

# **Kognitiivinen oppipoikamalli tietojenkäsittelytieteessä**

Pessi Moilanen

Kandidaatintutkielma  
HELSINGIN YLIOPISTO  
Tietojenkäsittelytieteen laitos

Helsinki, 21. helmikuuta 2014

Tiedekunta — Fakultet — Faculty		Laitos — Institution — Department	
Matemaattis-luonnontieteellinen		Tietojenkäsittelytieteen laitos	
Tekijä — Författare — Author			
Pessi Moilanen			
Työn nimi — Arbetets titel — Title			
Kognitiivinen oppipoikamalli tietojenkäsittelytieteessä			
Oppiaine — Läroämne — Subject			
Tietojenkäsittelytiede			
Työn laji — Arbetets art — Level		Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Kandidaatintutkielma		21. helmikuuta 2014	6
Tiivistelmä — Referat — Abstract			
:D			
Avainsanat — Nyckelord — Keywords			
avainsana 1, avainsana 2, avainsana 3			
Säilytyspaikka — Förvaringsställe — Where deposited			
Muita tietoja — Övriga uppgifter — Additional information			

## Sisältö

<b>1</b>	<b>Johdanto</b>	<b>1</b>
<b>2</b>	<b>Kognitiivinen kisällioppiminen</b>	<b>2</b>
<b>3</b>	<b>Ohjelmoinnin opetus</b>	<b>4</b>
<b>4</b>	<b>Opettamistapojen soveltaminen käytännössä</b>	<b>4</b>
4.1	Muotoillut esimerkit . . . . .	4
4.2	Kysymysten muodostaminen . . . . .	5
4.3	Valmentaminen käytännössä . . . . .	5
<b>5</b>	<b>Yhteenveto</b>	<b>5</b>
	<b>Lähteet</b>	<b>5</b>

# 1 Johdanto

Ihmiset oppivat asioita seuraamalla ja matkimalla toistensa toimintoja. Tiedon ja taidon eteenpäin siirtäminen jälkipolville on ollut avaimena ihmiskunnan kehittymiselle ja selviytymiselle. Siirrettävän tiedon määrä kasvaa vuosi vuodelta ja tiedon ylläpitoon ja omaksumiseen kuluu enemmän aikaa. Ihmisellä on rajallinen aika oppia asioita, joten oppiminen on tehtävä mahdollisimman tehokkaasti ajankäytön suhteen.

Opettamiseen on monia erilaisia tekniikoita. Tässä aineessa käymme läpi kisällioppimisen ja kognitiivisen kisällioppimisen perusideat, kognitiivisen kisällioppimisen erilaiset menetelmät ja kuinka niitä pystyy soveltamaan käytännössä. Samassa yhteydessä käsittelemme myös muutaman tilanteen joissa kognitiivista kisällioppimista on käytetty tietojenkäsittelytieteen opettamisessa.

Mestari on henkilö, joka on alansa ammattilainen eli erittäin kokenut jossakin tietyssä asiassa. Mestarilla voi olla oppipoikia, jotka hänen opastuksellaan pyrkivät saamaan kokemusta ja hiljaista tietoa. Oppipoika eli kisälli on henkilö, joka pyrkii oppimaan mestarilta. Mestari ohjeistaa kisälliä työn suorituksessa soveltaen erilaisia opetustekniikoita. Tarpeeksi harjoiteltuaan ja opittuaan kisällistä tulee mestari.

Perinteisessä kisällioppimisessa (apprenticeship) on mestarin suorittaman työn eri vaiheita konkreettisesti näkyvissä oppipojalle. Oppipojalle yksi oppimistavoista on vierestä tarkkaileminen, kun mestari suorittaa jotakin työtehtävää. Työtehtävä voi olla esimerkiksi haarniskan takominen, kakun leipominen tai paidan kutominen. Mestari pystyy antamaan työnvaiheissa erilaisia neuvoja, miten ja miksi asiat tapahtuvat. Oppilas pystyy tekemään näistä vaiheista muistiinpanoja, joita hän voi hyödyntää harjoittelussaan.

