseva, tai onko funktionaalisen näkökulman painottaminen vain kätevä keino koodin muuntamisessa oppimistuotteiksi. Tutkimus ei myöskään antanut tietoa siitä, missä määrin ja miten eri oppimateriaaleja luokkahuoneissa hyödynnetään tai millä perusteella opettajat ohjelmoinnin oppimateriaaleja ylipäätään valitsevat. Materiaalien valintaperusteet ja niiden käyttäminen ovatkin selkeä ja tarpeellinen jatkotutkimuksen aihe.

Olen tarkastellut aineistoa tietyssä ajallisessa ja kulttuurisessa kontekstissa, joten teemat olisivat kenties erilaisia, jos tutkimus toistettaisiin viiden vuoden päästä silloisia ohjelmoinnin oppimateriaaleja analysoiden. Tulosten perusteella on havaittavissa, että oppimateriaalien käsitykset ohjelmoinnista ovat muuttumassa alan tutkimuksen (esim. Fagerlund, 2021; Mertala, 2021; Dufva & Dufva, 2016) myötä neutraalista ongelmanratkaisutyökalusta yhteiskunnalliseksi ilmiöksi, jolla on sekä kielteisiä että myönteisiä vaikutuksia ihmisten toimintaan. Jatkotutkimuksen avulla olisi mahdollista tarkastella, heijastuuko tämä kehityssuunta myös oppimateriaaleihin.

Lopuksi on aiheellista muistuttaa, että Biestan (2010) kasvatuksen tehtävät ja Dufvan ja Dufvan (2016) koodimetaforat ovat vain yksi (joskin perusteltu) mahdollinen teoreettinen viitekehys ohjelmoinnin opetuksen oppimateriaalien tutkimiseen. Myöhempi tutkimus voi laajentaa näkökulmia analysoimalla oppimateriaaleja muiden käsitteiden ja viitekehysten läpi. Esimeriksi Uudet lukutaidot -kehittämisohjelman (Opetushallitus & Kansallinen audiovisuaalinen instituutti, 2021) määrittämien ohjelmointiosaamisen osa-alueiden (ohjelmoinnillinen ajattelu, tutkiva työskentely ja tuottaminen sekä ohjelmoidut ympäristöt ja niissä toimiminen) yksityiskohtaisempi tarkastelu suhteessa ohjelmointiin liittyviin sosiaalisiin, poliittisiin[23] ja eettisiin kysymyksiin olisikin mielenkiintoinen jatkotutkimuksen aihe.

[23] Ohjelmistosuunnittelija ja taiteilija Ramsay Nasser on esimerkiksi kyseenalaistanut englannin kielen valta-asemaa suunnittelemalla täysin arabiankielisen ohjelmointiympäristön (nas.sr).

8 LOPUKSI

Ohjelmoinnin sisällyttäminen perusopetuksen opetussuunnitelman perusteisiin, ja etenkin osaksi arvostettua matematiikan oppiainetta välittää viestin, että myös ohjelmointi on tärkeää. Ohjelmointia ei tule kuitenkaan nähdä yksinomaan matemaattisena taitona, sillä se edesauttaa koodiutunutta ajattelua (Kitchin, 2014; Vadén, 2002). Pelkissä funktionaalisissa taidoissa pitäytyminen myös ylläpitää vallitsevia olosuhteita.[24]

Digitaalisuuden kaikkiallisuus tekee ohjelmoitujen ympäristöjen tarkastelusta kuitenkin tärkeän oppisisällön (Robo2, s. 11–12), sillä tekoälyn yleistyessä olemme digitaalisuuden ympäröimiä yhä useammalla elämänalueella (Fagerlund, 2022). Joudumme esimerkiksi jatkuvasti todistamaan olevamme ihmisiä emmekä botteja (tietokoneohjelmia), vaikka internet on ihmisten luomus. Open AI -säätiö julkaisi vuonna 2022 ChatGPT -tekoälymallin,[25] jolla on kyky ymmärtää ihmisen kirjoittamaa vapaamuotoista tekstiä. Sovellus ei kuitenkaan ymmärrä asioiden merkityksiä, vaan pyrkii muodostamaan vastauksen, joka kuulostaa oikealta. Funktionaaliset ohjelmointitaidot ja kone-metaforaan liittyvät koodikäsitykset eivät näin ollen riitä ChatGPT:n kaltaisten ilmiöiden ymmärtämiseen tai niistä keskustelemiseen, vaan siihen tarvitaan erilaisia ja keskenään ristiriitaisiakin näkökulmia.

