Samu Peltonen

Ohjelmoinnin opiskelu verkossa

Tietotekniikan kandidaatintutkielma

12. toukokuuta 2017

Jyväskylän yliopisto

Tietotekniikka

Tekijä: Samu Peltonen

Yhteystiedot: samu.m.peltonen@student.jyu.fi

Työn nimi: Ohjelmoinnin opiskelu verkossa

Title in English: Studying programming online

Työ: Kandidaatintutkielma

Sivumäärä: 19+0

Tiivistelmä: Ohjelmoinnin opiskeluun on enemmän tapoja kuin koskaan, sekä aloittelijoille että kokeneemmille harrastelijoille ja ammattilaisille. Syy tähän on verkossa tarjottavan opetusmateriaalin määrän kasvu. Tutkielmassa selvitetään miten ohjelmointia voi opiskella verkossa ja perehdytään tarkemmin varsinkin MOOCeihin sekä tutoriaaleihin. Tämän jälkeen tutkitaan mitä asioita verkko-opiskelussa voi ottaa huomioon muun muassa opiskelun laadun parantamiseksi.

Avainsanat: ohjelmointi, verkko-opiskelu, etäopiskelu, MOOC

Abstract: Beginners as well as more experienced programmers and professionals have more ways to study programming than ever. The reason behind that is the increase in learning material on internet. This study finds out how can we learn programming online and it familiarizes ourselves especially with MOOCs and tutorials. After that we will study what things can be taken into consideration in online studying, for example to improve the quality of studying.

Keywords: programming, self-study, distant learning, MOOC

Kuviot

Kuvio 1. Ohjelmointitutoriaalien ominaisuudet 6
Kuvio 2. Kolbin oppimistyylit (https://www.simplypsychology.org/learning-kolb.html) 10

Sisältö

1 JOHDANTO 1

2 OPISKELU VERKOSSA 3
2.1 Massiiviset avoimet verkkokurssit 3
2.2 Ohjelmointiutoriaalit 5
2.3 Muut 7

3 VERKKO-OPISKELUSSA HUOMIOITAVIA ASIOITA 8
3.1 Mediaformaatit 8
3.2 Oppimistyylit 9
3.3 Ohjelmointistandardit ja -käytänteet 11
3.4 Palaute 12

4 POHDINTA 14

KIRJALLISUUTTA 15

1 Johdanto

Ohjelmoinnin opiskeluun on enemmän tapoja kuin koskaan, sekä aloittelijoille että kokeneemmille harrastelijoille ja ammattilaisille. Syy tähän on verkossa tarjottavan opetusmateriaalin määrän kasvu (Ada S. Kim & Andrew J. Ko (2017)). Tässä kirjallisuuskatsauksessa kartoitetaan, millaisia mahdollisuuksia verkko-/etäopiskelulle on, ja tutkitaan mitä asioita verkko-opiskelussa voi ottaa huomioon. Esille nousevia kysymyksiä ovat, mitkä ovat ohjelmoinnin verkko-opiskelun mahdolliset vahvuudet sekä heikkoudet, vaikuttavatko oppimistyylit opiskelun tehokkuuteen, onko opiskelussa käytettävän median formaatilla väliä ja miten opiskelijan saama palaute vaikuttaa oppimistuloksiin. Tutkielmassa ei käsitellä ohjelmointilogiikan oppimista tai ohjelmointikielen rakenteiden oppimista. Aihe on tärkeä, sillä on ihmisiä jotka eivät käy ohjelmointia opettavilla kursseilla, mutta haluavat opiskella ohjelmointia joko taustalla olevan kokemuksen tuella, tai täysin ilman aiempaa kokemusta. Työelämässä ohjelmointiin liittyvällä alalla oleva henkilö voi myös kokea tarvetta osaamisen laajentamiselle, eikä työnantaja välttämättä järjestä koulutusta, jolloin verkkomateriaalista opiskelu itsenäisesti on mahdollisesti varteenotettava vaihtoehto.

Tutkimusstrategiana on hakea tietoa jo suoritetuista kokeista ja tutkimukista joissa tutkittavat, jotka suurimmaksi osaksi ovat opiskelijoita, ovat käyttäneet verkkomateriaaleja ja ympäristöjä hyödykseen ohjelmoinnin opiskelussa, ja näistä saatuja tuloksia on joko arvioitu suoraan, tai verrattu kontrolliryhmään. Tulosten ja havaintojen pohjalta voidaan tehdä tässä kirjallisuuskatsauksessa johtopäätöksiä ja yhteenvetoa olennaisista asioista ohjelmoinnin verkko-opiskelussa. Vaikka ohjelmoinnin verkko-opiskelu on internetin aikakaudella aina ollut mahdollinen tapa opiskella ohjelmointia, tutkimustulokset aiheesta ovat osin rajattuja, koska osassa opiskelutavoista opiskelu on täysin itsenäistä, jolloin tutkijan on vaikea saada dataa oppimistuloksista.