Kisälli voi saada suoritettavakseen mestarin avustuksella pienempiä tehtäviä, jotka ovat sopivia kisällin sen hetkiseksi taitotasolle. Syynä tähän on monissa tilanteissa raaka-aineiden suuri hinta, mikä ei jätä tilaa virheille. Täysin uusi kisälli tuskin saa tarpeeksi luottoa mestarilta suorittaakseen kalliin silkin leikkaamista. Pienistä tehtävistä kisälli kerryttää kokemusta. Kokemuksen karttuessa kisälli alkaa tekemään laajempia ja haastavampia tehtäviä.

\*Oppipoika voi oppia kuitenkin vain rajallisen määrän mestarin alaisena. Tarpeeksi opittuaan on hän valmis täysin itsenäiseen työskentelyyn. Alansa mestariksi päästäkseen on hänen saatava oman alansa killalta tunnustus mestarin taidoista. Tämä tunnustus vaatii työn esittämistä killalle, jonka perusteella arvioidaan onko kisälli valmis mestariksi.\*

## 2 Kognitiivinen kisällioppiminen

Kognitiivisessa kisällioppiminen (cognitive apprenticeship) on teoria prosessista, jossa mestari opettaa taitoa kisällille. Kognitiivisessa kisällioppimisessa käsitellään opettamista tilanteissa, joissa usein työvaiheet opetettavasta asiasta eivät ole konkreettisesti näkyvillä oppipojalle. Tällöisiä asioita voivat olla esimerkiksi yleinen ongelmanratkaisu, algoritmin valinta tai luetun ymmärtäminen. Kisällin on vaikea oppia mestarilta, jos ainoat näkyvät vaiheet ovat kaavojen pyörittelyt ja ratkaisun ilmestyminen paperille. Mestarin on saatava ajatuksensa näkyviin, jotta oppipoika hyötyy niistä. Toisinaan ajattelu on kuitenkin abstrahoitunut ja yhdistynyt muihin käsitteisiin, jotka eivät ole oppilaalle vielä tuttuja. Tällöin opettajan on selitettävä asia kisällille, tasolla mikä on kisällille ymmärrettävissä. Ajatusten näkyviin tuontiin on erilaisia tekniikoita, joita voidaan soveltaa tilanteesta riippuen. Kognitiivisessä kisällioppimisessa on samat periaatteet kuin perinteisessä kisällioppimisessa, poikkeavuutena on opetettavien asioiden luonne ja kuinka oppi saadaan perille oppilaalle.

Kognitiiviseen kisällioppimiseen on määritelty kuusi erilaista opetusmenetelmää. Opetusmenetelmät sulautuvat toisiinsa eli ne hyödyntävät ja ovat vahvasti sidoksissa toisiinsa.

Mallintamisessa (Modelling) työn suorittamisen eri vaiheet pyritään saamaan näkyviin oppilaalle. Työn suorittaa normaalisti opettaja tai muu oppilasta kokeneempi henkilö. Oppilas pyrkii saamaan kokemusta ja muodostamaan käsitteellisen mallin suoritetuista vaiheista. Oppilas näkee syy seuraussuhteen ja saa käsityksen miten suoritettut vaiheet johtavat onnistuneeseen lopputulokseen.

Kognitiivisessä kisällioppimisessa ongelmana on ajatustyön esille saanti oppilaalle, joten on pyrittävä käyttämään keinoja, joilla oppilas hahmottaa ne. Ongelmanratkaisuprosessia mallintaessa mestari kertoo kussakin työn vaiheessa ääneen mitä hän tekee jatkaksi, jolloin oppilaat voivat kirjoittaa ongelmanratkaisuprosessin vaiheet muistiin.

