Julia Tytti Karoliina Sippola

Katsaus ohjelmoinnin opetustyyleihin ja motivointiin

Tietotekniikan kandidaatintutkielma

13. toukokuuta 2020

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Julia Tytti Karoliina Sippola

Yhteystiedot: julia.t.k.sippola@student.jyu.fi

Ohjaaja: Antti-Jussi Lakanen

Työn nimi: Katsaus ohjelmoinnin opetustyyleihin ja motivointiin

Title in English: Overview of teaching methods and motivation in programming

Työ: Kandidaatintutkielma

Sivumäärä: 23+0

Tiivistelmä: Tämän kandidaatintutkielman kartoittaa, tarkastelee ja tutkii erilaisia tapoja ja menetelmiä, miten ja millä tavoin ohjelmointia pyritään opettamaan ja motivoimaan peruskursseilla. Tavoitteena on saada kokonaisvaltainen kuva erilaisista opetuskäytännöistä ja motivointitavoista. Tutkimus toteutettiin kirjallisuuskatsauksena.

Avainsanat: opetus, ohjelmointi, motivaatio

Abstract: This Bachelor's thesis reviews different teaching methods and approaches to motivate and keep students engaged during their first computer science and programming courses.

Keywords: teaching, education, computer science, motivation

Kuviot

Sisältö

1 Johdanto

Tässä kandidaatintutkielmassa kartoitetaan keinoja ja tekniikoita ohjelmoinnin kiinnostavaan ja motivoivaan opetukseen. Tutkielmassa pureudutaan siihen, millaisin opetusmenetelmin ohjelmointia on opetettu ja millä keinoin opiskelijoita pyritään motivoimaan. Tutkimuskohteena on ohjelmoinnin perusteisiin keskittyvät kurssit korkeakouluympäristössä. Aiemmista tutkimuksista on esimerkiksi noussut esille se, ettei ohjelmoinnin perusteiden opetukseen ole kehitetty yhtenäistä universaalia mallia, vaan kurssitoteutukset ovat usein lähteneet oppilaitosten omien tarpeiden pohjalta (Sheard ym. 2009). Tämä varmasti palvelee oppilaitosten omia tarkoitusperiä, mutta esimerkiksi oppimistulosten vertailu oppilaitosten välillä saattaa olla haasteellista. Myös peruskursseilla opetettavat asiat saattavat erota yliopistojen välillä huomattavasti (Dale 2005). Esimerkiksi opetukseen valittu ohjelmointikieli vaihtelee yliopistojen ja kurssien välillä, ja kursseilla painotetut asiat voivat olla hyvinkin erilaisia. Silmukat ja ehtolauseet ovat ainakin vielä suuri osa peruskurssien opetusta (Dale 2005).

Ohjelmointia pidetään yleisesti hankalana oppia ja opettaa (Sheard ym. 2009). Yhdeksi syyksi on esitetty perusteiden monimuotoisuutta: jotta voi sanoa osaavansa ohjelmoida, yksilön täytyy osata hallita, soveltaa ja pystyä yleistämään suuri määrä erilaisia tietojenkäsittelyyn liittyviä tarkkoja konsepteja (Robins, Rountree ja Rountree 2003). Edellä mainitut asiat voivat luoda opiskelijoille tunteen ohjelmoinnin sirpaleisuudesta ja vaikeaselkoisuudesta, ja tätä kognitiivista kuormaa voidaan kenties helpottaa varsinkin peruskursseilla erilaisin työkaluin. Malmi ym. (2019) mukaan erilaisia teorioita pyritään aktiivisesti kehittämään tietojenkäsittelytieteiden opettamiseen, ja myös moni näistä teorioista lainataan myös kasvatus- ja sosiaalitieteiden puolelta.

Tässä tutkielmassa käsitellään millaisia käsityksiä oppimisesta on ja millaisia kurssitoteutuksia ohjelmoinnin peruskursseilla käytetään. Lisäksi tutkitaan opiskelijoille tarkoitettuja motivointitapoja, keskittyen visualisointityökaluihin, teemoittamiseen ja kontekstualisointiin sekä osaamismerkkeihin. Korkeakouluun ja ohjelmoinnin opintoihin hakeutuminen viittaa siihen, että opiskelijoilla on jo olemassa jonkintasoinen mielenkiinto sekä kiinnostus oppia tietojenkäsittelyä sekä eritoten ohjelmointia. Pitkällä tähtäimellä heillä on mahdollisesti tavoitteenaan suorittaa tutkinto informaatioteknologian alalta ja löytää tulevaisuudessa paikka

alan asiantuntijoina. Toisaalta taas opiskelijoiden motivaation syyt saattavat vaihdella suurestikin, vaikka tähtäin olisikin sama. Perusteluita ja syitä juurikin ohjelmoinnin opiskeluun on tutkinut esimerkiksi Jenkins (2001), ja kahdeksi pääsyyksi osoittautui opiskelijoiden toive jonkin tulevan mahdollistamiseksi, kenties taloudellisesti luotettavan ammatin takia, sekä oppimisen halun takia (Jenkins 2001). Samassa tutkimuksessa todettiin myös opiskelijoiden antavan vähän arvoa pelkästään kurssin läpipääsylle, mikä osoittaisi opiskelijoiden olevan nimenomaan sitoutuneita oppimaan.

Yleinen käsitys tietojenkäsittelytieteiden ja ohjelmoinnin kursseista on, että niitä keskeytetään keskimäärin enemmän ja useammin kuin muiden alojen peruskursseja. Tätä ei kuitenkaan todellisuudessa ole kuitenkaan tutkittu niin paljoa kuin kirjallisuus antaisi ymmärtää, ja kattavaa empiiristä näyttöä väitteelle ei ole (Watson ja Li 2014). On myös hyvin epätodennäköistä, että kukaan opettaja tekisi kursseistaan tarkoituksenmukaisesti vaikeita tai epämotivoivia. Kirjallisuutta tutkiessa yritettiin siis löytää ne keinot, joiden on todettu tehokkaasti ylläpitävän motivaatioita ja innostusta vaikeaksi koetun asian opetteluun ja opiskeluun. Tavoitteena oli löytää vastaukset seuraaviin kysymyksiin: millä tavoin ohjelmointia opetetaan sekä millaisia erilaisia motivointitapoja ja työkaluja kursseilla suositaan.