Kirjallisuuskatsauksen ensimmäisessä osassa tutkitaan erilaisia konkreettisia tapoja opiskella ohjelmointia verkossa. Toisessa osassa perehdytään tutkimuksiin, jotka tarjoavat tuloksia siitä, miten yksittäiset huomioonotettavat asiat vaikuttavat ohjel-

moinnin verkko-opiskeluun. Toisessa osassa myös analysoidaan ensimmäisen osan opiskelutapoja lukujen teemojen näkökulmista.

2 Opiskelu verkossa

Tässä luvussa esitellään konkreettisia tapoja ohjelmoinnin verkko-opiskeluun. Opiskelutavat jaetaan kolmeen ryhmään: MOOCit, tutoriaalit, sekä muut verkkoa hyödyntävät opiskelutavat. Esittelyn lisäksi perehdytään opiskelutapoihin liittyviin tutkimuksiin, jotka tarjoavat statistiikkoja ja auttavat jakamaan aiheita pienempiin osiin syvällisempää tarkastelua varten.

2.1 Massiiviset avoimet verkkokurssit

Massiiviset avoimet verkkokurssit (Massive Open Online Courses, jatkossa MOOCs) ovat verkkokursseja, jotka ovat avoinna rajatun ajan, usein 6 – 10 viikkoa (René F. Kizilcec & Chris Piech & Emily Schneider (2013)). Nimensä mukaisesti ne ovat kaikille avoimia opiskeluympäristöjä. Näitä kursseja järjestävät varsinkin oppilaitokset, koska luentotyylinen formaatti on tällaisille tahoille luonteva toteuttaa (René F. Kizilcec & Chris Piech & Emily Schneider (2013)). Kurssien rakenne on usein lineaarinen, kuten normaaleissa luentokursseissa, ja kurssien luentoja voi katsoa usein sekä reaaliaikaisesti suoratoistona verkossa, että ladata videoina tietokoneelleen myöhempää katselua varten (René F. Kizilcec & Chris Piech & Emily Schneider (2013)). Näitä luentoja seuraa säännöllisesti tehtäviä, joista saadut pisteet vaikuttavat kurssista saatavaan arvosanaan, joka saattaa vaikuttaa saataviin opintopisteisiin, mikäli opiskelija on kirjoilla oppilaitoksessa joka hyväksilukee kyseenomaisen verkkokurssin (René F. Kizilcec & Chris Piech & Emily Schneider (2013)).

Videoiden ja tehtävien lisäksi kursseilla on usein käytössä myös jonkinlaisia viestintäkanavia, kuten foorumeita, joilla opiskelijat voivat hakea muun muassa vertaistukea (René F. Kizilcec & Chris Piech & Emily Schneider (2013)). Ohjaajien määrä on kuitenkin kurssin luonteesta johtuen rajallinen opiskelijoiden määrään verrattuna, joten henkilökohtaista ohjausta opiskelija ei välttämättä saa. Edellisen lisäksi eroja MOOCien ja esimerkiksi yliopistotason suppeampien ohjelmointikurssien käytänteiden välillä on melko vähän, suurimpana erona on mahdollisesti harjoitustöiden

puute, jolloin yhtenäinen käytännön kokonaisuus pitää muodostaa pelkillä viikkotehtävillä.

René F. Kizilcec & Chris Piech & Emily Schneider (2013) tutkivat MOOCien suorittajia, seuraamalla heidän käyttäytymistään ja tuloksiaan kurssin ajalta. Tutkittavana oli kolme eri tasoista tietotekniikan kurssia. Opiskelijoiden oletettiin kuuluvan neljään selkeään ja yleisesti toistuvaan kategoriaan heidän osallistumiskäyttäytymisensä perusteella. Kategoriat olivat 'suorittava' (englanniksi completing), 'seuraava' (auditing), 'putoava' (disengaging) ja 'valikoiva' (sampling). Suorittavat opiskelijat seurasivat kurssimateriaaleja ja palauttivat harjoitustyönsä aikataulun mukaisesti, eli he suorittivat kurssin tavalla, jolla he saivat mahdollisesti opintopisteitä ja kurssisuoritusmerkinnän. Seuraavat opiskelijat seurasivat kurssimateriaaleja koko kurssin ajan ja tekivät tehtäviä joko vähän tai ei ollenkaan, mutta suorittamisen määrä oli kuitenkin tasaista läpi kurssin. Putoavat opiskelijat suorittivat tehtäviä ja katsoivat videoita kurssin alussa, mutta jossakin kohtaa kurssiaktiivisuus oli kääntynyt laskuun, ja loppukurssista nämä opiskelijat olivat joko jättäneet kurssin kokonaan kesken, tai katsoneet vain videomateriaalit. Valikoivat opiskelijat joko seurasivat videomateriaalia muutaman luennon verran ja jättivät kurssin sen jälkeen kesken, tai opiskelivat jonkin verran materiaaleja valikoivasti, mahdollisesti aloittaen opiskelun vasta kun kurssi oli jo alkanut.