Biesta (2017, s. 39) peräänkuuluttaakin kriittisen ajattelun tärkeyttä pelkkien funktionaalisten taitojen sijaan, jotta lapset ja nuoret oppivat puolustamaan omia rajojaan ja arvioimaan, mitä vaikutteita he lähtevät seuraamaan ja mitä eivät. Kun oppilaille tarjotaan erilaisia ajattelun välineitä heitä ympäröivien digitaalisen todellisuuden tarkastelemiseen, heille tarjoutuu yhtäläiset mahdollisuudet yhteiskunnalliseen osallisuuteen ja omien unelmien tavoitteluun.

[24] Esimerkiksi Rogers (1995, s. 295) on esittänyt, että internetin kaltaisten teknologisten innovaatioiden hyöty johtaa sosioekonomisten erojen kasvuun, sillä hyväosaiset pystyvät hyödyntämään resursseja oman sosiaalisen, taloudellisen ja kulttuurisen pääomansa kasvattamiseen.

[25] https://openai.com/

LÄHTEET

Albion, P. R. (2016). If this is the second coming of coding will there be rapture or rejection? Teoksessa S. Prestridge & A. P. Albion (toim.), *Proceedings of the Australian Council for Computers in Education conference* (s. 1–8). Australian Council for Computers in Education. https://research.usq.edu.au/download/694f76e4c763c4b1720a89a06f76b88f4b3adde1bc13d7cb777fad7a4803fa1e/198819/code03.pdf

Antikainen, A., Rinne, R. & Koski, L. (2021). *Kasvatussosiologia*. (6. uudistettu painos). PS-kustannus.

Anundi, M., Kuitunen, P-P., Olin, H., Järvenpää, T. & Hurskainen, T. (2018). *Ohjelmoinnin, robotiikan, tekoälyn ja esineiden internetin opetussuunnitelman toteuttamisen tukimateriaali Vantaa 2018*. Haettu 4.8.2022 osoitteesta https://sites.google.com/eduvantaa.fi/ohjelmointi-ops-vantaa-2018/etusivu

Balanskat, A. & Englehart, K. (2014). *Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe.* Brussels: European Schoolnet. https://www.researchgate.net/publication/278671873_Computing_our_Future

Berry, M. (2013). *Computing in the national curriculum: A guide for primary teachers.* Nottingham: Naace. https://www.grayrigg.cumbria.sch.uk/assets/files/general/CASPrimaryComputing.pdf

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *The Nordic approach to introducing computational thinking and programming in compulsory education*. Brussels: European Schoolnet. https://doi.org/10.17471/54007

Bridle., J. (2018). *New Dark Age. Technology and the End of the Future.* Verso.

Bridle., J. (2022). *Ways of Being.* Allen Lane.

Biesta, G. (2010). *Good Education in an Age of Measurement. Ethics, Politics, Democracy.* Routledge.

Biesta, G. (2016). ICT and Education Beyond Learning. A framework analysis, Development and Critique. Teoksessa E. Elstad (toim.), *Digital Expectations and Experiences in Education* (s. 29–43). Sense Publishers. https://link.springer.com/chapter/10.1007/978-94-6300-648-4_2

Biesta, G. (2017). *The Rediscovery of Teaching*. Routledge.

Biesta, G. (2020). Risking Ourselves in Education: Qualification, Socialization, and Subjectification Revisited. *Educational Theory, 70*(1), 89–104. https://doi.org/10.1111/edth.12411

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., & Rumble, M. (2012). Defining Twenty-First Century Skills. Teoksessa P., Griffin, B. McGaw & E. Care (toim.), *Assessment and Teaching of 21st Century Skills* (s. 17–66). Springer. http://dx.doi.org/10.1007/978-94-007-2324-5_2

Brainin, E., Shamir, A. & Eden, S. (2022). Promoting Spatial Language and Ability Among SLD Children: Can Robot Programming Make A Difference? *Journal of Educational Computing Research*, *60*(7), 1742–1762. https://doi.org/10.1177/07356331221083224

Brunila, K. (2011). The Projectisation, Marketisation and Therapisation of Education. *European Educational Research Journal, 10*(3), 421–432. https://doi.org/10.2304/eerj.2011.10.3.421

Carr, N. (2010). Pinnalliset. Mitä internet tekee aivoillemme. Terra Cognita.