Luvussa 2 tutustutaan tarkemmin ohjelmoinnin opetukseen sekä yleisesti opettajien lähestymistyyleihin sekä aktiiviseen oppimiseen. Siinä esitellään myös opiskelijoiden kokemia haasteita ohjelmoinnissa. Luvussa 3 avataan erilaisia käytössä olevia motivointityökaluja ja esitellään esimerkkejä käytössä olevista ratkaisuista. Kappaleissa pohditaan myös niiden vaikutusta ohjelmoinnin opiskelun kiinnostavuuteen ja osallisuuteen sekä näin ollen myös motivaatioon. Luku 4 tarjoaa yhteenvedon koko tutkielmasta.

2 Ohjelmoinnin oppiminen ja opetus

Seuraavissa alaluvuissa käsitellään käsityksiä opetuksesta ja oppimisesta. Oppiminen on niin opettajasta kuin oppilaastakin kiinni, ja oppimiseen vaikuttaa myös monet asiat tässä esitellyn ulkopuolelta. Esimerkiksi opiskelijan henkilökohtaiset huolet ovat suuri tekijä, johon opettajan on hankala puuttua tai helpottaa. On hyvä muistaa, että kursseilla on hyvin paljon erilaisia opiskelijoita, ja opettajan näkemykset opiskelijoistaan ja oppimisesta vaikuttavat kurssin käytäntöihin. Kurssin suoritustapa sekä valitut opetusmetodeihin ovat hyvä esimerkki. Opetuksen käytössä olevat resurssit vaikuttavat niin opiskelijoiden kuin opettajienkin motivaatioon, sillä riittämättömät resurssit pienentävät mahdollisuutta esimerkiksi henkilökohtaiseen ohjaukseen tai avunantoon.

2.1 Aktiivinen oppiminen

Opiskelijat ovat monipuolinen joukko erilaisia ihmisiä, ja heillä kaikilla on erilaiset tavoitteet ja motivaatio tiettyä kurssia tai opintokokonaisuutta kohtaan: on hankala siis määrittää sellaista yksittäistä opetusmetodia, joka palvelisi kaikkia. Opetusmetodin valintaan taas vaikuttaa opettajan käsitys siitä, miten oppiminen tapahtuu ja millaisia opiskelijoita hän mielestään opettaa. Biggs ja Tang (2011) jakavat opettajien ajatusmallit oppimisesta kolmeen tasoon.

Tasolla 1 opettaja vain jakaa tietoa eteenpäin opiskelijoilleen, ja opettaja luottaa siihen, että opiskelijat ovat tarpeeksi motivoituneita ja itseohjautuvia. Opiskelijoiden oletetaan omaavat tarpeelliset ja hyvät akateemiset taidot ja he ovat kiinnostuneita aiheesta syvemmin jo ennen kurssin alkamista. Ajattelumallin voisi tiivistää kysymykseen *millainen opiskelija on*: opiskelijalla itsellään on joko hyvät akateemiset taidot tai hänen tulisi itsenäisesti harjoitella ja parantaa niitä. Hyvät akateemiset taidot omaava opiskelija tietää, miten oppii ja hän on jo valmiiksi motivoitunut. Opiskelija voisi näin oppia periaatteessa mitä tahansa, jopa ilman opettajan apua. Jos opiskelijan akateemiset taidot eivät ole riittävällä tasolla, siihen opettaja ei voi tai tarvitse vaikuttaa. Tämä johtaa helposti ajattelutapaan, jossa vastuu oppimisesta on opiskelijalla itsellään: on opiskelijan oma syy, jos hän ei ole oppinut.

Seuraavana esitellään taso 2, jossa keskitytään siihen, *mitä opettaja itse tekee*: kenties opettaja luo mielenkiintoisia ja jännittäviä tiedonjakostrategioita, joita on helppo ja ymmärrettävä seurata, mutta opiskelijoita ei osallisteta erityisesti opetuksessa. Vastuu oppimisesta siirretään opettajalle: hän kenties osaa opettaa mielenkiintoisesti ja selkeästi, mutta tehtävät ovat opettajapainotteisia, ja ne eivät välttämättä vaadi oppilaalta aktiivisuutta tai syvempää paneutumista aiheeseen. Vastaukset saatetaan antaa oppilaille ilman, että heidän tulisi pohtia aihetta itsenäisesti riittävällä tasolla, jotta oivallus ja ymmärrys voisivat kehittyä tarpeelliselle tasolle. Ajattelumalli kääntää syytökset oppimattomuudesta opettajaan: opettaja ei osaa opettaa, kun opiskelijat eivät ole ymmärtäneet tai eivät hallitse jotain kurssin aihealuetta. Oppilaille ei anneta kovin aktiivista roolia opetuksessa.

Tasolla 3 ajattelumalli yhdistää molemmat mallit: sekä opiskelijalla ja opettajalla on oma roolinsa opetuksen onnistumisessa. Opettajan tulee pyrkiä *aktivoimaan opiskelijoita ajattelemaan itse* ja näin kursimaan kasaan oppilaiden lähtökohtaisia eroavaisuuksia aktiivisuudessa (Biggs ja Tang 2011). Aktivoiva opetustyyli pyrkii saamaan opiskelijoita tavoittelemaan syvempää ymmärrystä opetettavasta aiheesta. Syvempi ymmärrys luo yleensä myös positiivisen kuvan opiskelijan omasta osaamisestaan, jolloin myös osallisuus ja mielenkiinto kehittää omaa osaamistaan kasvaa. Näin myös motivaatio pysyy yllä tai lähtee kukoistamaan.

Aktiivinen malli palvelee niin valmiiksi aktiivisia opiskelijoita kuin myös epäaktiivisia: tehtävät ja kurssin suoritusvaatimukset suunnitellaan sillä tavalla, että ilman osallistumista ja aktiivisuutta niitä ei voi suorittaa hyväksytysti. Jako perustuu vahvasti konstruktivistiseen opetusteoriaan, jossa oppijat rakentavat uutta tietoa jo olemassa olevan puitteille (Biggs ja Tang 2011).

