Moduuli 8

Ohjelmoinnillista ajattelua ja monialaista opetusta tukevat digitaaliset ympäristöt

Tekijät:
Peter Larsson, Turun yliopisto (Suomi)
Ashok Veerasamy, Turun yliopisto (Suomi)

Arvioijat:
CARDET (Kypros)
Tallinnan yliopisto (Viro)
Vilnan Yliopisto (Liettua)

Ulkoiset arvioijat:
Dr. Filiz Kalelioğlu, Ankaran yliopisto (Turkki)
Andreas Mühling, Paderbornin yliopisto (Saksa)

Pilotointi:
CARDET (Kypros)
Vilnan yliopisto (Liettua)

Design (graafiset elementit):
Vaidotas Kinčius (Liettua)

Moduulin *kuvaus* on osa projektia “Future Teachers Education: Computational Thinking and STEAM” (TeaEdu4CT). Koordinaattori: Prof. Valentina Dagienė, Vilnan Yliopisto, Liettua. Osallistujat: Wienin teknillinen yliopisto (Itävalta), CARDET (Kypros), Tallinnan yliopisto (Viro), Turun yliopisto (Suomi), Paderbornin yliopoisto (Saksa), CESIE (Italia), Radboud Yliopisto (Alankomaat), KTH Kungliga Tekniska Högskolan (Ruotsi), Ankaran yliopisto (Turkki). Projekti on saanut rahoitusta Erasmus+ ohjelmasta KA2.

Sisältö

Tiivistelmä

Moduulin aiheena on ohjelmoinnillisen ajattelun yhdistämistä STEAM (Science/tiede, Technology/teknologia, Engineering/tekniikka, liberal Arts/ihmis- ja yhteiskuntatieteet ja Mathematics/matematiikka) -opetukseen tukevien oppimisympäristöjen luominen. Oppimisympäristöissä hyödynnetään opetusteknologioita, jotka tukevat ohjelmoinnillisen ajattelun soveltamista. STEAM-opetus yhdistää LUMA (luonnontiede, matematiikka ja teknologia; engl. STEM) -aineet ja ihmis- sekä yhteiskuntatieteisiin tavoitteena edistää tasa-arvoa ja tehdä LUMA-aineista helpommin lähestyttäviä. Ihmis- ja yhteiskuntatieteet tarjoavat LUMA-aineille kontekstin ja tukevat luovuutta, kun taas ohjelmoinnillinen ajattelu tarjoaa uuden menetelmän niiden oppimiseen. Ohjelmoinnillinen ajattelu tulkitaan moduulin kontekstissa yleiseksi asenteeksi ja taidoksi, joka tukee tietojenkäsittelytieteen menetelmien soveltamista tieteellisiin, teknisiin ja matemaattisiin aineisiin.

Tietojenkäsittelytieteen kolme näkökulmaa (matematiikka, tekniikka, ja tiede) yhdistettynä ohjelmoinnnilliseen ajatteluun muodostavat moduulin viitekehyksen. Viitekehys tukee opettajia opetusteknologioiden valinnassa ohjelmoinnillisen ajattelun ja STEAM-opetuksen yhdistämiseksi. Kurssin päättävässä projektissa tulevat opettajat muodostavat projektiryhmiä kehittämään STEAM-opetusta, jossa hyödynnetään ohjelmoinnillista ajattelua ja sitä tukevia oppimisympäristöjä.

Tämä moduuli on tarkoitettu tuleville LUMA-aineiden opettajille.

Kohderyhmä ja esitiedot

Kohderyhmänä ovat tulevat LUMA-aineiden opettajat. Moduulissa painotetaan monialaisuutta, mutta esitettyjä opetusmenetelmiä ja -teknologioita voidaan myös hyödyntää yksittäisen aineen opetuksessa.

Ennakkotiedoiksi riittää oman alan tai alojen tuntemus sekä pedagogiset opinnot.

Aikaisempaa ohjelmointikokemusta ei vaadita, mutta siitä on hyötyä.

Avainsanat

opetuksen suunnittelu, opetusteknologia, ohjelmoinnillinen ajattelu, LUMA-aineet, monialainen opetus.

Soveltuvat osaamismallit

Tämän moduulin sisältö tukee Opettajien Digitaalisen Osaamisen Viitekehystä (engl. Digital Competence Framework for Educators; DigCompEdu). Alla on lueteltu ne aiheet

viitekehyksestä, jotka sisältyvät moduuliin. Lisäksi kerrotaan miten aihetta käsitellään moduulissa.