Carter, N., Bryant-Lukosius, D., DiCesco, A., Blythe, J. & Neville, A. J. (2014). *The use of The Use of Triangulation in Qualitative Research.* Oncology Nursing Forum, 41(5), 545–547. doi: 10.1188/14.ONF.545-547

Dayarathna, M., Yonggang, W., Rui, F. (2016). Data Center Energy Consumption Modeling – A Survey. *IEEE Communications surveys and tutorials*, 18(1), 732–794. DOI:10.1109/COMST.2015.2481183

Denning, P. (2017). Remaining trouble spots with computational thinking. *Communications of the Association for Computing Machinery, 60*(6), 33–39. https://doi.org/10.1145/2998438

Drewett, A., & O'Reilly, M. (2021). Examining the value of using naturally occurring data to facilitate qualitative health research with 'seldom heard'

'vulnerable' groups: A research note on inpatient care. *Qualitative Research*, *0*(0). https://doi.org/10.1177/14687941211039971

Dufva, T. (2015). Digitaalisen maailman ymmärtämisestä: Miten elektroniikan avulla voidaan kasvattaa digitaaliseen yhteiskuntaan? Teoksessa R. Ruuskanen, H. Vähänissi & I. Tujula (toim.), *Robo1 – Elektroniikkaa ja askartelua* (s. 2–3). Mediataide kartalle! -hanke (2013–2015). Live Herring ry & Käsityökoulu Robotti ry. https://www.kasityokoulurobotti.fi/wp-content/uploads/2014/06/Robo1_elektroniikkaa_ja_askartelua.pdf

Dufva, T. & Dufva, M. (2016). Metaphors of code – structuring and broadening the discussion on teaching kids to code. *Thinking Skills and Creativity, 22*, 97–110. https://doi.org/10.1016/j.tsc.2016.09.004

Dufva, T. (2017a). Kasvamassa digitaalisessa maailmassa. Teoksessa I. Tujula & R. Ruuskanen (toim.), *Robo2 – Elektroniikan ja ohjelmoinnin alkeita varhaiskasvatukseen ja alkuopetukseen* (s. 8–13). Käsityökoulu Robotti ry. https://www.kasityokoulurobotti.fi/wp-content/uploads/2017/02/robo2_nettiversio_2017_01_28.pdf

Dufva, T. (2017b). Maker Movement: Creating knowledge through basic intention. *Techne Series – Research in Sloyd Education and Craft Science*, *24*(2), 129–141. https://www.researchgate.net/publication/322696323_Maker_Movement_creating_knowledge_through_basic_intention

Dufva, T. & Dufva, M. (2019). Grasping the future of the digital society. *Futures, 107*, 17–28. https://doi.org/10.1016/j.futures.2018.11.001

Dufva, S. T. (2021). Creative Coding as Compost(ing). In K. Tavin, G. Kolb & J. Tervo (toim.), *Post-Digital, Post-Internet Art and Education* (s. 269-283). Palgrave Macmillan. https://doi.org/10.1007/978-3-030-73770-2_16

Eskola, J. & Suoranta, J. (2000). *Johdatus laadulliseen tutkimukseen*. (4. painos). Vastapaino.

Fagerlund, J. (2021). *Teaching, Learning and Assessing Computational Thinking through Programming with Scratch in Primary Schools* [väitöskirja, Jyväskylän yliopisto].

https://jyx.jyu.fi/bitstream/handle/123456789/78190/978-951-39-8882-1_vaitos05112021.pdf?sequence=1&isAllowed=y

Fagerlund, J. (10.3.2022). Tuoreita näkökulmia ohjelmoinnin ja ohjelmoinnillisen ajattelun opetukseen koulussa. *Dimensio*. https://dimensiolehti.fi/tuoreita-nakokulmia-ohjelmoinnin-ja-ohjelmoinnillisen-ajattelun-opetukseen-koulussa/

Flick, U. (2002). *An Introduction to Qualitative Research*. Sage Publications.