Opetuksessa tulisi pyrkiä pois pintapuolista oppimesta ja kannustaa opiskelijoita oppimaan niin sanotusti syvemmällä tasolla, jotta kurssisuoritukset eivät kulminoituisi pelkästään läpäisyyn ja näennäisesti oppimistavoitteiden saavuttamiseen (Biggs ja Tang 2011). Ohjelmoinnin kontekstissa kuvailtu syvempi osaaminen ja ymmärrys on kriittistä: pelkällä pintapuolisella oppimisella on todella työlästä läpäistä edes peruskursseja, sillä yksinkertaisimmatkin tehtävät ovat yleensä jo jollain tasolla soveltavia pelkän tiedon toistamisen sijaan. Peruskursseilla opetetaan asiat, mitä jatkokursseilla tullaan tarvitsemaan: jos nämä asiat eivät ole hallussa, oppilas tuskin tuntee luottamusta ja uskoa omaan osaamiseensa, milloin

myös motivaatio ja kiinnostus ovat vaarassa kadota.

Myös Robins, Rountree ja Rountree (2003) on tullut samanlaisiin tuloksiin syvemmän oppimisen painotuksesta opetuksesta, ja lisää lisäksi vielä hyvän kommunikaation oppilaan ja opettajan välillä olevan myös tärkeää. Tutkimuksessa todetaan, että kannustavat metodit elinikäiseen ja syvempään oppimiseen luodaan selvästi kerrotuilla kurssitavoitteilla ja konkreettisilla päämäärillä, esimerkiksi kertomalla opiskelijoille, että tämän kurssin jälkeen osaatte tehdä oman pelin alusta loppuun. Myös kurssin sisällä tapahtuvien tavoitteiden selkeys on suotavaa: ensimmäisellä viikolla opetellaan muuttujat, toisella viikolla ehtolauseet. Opiskelijoiden sitouttaminen esimerkiksi viikoittaisiin tehtäviin sekä tarkoituksenmukainen arviointi ja palaute opettajalta opiskelijalle lisää toivottua aktiivisuutta ja osallisuutta. Myös kommunikaatio opettajan ja opiskelijan välillä lisääntyy. Toiset opiskelijat tarvitsevat enemmän tukea tarvittavan aktiivisuuden tason saavuttamiseen, ja jos tukea ei ole saatavilla kurssin aikana tarpeeksi, saattaa oppilaan kiinnostus ja motivaatio lopahtaa hyvinkin nopeasti.

Pintapuolinen opettelu ei myöskään saa aikaan positiivisia tunteita oppiainetta kohtaan, ja vaikutus voi itseasiassa olla päinvastainen: kurssimateriaalin ylimalkainen vilkuilu ja huolimaton paneutuminen saattaa aiheuttaa esimerkiksi ahdistuneisuutta, kyynisyyttä tai turhautuneisuutta opiskelijassa (Biggs ja Tang 2011). Syvempi lähestymistapa taas yleensä luo tunteen nautinnosta, ja opiskelu ikään kuin vie mukanaan ja aktivoi osallistumaan ja keskustelemaan esimerkiksi luennoilla opettajan kanssa.

Kuten ylempänä todetaan, aktiivinen pohdinta ja osallisuus parantavat opiskelijoiden oppimista sekä motivaatiota. Tätä aktiivista oppimista pyritään tukemaan ohjelmointikurssien suoritustavoista lähtien: varsinkin ohjelmoinnin opetuksessa yleisesti ”learning by doing” eli ”tekemällä oppii” on todettu olevan tehokas. Perinteisesti peruskursseilla suositaan viikoittaisia tehtäviä sekä luento-opetusta, ja osaaminen testataan tentissä tai muunlaisessa todistuksessa osaamisesta. Myös harjoitustyöt ovat osa joitain kursseja. Ohjelmointikurssien tehtävät ovat yleensä hyvin käytännönläheisiä, vaikka opetustavoitteena pyritään opettamaan yleistettäviä malleja ja toimintatapoja, joita voidaan soveltaa monenlaiseen ongelmaan ja antamaan tarvittavat pohjatiedot soveltavampia ongelmia varten.

Myös muita tapoja peruskurssien suorittamiseen on olemassa. Alla on listattuna joitain vaih-

toehtoisia tapoja, mutta joihin ei tässä tutkielmassa mennä tarkemmin. Kurssisuoritukset voivat olla myös sekoitus monesta erilaisesta tavasta.

- Kontakti- tai lähiopetus sekä viikkotehtävät
- Itseopiskeltava materiaali ja viikkotehtävät
- Pelkkä tentti / muu loppukoe tai harjoitustehtävä
- Itsenäisesti opiskeltavat kokonaisuudet ilman aikarajoituksia

2.2 Motivaatio ja kiinnostus

Korkeakouluissa ohjelmointikurssin valinneilla henkilöillä on yleensä jo valmiiksi jonkinlainen motivaatio ja kiinnostus ohjelmointia kohtaan: sitä pitäisi siis kasvattaa ja ylläpitää kurssin aikana. Tehtävien ei tulisi olla liian haastavia tai helppoja, vaan juuri sopivia. Massakursseilla yksilölliset erot ovat kenties vaikea ottaa huomioon.

Ohjelmoinnin opetuksessa opiskelijoiden motivaation syitä on tutkinut esimerkiksi Jenkins (2001). Tutkimuksessa hän jakoi motivaation lähteet kahdeksaan eri osaan. Tutkimuksen mukaan ohjelmoinnin opiskelijoiden motivaatio oli suurella osalla tulevaisuuden edun saamiseen, kuten hyväpalkkainen työ. Toisena motivaattorina nousi vahvasti esille tarve oppia. On huomattavaa, että opiskelijat antoivat vain vähän painoarvoa ainoastaan kurssin läpäisemiselle.

Opiskelijoiden motivaatioita on hankala suoranaisesti mitata. On mahdollista tutkia ja tarkastella oppilaan käytöstä ja näin koettaa päätellä syitä sen takana, mutta varmaksi ei niitä voida todeta. Motivaatiota voidaan tarkastella esimerkiksi odotus-arvo -teorian (eng. expectancy-value theory) (Wigfield ja Eccles 2000) kautta, eli opiskelijoiden tulee odottaa onnistuvansa tietyssä tehtävässä tai kurssissa, ja heidän tulee kokea saavutus arvokkaana ja tarpeellisena. Jos jompikumpi oletus on ei täyty tai sillä ei opiskelijasta ole väliä, opiskelijan into tippuu hyvinkin nopeasti.