Tutkimuksen tuloksista pystytään vahvistamaan oletetut neljä opiskelijoiden kategoriaa. Lisäksi tutkimuksesta käy ilmi, että vaikka seuraavat opiskelijat eivät läpäise kurssia perinteisessä mielessä, saavat he silti mahdollisesti omiin tarpeisiinsa kursssin aiheesta tarpeeksi tietoa. Joissakin tapauksissa, riippuen kurssin tasosta, seuraavat opiskelijat hyötyivät kurssista jopa yhtä paljon kuin suorittavat opiskelijat. Tämä tarkoittaa sitä, että MOOCeille on useampia mahdollisia suoritustyylejä erilaisille opiskelijatyypeille tai erilaisissa lähtökohdissa oleville opiskelijoille, joilla päästään mahdollisesti samaan lopputulokseen, tai ainakin opiskelijan alunperin tavoittelemaan lopputulokseen. Tilastot (René F. Kizilcec & Chris Piech & Emily Schneider (2013)) osoittavat, että suorittavan tyylinen opiskelu on yleensä paras valinta opiskelijalle, mutta muiden opiskelutyylien ja opiskelun tavoitteiden takia kurssien

järjestäjien olisi hyvä harkita useampia suoritustapoja, varsinkin jos suoritukseen liittyy opintopisteitä.

MOOCien keskeyttämisen syyt olivat myös keskeisiä tutkimuksen kohteita. Tämä koski lähinnä putoavia opiskelijoita, tai valikoivia opiskelijoita joiden osallistuminen kurssin aktiviteetteihin sijoittui kurssin alkupuolelle. Tuloksista ja kyselyistä selviää, että suuri yhdistävä tekijä kurssien keskeyttämiselle on usein aikataulutukseen liittyvät ongelmat. Parempi vaihtoehto opiskelijalle voisi olla, jos kurssien kestoa voitaisiin pidentää, tai jättää takaraja tehtävien palautuksista kokonaan pois. On toki mahdollista että tästä seuraisi uusia ongelmia, kuten opintojen tarpeeton pitkittyminen. Seuraavissa alaluvuissa on esitelty ohjelmoinnin opiskeluun vaihtoehtoisia tapoja, jotka saattavat olla ajankäytöllisesti joustavampia.

2.2 Ohjelmointiutoriaalit

Suurta verkko-ohjelmointimateriaalien tarvetta täyttämään on syntynyt kampanjoiden ja muun muassa valtioiden rahoitusten avulla paljon erilaisia ohjelmointitutoriaaleja, joilla voidaan opettaa suuria massoja opiskelijoita, ilman aikataulurajoitteita (Ada S. Kim & Andrew J. Ko (2017)). Ohjelmointitutoriaaleilla ei ole tarkkaa määritelmää, mutta yleensä on kyse jonkinlaisesta askeleittain etenevästä opiskelumateriaalista, joka on saatavilla verkossa. Ada S. Kim & Andrew J. Ko (2017) jakavat tutoriaalit viiteen kategoriaan: Interaktiiviset tutoriaalit (opiskelijan pitää tuottaa omaa koodia edetäkseen tutoriaalissa eteenpäin), 'verkkolähteet' (web references, toimivat eräänlaisina sanakirjoina tai materiaalipankkeina), MOOCit, opettavat pelit (pelillistävät jollakin tavalla ohjelmoinnin opiskelun) ja luovat alustat (tarjoavat alustan ohjelmoinnille ja sen opiskelulle, mutteivät anna selviä tavoitteita). Tässä lajittelussa siis jo edellä käsitellyt MOOCit kuuluvat myös tutoriaaleihin, mutta tässä luvussa keskitytään enemmän muihin esiteltyihin kategorioihin.

Ohjelmoinnin verkkotutoriaaleissa suurimmassa osassa on bottom-up rakenne, jolloin opetuksessa siirrytään pienistä ja yksinkertaisista, mutta tärkeistä asioista suurempiin kokonaisuuksiin. Tämän lisäksi on myös tutoriaaleja, jotka interaktiivisuu-

dellaan mahdollistavat up-bottom rakenteen. Usein tämä tarkoittaa sitä että opiskelijalle annetaan valmiiksi toimivia komponentteja, joihin lisätään toiminnallisuutta esimerkiksi selaimesta löytyvällä editorilla, jolloin muutokset näkyvät reaaliajassa.

Ada S. Kim & Andrew J. Ko (2017) analysoivat kolmeakymmentä suosittua ohjelmointitutoriaalia ja selvittivät, mitä ne opettavat ja miten. Oheisessa taulukossa näkyy osa selvityksen tuloksista.