Giordano, D., Maiorana, F., Csizmadia, A., Marsden, S., Riedesel, C., Mishra, S. & Vinikiene, L. (2015). New horizons in the assessment of computer science at school and beyond: leveraging on the ViVA platform. *Proceedings of the 2015 ITiCSE on Working Group Reports* (s. 117–147). Association for Computing Machinery. https://doi.org/10.1145/2858796.2858801

Gray, A. (24.11.2016). *This map shows how undersea cables move internet traffic around the world.* World Economic Forum. https://www.weforum.org/agenda/2016/11/this-map-shows-how-undersea-cables-move-internet-traffic-around-the-world/

Hatch, M. (2014). The maker movement manifesto: rules for Innovation in the new world of crafters, hackers and tinkerers. McGraw-Hill Education.

Heikkilä, V.-M. (18.11.2014). Ohjelmoinnin opintiet. *Skrolli, 2*, 12–18. https://skrolli.fi/2014/09/ohjelmoinnin-opintiet/

Heiskala, R. (1990). Tulkinnan koeteltavuus ja aikakauslehtien analyysi. Teoksessa K. Mäkelä (toim.) *Kvalitatiivisen aineiston analyysi ja tulkinta* (s. 242–262). Gaudeamus.

Hirsjärvi, S., Remes, P. & Sajavaara, P. (2007). *Tutki ja kirjoita.* (13. uudistettu painos). Tammi.

Huttunen, T. (2022). Uudet lukutaidot -kehittämisohjelma. *Opetus- ja kulttuuriministeriö*. Haettu 19.11.2022. https://okm.fi/uudet-lukutaidot

Ideland, M. (2021). Google and the end of the teacher? How a figuration of the teacher is produced through an ed-tech discourse. *Learning, Media and Technology, 46*(1), 33–46.

https://www.tandfonline.com/doi/pdf/10.1080/17439884.2020.1809452?needAccess=true

Ideland, M., Jobér, A. & Axelsson, T. (2021). Problem solved! How eduprenuers enact a school crisis as business possibilities. *European Educational Research Journal*, *20*(1), 83–101. https://doi.org/10.1177/1474904120952978

Jokisaari, O.-J. (2017). *Kasvatus tuotemaailmassa – Günther Anders kasvatusfilosofina.* [väitöskirja, Tampereen yliopisto]. https://trepo.tuni.fi/bitstream/handle/10024/100826/978-952-03-0403-4.pdf?sequence=1&isAllowed=y

Järvenpää, J. (2018). *Raapaisu. Ohjelmoinnin opas.* Edukustannus.

Kaarakainen, M.-T. (2019). *Education and inequality in digital opportunities – Differences in digital engagement among Finnish lower and upper secondary school students.* [väitöskirja, Turun yliopisto]. https://urn.fi/URN:ISBN:978-951-29-7819-9

Kafai, Y., & Burke, Q. (2013). The social turn in K-12 programming: moving from computational thinking to computational participation. *Teoksessa SIGCSE'13 – Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (s. 603–608). Association for Computing Machinery. https://doi.org/10.1145/2445196.2445373

Kazmertsuk, A. (2019). Koulujen digitalisaatio digifanin silmin. *Natura*, *2*, 8–9.

Kiilakoski, T. (2012). *Kasvatus teknologisessa maailmassa. Tutkimus teknologisoituvasta kasvatuksesta* [väitöskirja, Oulun yliopisto].

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, *41*(2), 75–86. https://doi.org/10.1207/s15326985ep4102_1

Kitchin, R. & Dodge, M. (2011). *Code/Space: Software and Everyday Life.* Massachusetts: The MIT Press. https://www.researchgate.net/publication/32884850_Code_space_and_everyday_life

Kitchin, R. (2014). The real-time city? Big Data and Smart Urbanism. *GeoJournal, 79*, 1–14. https://sharingcitiesalliance.knowledgeowl.com/help/the-real-time-city-big-data-and-smart-urbanism

Knochel, A. ja Patton, R. (2015). If Art Education Then Critical Digital Making: Computational Thinking and Creative Code. *Studies in Art Education*, *57*(1), 21–38. https://www.academia.edu/22357407/If_Art_Education_Then_Critical_Digital_Making_Computational_Thinking_and_Creative_Code