Valmentamisessa (Coaching) opettaja tarkkailee oppilaan työskentelyä ja antaa hänelle henkilökohtaisia neuvoja ja pystyy avustamaan oppilasta kriittisillä hetkillä. Opettaja pyrkii samaistumaan oppilaan osaamiseen ja pyrkii ohjaamaan tätä oikeaan suuntaan. Opettaja voi rakentaa oppilaalle tehtäviä, jotka ovat oppilaan osaamiselle sopivia.

Oppimisen oikea-aikainen tukeminen (scaffolding) käsittää erilaisia tekniikoita ja strategioita oppilaan oppimisen tukemiseen. Oppilaalle voidaan antaa tehtäviä, jotka sisältävät tekniikoita, joita oppilas ei entuudestaan osaa.

Opettajalle on tärkeää, että hän osaa arvioida oppilaan taitotasoa, kykyä oppia ja käytettävissä olevaa aikaa. Opettajan on tarkkailtava oppilaan työskentelyä varmistaakseen, että oppilas ei jää jumiin työssään, josta seuraa oppilaan turhautuminen ja motivaation menettäminen [1]. Opettajan on

annettava oppilaalle oikea-aikaisia neuvoja tehtävässä etenemiseen estääkseen jumiin jääminen. Opettaja antaa oppilaalle apua tehtävän vaiheissa, joita oppilas ei vielä itsenäisesti osaa suorittaa. Opettaja pystyy antamaan sekä suullista ja kirjoitettua palautetta oppilaalle, jonka perusteella oppilas pystyy parantamaan työskentelyään [1]. Annetun avun määrä ja neuvojen on oltava sopiva, ne eivät saa pilata tehtävän tuomaa oppimismahdollisuutta, eivätkä ne saa olla liian vähäistä.

Opettaja vähentää antamaansa tukea hiljalleen (fading), jättäen oppilaalle enemmän vastuuta. Nämä keinot kehittävät oppilaan itsetietoisuutta ja oppilas oppii korjaamaan itse virheitään, joidenka seurauksena hän ei tarvitse enää niin paljoa ulkopuolista apua.

Artikuloinnissa (articulation) oppilaiden on muutettava ajatuksensa, tietonsa ja ajatusprosessinsa sanoiksi. Artikuloinnin seurauksena oppilaat voivat vertailla paremmin omaa ajatteluaan opettajan ajatteluun ja opettaja pääsee käsiksi oppilaan ongelmanratkaisuprosessiin. Opettaja voi pyytää oppilaita puhumaan ääneen oppilaan suorittaessa jotakin tehtävää, näin oppilas joutuu muotoilemaan ajatuksensa sanoiksi.

Oppilaan voi laittaa pariohjelmointiin toisen henkilön kanssa, tässä tilanteessa oppilaan on kommunikoidava toisen osapuolen kanssa ja esitettävä tälle ajatuksensa.

Oppilaalle voidaan esittää muotoiltuja kysymyksiä. Kuinka-kysymysten sijaan oppilaille kannattaa esittää miksi-kysymyksiä. Oppilas voidaa pyytää ohjelmointiin ratkaisu johonkin tehtävään ja sen lisäksi kysyä miksi kyseinen koodi toimii. Oppilas joutuu tehtävän ratkaisun ohella miettimään omaa työskentelyään syvemmin. [3]

Peilaamisessa (reflection) oppilas vertailee työskentelyään opettajaan, kokeneempaan henkilöön tai toiseen oppilaaseen. Oppilas saa tätä kautta tietoa siitä, miten hänen työskentelynsä poikkeaa tämän toisen henkilön työskentelystä. Näin oppilas näkee mihin asioihin hänen on panostettava tullakseen paremmiksi. Oppilas voi muistella aiempaa työskentelyään ja miettiä mitä hän on oppinut. Peilaamisen seurauksena oppilaan itetietoisuus ja itsekriittisyys kehittyy ja hän pystyy vertailemaan ymmärrystään toisten ymmärrykseen.