Tutorials	1. Personalization			2. Utilization	3. Contents									4. Organization		5. Context			6. Actionability	7. Feedback		8. Transfer learning			9. Support
	a. Age	b.Educational status	c. Coding experiences	Subsequent knowledge	a. Variables	b.Arithmatic	c. Logical	d.Conditional	e. Loops	f. Arrays	g. Function	h. Object		a. Bottom-up	b. Hierarchial structure	a. Lecture-based	b. Project-based	c. Storyline	Learners write code	a. Feedback	b. Immediate feedback	a. How to use	b. When to use	c. Why to use	Additional materials for self-monitoring
Codecademy Python		✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔	✔	✔	✔	✔	✔	✔	✔			✔
CodeSchool				✔	✔	✔	✔	✔						✔	✔	✔		✔	✔	✔	✔	✔			✔
Khan Academy		✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔		both	✔	✔	✔		✔	✔	✔	✔			✔
Online Python Tutor														top-down					✔	✔	✔				
Codingbat Python				✔	✔		✔	✔	✔	✔				✔	✔				✔	✔	✔				✔
Regex Golf														✔					✔	✔	✔				
Regex 101		✔												✔					✔	✔	✔	✔			✔
W3School JS					✔	✔	✔	✔	✔	✔	✔	✔		✔					✔	✔	✔	✔			✔
LearnPython					✔	✔	✔	✔	✔	✔	✔	✔		✔					✔	✔	✔				
Learnjavascript					✔	✔	✔	✔	✔	✔	✔	✔		✔					✔	✔	✔				
Funprogramming			✔	✔				✔	✔	✔	✔			✔					✔		✔	✔	✔		✔
TutorialsPoint Python		✔	✔		✔	✔	✔	✔	✔	✔	✔	✔		✔								✔			✔
Cprogramming		✔			✔	✔	✔	✔	✔	✔	✔			✔								✔			✔
Codingunit C tutorial		✔	✔		✔	✔	✔	✔	✔	✔	✔			✔								✔	✔		✔
Wired.com Javascript			✔		✔			✔	✔	✔	✔	✔		✔	✔							✔			✔
Pythonprogramming					✔	✔		✔	✔	✔	✔			both						✔		✔			✔
Stack Overflow					N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A		top-down						✔		✔	✔	✔	✔
edX Microsoft JS				✔	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔	✔				✔		✔			✔
Lynda.com		✔	✔		✔	✔	✔	✔	✔	✔	✔	✔		top-down	✔	✔				✔		✔	✔	✔	
Coursera Javascript				✔	✔	✔	✔	✔	✔	✔	✔	✔		top-down	✔	✔			✔	✔	✔	✔			✔
Coursera Python				✔	✔	✔	✔	✔	✔	✔	✔	✔		both	✔	✔			✔	✔	✔	✔			✔
Code.org	✔	✔	✔	✔	✔	✔	✔	✔	✔		✔			top-down	✔	✔	✔	✔	✔	✔	✔	✔			✔
Gidget				✔	✔	✔	✔	✔	✔	✔	✔			✔	✔		✔	✔	✔	✔	✔	✔			✔
CodeCombat		✔		✔	✔	✔	✔	✔	✔	✔	✔			both	✔			✔	✔	✔	✔	✔	✔		✔
LightBot	✔	✔	✔	✔	✔				✔					both	✔		✔	✔	✔	✔	✔				
CodeHunt			✔	✔	✔	✔	✔	✔	✔	✔				both	✔		✔	✔	✔	✔	✔				✔
Code Avengers		✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔	✔			✔	✔	✔	✔			✔
Grok Learning		✔	✔	✔	✔	✔	✔	✔	✔	✔	✔			✔	✔	✔			✔	✔	✔	✔		✔	✔
Scratch	✔	✔	✔		✔	✔	✔	✔	✔	✔	✔	✔		top-down			✔		✔	✔	✔	✔			✔
Alice		✔	✔		✔	✔	✔	✔	✔	✔	✔	✔		top-down			✔		✔	✔	✔	✔			✔

Kuvio 1. Ohjelmointitutoriaalien ominaisuudet

Tuloksista selviää, että vain muutamat tutoriaalit opettivat asioita tulevaisuutta ajatellen, eli ohjasivat opiskelijoita käyttämään opittua tietoa tutoriaalien ulkopuolella. Lisäksi ratkaisuilla oli harvoin vaihtoehtoisia malleja tai opiskelijoille ei kerrottu, miksi ja milloin erilaisia ratkaisumalleja tulisi käyttää. Opiskelija voi ottaa tämän huomioon valitessaan itselleen ohjelmointitutoriaalia, jolloin vaihtoehtojen määrä vähenee suuresti ja ymmärrys ohjelmoinnista saattaa kasvaa enemmän kuin karsi-

tuilla vaihtoehdoilla. Tuloksista havaitaan myös, että kaikissa tutoriaaleissa vaadittiin aktiivista koodin tuottamista, eli mitään tutoriaaleista ei ole suunniteltu pelkän lukemisen tai videomateriaalien katsomisen ympärille.

2.3 Muut

Siinä missä edellisissä luvuissa käsiteltyjä MOOCeja sekä tutoriaaleja voidaan käyttää myös muiden asioiden kuin ohjelmoinin opiskeluun, ohjelmointiin on myös erityisiä opiskelutapoja, joita ei välttämättä voida muussa opiskelussa hyödyntää.