Koivisto, J., Toikkanen, T. ja Palsa, L. (2022a). Polkuja ohjelmointiosaamiseen. Opas vuosiluokille 1–6. Uudet lukutaidot -kehittämisohjelma (2020–2023). https://www.mediataitokoulu.fi/ohjelmointi_alakoulu.pdf

Koivisto, J., Toikkanen, T. ja Palsa, L. (2022b). *Polkuja ohjelmointiosaamiseen. Opas vuosiluokille 7–9.* Uudet lukutaidot -kehittämisohjelma (2020–2023). https://www.mediataitokoulu.fi/ohjelmointi_ylakoulu.pdf

Konle-Seidl, R. & Danesi, S. (2022). *Digitalisation and changes in the world of work.* Policy Department for Economic, Scientific and Quality of Life Policies Directorate-General for Internal Policies. European Parliament. https://www.europarl.europa.eu/RegData/etudes/STUD/2022/733986/IPOL_STU(2022)733986_EN.pdf

Koodikoulu (2017–2023): Miksi koodaus? *Suomen koodikoulu*. Haettu 1.12.2022 osoitteesta https://www.suomenkoodikoulu.fi/miksi-koodaus-2/

Kupiainen, R. (2011). Visuaalinen maailma, koulu ja oppiminen. Teoksessa K. Pohjola (toim.), *Uusi koulu. Oppiminen mediakulttuurin aikakaudella.* Koulutuksen tutkimuslaitos (s. 99–108). Jyväskylän yliopisto

Kupiainen, R., Kulju, P. & Mäkinen, M. (2015). Mikä monilukutaito? Teoksessa T. Kaartinen (toim.), *Monilukutaito kaikki kaikessa*. (Artikkelit 6115, s. 13–24). Tampereen yliopiston normaalikoulu. https://urn.fi/URN:NBN:fi:uta-201509292320

Kupiainen, R. (2019). Monilukutaidon pedagogiikan lähtökohtia. Teoksessa M. Harmanen & M. Hartikainen (toim.), *Monilukutaitoa oppimassa*. (Oppaat ja käsikirjat 2019: 2, s. 23–32). Opetushallitus.

Kärki, I. (2015). *Uusliberialismia vai ei? Suomalaista perusopetusta koskevien koulutuspoliittisten kasvatus- ja opetustavoitteiden arvot vuosina 1994-2012 Shalom Schwartzin arvoteorian valossa.* [väitöskirja, Helsingin yliopisto]. http://urn.fi/URN:ISBN:978-951-51-1130-2

Labusch, A., Eickelmann, B. & Vennemann, M. (2019). Computational Thinking Processes and Their Congruence with Problem-Solving and Information Processing. Teoksessa S.-C., Kong & H. Abelson (toim.), *Computational Thinking Education* (s. 65–78). Springer. https://www.researchgate.net/publication/334112423_Computational_Thinking_Processes_and_Their_Congruence_with_Problem-Solving_and_Information_Processing

Lanier, J. (2010). *Et ole koje – Manifesti.* Terra Cognita.

Lappi, O., Rusanen, A.-M. & Pekkanen, J. (2018). Tekoäly ja ihmiskognitio. *Tieteessä tapahtuu* (1), 42–46.

Leino, K., Rikala, J., Puhakka, E., Niilo-Rämä, M., Siren, M., & Fagerlund, J. (2019). *Digiloikasta digitaitoihin – kansainvälinen monilukutaidon ja ohjelmoinnillisen ajattelun tutkimus (ICILS 2018)*. Koulutuksen tutkimuslaitos. https://jyx.jyu.fi/bitstream/handle/123456789/66250/978-951-39-7937-9.pdf?sequence=1&isAllowed=y

Lessig, L. (1999). *Code and Other Laws of Cyberspace.* Basic Books.