Motivaatio voidaan jakaa neljään kategoriaan: ulkoinen motivaatio (extrinsic), joka perustuu tulevaan uraan ja palkintoon, minkä kurssin suorittamisesta saa. Sosiaalinen (social) motivaatio, missä päämotivaattori on saada hyväksyntää joltain kolmannelta osapuolelta. Saa-

vutusmotivaatio (achievement), jossa pääpaino on vain halussa pärjätä hyvin henkilökohtaisen mieltymyksen takia sekä sisäinen (intrisic) motivaatio, jossa motivaatio lähtee sisäistä kiinnostuksesta ja innostuksesta tietojenkäsittelyyn. Sisäisen motivaation katsotaan olevan yleisesti pitkäaikaisempi ja luonteeltaan pysyvämpi verrattuna esimerkiksi ulkoiseen motivaatioon. Yllämainitut kategoriat ovat ensisijaisia, mutta eivät toisiaan poissulkevia: yksilön päätöksiin vaikuttaa monet syyt, ja opiskelija painottaa yllä olevia motivaation lähteitä yksilöllisesti. Tämä jaottelu pohjautuu Jenkinsin tutkimuksessa käytettyyn jakoon (Jenkins 2001).

Pitkäaikainen motivaatio syntyy sisäisestä kiinnostuksesta ja omakohtaisesta mielenkiinnosta aihetta kohtaan (Biggs ja Tang 2011). Valitettavasti kiinnostuksen syntyminen ei aina takaa sen pysyvyyttä, ja motivaation synnyttäminen tai ylläpitäminen voi olla myös työlästä. On hyvä huomata, että kiinnostus ei ole pysyvä asia: se itseasiassa muokkautuu ihmisen ja ympäristön yhteisvaikutuksesta, ja opetuksessa opettajan tulisi olla tästä tietoinen (Lakanen 2016, s. 15).

Opiskeluun sitoutuneet opiskelijat saavat myös motivaation lisäksi muita hyötyjä. Sitoutuneilla ja aktiivisilla opiskelijoilla on tunne siitä, että he kuuluvat opiskeluyhteisöön, ja tämä sitoutuneisuus on linkitetty myös osoittavan positiivisia vaikutuksia opintojen loppuunsuorittamiseen. Toisaalta epäaktiivisuus ja osallistumattomuus näyttäisivät vaikuttavan akateemiseen suoriutumiseen negatiivisesti. Kiinnostuksella näyttäisi siis olevan roolinsa opintojen loppuunvientiin, mutta suoraa korrelaatiota väitteelle ei ole löydetty (Lakanen 2016, s. 20).

2.3 Oppimisen haasteet aloittelijoille ohjelmoinnissa

Sorva, Karavirta ja Malmi (2013) ovat koostaneet seuraavat yleiset haasteet ohjelmoinnin aloittelijoille.

Ohjelmoinnin staattinen hahmotus: aloittelijat saattavat mieltää ohjelmoinnin opettelun tarkoittavan pelkästään koodin tuottamista tietyllä, spesifillä ohjelmointikielellä, vaikkakin ohjelmoinnin opetus itsessään pyrkii antamaan opiskelijoille tietoa ja taitoa suunnitella tietokoneiden toimintaa sekä ohjelmien suorittamia dynaamisia prosesseja.

Haasteet tietokoneen ymmärtämisessä: niin sanottu ”The Notional machine” eli ajatus siitä, miten tietokone toimii ja toiminnan ymmärtäminen on myös osoittautunut hankalaksi. Nämä mentaaliset mallit tietokoneesta ja sen toiminnasta tulisi olla täsmälliset ja toimintavarmat, mutta intuitiivinen ymmärrys tietokoneesta saattaa olla epätäydellinen ja moni saattaa vain arvailla, miksi jokin toimii kuten toimii.

Ohjelmoinnin perusteiden väärinymmärrys: Monet aloittelevien ohjelmoijien väärinymmärrykset koskevat piilotettua ajonaikaista tietokoneen toimintaa. Koska taustalla tapahtuvat asiat ovat piilotettuja, ja opiskelijalle ei välttämättä ole selvää kuvaa mitä koneen alla tapahtuu, opiskelijan intuitio luo väärinymmärryksiä käsiteltävän aiheen toiminnasta. Väärinymmärrys johtaa taas käsitykseen, että asia on hankala tai vaikea opettaa, sillä väärinymmärrettyä asiaa on hankala itse korjata. Monet näistä helposti väärinymmärretyistä konsepteista liittyy esimerkiksi viitteisiin, referensseihin, objektin tai tietokoneen tilaan, rekursioon taikka automaattiseen päivitykseen ohjelmakoodissa.

Hankaluudet ohjelman jäljittämisessä ja tilan havainnoinnissa: Ohjelmien jäljitys askel askeleelta on myös ollut hankalaa aloitteleville ohjelmoijille. Kun ohjelmaa jäljitetään askel askeleelta, on tärkeää tietää missä tilassa ohjelma on. Ohjelman tila on hyvin arkipäiväinen ja luonnollinen ilmiö tietojenkäsittelytieteessä, mutta se on ajatuksena ja konseptina hyvin uusi aloitteleville ohjelmoijille, joten sen omaksumiseen ja sujuvaan käyttöön menee oma aikansa. Tämä vaatii myös opiskelijan mielessä olevan tietokoneen toiminnan mallin muokkaamista, erilaisten ohjelman tilojen hyväksymistä ja ymmärtämistä. Jos opiskelijan mielessä oleva malli ei vastaa todellisuutta, saattaa se helposti johtaa väärinymmärryksiin sekä arvauksiin, jotka joko ovat totta vahvistaen väärää mallia tai epätosia, jolloin turhautuneisuus aiheeseen saattaa kasvaa. Ideaalitilanteessa tästä toki opiskelija huomaa olevansa väärässä, ja pyrkii korjaamaan ajattelunsa vinoumat. On kuitenkin huomattava, että oman ajattelun vinoumia on hankala huomata saatikka korjata.