Verkon ohjelmointiyhteisöjä voidaan käyttää valmiiden ratkaisujen etsimiseen, koska on todennäköistä että jollain muulla ohjelmoijalla on ollut jo sama ongelma ja sitä on jo kysytty jollakin julkisella palstalla. Opiskelussa voi myös hyödyntää ryhmien tai yhteisöjen välisiä projekteja, joista varsinkin avoimeen lähdekoodiin perustuvat projektit ovat ohjelmoinnin opiskelun kannalta relevantteja. Näiden lisäksi internettiä voi käyttää kattavien ohjelmointikielten ja -työkalujen dokumentaatioiden etsimiseen, omien skriptien kehitämiseen ja ajamiseen yleisillä sivuilla, sekä verkkosivujen lähdekoodien opiskeluun.

Yleisesti ottaen tässä luvussa esitellyt menetelmät toimivat ohjelmoinnin opiskelun tukena ja ne palvelevat selkeitä yksittäisiä päämääriä.

3 Verkko-opiskelussa huomioitavia asioita

Tässä luvussa eritellään asioita, joita opiskelija voi ottaa huomioon tai joita tulisi ottaa huomioon oppimisen varmistamiseksi ja tehostamiseksi, sekä opitun laadun takaamiseksi. Tämän lisäksi tutkitaan niiden toteutusta MOOCeissa, verkkotutoriaaleissa ja muissa opiskelutavoissa.

3.1 Mediaformaatit

Lisääntynyt verkossa olevan opiskelumateriaalin määrä on aiheuttanut pohdintaa siitä, mitkä ovat onnistuneen opiskelumateriaalin keskeisiä ominaisuuksia. Vikas Sahasrabudhe & Shivraj Kanungo (2014) ovat tutkineet asiaa mediaformaatin valinnan näkökulmasta. Verkossa opiskellessa on mahdollista hyödyntää monia erilaisia mediaformaatteja. Näitä voidaan pelkistäen vertailla esimerkiksi akselilla yksinkertaisimmasta tai "pelkistetyimmästä"monimutkaisempaan tai "korkeamman tason mediaan". Toisessa päässä akselia on käytännössä siis pelkkä teksti ja toisessa päässä esimerkiksi videota, audiota ja vaikkapa graafeja sisältävää materiaalia. Edellä mainitussa tutkimuksessa tutkittiin neljää mediaformaatin kategoriaa: tekstiä ja grafiikkaa (TG), tekstiä, grafiikkaa ja ääntä (TGS), tekstiä, grafiikkaa ja puhuvaa päätä (TGTH), sekä tekstiä, grafiikkaa ja videota tai animaatiota (TGVA). TG-kategoriassa on käytetty siis tekstin lisäksi vain kuvia ja graafeja, eli rajoitukset ovat samat kuin normaalilla oppikirjalla. TGS-kategoriassa edelliseen on lisätty kommentointiraita esimerkiksi diaesitykseen. TGTH-kategoriassa äänellä on näkyvä lähde, eli ohjeistajan pää. TGVA-kategoriassa video ei rajoitu pelkästään ohjeistajan päähän vaan videokuvaa on muustakin opiskelumateriaaliin liittyvästä.

Vaikka "korkeamman tason"mediaformaattija suositaan nykyisin laajasti, tutkimus osoittaa että tehokkain oppiminen saavutetaan jo TGTH-kategoriassa, ja TGVA-kategoriassa oppimistulokset saattavat jo olla marginaalisesti huonompia. TG- ja TGS-kategorioissa tulokset ovat samaa tasoa. Kaikenkaikkiaan kategoriat sijoittuivat oppimistulosten vertailussa alle kymmenen prosenttiyksikön sisään, joten erot ovat havainnoitavia,

mutteivät järin merkityksellisiä.