1. Ammatillinen sitoutuneisuus	
1.3. Reflektointi	Omien ja opetusyhteisön digitaalisen pedagogiikan käytäntöjen arviointi, reflektointi ja kehittäminen.
	Moduuli tarjoaa selkeän viitekehyksen digitaalisen opetusteknologian hyödyntämisen mahdollisuuksien arvioimiseksi ohjelmoinnillista ajattelua ja LUMA(STEM)-aineita yhdistävässä opetuksessa.
2. Digitaaliset resurssit	
2.1. Digitaalisten resurssien valinta	Digitaalisten resurssien tunnistaminen, arviointi ja valinta opetuksen sekä oppimisen tukemiseksi. Resurssien käytön suunnittelussa on huomioitava oppimistavoite, konteksti, pedagoginen lähestymistapa, ja kohderyhmä.
	Opetukseen hyödynnettävissä olevia teknologioita on paljon. Valinnan tukemiseksi moduulissa painotetaan opetettavan aiheen keskeisten piirteiden tunnistamista, joiden perusteella voidaan valita oikeanlainen opetusteknologia.
2.2.Digitaalisten resurssien luonti ja muokkaaminen	Osaa muokata ja kehittää avoimilla lisensseillä tai muuten vapaasti käytettävissä olevia digitaalisia resursseja. Myös digitaalisten resurssien luonti, itsenäisesti tai yhteistyössä, kuuluu taitoihin. Huomioi digitaalisten resurssien ja niiden käytön suunnittelussa oppimistavoitteet, kontekstin, pedagogisen lähestymistavan ja kohderyhmän.
	Vaikka opetusteknologiat ovat jo olemassa, niin niitä on yleensä muokattava käyttötarkoitukseen sopivaksi. Tässä moduulissa opiskelijat oppivat suunnittelemaan teknologian käytön ennen toteutusta.
3. Opetus ja oppiminen	
3.1. Opetus	Osaa suunnittella, miten digitaalisia laitteita ja resursseja yhdistetään opetusprosessiin tehokkuuden lisäämiseksi. Pystyy asiantuntevasti toteuttamaan ja ohjamaan digitaalisia opetuskokeiluja. On kiinnostunut uusien opetusmuotojen ja pedagogisten menetelmien kokeilemisesta ja kehittämisestä.
	Onnistuakseen opetusteknologian on sovittava opetettavaan aiheen kontekstiin ja tuettava oppimisprosessia. Tämän moduulin käytännön osa sisältää opetusteknologian käytön suunnittelun.

3.4. Itseohjautuva oppiminen	Osaa hyödyntää digitaalisia teknologioita oppilaiden itseohjautuvuuden edistämiseen eli tukea heidän oman oppimisen suunnittelua, seurantaa ja reflektointia. Itseohjautuvuuden edistämiseksi on oppilaalla oltava mahdollisuus tuoda esille osaamistaan, ilmaista mielipiteitään ja valita luovia ratkaisuja.
	Tämä moduuli tarjoaa periaatteet, joita opettaja voi käyttää innovatiivisten ja itseohjautuvaa oppimista tukevien opetussuunnitelmien luomiseen. Käytetyn opetusteknologian tulisi tukea oppimista, mutta välttää valmiita ratkaisuja. Näin mahdollistetaan luovuus ja ratkaisun löytämisen ilo.
4. Arviointi	
4.1. Arviointi strategiat	Osaa hyödyntää digitaalisia teknologioita formatiiviseen ja summatiiviseen arviointiin. Pystyy kehittämään monipuolisia arviointitapoja.
	Opetusteknologioiden hyödyntämisen ohjelmoinnillisen ajattelun ja STEAM-opetuksen yhdistämiseksi tarkoituksena on tukea oppilaita ratkaisujen löytämisessä tai keksimisessä. Ratkaisu luodaan askel kerrallaan ja siksi formatiivinen arviointi sopii hyvin. Monialaisessa (STEAM) oppimisessa lopputulos on yleensä kuitenkin enemmän kuin osiensa summa, ja siksi on hyödyllistä käyttää myös summatiivista arviointia.
5. Oppilaiden voimaannuttaminen	
5.3. Oppilaiden aktiivinen sitouttaminen	Digitaalisten teknologioiden käyttäminen oppilaiden sitouttamiseksi oppiaineen aktiiviseen opiskeluun. Pedagogisissa strategioissa huomioidaan myös digitaalinen teknologia, jolla pyritään edistämään monialaisia taitoja ja valmiutta opitun soveltamiseen uusissa konteksteissa. Tavoitteena on lisätä oppilaiden aktiivisuutta, luovuuttta ja sitoutuneisuutta ottamalla heidät mukaan käytännön toimintaan, tieteelliseen tutkimukseen tai monipuolisten ongelmien ratkaisemiseen.
	Kohdan tavoitteet ovat yhteneviä STEAM-opetuksen kanssa. Ohjelmoinnillisen ajattelun -näkökulmat tukevat digitaalisten teknologioiden monialaista soveltamista.
6. Oppilaiden digitaalisen osaamisen edistäminen	
6.5. Digitaalinen ongelmanratkaisu	Käytetään tehtäviä ja arviointimenetelmiä, joissa mitataan oppilaan kykyä ratkaista teknisiä ongelmia tai taitoja soveltaa teknologiaa uusiin tilanteisiin.