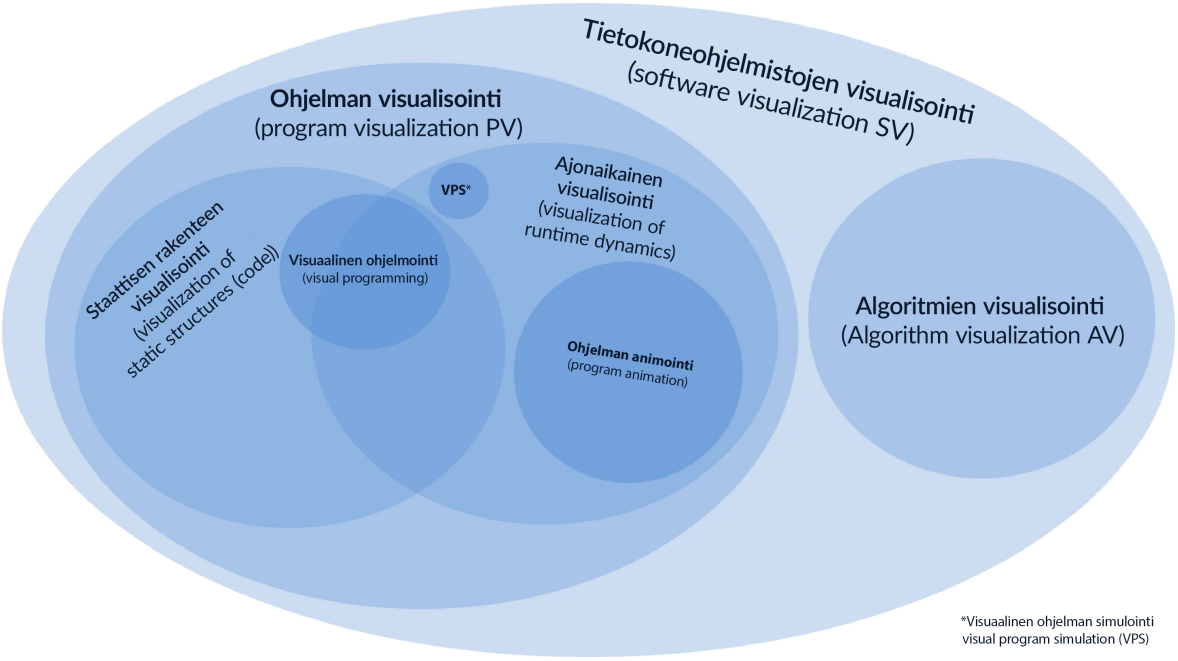
Opiskelijoiden erilaiset piirteet vaikuttavat myös haasteeksi koettuihin asioihin. Lähestymistavat oppimiseen ja oppimistavoitteet muokkaavat oppimisorientaatiota: innostuneet ja hyvät akateemiset taidot omaavat opiskelijat tarttuvat hanakammin opettajan jakamaan tietoon ja yhdistelevät niitä aktiivisesti aikaisemmin opittuun, kun taas hieman passiivisemmat opiskelijat saattavat keskittyä muihin asioihin, jolloin opiskelusta tulee enemmänkin pintapuolista

ja he turvautuvat vain muistamaan kaikista keskeisimmät aihealueet, mutta yhdistäminen aikaisempaan tietoon saattaa jäädä kehnoksi.

3 Opetus- ja motivointityökalut

Seuraavissa alaluvuissa esitellään ja tutkitaan erilaisia työkaluja, joilla tuetaan opetusta sekä oppimista varsinkin motivaation kannalta aktivoimalla ja osallistamalla. Keskiössä ovat visualisointityökalut, kontekstualisointi ja teemoittaminen sekä osaamismerkit.

3.1 Visualisointityökalut



Kuvio 1. Sorva, Karavirta ja Malmi (2013):n kuvasta mukailtu suomenkielinen versio visualisointityökalujen jaottelusta.

Sorva, Karavirta ja Malmi (2013) jakaa ohjelmistojen visualisoinnin kahteen isompaan kokonaisuuteen: algoritmien visualisointiin sekä ohjelman visualisointiin 1. Ohjelmien visualisointi jakautuu edelleen kahteen: staattiseen rakenteen visualisointiin ja ajonaikaiseen visualisointiin ja ohjelman suoritukseen. Ajonaikaiseen suoritukseen voidaan luokitella esimerkiksi ohjelman animointi, joka tarkoittaa esimerkiksi ohjelman dynaamista visualisointia ohjelman ajon aikana. Visuaaliset debuggerit ovat yksi esimerkki. Staattisella koodilla taas tarkoitetaan lähdekoodin rakennetta sekä lähdekoodin riippuvaisuuksia, esimerkiksi luokat lähdekoodin sisällä tai ohjelman ajamiseen tarvittavat ulkopuoliset tiedostot.

Visualisointityökaluja voidaan hyödyntää erityisesti erilaisten tietojenkäsittelytieteiden konseptien ja dynaamisten tietorakenteiden opetukseen sekä ohjelman toiminnan havainnollistamiseen. Visualisointityökaluilla voidaan myös pienentää aloittelevien ohjelmoijien kokemia haasteita, sillä suuri osa haasteista liittyy opiskelijoiden ajatuksiin ja sisäisiin malleihin tietokoneiden toiminnasta, kuten kappaleessa 2.3 käydään läpi. Perinteisesti visualisoinnin avulla opettajat täydentävät opetustaan, helpottaen aiheen seuraamista sekä kuvantaen omaa ajatusmaailmaansa opiskelijoille. On kuitenkin huomattava, että dynaamisia rakenteita on hankala kuvata kuvien avulla, jotka ovat staattisia.

Yleisen käsityksen mukaan visualisointi auttaa tietorakenteiden ymmärtämisessä, mutta tutkimusten mukaan pelkällä passiivisella visualisoinnilla ei ole niinkään vaikutusta oppilaiden aktiivisuuteen tai osallisuuteen (Naps ym. 2002). Passiivisella visualisoinnilla tarkoitetaan tässä sellaista tapaa seurata opetusta tai visualisointia, missä opiskelijaa ei velvoiteta mitenkään pohtimaan käsillä olevan aiheen, esimerkiksi tietorakenteen, seuraavaa tilaa tai tapahtumaa: tällöin opiskelija saattaa turvautua vain ulkoaopetteluun, eikä hänen ymmärryksensä aiheesta lisäänny. Ilman visualisointityökaluja on kuitenkin täysin mahdollista oppia, vaikkakin osallistuminen ja sitoutuminen onkin tällöin vähäisempää. Oppiminen ei aina edellytä kiinnostusta aiheeseen.