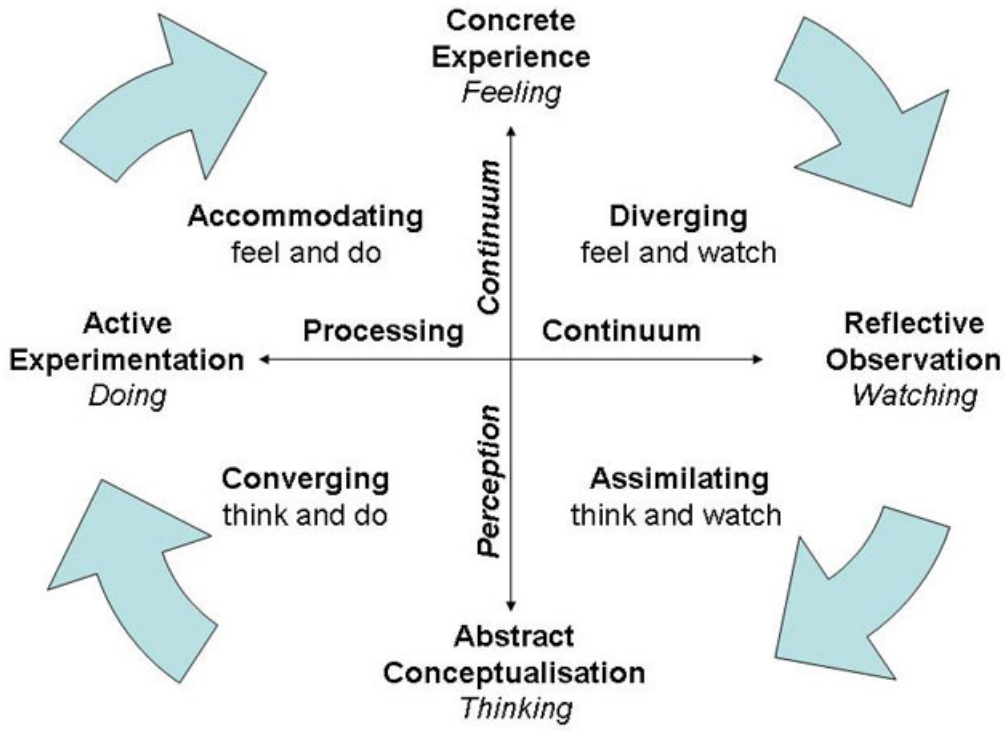
MOOCeissa tarjottavien mediaformaattien määrä on rajallinen, mutta luentovideoilla (TGVA) tarjotaan hyvät opiskeluedellytykset. Erilaisissa ohjelmointitutoriaaleissa mediaformaatit vaihtelevat laajasti. Skaalan toisessa päässä ovat pelkkää tekstiä ja ohjelmakoodia sisältävät tutoriaalit, jotka muistuttavat paljon ohjelmointia opettavia kirjoja. Toisessa päässä ovat laajat tutoriaalit jotka tarjoavat useita erilaisia mediaformaatteja, mukaanlukien interaktiivisia formaatteja, joita Vikas Sahasrabudhe & Shivraj Kanungo (2014) eivät ole huomioineet. Luvussa 2.3 käsitellyt opiskelutavat sisältävät suurimmaksi osaksi pelkkää tekstiä (TG) niiden käytännönläheisyyden ja pelkistetyn luonteen takia, joten Vikas Sahasrabudhe & Shivraj Kanungo (2014) mukaan ne soveltuvat yksittäisinä opiskelutapoina hieman heikommin ohjelmoinnin opiskeluun.

3.2 Oppimistyylit

Erilaisten oppimistyylien on havaittu vaikuttavan verkossa tapahtuvan opiskelun tehokkuuteen ja niiden välisiä eroja on tutkittu Ruey-Shiang Shaw (2012) suorittamassa tutkimuksessa. Oppimistyylit jaetaan siinä neljään eri kategoriaan: 'osallistuja' (diverger), 'tarkkailija' (assimilator), 'päättelijä' (converger) ja 'toteuttaja' (accommodator). Nämä nähdään havainnollistettuna alla kuviossa 2.

Tutkimuksesta saadut tulokset osoittavat, että toteuttaja oppijatyypit saivat aikaan parhaita oppimistuloksia. Hieman tämän ryhmän jälkeen tulivat päättelijä ja osallistuja oppijatyypit. Eroa näiden kolmen ryhmän välillä oli vain vähän. Selkeästi viimeiseksi jäi tarkkailijoiden ryhmä. Eli siis oppimistyyli joka luottaa paljon asioiden konseptualisointiin ja tarkkailuun, jättäen käytännönläheisemmät aspektit vähemmälle huomiolle, toimii ohjelmoinnin opiskelussa huonommin.

Kuten René F. Kizilcec & Chris Piech & Emily Schneider (2013) havaitsivat tutkimuksessaan, käytännönläheisemmät opiskelutavat olivat MOOCeissa usein oppimistulosten kannalta parempia, jolloin ainakin osallistuja ja toteuttaja oppimistyyleillä saadaan myös näissä aikaan todennäköisesti parhaita tuloksia. Toisaalta kuten



Kuvio 2. Kolbin oppimistyylit (https://www.simplypsychology.org/learning-kolb.html)

luvussa 2.1 mainittiin, myös luentomateriaalien seuraaminen ilman tehtävien tekemistä oli joissain tapauksissa pätevä opiskelumetodi, mikä tekee tarkkailija ja päättelijä oppimistyyleistä kuitenkin mahdollisesti toimivan MOOCien suorittamisessa. Ada S. Kim & Andrew J. Ko (2017) mukaan ohjelmointitutoriaaleissa taas vaaditaan aina aktiivista koodin tuottamista, joten osallistuja ja toteuttaja oppimistyylit soveltuvat selkeästi tähän opiskelutapaan parhaiten. Myös luvussa 2.3 esitetyt opiskelutavat ovat käytännönläheisiä, jolloin osallistuja ja toteuttaja oppimistyylit toimivat niissäkin.