	Ohjelmoinnillinen ajattelu on ongelmanratkaisumenetelmä. Se luo mahdollisuuden käyttää tietotekniikkaa ongelmien ratkaisemiseksi. Luomalla mielenkiintoisia ja haastavia tehtäviä opettaja voi välittää tietoaan oppilailleen.

Oppimistulokset ja arviointi

Hyvä oppimistulos vaatii tulevalta opettajalta kykyä…	**Arviointimenetelmät**
tunnistaa ohjelmoinnillisen ajattelun soveltamismahdollisuudet LUMA (STEM)-aineissa (analyysi)	Ohjelmoinnillisen ajattelun ja LUMA(STEM)-aineen tai aineiden yhdistämisen suunnitelman vertaisarviointi
valita sopiva opetusteknologia tukemaan ohjelmoinnillisen ajattelun soveltamista STEAM-opetuksessa (soveltaminen)	LUMA(STEM) -aineen tai aineiden opetusteknologian valintaa perustelevan esseen vertaisarviointi
opetusintervention suunnitteluun, jossa hyödynnetään opetusteknologiaa ohjelmoinnillisen ajattelun ja STEAM-opetuksen yhdistämiseksi (luovuus)	Luennoitsija arvioi opetusintervention suunnitelmaa kuvaavan raportin ja siihen perustuvan esityksen

Moduulin suunnitelma ja opetusmenetelmät

Moduuli perustuu Code, Connect and Create (3C) ohjelmoinnin täydennyskoulutusmalliin (Jocius et al., 2020), jonka tarkoituksena on tukea opettajia ainesisällön ja ohjelmoinnillisen ajattelun yhdistämisessä. Mallia sovelletaan tulevien opettajien kouluttamiseen. Ohjelmointi on yleistetty ohjelmoinnillista ajattelua tukevien opetusteknologioiden käytöksi. 3C käyttää ohjelmoinnillisen ajattelun PRADA-mallia (Dong et al., 2019), jonka käsitteiden valinnassa on huomioitu, että opettajilta puuttuu usein ohjelmoinnin tai tietojenkäsittelytieteen tuntemus. Vaikka tuttu käsitteistö helpottaa oppimista, saattaa se jättää pois osan ohjelmoinnillisen ajattelun ymmärtämiseen vaadittavasta taustatiedosta. Tässä moduulissa terävöitämme PRADA-mallin käsitteitä tarjoamalla taustatietoa tietojenkäsittelytieteestä.

Moduulin rakenne

Osio 1 - Kolme näkökulmaa tietojenkäsittelytieteeseen

- Tietojenkäsittelytiede (Tkt) sisältää matematiikan, tekniikan ja tieteen näkökulmat. Näkökulmat tukevat Tkt:n soveltamista näillä aloilla.

Osio 2 - OA:n näkökulmat STEAM-opetukseen viitekehys

- Hahmontunnistuksen, Abstrahoinnin, Analyysin ja Algoritmien taidot muodostavat Ohjelmoinnillisen Ajattelun (OA) mallin. Kolmea Tkt:n näkökulmaa käytetään apuna tunnistamaan LUMA (STEAM lyhenteestä STEM) aineiden piirteitä johon OA:a voidaan soveltaa.

Osio 3 - Opetusteknologian valinta OA:n näkökulmien avulla

- OA:n etuna ongelmanratkaisussa on mahdollisuus käyttää Tkt:n menetelmiä ja työkaluja. OA:ta voidaan käyttää ongelmien ratkaisemiseen, tehtävien automatisointiin ja ilmiöiden ymmärtämiseen käyttäen tietotekniikkaa. Opetusteknologiat yhdistävät tietotekniikan ja tuetun käytön tehden työkaluista soveltuvia oppimiseen.

Osio 4 -Opetussisällön luominen STEAM-opetukseen hyödyntäen OA:a

- Osiossa tulevat opettajat oppivat tekemään yhteistyötä monialaisen opetussissällön luomisessa ja hyödyntämään STEAM-opetuksen mallia (projekti). OA:n näkökulmat STEAM-opetukseen -viitekehys tukee OA:n hyödyntämisen mahdollisuuksien havaitsemista.