Maloney, J., Resnick, M., Rusk, N., Silverman, B. & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education*, *10*(4):16, 1–15. https://web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf

Manches, A., & Plowman, L. (2017). Computing education in children's early years: A call for debate. *British Journal of Educational Technology*, *48*(1), 191–201. https://doi.org/10.1111/bjet.12355

Mertala, P. (14.12.2017). Näkökulmia monilukutaitoon – opettajuus ja situationaaliset lukutaidot. *Kieli, koulutus ja yhteiskunta, 8*(6). https://www.kieliverkosto.fi/fi/journals/kieli-koulutus-ja-yhteiskunta-

joulukuu-2017/nakokulmia-monilukutaitoon-opettajuus-ja-situationaaliset-lukutaidot

Mertala, P. (2019). (Vasta)kertomuksia koulutuksen digitalisaatiosta. *Kasvatus & Aika, 13*(3), 26–45. https://doi.org/10.33350/ka.76593

Mertala, P. (11.11.2020). Luokkahuoneesta datatehtaaksi. *Alusta! Tampereen yliopiston yhteiskuntatieteiden tiedekunnan verkkojulkaisu*. https://www.tuni.fi/alustalehti/2020/11/11/luokkahuoneesta-datatehtaaksi/

Mertala, P., Palsa, L. & Dufva, T. S. (2020). Monilukutaito koodin purkajana. *Media & viestintä*, *43*(1), 21–46. https://journal.fi/mediaviestinta/article/download/91079/50116

Mertala, P. (2021). The pedagogy of multiliteracies as a code breaker: A suggestion for a transversal approach to computing education in basic education. *British Journal of Educational Technology, 52*(6), 2227–2241. https://doi.org/10.1111/bjet.13125

Mertala, P. (2022a). Digitarinoita – Mitä on koulutusteknologiapuhe ja miksi siihen tulee suhtautua epäillen. *Kieli, koulutus ja yhteiskunta*, *13*(3), 1–9. https://jyx.jyu.fi/bitstream/handle/123456789/80898/Digitarinoita_%2520mit%25C3%25A4%2520on%2520koulutusteknologiapuhe.pdf?sequence=1&isAllowed=y

Mertala, P. (2022b). Koneluettava lapsi? Opetuksen datafikaatio ja automaatio Minä–Se-suhteiden tuotantokoneistona. *Kasvatus & Aika*, *16*(4), 26–42. https://doi.org/10.33350/ka.117096

Mertala, P. (2022c). Onko etäkoulu mahdollinen? *Kasvatus & Aika, 16* (4), 143–157. https://doi.org/10.33350/ka.119480

Meyer, J. (2018). *Programming 101 – The How and Why of Programming Revealed Using the Processing Programming Language.* Apress. https://link.springer.com/content/pdf/bfm:978-1-4842-3697-0/1

Mykkänen, J. & Liukas, L. (2014). *Koodi 2016 – Ensiapua ohjelmoinnin opettamiseen peruskoulussa*. https://s3-eu-west-1.amazonaws.com/koodi2016/Koodi2016_LR.pdf

Mäkelä, K. (1990). Kvalitatiivisen analyysin arviointiperusteet. Teoksessa K. Mäkelä (toim.), *Kvalitatiivisen aineiston analyysi ja tulkinta* (s. 42–59). Gaudeamus.

O´Neill, K. (2017). *Matikkatuhoaseet, Miten suuraineisto lisää eriarvoisuutta ja uhkaa demokratiaa.* Terra Cognita.

Opetushallitus (2014). *Perusopetuksen opetussuunnitelman perusteet*. Määräykset ja ohjeet 96. Opetushallitus. https://www.oph.fi/sites/default/files/documents/perusopetuksen_opetussuunnitelman_perusteet_2014.pdf

Opetushallitus & Kansallinen audiovisuaalinen instituutti (2021). Ohjelmointiosaaminen. *Opetus- ja kulttuuriministeriön Uudet lukutaidot -kehittämishanke (2020–2022)*. Haettu 18.10.2022 osoitteesta https://eperusteet.opintopolku.fi/#/fi/digiosaaminen/8706410/osaamiskokonaisuus/8709075

Paananen, J. (ei pvm.). Opettajille. *Koodikirja*. https://www.koodikirja.fi/opettajille/

Patton, M. Q. (2002). *Qualitative research and evaluation methods* (3. painos). Sage.