3.3 Ohjelmointistandardit ja -käytänteet

Ohjelmoinnin standardit ovat ohjelmoinnin parissa työskentelevien ihmisten yhdessä säätämiä ohjeita ja linjauksia, joita ohjelmoinnissa tulisi noudattaa. Tähän kuuluvat muun muassa nimeämiskäytännöt, sisennykset, rivitykset, sekä myös koodin toimintaan vaikuttavia ohjeita, esimerkiksi funktion paluuarvojen palautus. Monilla kielillä ohjelmoinnin standardit eroavat hieman toisistaan. Xiaosong Li & Christine Prasad (2005) mukaan on todistettu, että ohjelmointistandardien noudattaminen ohjelmistokehityksessä parantaa kehitystiimin kommunikaatiota, vähentää ohjelmakoodissa esiintyviä virheitä ja parantaa ohjelmakoodin yleistä laatua. Koska standardien ja käytänteiden noudattaminen ei ole välttämätöntä toimivan ohjelmakoodin aikaansaamiseksi, on verkkolähteiden välillä eroja siinä miten paljon näitä opetetaan. Kuten Ada S. Kim & Andrew J. Ko (2017) selvittävät, melko harva verkkotutoriaali selittää, miksi esitettyjä ratkaisuja tulisi käyttää. Eli vaikka mallivastaukset olisivat standardien mukaista koodia, opiskelijaa ei välttämättä informoida siitä, mikä niistä tekee standardien mukaista.

Xiaosong Li & Christine Prasad (2005) tutkivat opiskelijoiden suhtautumista ohjelmoinnin standardeihin. Tuloksista havaitaan, että kokeneemmat opiskelijat pitävät standardien opiskelua hyödyllisempänä. Huolimatta siitä, että ylemmän vuosikurssin opiskelijat olivat jo opiskelleet standardeja, löytyi tästä ryhmästä enemmän ihmisiä jotka kokivat, että ohjelmointikurssista vähintään 15% ajasta tulisi käyttää ohjelmoinnin standardien opiskeluun. Tämä tarkoittaa sitä, että jollei verkko-opiskelussa ole ohjaajaa, joka valmiiksi miettii standardien opettamisen opiskelijalle, täytyy niiden opiskeluun kiinnittää itse erityistä huomiota, varsinkin jos opiskelija on kokemattomampi ohjelmoija.

Koska MOOCit ovat usein luentopohjaisia kursseja ja luennoitsija tai luennoitsijat ovat ohjelmoinnin asiantuntijoita, on vastuu standardien opettamisesta suurelta osin heillä. Verkkotutoriaaleissa materiaalien valmistajat päättävät, kuinka paljon he sisällyttävät opiskelumateriaaleihin asiaa standardeista. Sekä MOOCeista että verkkotutoriaaleista opiskellessa opiskelijalla on tietysti mahdollisuus täydentää tietämystään standardeista muista lähteistä. Ongelmaksi saattaa muodostua se että

aloittelevat ohjelmoijat, jotka hyötyisivät eniten standardien opiskelusta, eivät osaa etsiä tietoa niistä muualta tai eivät edes näe sitä tarpeelliseksi, kuten Xiaosong Li & Christine Prasad (2005) ovat tutkimuksessaan selvittäneet.

3.4 Palaute

Perinteisen lähiopetuksen etuna on se, että opiskelija saa lähes poikkeuksetta opettajalta tai ohjaajalta palautetta opiskelustaan. Ohjelmoinnissa palautetta ja neuvoja saa muun muassa syntaksivirheistä ihmiskielellä, kun taas verkossa opiskellessa ja ohjelmakoodia ajaessa ainoa palaute mitä opiskelija syntaksivirheestä saa, on kääntäjän virheilmoitus, joka ympäristöstä riippuen saattaa olla melko kryptinen varsinkin aloittelevalle ohjelmoijalle. Osa ohjelmointiympäristöistä saattaa tarjota myös varoituksia, jos tuotettu ohjelmakoodi on esimerkiksi kielen standardien vastaista, vaikka se menisi kääntäjästä läpi. Tällöin palaute on usein selkeää mutta jää pinnalliseksi (Michael J. Lee & Andrew J. Ko (2011)).

Ada S. Kim & Andrew J. Ko (2017) mukaan tavoitteellisessa opiskelussa tulisi aina olla mukana kohdennettua palautetta, joka myös antaa opiskelijalle tietoa opiskelun edistymisestä. Oppimisen kannalta on myös hyödyllistä että palaute on välitöntä, mitä ei välttämättä saavuteta verkon opiskeluympäristöissä, koska opiskelija joutuu odottamaan että joko verkkokurssin ohjaaja tai mahdollinen vertaisopiskelija vastaa kysymykseen.