Osio 5 - STEAM-opetusta ja OA:a tukevien oppimisympäristöjen suunnittelu

- Tulevat opettajat jatkavat yhteistyötä kehittämällä oppimisympäristön tukemaan opetussisältöä (projekti). OA:a tukevan opetusteknologian valinta on keskeinen osa oppimisympäristön suunnittelua.

Osio 6 - Projektien esittely

- Tulevien opettajien projektiryhmät esittelevät suunnitelmansa monialaisesta opetuksesta ja niitä tukevista oppimisympäristöistä.

Osioiden opetusmenetelmät ja ajankäyttö

ECTS 1 = 27 tuntia

Osio 1 – Kolme näkökulmaa tietojenkäsittelytieteeseen

Luento
Kolme näkökulmaa tietojenkäsittelytieteeseen: 90 min

Tehtävä 1.1 kotitehtävä
Oman aiheen tulkinta siihen sopivan tietojenkäsittelytieteen näkökulman kautta: 60 min

Osio 2 – Ohjelmoinnillisen ajattelun näkökulmat STEAM-opetukseen

Luento
Ohjelmoinnillisen ajattelun näkökulmat STEAM-opetukseen: 90 min

Tehtävä 2.1 kotitehtävä
Omaan aiheeseen liittyvän tehtävän kuvaus käyttäen Ohjelmoinnillisen ajattelun näkökulmia STEAM-opetukseen -viitekehystä: 60 min

Tehtävä 2.2 kotitehtävä
Kolmen tehtävän 2.1 palautuksen vertaisarviointi: 90 min

Osio 3 – Opetusteknologian valinta Ohjelmoinnillisen ajattelun näkökulmien avulla

Luento
Opetusteknologian valinta Ohjelmoinnillisen ajattelun näkökulmien avulla: 90 min

Tehtävä 3.1 Kotitehtävä
Opetusteknologian tuen suunnittelu omalle aiheelle: 2 tuntia

Tehtävä 3.2 Kotitehtävä
Kolmen tehtävän 3.1 palautuksen vertaisarviointia: 90 min

Osio 4 – Opetussisällön luominen STEAM-opetukseen hyödyntäen ohjelmoinnillista ajattelua

Luento
Opetussisällön luominen STEAM-opetukseen hyödyntäen ohjelmoinnillista ajattelua 90 min

Tehtävä 4.1
Kurssiprojekti osa 1/2: 5 tuntia

Tehtävä 4.2
Ryhmä keskustelee ohjaajan kanssa projektin suunnitelmasta: 30 min

Osio 5 – STEAM-opetusta ja ohjelmoinnillista ajattelua tukevien oppimisympäristöjen suunnittelu

Luento
STEAM-opetusta ja ohjelmoinnillista ajattelua tukevien oppimisympäristöjen suunnittelu 2/2: 90 min

Tehtävä 5.1
Kurssiprojekti 2/2: 5 tuntia

Osio 6 – Projektien esittely

Tehtävä 6.1
Projektiryhmien esitykset ja keskustelu (15 min/ryhmä)

Osiot ja tehtävät

Osio 1 – Kolme näkökulmaa tietojenkäsittelytieteeseen

Yksi ohjelmoinnillisen ajattelun haasteista, on sen erinomainen yhteensopivuus LUMA-aineiden kanssa. Se näyttäisi olevan kyseisten aineiden osa, mutta jos samat ajattelumallit ovat jo olemassa, niin mikä on ohjelmoinnillisen ajattelun hyöty (Pears, 2019). Ohjelmoinnillinen ajattelu voidaan tulkita kyvyksi havaita mahdollisuuksia tietojenkäsittelytieteen menetelmien ja työkalujen hyödyntämiseen yhdistettynä taitoon soveltaa niitä. Jotta tämä olisi mahdollista, on ohjelmonnillisen ajattelun hyödyntäjän oltava myös perehtynyt tietojenkäsittelyyn, joka on linjassa käsitteen tunnetuksi tehneen artikkelin ajatusten kanssa (Wing 2006). Tietojenkäsittelytieteen määritelmästä ei kuitenkaan ole yksimielisyyttä, vaan sen voidaan sanoa koostuvan kolmesta keskeisestä näkökulmasta: matematiikka, tekniikka ja tiede (Tedre, 2018). Moduulin aiheen kannalta tämä on hyödyllistä, koska LUMA-aineiden näkökulmat ovat jo edustettuna. Tässä osiossa esitellään tietojenkäsittelytiede kolmen näkökulman kautta.