Monet opettajat käyttävät erilaisia tapoja ja ohjelmia kuvantaakseen ja mallintaakseen eri osia ohjelmissa tai rakenteissa. Esimerkiksi ohjelman ajonaikaisen toiminnan selittämiseen käytetään avuksi debuggerien lisäksi kutsuhierarkioiden piirtämistä. Ohjelmakoodin rakenteen mallintaminen, metodien ja suhteellisuuden esilletuominen sekä kutsuhierarkian selventäminen ovat arkipäiväisiä esimerkkejä ohjelmistojen visualisoinnista. Naps ym. (2002) kyselyssä opettajien mielestä visualisoinnista saatuja hyötyjä oli monia: 86% vastanneista oli sitä mieltä, että visualisointi paransi opiskelijoiden osallisuutta luentojen aikana. Myös muita positiivisesti osallisuuteen vaikuttavia tekijöitä tuotiin esille. Oppilaat kokivat luennot hauskemmiksi sekä se helpotti opiskelijoiden väärinymmärryksien huomaamista, sillä nyt opettajalla ja oppilaalla oli tehokas ilmaisutapa keskustella käsillä olevasta aiheesta tarkemmin.

Aktivoimalla opiskelijat visualisoinnin avulla passiivisen informaation toistamisen tai kuuntelemisen lisäksi on siis positiivisia vaikutuksia opiskelijoiden osallisuuteen, ja ne tukevat

myös pidempiaikaista kiinnostusta ohjelmointiin. Opiskelijat hyötyvät myös itse piirtämällä tai mallintamalla omaa käsitystään esimerkiksi tietyn algoritmin toiminnasta, sillä silloin aukot omassa ajattelussa tulevat paremmin esille, niin sanotusti visualisoiduksi.

Algoritmien visualisoiminen kurssitehtävänä on koettu myös motivoivaksi ja opettavaiseksi. Oppilaat nauttivat nähdessään omien ratkaisujensa tulokset nopeasti ja helposti. Opetettavan asian visualisoiminen syö kuitenkin opetusaikaa paljon verrattuna kevyempiin tehtäviin, ja monella kurssilla ei ole tarpeeksi aikaa uhrattavaksi visualisointiin kuluvan ajan takia. Myös opettajat kokevat visualisoinnin tehokkaaksi, mutta visualisoinnin integrointi oppitunteihin sekä kurssiin tuntuu vievän liikaa aikaa (Naps ym. 2002).

3.2 Teemoittaminen ja kontekstualisointi

Ohjelmoinnin opetuksen tavoitteena on usein opettaa yleisiä ja abstrakteja ratkaisumalleja, joita voi soveltaa moneen eri tilanteeseen ja ongelmaan. Siksi opiskelijoiden joskus hankala löytää omakohtaista tarttumapintaa aiheeseen, jolloin opiskelijan sitoutuminen aiheeseen laskee. Opiskelijat myös käyttävät vähemmän aikaa ja panostavat kurssiin vähemmän, jos opetettava aihe ei ole heidän mielestään ajankohtainen tai hyödyllinen.(Lukkarinen ja Sorva 2016) Tämän takia ohjelmointikurssien kontekstualisointi ja aihealueiden tuominen lähelle opiskelijoita on tärkeää: kun aihe on mielenkiintoinen ja inspiroiva, siihen uppoutuminen on paljon luontevampaa. Aktivoimalla ja osallistamalla kurssin sisältöihin myös syvempi osaaminen ja ymmärrys kasvaa, ja näin ollen vältetään vain pintapuolista oppimista ja ulkoaopettelua.

Kontekstualisoinnin teemoittaminen kurssilaisille voi osoittautua hankalaksi: kasvavat sisäänottomäärät ja opiskelijajoukkojen sisäinen erilaisuus tuovat hyvin paljon monipuolisuutta peruskurssille. Jos aikaisemmin on voitu sanoa, että yleisemmin tietojenkäsittelyn opiskelijat innostuvat tietynlaisista teemoista kurssilla, niin enää se ei välttämättä ole totta. Liian raskaalla kontekstualisoinnilla riskeerataan myös ohjelmoinnin opetuksen päätavoite: saada opiskelijalle taitoja, joita he voisivat hyödyntää mahdollisimman monissa erilaisissa ongelmissa ratkaisun saavuttamiseksi. Jos kontekstualisointi on liian aihekohtaista ja spesifiä, taidot eivät välttämättä siirrykään oikean maailman ongelmiin (Lukkarinen ja Sorva 2016).

Opiskelijat kaipaavat oikeita, tosielämän esimerkkejä sekä projekteja - jos tehtävän koetaan vain tehtäväksi ilman suurempaa hyötyä, se nähdään vain suoritettava pahana eikä motivoivana. Lukkarinen ja Sorva (2016) ovat listanneet yleisiä teemoja ohjelmoinnin opetukseen, joiden ympärille mahdolliset kontekstualisoinnit ovat rakentuneet:

- Oppilaiden arkipäiväiset kokemukset tietotekniikan saralla, kuten sosiaalinen media tai mobiililaitteet
- Oikean maailman data, kuten pandemioiden tartuntalukujen käsittely, maantieteellinen data tai vaalitulokset
- Robotit
- Tietokonepelit
- Luonnontieteelliset sovellukset, kuten fysiikka, matematiikka, maantiede tai biologia
- Tietojenkäsittelyn alakategoriat itsessään
- Oikean maailman ongelmat, joissa ammattilaiset hyödyntävät taitojaan
- Muut ammattialat, kuten tutkimus
- Multimedian konteksti ja käyttö

Kurssien kontekstualisointia voidaan soveltaa esimerkiksi kurssitehtäviin tai harjoitustyöhön. Viikoittaiset tehtävät voivat liittyä vaikkapa tiettyyn oikean maailman ongelmaan, joka tulee ratkaista kurssilla annetuin työkaluin. Toisaalta koko kurssi voidaan suorittaa tietyssä kontekstissa tai teemassa. Esimerkiksi oman pelin tekeminen, samalla oppien ohjelmoinnin perusteet ja käytännön. Innostavat teemat sekä mielenkiintoiset tehtävät ylläpitävät mielenkiintoa ja sitoutumista oppimiseen, jos aihe muutoin ei niin kiinnostava olekaan. Teoria niin ikään naamioidaan hauskan tehtävän päälle.