Palinkas, L. A., Horwitz, S. M., Green, C. A., Wisdom, J. P., Duan, N. & Hoagwood, K. (2015). Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. *Administration and Policy in Mental Health and Mental Health Services Research, 42* (5), 533–544. DOI:10.1007/s10488-013-0528-y

Palsa, L. & Ruokamo, H. (2015). Behind the concepts of multiliteracies and media literacy in the renewed Finnish core curriculum: A systematic literature review of peer-reviewed research. *Seminar.Net*, *11*(2), 101–119. https://doi.org/10.7577/seminar.2354

Palsa, L., Pekkala, L., & Salomaa, S. (2019). Monilukutaito ja mediakasvatuksen pedagogiikka. Teoksessa M. Harmanen & M. Hartikainen (toim.), *Monilukutaitoa oppimassa*. (Oppaat ja käsikirjat 2019: 2, s. 41–50). Opetushallitus.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books. http://dl.acm.org/citation.cfm?id=1095592

Pariser, E. (2012). *The Filter Bubble*. Penguin.

Parviainen, M., Tamminen, E., Aho, A. & Järvinen, J.-P. (2016). *ViLLE opintopolku – Alakoulun ohjelmointiopas*. Turun yliopisto. https://ville.cs.utu.fi/opintopolku/ville_alakoulun_ohjelmointiopas.pdf

Potter, J. (2002). Two kinds of natural. *Discourse Studies*, *4*(4): 539–542. https://doi.org/10.1177/1461445602004004090

Puolimatka, T. (1996). *Kasvatus ja filosofia.* Kirjayhtymä.

Rantala, J. (2018). Koodin ja ohjelmiston filosofisista. *niin & näin, 1*, 101–108. https://www.researchgate.net/publication/337670319_Koodin_ja_ohjelmiston_filosofiasta

Resnick, M., Flanagan, M., Kelleher, C., MacLaurin, M., Ohshima, Y., Perlin, K. & Torres, R. (2009). Growing up programming: Democratizing the creation of dynamic, interactive media. *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems* (s. 3293–3296). Association for Computer Machinery. https://typeset.io/papers/growing-up-programming-democratizing-the-creation-of-dynamic-2biuoarnm3

Rinne, E. (2019). *(Moni)kulttuurinen maailmankuva ja kuulumisen politiikka suomalaisissa peruskoulun oppikirjoissa ja nuorten kokemuksissa*. [väitöskirja, Tampereen yliopisto]. https://trepo.tuni.fi/bitstream/handle/10024/116296/978-952-03-1182-7.pdf?sequence=2&isAllowed=y

Rogers, E. M. (1995). *Diffusion of Innovations.* Free Press.

Román-González, M., Pérez-González, J.-C. & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Ruuska, H. (2014). Opettajan ei tarvitse tehdä työvälineitään. Teoksessa H. Ruuska, M. Löytönen & A. Rutanen (toim.) *Laatua! Oppimateriaalit muuttuvassa tietoympäristössä* (s. 41–46). Suomen tietokirjailijat ry.

Ruuskanen, R., Vähänissi, H. & Tujula, I. (2015). *Robo1-opas*. Mediataide kartalle! -hanke (2013–2015). Live Herring ry & Käsityökoulu Robotti ry.

https://www.kasityokoulurobotti.fi/wp-content/uploads/2014/06/Robo1_elektroniikkaa_ja_askartelua.pdf

Simons, M., Lundahl, L. & Serpieri, R. (2013). The Governing of Educational in Europe: commercial actors, partnerships and strategies. *European Educational Research Journal, 12*(4), 416–424.

Soronen, A. & Koivunen, A. (2022). Platformed intimacies: Professional belonging on social media. *European Journal of Cultural Studies, 25*(5), 1344–1360. https://doi.org/10.1177/13675494221079854

Säntti, J. (2020). Joukkoviestinnästä digiaikaan – Tieto- ja viestintätekniikka suomalaisen perusopetuksen opetussuunnitelmien perusteissa 1970–2014. *Kasvatus & Aika, 14*(3), 60–79. https://doi.org/10.33350/ka.82657

Säntti, J., Hansen, P. & Saari, A. (2021). Future jamming: Rhetoric of new knowledge in Finnish educational policy texts. *Policy Futures in Education, 19*(1), 1–18. https://doi.org/10.1177/147821032098570

Tanhua-Piiroinen, E., Kaarakainen S-S., Kaarakainen M.-T., Viteli, J., Syvänen, A. ja Kivinen A. (15.2.2019). *Digiajan peruskoulu*. (Valtioneuvoston selvitys- ja tutkimustoiminnan julkaisusarja 6/2019). Valtioneuvosto. http://julkaisut.valtioneuvosto.fi/handle/10024/161383