Tietojenkäsittelytiede on nuori tieteenala. Ensimmäinen tietojenkäsittelytieteen laitos perustettiin Purduen yliopistoon 1962. Tietojenkäsitttelytieteen avulla tietokoneet ja niihin liittyvät teknologiat ovat kuitenkn kehittyneet nopeasti ensimmäisestä ENIAC-tietokoneesta 1945, joka vei tilaa 167 neliömetriä, aina miljardeihin toisiinsa yhteydessä oleviin matkapuhelimiin 2020, jotka sopivat omistajiensa taskuihin. Ennen 1990 tietojenkäsittelytieteen tutkimus voitiin jakaa tietotekniikkaan, joka keskittyi laitteistoihin, teoreettiseen tietojenkäsittelytieteeseen, joka tutki ohjelmointia ja tietojärjestelmätieteeseen, jonka aiheena oli tietokoneiden käyttö liiketoiminnassa. Vuoden 1990 jälkeen syntyi vielä kaksi osa-aluetta ohjelmistotekniikka, joka tutki ohjelmistojen tuotantoa ja tietohallinto, joka keskittyi organisaation tietotekniikan hallintaan ja käytön tukemiseen. Tietojenkäsittelytieteen tutkimusalueiden määrä kasvoi vuoden 1996 12:sta 26:een vuonna 2005. Vaikka tieteenala näyttää jakautuvan yhä useampiin

tutkimusalueisiin, on kaikilla kuitenkin jotain yhteistä. Tietojenkäsittelytiede on ainutlaatuinen yhdistelmä matematiikkaa, tekniikkaa, ja tiedettä.

Matematiikka

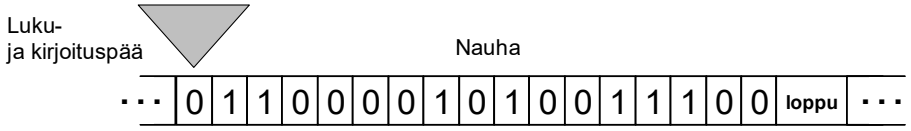
Englannin kielen sana computer, eli suomeksi tietokone, viittasi alun perin laskijan ammattiin, jossa tehtävänä oli suorittaa laskutoimituksia annetun suunnitelman mukaisesti. Yleensä yhden laskijan tulos yhdistettiin muiden kanssa, aiotun lopputuloksen muodostamiseksi. Vaikka ihmistietokoneilta vaadittiin matemaattisia taitoja oikein laskemiseksi, työ oli muuten enimmäkseen mekaanista. Laskeminen oli pilkottu selkeästi määriteltyihin osiin, jotka olivat luonteeltaan toistuvia. Tämä poikkesi matemaatikon työstä, jossa tutkitaan matemaattisten järjestelmien ominaisuuksia tai niiden sovelluksia. Matemaatikot suunnittelivat, miten laskijoiden tulisi tehdä työnsä. Ensimmäinen tietokone luotiin tarjoamaan lisää laskentakapasiteettia. Osa ihmistietokoneista jatkoi tietokoneiden ohjelmoijina.

Matematiikka on keskeisessä asemassa tietojenkäsittelytieteessä ja sen voidaan sanoa muodostavan tietojenkäsittelyn teorian. Matematiikan eri osa-alueet tarjoavat tietojenkäsittelyssä tarvittavia käsitteitä ja menetelmiä. Yleiskäyttöisen tietokoneen matemaattinen malli keksittiin ennen sen fyysistä vastinetta. Erilaisilla matemaattisilla apuvälineillä ja laskukoneilla on pitkä historia ja oma tekninen kehitys. Ainoastaan nykyaikaisen tietokoneen keksimisen myötä voidaan sanoa, että teoria ja käytäntö yhdistyivät. Tietokoneen teoreettinen malli luotiin selvittämään, onko kaava laskettavissa vai ei. Kysymyksen esitti matematiikan jättiläinen David Hilbert ja vastauksen antoi 24-vuotias brittiopiskelija Alan Turing.

Hilbertin esittämä kysymys etsi ratkaisua päätösongelmaan (saksankielellä Entscheidungsproblem). Haasteena oli suunnitella mekaaninen menetelmä, jonka avulla voitiin laskea, oliko predikaattilogiikalla esitetty lause todistettavissa vai ei. Jotta tämä olisi mahdollista, tarvittiin laskutoimituksen vaiheiden eksplisiittinen määritelmä eli algoritmi. Turing lähti ratkaisemaan ongelmaa mallintamalla laskemista kynällä ja ruutupaperilla. Usean rivin sijasta laskutoimitus voitiin tehdä yhdellä äärettömällä rivillä. Numeroiksi riittivät nolla ja yksi, koska mikä tahansa luku tai symboli voitiin koodata em. muodostetuilla numerosarjoilla. Vain yksi numero ruutua kohti oli sallittu ja laskeminen tapahtui ruutu kerrallaan. Ihminen voi luottaa päättelykykyynsä, mutta tässä päättelyn korvasivat ekplisiittiset säännöt. Lopputuloksena oli Turingin Kone (TK), jossa oli ääretön ruutuihin jaettu paperinauha, säännöt sekä lisäksi luku-/kirjoituspää ruutujen lukemiseen ja niihin kirjoittamiseen (katso Kuva 1).