Kuvio 2. Esimerkkikuva badgesta (motivointityökalu)

3.3 Osaamismerkit

Osaamismerkki (eng. achievement badge) on yleensä graafinen ikoni, jonka saa merkiksi jonkun tietyn asian saavuttamisesta, esimerkiksi tehdystä tehtävästä tai saavutetusta tarkistuspisteestä. 2 on yksinkertainen esimerkki tälläisestä ikonista. Saavutetulla merkillä ei ole käytännössä esimerkiksi rahallista arvoa, mutta ne voivat toimia todisteena tietystä osaamistasosta. Osaamismerkit ovat myös hyvin kontekstiriippuvaisia, varsinkin jos ne ovat vain tietyllä kurssilla käytössä.

Osaamismerkit ovat eräänlainen pelillistämisen ilmenemismuoto. Pelillistäminen määritellään erilaisten pelielementtien ja tekniikoiden käytöksi ei-pelien konteksteissa (Deterding ym. 2011). Pelillistämisellä opetuskontekstissa pyritään lisäämään opiskelijoiden motivaatiota ja osallisuutta sekä sitoutuneisuutta tiettyyn tehtävään tai kurssiin. Osaamismerkkien saaminen tietystä osa-alueen tehtävän suorittamisesta kielii siitä, että opiskelija on saavuttanut tarvittavat taidot sen läpäisyyn tai saavuttanut tietyn pisteen kurssista, ja yleensä nämä kriteerit ovat myös selkeästi esitettyinä opiskelijoille.

Osaamismerkit ja erilaiset osaamisen tunnustusmerkit antavat uskoa omaan asiantuntijuuteen, toimijuuteen ja minäkuvaan, ja ovat eräänläinen ulkoinen palkkio. Osaamismerkit, jos ne ovat julkisia esimerkiksi opiskelijan profiilissa, toimivat myös saavutuksina muiden silmissä. Niiden tavoitteleminen lisää osallisuutta ja sitoutumista kurssiin. Tämä ei kuitenkaan toteudu jokaisen opiskelijan kohdalla, ja riippuu opiskelijan asenteista kokeeko hän osaamis-

merkit positiivisena vai negatiivisena (Hakulinen, Auvinen ja Korhonen 2015).

Hakulinen, Auvinen ja Korhonen (2015) kertovat, että osaamismerkkien hyödyntäminen opetuksessa muokkaa opiskelijoiden käytöstä kurssin aikana. Kun osaamismerkkien saamisen kriteerit ja tavoitteet ovat tiedossa, opiskelijat koettivat saavuttaa niitä aktiivisesti. Vaikeampien merkkien saavuttaminen koettiin palkitsevaksi, ja opiskelijat kokivat merkkien saavuttamisen olleen motivoivaa ja innostavaa.

Osaamismerkkien myöntämisen ei tulisi olla toisten merkkien poissulkemista: kriteerit niiden myöntämiselle ja saavuttamiselle tulisi olla mahdollista ilman, että jonkun toisen merkin saaminen muuttuisi mahdottomaksi. Jos merkin saavuttaminen on mahdotonta, opiskelijan into sen oppimiseen saattaa kadota, vaikka aihe olisikin tärkeä. Myös merkkien saavuttamisen malli vaihtelee opiskelijoiden välillä: erilaiset pohjatiedot ja motivaation syyt näyttivät olevan eräs tekijä. Osaamismerkkejä on myös kritisoitu siitä, että ne saattavat kannustaa enemmänkin ulkoiseen motivaatioon kuin sisäiseen (Abramovich, Schunn ja Higashi 2013), (Hakulinen, Auvinen ja Korhonen 2015).

Osaamismerkkejä laajemmin oppimisprosessissa hyödyntävää mallia kutsutaan osaamismerkein ohjautuvaksi oppimiseksi (eng. Digital Open Badge-driven Learning).

4 Yhteenveto

Motivaatiota voi kasvattaa, ja yleisin toimivaksi todettu metodi on saada opiskelijat aktiivisiksi osallisiksi kurssin aikana. Opettajan tulisi saada opiskelijat ohjelmoinnissa sitoutumaan tehtäviin ja aiheeseen, ja kenties kääntämään opiskelijoiden ajatusmaailma pelkästä läpipääsystä ja opintoppisteiden saavuttamisesta kohti oikeaa mielenkiintoa ja innostusta aiheeseen. Osallisuutta pyritään tukemaan opetuksessa monin eri tavoin. Kiinnostavat ja ajankohtaiset teemat, kommunikaatio, osaamismerkkien myöntäminen sekä palaute ovat todettu toimiviksi motivaation ylläpitäjiksi. Opiskelijoiden tulisi myös pyrkiä aktiivisuuteen, sekä tuoda esille heidän ajattelutapaansa ja sitä, ollaanko aiheesta oikeasti motivoituneita ja kiinnostuneita.

Aktivoiva ja osallistava opetus, jossa opettaja ei ainoastaan ”jaa tietoa” vaan haastaa opiskelijat olemaan vastuussa itse aktiivisesta pohdinnasta helpottaa myös opettajan vastuuta. Vuoropuhelussa esille tulevat ongelmat opiskelijoiden ymmärryksessä tulevat nopeammin esille, ja niihin on helppo tarjota korjausta.

Motivointityökalut, jotka tutkielmassa esiteltiin, näyttävät tukevan aktiivista osallisuutta hyvin ja motivoimaan erilaisia opiskelijoita. Visualisointityökalut selventävät erilaisia konsepteja ja ajattelumalleja sekä rakentavat tarkempaa kuvaa tietokoneen, lähdekoodin ja ohjelmistojen toiminnasta, ja ne tarjoavat mielenkiintoista seurattavaa opetustilaisuuteen pelkän puheen lisäksi. Lisäksi opettajan ja opiskelijoiden ajattelun läpikäynti visualisoinnin avulla antaa mahdollisuuden huomata oman ajattelun vinoutumat tehokkaasti.

Teemoittaminen ja konteksti tuo ohjelmoinnin lähemmäksi ja esittelee arkipäiväisiä ongelmia, johon ohjelmointi on ratkaisu. Kontekstualisointi myös ylläpitää motivaatiota tuomalla opetettavat aiheet ajankohtaisiksi ja oppilaita koskettavaksi.