Michael J. Lee & Andrew J. Ko (2011) ovat tutkineet ohjelmointityökalujen antamaa palautetta ja sen vaikutusta opiskelijoihin ja opiskelijoiden motivaatioon. Tutkimusta varten kehitettiin Gidget niminen peli, jossa pelaaja ohjastaa nimikkohahmoa ja debuggaa viallista ohjelmakoodia. Mikäli virheitä syntyy, Gidget osaa hypätä virheellisten komentojen yli ja kertoo pelaajalle ihmismäisesti ettei tunne viallista komentoa. Tutkimuksessa keskityttiin Gidgetin testaamiseen aloittelevilla ohjelmoijilla. Ohjelmoijia jaettiin sattumanvaraisesti joko kontrolliryhmään tai kokeelliseen ryhmään; kontrolliryhmän käyttämä Gidget oli persoonaton ja palaute oli neutraalia, kun taas kokeellisen ryhmän käyttämä versio oli persoonallinen ja palaute oli

ihmismäisempää. Tutkimuksessa kerätyistä logeista ja kyselyistä selviää, että ohjelmointiympäristön ihmismäinen palaute voi lisätä aloittelevien ohjelmoijien motivaatiota ohjelmoida. Lisäksi käy ilmi, että kokeneempien ohjelmoijien keskuudessa tälläistä ilmiötä ei havaita, joten ihmismäisen palautteen tuomat edut häviävät kokemuksen karttuessa.

Laadukkaan palautteen saaminen MOOCeissa ja tutoriaaleissa epävarmaa. Kuten luvussa 2.1 todettiin, MOOCeissa ohjaajien määrä on kurssin luonteesta johtuen rajallinen opiskelijoiden määrään verrattuna, joten opiskelija ei välttämättä saa riittävää palautetta tai palautteen saamiseen kuluu liian kauan aikaa. Tutoriaaleissa ihmismäisen palautteen saaminen on vielä epävarmempaa, koska opiskelija ei ole välttämättä ihmisten kanssa tekemisissä ollenkaan.

4 Pohdinta

Tässä kirjallisuuskatsauksessa tutkittiin erilaisia tapoja opiskella ohjelmointia verkossa ja selvitettiin, mitä asioita verkko-opiskelussa voi ottaa huomioon.

Verkon opiskelutavoista perehdyttiin suurimmaksi osaksi MOOCeihin ja tutoriaaleihin, sekä tuotiin hieman esille opiskelua mahdollisesti tukevia opiskelutapoja. Suosittuja opiskeluympäristöjä on verkossa paljon, mutta kuten Ada S. Kim & Andrew J. Ko (2017) selvittivät, suuressa osassa niistä on puutteita opetuksen laadussa. Tällöin siis opiskelun onnistuneisuus riippuu paljolti laadukkaiden opetusympäristöjen löytämisestä. 2.3 luvussa esitellyt opiskelutavat taas toimivat muiden opiskelutapojen tukena.

Opiskelija voi halutessaan ottaa huomioon erittäin monta asiaa, joilla voi tehostaa tai varmistaa omaa ohjelmoinnin opiskeluaan verkkoympäristöissä. Näihin kuuluvat erilaisten mediaformaattien ja oppimistyylien vaikutus, sekä standardien, käytänteiden ja palautteen merkitys. Näiden lisäksi on myös mahdollisesti määräämättömän paljon muita alueita, kuten kokemuksen vaikutus ohjelmoinnin verkko-opiskeluun, joita ei tässä tutkimuksessa käsitelty tai jotka eivät saaneet omaa käsittelylukuaan.

Verkossa ohjelmoinnin opiskelu sisältää jonkin verran riskitekijöitä, varsinkin kokemattomalle ohjelmoijalle. Näistä suurin osa liittyy siihen että opiskelija ei välttämättä saa riittävää palautetta eikä verkkomateriaaleissa välttämättä opeteta tarpeeksi ohjelmakoodin laatuun liittyviä asioita. Valinnanvaraa on kuitenkin laajasti ja opiskelu on opiskelijalle joustavaa. Jatkotutkimusta aiheesta voidaan saada tutkimalla sitä, miten merkittäviä tutkitut asiat ovat suuressa mittakaavassa opiskelun tulosten kannalta, kun mukaan vertailuun otetaan lisää näkökulmia, kuten opiskelumotivaation merkitys.

Kirjallisuutta

Ruey-Shiang Shaw, K. 2012. *A study of the relationships among learning styles, participation types, and performance in programming language learning supported by online forums*. Computers & Education Volume 58, s. 111–120.

Xiaosong Li & Christine Prasad 2005. *Effectively teaching coding standards in programming*. SIGITE '05 Proceedings of the 6th conference on Information, s. 239–244.

Vikas Sahasrabudhe & Shivraj Kanungo 2014. *Appropriate media choice for e-learning effectiveness: Role of learning domain and learning style*. Computers & Education Volume 76, s. 237–249.

René F. Kizilcec & Chris Piech & Emily Schneider 2013. *Deconstructing Disengagement: Analyzing Learner Subpopulations in Massive Open Online Courses*. LAK '13 Proceedings of the Third International Conference on Learning Analytics and Knowledge, s. 170–179.

Ada S. Kim & Andrew J. Ko 2017. *A Pedagogical Analysis of Online Coding Tutorials*. SIGCSE'17 March 8–11, s. 321–326.

Michael J. Lee & Andrew J. Ko 2011. *Personifying Programming Tool Feedback Improves Novice Programmers' Learning*. ICER 11 Proceedings of the seventh international workshop on Computing education research, s. 109–116.