Säännöt

Tila	Symboli nauhalla	Kirjoitettava symboli	Liiku - vasen / oikea / ei	Seuraava tila
T_1	0	0	*oikea*	T_1
T_1	1	1	*oikea*	T_2
T_1	*loppu*	–	*ei*	–
T_2	0	0	*oikea*	T_2
T_2	1	1	*oikea*	T_3
T_2	*loppu*	*pariton*	*ei*	–
T_3	0	0	*oikea*	T_3
T_3	1	1	*ei*	T_2
T_3	*loppu*	*parillinen*	*oikea*	–



Kuva 1. TK koostui ruutuihin jaetusta äärettömästä paperinauhasta, luku-/kirjoituspäästä ja säännöistä, jotka ohjasivat sen suoritusta. Kuvan säännöt määrittelevät algoritmin, joka tarkistaa onko ykkösten määrä nauhalla pariton vai parillinen.

Säännöt olivat keskeisiä TK:ssa. Jokaisella säännöllä oli tunniste ja eri versiot riippuen luku-/kirjoituspään alla olevan ruudun sisällöstä. Sääntö pystyi ohjamaan luku-/kirjoituspään kirjoittamaan symbolin ruutuun ja siirtämään luku/kirjoituspään ruudun vasemmalle, oikealle tai pitämään sen paikallaan. Sääntö sisälsi myös viittauksen seuraavaan sääntöön. Koneella ja säännöillä voitiin määritellä kaikki laskutoimitukset yksiselitteisesti. Toinen keksintö oli käyttää yhteistä tunnistetta ryhmälle sääntöjä, jonka mahdollistamalla modularisuudella voidaan uudelleen käyttää jo kerran määriteltyä laskutoimituksia. Turing hyödynsi tätä ominaisuutta koneen määrittelemiseen, jolla voitiin simuloida minkä tahansa TK:n toimintaa. Universaalin TK:n (UTK) avulla Turing pystyi todistamaan, että kone, joka pystyi tarkistamaan tuottiko toinen TK tuloksen, oli mahdoton. Tämä tarkoitti, että ei ole olemassa yleistä menetelmää, jolla voitaisiin todeta, onko kaava laskettavissa.

Alan Turing huomasi, kun oli julkaisemassa TK:n ja UTK:n mallejaan vuonna 1936, että amerikkalainen Alonzon Church oli ehtinyt ensin. Church käytti erilaista mallia, joka perustui funktioihin ja niiden sieventämiseen. Turingin artikkeli julkaistiin kuitenkin mallin ainutlaatuisuuden vuoksi, mutta hän julkaisi myöhemmin lisäyksen, jossa hän myönsi Churchin mallin olevan ensimmäinen. Lisäyksessä hän kuvasi myös TK:n, joka toteutti Churchin mallin. Molemmat mallit olivat ominaisuuksiltaan vastaavat. Samaan aikaan keksittiin useita muita algoritmeja määritteleviä malleja, jotka erosivat laskennan toteutustavalla. Ne olivat kaikki yhtä ilmaisukykyisiä (niillä voitiin määritellä samat asiat) ja oletuksena on, että ne määrittelevät laskennan rajat.

Algoritmeja määrittelevät mallit edelsivät nykyaikaisen tietokoneen keksimistä, mutta monet nykyään tietojenkäsittelytieteessä käytössä olevat ideat ovat lähtöisin laskennan formalisoinnista (Davis, 2015). Esimerkiksi TK sisälsi ajatuksen ohjelmointikielestä (säännöt),

modulaarisuudesta, ohjelmasta ja datasta samassa muistissa, virtuaalikoneesta ja tulkattavista ohjelmointikielistä. Turing keksi myös idean hajautetusta tietojenkäsittelystä, jossa TK voisi saada osan ratkaisusta ulkopuolisesta lähteestä. Turingin kone oli ihmisen laskutoimituksen suorittamisen mekanisointi, joten se ei ollut vain teoreettisesti vaan myös intuitiivisesti laskennan malli. Merkki Alan Turingin työn merkityksestä on A. M. Turing Award -palkinto, jonka myöntää Association for Computing Machinery (ACM), joka on toinen kahdesta johtavasta tietojenkäsittelytieteilijöiden järjestöstä. Turingin keksimää mallia käytetään yhä teoreettisen tietojenkäsittelytieteen todistuksissa.