Osaamismerkit antavat konkreettisia tavoitteita pitkin kurssia opiskelijoille, joita on helppo seurata. Kurssin osaamistavoitteet on osaamismerkkien avulla pilkottu mukaviksi välietapeiksi, joiden saavuttaminen edistää koko kokonaisuutta samalla antaen onnistumisen tunteita koko kurssin ajan. Ne myös pitävät opiskelijat pääosin sitoutuneina kurssiin ja sen materiaaliin koko suorituksen ajan.

Nämä esitellyt motivointityökalut edistävät siis osallisuutta ja motivaatiota eri tavoin onnistuneesti, sekä osa pienentää myös aloittelevien ohjelmoijien kokemia haasteita omista taidoistaan. Haasteiden ylipääseminen nostaa opiskelijan käsitystä omista taidoistaan, ja näin ollen hänen odotuksensa nousee positiivisesti , esimerkiksi kurssin läpäisyä kohtaan.

Ohjelmoinnista potentiaalisesti kiinnostunut joukko on myös laajentunut viime vuosien aikana, mikä tarkoittaa laajenevaa motivaation ja kiinnostuksen kohteiden kirjoa: tämä luo niin haasteita kuin myös mahdollisuuksia esimerkiksi mielenkiintoisten kurssisisältöjen luomiseen sekä kehittämiseen. Tämä katsaus tuntui olevan vasta pintaraapaisu ohjelmoinnin opetukseen ja sen motivaation ylläpitämiseen sekä ylläpitämisen työkaluihin, ja kirjallisuutta aiheesta löytyi runsaasti ja monipuolisesti. Tutkielma antoi aihetta lähteä kartoittamaan muita motivointikeinoja opetukseen liittyen. Pelillisyyteen liittyvät motivointityökalut opetuksessa olivat varsinkin esillä osaamismerkkien ja teemoittamisen yhteydessä. Osaamismerkein ohjautuva oppiminen oli uusi käsite, joka vaikutti mielenkiintoiselta ja tutustumisen arvoiselta. Myös visualisointiin liittyvistä haasteista löytyisi varmasti aihe jatkotutkimukselle.

Lähteet

Abramovich, Samuel, Christian Schunn ja Ross Mitsuo Higashi. 2013. “Are Badges Useful in Education?: It Depends upon the Type of Badge and Expertise of Learner”. *Educational Technology Research and Development* 61 (2): 217–232. ISSN: 1042-1629. https://www.learntechlib.org/p/114262.

Biggs, John B., ja Catherine So-kum Tang. 2011. *Teaching for Quality Learning at University : What the Student Does.* Nide 4th ed. McGraw-Hill Education. ISBN: 9780335242757.

Dale, Nell. 2005. “Content and Emphasis in CS1”. *SIGCSE Bull.* (New York, NY, USA) 37 (4): 69–73. ISSN: 0097-8418. doi:10.1145/1113847.1113880.

Deterding, Sebastian, Dan Dixon, Rilla Khaled ja Lennart Nacke. 2011. “From Game Design Elements to Gamefulness: Defining “Gamification””. Teoksessa *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments,* 9–15. MindTrek ’11. Tampere, Finland: Association for Computing Machinery. ISBN: 9781450308168. doi:10.1145/2181037.2181040.

Hakulinen, Lasse, Tapio Auvinen ja Ari Korhonen. 2015. “The Effect of Achievement Badges on Students’ Behavior: An Empirical Study in a University-Level Computer Science Course”. *International Journal of Emerging Technologies in Learning (iJET)* (Kassel, Germany) 10 (1): 18–29. ISSN: 1863-0383. https://www.learntechlib.org/p/151161.

Jenkins, Tony. 2001. “The Motivation of Students of Programming”. *SIGCSE Bull.* (New York, NY, USA) 33 (3): 53–56. ISSN: 0097-8418. doi:10.1145/507758.377472.

Lakanen, Antti-Jussi. 2016. “On the impact of computer science outreach events on K-12 students”. Viitattu 29. huhtikuuta 2020. https://jyx.jyu.fi/handle/123456789/49729.

Lukkarinen, Aleksi, ja Juha Sorva. 2016. “Classifying the Tools of Contextualized Programming Education and Forms of Media Computation”. Teoksessa *Proceedings of the 16th Koli Calling International Conference on Computing Education Research,* 51–60. Koli Calling ’16. Koli, Finland: Association for Computing Machinery. ISBN: 9781450347709. doi:10.1145/2999541.2999551.

Malmi, Lauri, Judy Sheard, Päivi Kinnunen, Simon ja Jane Sinclair. 2019. “Computing Education Theories: What Are They and How Are They Used?”: 187–197. doi:10.1145/3291279.3339409.

Naps, Thomas L., Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen ym. 2002. “Exploring the Role of Visualization and Engagement in Computer Science Education”. *SIGCSE Bull.* (New York, NY, USA) 35 (2): 131–152. ISSN: 0097-8418. doi:10.1145/782941.782998.

Robins, Anthony, Janet Rountree ja Nathan Rountree. 2003. “Learning and Teaching Programming: A Review and Discussion”. *Computer Science Education* 13 (2): 137–172. doi:10.1076/csed.13.2.137.14200.

Sheard, Judy, S Simon, Margaret Hamilton ja Jan Lönnberg. 2009. “Analysis of research into the teaching and learning of programming”. 8:175–191. doi:10.1145/1584322.1584334.

Sorva, Juha, Ville Karavirta ja Lauri Malmi. 2013. “A Review of Generic Program Visualization Systems for Introductory Programming Education”. *ACM Trans. Comput. Educ.* (New York, NY, USA) 13 (4). doi:10.1145/2490822.

Watson, Christopher, ja Frederick W.B. Li. 2014. “Failure Rates in Introductory Programming Revisited”. Teoksessa *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education,* 39–44. ITiCSE ’14. Uppsala, Sweden: Association for Computing Machinery. ISBN: 9781450328333. doi:10.1145/2591708.2591749.

Wigfield, Allan, ja Jacquelynne S. Eccles. 2000. “Expectancy–Value Theory of Achievement Motivation”. *Contemporary Educational Psychology* 25 (1): 68–81. ISSN: 0361-476X. doi:https://doi.org/10.1006/ceps.1999.1015.