Tero, A., Takagi, S., Saigusa, K., Ito, K., Bebber, D. P., Fricker, M. D. & Nagakaki, T. (2010). Rules for Biologically Inspired Adaptive Network Design. *Science*, *327*(5964), 439–442. https://pdodds.w3.uvm.edu/files/papers/others/2010/tero2010a.pdf

Tervonen, L., Kortelainen, M. & Kanninen, O. (2018). Eliittilukioiden vaikutukset yliopisto-opintojen aloittamiseen ja koulutusalan valintaan. *Yhteiskuntapolitiikka 83*(4), 374–386. https://urn.fi/URN:NBN:fi-fe2018092036138

Tomperi, T. (2017). Kriittisen ajattelun opettaminen ja filosofia. Pedagogisia perusteita. *niin & näin, 4*, 95–112. https://netn.fi/sites/www.netn.fi/files/netn174-17.pdf

Tujula, I. & Ruuskanen, R. (2017). *Robo2 – Elektroniikan ja ohjelmoinnin alkeita varhaiskasvatukseen ja alkuopetukseen.* Käsityökoulu Robotti ry.

https://www.kasityokoulurobotti.fi/wp-content/uploads/2017/02/robo2_nettiversio_2017_01_28.pdf

Tuomi, J. & Sarajärvi, A. (2009). *Laadullinen tutkimus ja sisällönanalyysi.* 6. uudistettu painos. Kustannusosakeyhtiö Tammi.

Tuomi, J. & Sarajärvi, A. (2018). *Laadullinen tutkimus ja sisällönanalyysi*. Uudistettu painos. Kustannusosakeyhtiö Tammi.

Tutkimuseettinen neuvottelukunta (2023). *Hyvä tieteellinen käytäntö ja sen loukkausepäilyjen käsitteleminen Suomessa.* (Tutkimuseettisen neuvottelukunnan julkaisuja 2/2023.) Tutkimuseettinen neuvottelukunta. https://tenk.fi/sites/default/files/2023-03/HTK-ohje_2023.pdf

Petzold, C. (2000). *Code – Tietokonelaitteiden ja ohjelmien salainen maailma.* IT Press.

Vadén, T. (2002). Digitaalinen nominalismi – koodiutuminen ja tiedon omistajuuden ongelma tietoyhteiskunnassa. *niin & näin, 4*, 9–15. https://netn.fi/sites/www.netn.fi/files/netn024-06.pdf

Värri, V.-M. (2000). *Hyvä kasvatus – kasvatus hyvään. Dialogisen kasvatuksen filosofinen tarkastelu erityisesti vanhemmuuden näkökulmasta.* [väitöskirja, Tampereen yliopisto]. https://trepo.tuni.fi/bitstream/handle/10024/101122/varri_hyva_kasvatus_kasvatus_hyvaan.pdf?sequence=1&isAllowed=y

Värri, V.-M. (2014). Halun kultivointi ekologisen sivistyksen mahdollisuutena. Teoksessa A. Saari, O.-J. Jokisaari & V.-M. Värri (toim.), *Ajan kasvatus – kasvatusfilosofia aikalaiskritiikkinä* (s. 87–124). Tampere University Press. https://trepo.tuni.fi/handle/10024/103678

Williamson, B. (2016). Political computational thinking: policy networks, digital governance and 'learning to code'. *Critical Policy Studies 10*(1, 39–58. https://www.tandfonline.com/doi/pdf/10.1080/19460171.2015.1052003?needAccess=true

Williamson, B. (2017). *Big Data in Education: The digital future of learning, policy and practice.* SAGE Publications.

Williamson, B., Bergviken Rensfeldt, A., Player-Koro, C. & Selwyn, N. (2018). *Education recoded: policy mobilities in the international 'learning to code'*

agenda. Journal of Education Policy, 34(5), 705–725. https://doi.org/10.1080/02680939.2018.1476735

Williamson, B. & Hogan, A. (2020). *Commercialisation and privatisation in/of education in the context of Covid-19.* Education International. https://www.researchgate.net/publication/343510376_Commercialisation_and_privatisation_inof_education_in_the_context_of_Covid-19

Wing, J. (2006), Computational Thinking. *Communications of the Association for Computing Machinery*, *49*(3), 33–35. http://dx.doi.org/10.1145/1118178.1118215