Laskemisen mallien lisäksi myös muut matematiikan osa-alueet ovat keskeisiä tietojenkäsittelyssä. Tietojenkäsittelytiedettä voisi ajatella yhtenä matematiikan osa-alueista (Tedre, 2018). Kun on kyse laskennan teoreettisesta tutkimuksesta tai matematiikan osa-alueiden hyödyntämisestä ongelmien ratkaisussa, se on epäilemättä matematiikkaa. Kuitenkin, kun tietokonetta hyödynnetään matematiikan laskutoimitusten toteutuksessa, on myös kysymys fyysisestä toteutuksesta. Se mikä alkoi sähkötekniikasta kehittyi tietokonetekniikaksi ja ohjelmistotekniikaksi. Tietokonetekniikka kehittää laitteet ja ohjelmistotekniikka ohjelmat, niiden ohjaamiseen. Tietojenkäsittelytiede yhdistää teorian ja käytännön.

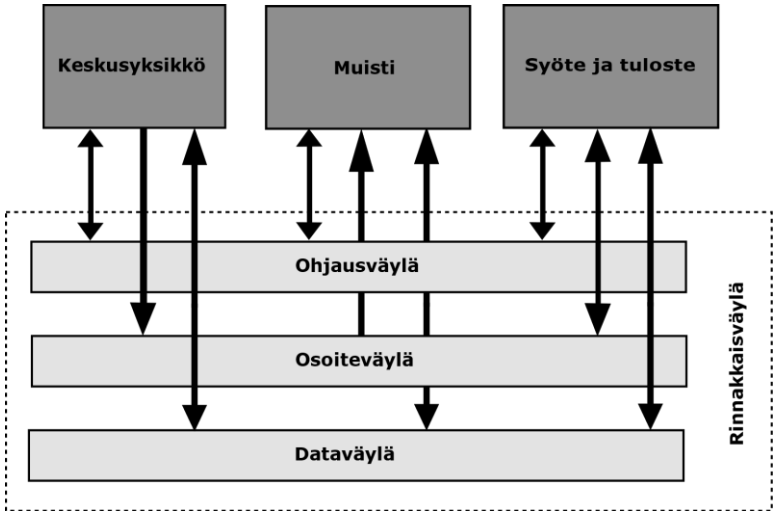
Tekniikka

Matematiikan lisäksi tekniikka on keskeinen tietojenkäsittelytieteen osa-alue. Yliopistot rakensiva ensimmäiset tietokoneet omaan käyttöön ja tästä työstä syntyi tietojenkäsittelytiede. Tekniikkaa voidaan yleisesti määritellä ”... alaksi, joka käyttää tieteellistä ja teknistä tietoa monipuolisten laitteiden, koneiden, rakenteiden, järjestelmien ja prosessien ideoimiseen, suunnitteluun, luomiseen, valmistamiseen, käyttämiseen, ylläpitämiseen ja purkamiseen ihmisten tavoitteiden toteuttamiseksi” (Blockley, 2012, s. xi). Ensimmäinen tietokone rakennettiin toisen maailmansodan aikana Yhdysvalloissa tarjoamaan enemmän laskentakapasiteettia tykkien ammusten lentoratataulukoiden laskemiseen. Jokainen uusi ase tarvitsi taulukoita, jotka sisälsivät yhteensä 3000 lukua. Taulukoiden laskeminen vei sadalta ihmiseltä tai yhdeltä varustettuna mekaanisella differentiaalianalysaattorilla kuukauden. Mekaanisia differentiaalianalysaattoreita oli kolme koko Yhdysvalloissa. Armeija teki yhteistyötä Mooren teknillisen korkeakoulun kanssa taulukoiden tuottamiseksi, ja heillä käytössä kaksi analysaattoria sekä kaksisataa laskijaa. Siitä huolimatta, he eivät pystyneet tuottamaan taulukoita tykkien kehityksen tahdissa.