Tutkimisessa (exploration) oppilas joutuu keksimään uusia tapoja, strategioita ja erilaisia keinoja lähestyä ongelmia. Oppilas joutuu kehittämään itse kysymyksiä ja ongelmia, joita hän pyrkii ratkaisemaan. Hän joutuu miettimään asioita eri näkökulmasta, kuin tilanteessa jossa hän saa tehtävän mestarilta. Muodostaessaan omia ongelmia kehittyy oppilaan taito määrittellä, käydä läpi ongelman vaatimat ominaisuudet. Samalla oppilaan taito itsenäiseen työskentelyyn kehittyy.

### 3 Ohjelmoinnin opetus

Kun ohjelmointia aletaan opettamaan täysin kokemattomalle, ei tavoitteena ole luoda oppilaita, jotka osaavat tehdä toimivia ohjelmia konemaisesti ilman syvempää ymmärrystä asiasta. Turhan usein opetuksessa keskitytään lopputulokseen. On pyrittävä luomaan oppilaita, jotka osaavat soveltaa ohjelmoinnin käsitteitä ja rakenteita ohjelmointiin liittyviin ongelmiin.

Jakaessa iso kokonaisuus pienempiin osiin saavat oppilaat niistä enemmän irti, koska he oppivat miksi jokaisella osalla tulee olemaan merkitys lopputuloksen kannalta.[3]

### 4 Opettamistapojen soveltaminen käytännössä

Kognitiivisen kisaopetuksen opetusmenetelmiä voidaan soveltaa käytännössä erilaisin tavoin. Käymme läpi muutamia tilanteita ja tapoja, kuinka menetelmiä on sovellettu käytännössä ja miten se on sujunut.

#### 4.1 Muotoillut esimerkit

Tutkimukset osoittavat, että opiskelijat keräävät tietoa mieluiten esimerkeistä, verrattuna muunlaisiin opiskelumateriaaleihin \*hae viite Instructional Design of a Programming Course... viite n.64\*. Opiskelijat oppivat myös enemmän tutkimalla esimerkkejä, kuin tekemällä samat tehtävät itse. He pystyvät tarkastelemaan ratkaisun erilaiset vaiheet ja muodostamaan niistä skeemoja.

Muotoillut esimerkit soveltuvat opiskelukohteisiin, joissa taidonhankkiminen on oleellista kuten musiikissa, shakissa ja ohjelmoinnissa \*hae viite Instructional Design of a Programming Course... viite n.2\*. Hunosti menestyvät oppilaat tarkastelevat monesti muotoillut esimerkit ainoastaan jos ne ovat tavanomaisista ongelmista, he voivat kokea nämä tehtävät pelottavina ja etäisinä. Joten jos esimerkki ei ole tällainen voi käydä niin, että pelkästään paremmat oppilaat katsovat ne läpi. \*hae viite Instructional Design of a Programming Course... viite n.13 ja [2]\*.

Muotoillut esimerkit auttavat skeemojen muodostamisessa ja oppimisen kehittämisessä. [2]

Esimerkkejä kannattaa olla samaan opetettavaan asiaan liittyen enemmän kuin yksi. Yhden esimerkin kannattaa kuvata yksinkertaista tilannetta, josta opetettavan asian perusidean ymmärtää. Toinen esimerkki puolestaan voi olla monimutkaisempi, kuin ensimmäinen. Oppilaat saavat näin enemmän irti opetettavasta asiasta, koska he saavat erilaisia näkökulmia \*hae viite Instructional Design of a Programming.. viite 54,22,25\*.

## 4.2 Kysymysten muodostaminen

Tämä aliluku pohjautuu artikkeliin. Kerromme kuinka kyseisessä artikkelissa käydään läpi miten muotoiltuja kysymyksiä pystyy käyttämään ohjelmoinnin opetuksessa.

Artikkelissa he tarkastelivat jonkin Yhdysvalloissa sijaitsevan yliopiston 7 eri CS0-, CS1- ja CS2-ohjelmointikurssien eli ohjelmoinnin peruskurssien opetusmateriaalejen sisältämien tehtävien rakennetta. Jokainen näistä kurseista arvosteltiin kurssikokeen perusteella, joten kurssien oppimistavoitteet pohjautuivat niihin.

Opetusmateriaalien tehtävistä noin viidesosa oli miksi-kysymyksiä ja loput kuinka-kysymyksiä. Kuinka-kysymyksissä opiskelija voi joutua kirjoittamaan toimivan koodin ohjelmointitehtävään, mutta miksi-kysymyksissä joutuu hän perustelemaan miksi koodi toimii kyseisessä tehtävässä. Kurssikokeissa puolestaan miksi-kysymyksiä oli vielä vähemmän, niitä oli 0%-15% välillä kokonaiskysymys määrästä.

Kuinka-kysymykset voivat luoda oppilaille harhaanjohtavan käsityksen millä tasolla asioita on tärkeää osata kokeessa. Oppilas opettelee muodostamaan vastaukset kysymyksiin ulkomuistista, ilman syvempää ymmärrystä miksi ja miten hänen vastauksensa oikeasti toimivat. Tämä kannustaa oppilaita ratkaisemaan ongelmia ymmärtämättä niitä, koska miksi nähdä ylimääräistä vaivaa ymmärryksen saamiseen, jos tehtävät on ratkaistavissa vähemmälläkin.

Miksi-kysymyksissä oppilas joutuu rakentamaan syvempää ymmärrystä opetettuun asiaan. Hän joutuu käsittelemään tehtävää syvemmin ja käymään läpi lopputulokseen johtavia vaiheita tarkemmin. Kun oppilas ymmärtää asian syvemmin pystyy hän perustelemaan vastauksiaan ja muodostamaan yhtenäisen konseptin lopputulokseen johtavista askeleista. Miksi-kysymyksiin tehtyjä vastauksia on kuitenkin haastavampi arvostella, koska vastaukset poikkeavat oppilaiden välillä ja heidän ajatusprosessinsa ovat yksilöllisiä.

Johtopäätöksinä he totesivat miksi-kysymyksiä olevan liian vähän. Heillä ei ole konkreettisia todisteita muotoiltujen kysymyksien toimivuudesta, mutta monet tekijät tukevat ideaa niiden toimivuudesta. Miksi-kysymyksiä tukevia tekijöitä löytyy kognitiivisen kisa-oppimisen ja oppimisen tilannesidonnaisuudesta.

## 4.3 Valmentaminen käytännössä

## 5 Yhteenveto

## Lähteet

- [1] Bareiss, Ray ja Radley, Martin: *Coaching via Cognitive Apprenticeship*. Teoksessa *Proceedings of the 41st ACM Technical Symposium on Compu-*



*ter Science Education*, SIGCSE '10, sivut 162–166, New York, NY, USA, 2010. ACM, ISBN 978-1-4503-0006-3. <http://doi.acm.org/10.1145/1734263.1734319>.

- [2] Caspersen, Michael E. ja Bennedsen, Jens: *Instructional Design of a Programming Course: A Learning Theoretic Approach*. Teoksessa *Proceedings of the Third International Workshop on Computing Education Research*, ICER '07, sivut 111–122, New York, NY, USA, 2007. ACM, ISBN 978-1-59593-841-1. <http://doi.acm.org/10.1145/1288580.1288595>.
- [3] Cutts, Quintin, Esper, Sarah, Fecho, Marlena, Foster, Stephen R. ja Simon, Beth: *The Abstraction Transition Taxonomy: Developing Desired Learning Outcomes Through the Lens of Situated Cognition*. Teoksessa *Proceedings of the Ninth Annual International Conference on International Computing Education Research*, ICER '12, sivut 63–70, New York, NY, USA, 2012. ACM, ISBN 978-1-4503-1604-0. <http://doi.acm.org/10.1145/2361276.2361290>.