Laskentakapasiteetin puute synnytti idean Mooren korkeakoulussa täysin sähköisestä laskimesta. Armeija suostui rahoittamaan hankkeen ja aloitti vuonna 1943 projektin elektronisen numeerisen integraattorin ja tietokoneen ENIACin (Electronic Numerical Integrator And Computer) kehittämiseksi. ENIAC:in suunnittelu sai inspiraation laskutehtävien jakamisesta ihmistietokoneille ja siitä miten differentiaalianalysaattori yhdisti osittaiset tulokset. Toiminnan pullonkaulojen välttämiseksi ENIAC:in oli oltava täysin sähköinen. Sen ohjausyksiköt, aritmeettiset piirit ja numeroiden tallennus tehtiin tyhjiöputkista. Nykyisten tietokoneiden lailla, ENIAC suunniteltiin yleiskäyttöiseksi (matemaattisesti), mutta sillä oli hajautettu rakenne (useita laskuyksiköitä) ja siinä käytettiin desimaaliesitystä. Hanke valmistui joulukuussa 1945. ENIAC oli tuhat kertaa nopeampi kuin tuon ajan parhaat laskentamenetelmät.

ENIAC oli menestys sekä teknisesti että toiminnallisesti, mutta sen ohjelmointi oli erittäin vaikeaa. ENIAC:in pituus oli kolmekymmentä metriä ja se koostui neljästäkymmenestä yksiköstä. Yksikköjä ohjattiin kytkintauluilla ja lisäksi eri yksiköt oli yhdistettävä johdoilla suunnitellun laskutoimituksen vaiheiden mukaisesti. Siksi ENIAC:in kehittänyt tiimi päätti yhteistyössä matemaatikko John von Neumannin kanssa suunnitella uudenlaisen tietokoneen. Uusi tietokone kuvattiin vuonna 1945 julkaistussa alustavassa raportissa, jossa oli vain von Neumannin nimi. Suunnitelma sai nimen von Neumannin arkkitehtuuri ja nykypäivän tietokoneet noudattavat yhä sen rakennetta. Siinä kuvataan tietokoneen pääkomponentit ja miten ne ovat vuorovaikutuksessa toistensa kanssa, tekniset yksityiskohdat jäävät toteuttajalle (mikä voi olla syy arkkitehtuurin pitkäikäisyyteen).

von Neumannin arkkitehtuuri kuvaa tietokoneen, joka koostuu syöteyksiköstä, aritmeettis-loogisesta yksiköstä, ohjausyksiköstä (nykyaikaisissa tietokoneissa aritmeettis-looginen yksikkö ja ohjausyksikkö muodostavat yhdessä keskusyksikön), muistiyksiköstä ja tulosteyksiköstä. Eri yksikköjä yhdistää rinnakkaisväylä (katso kuva 2). Tietokonetta ohjataan komennoilla, jotka ovat samassa muistissa datan kanssa. Sekä komennot että data esitetään binäärilukuina, jotka ovat sarja nollia ja ykkösiä. Pelkästään nollien ja ykkösten käyttö on teknisesti erittäin taloudellista verrattuna desimaaliesitykseen. Tietojenkäsittelyyn ja tietokoneen hallintaan liittyvät komennot suoritetaan peräkkäin. Fyysisellä tasolla komennot ovat vain tietokoneen osien sähköisiä tiloja, jotka voivat vaikuttaa muiden osien toimintaan. Abstraktisti komennot muodostavat kuvauksen aiotusta tietojenkäsittelystä. Fyysisen ja abstraktin tason välissä on ohjelman esitys tietokoneen muistissa (Dasgupta, 2016).



Kuva 2. (CC BY-SA 3.0) [1] von Neumann arkkitehtuuri määrittelee nykyaikaisen tietokoneen keskeiset komponentit ja niiden välisen vuorovaikutuksen. Pääkomponentteja ovat keskusyksikkö, muisti ja syöte- sekä tulosteyksikkö.

[1] Tekijänä W Nowicki - oma työ, joka perustuu kaavioon, joka näyttää puolestaan perustuvan sivulle 36 of The Essentials of Computer Organization and Architecture kirjoittajana Linda Null, Julia Lobur, https://books.google.com/books?id=f83XxoBC_8MC&pg=PA36, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=15258936

Ensimmäiset von Neumannin arkkitehtuurin mukaiset tietokoneet rakennettiin Iso-Britanniassa. Mark I valmistui huhtikuussa 1949 Manchesterin yliopistossa ja vain kuukautta myöhemmin EDSAC (Electronic Delay Storage Automatic Calculator) Cambridgessa. EDSAC-tietokoneen kehittäminen tapahtui yliopiston ja yksityisen yrityksen yhteistyönä. J. Lyons & Company oli Iso-Britannian suurin catering-yritys, jolla oli valtava talousosasto, jonka tehtävänä oli pitää yritys kannattavana. He etsivät koko ajan teknisiä apuvälineitä kirjanpidon tehostamiseksi ja he olivat kuulleet uudesta tietokoneesta. Sopimuksen mukaan he rahoittivat EDSAC-tietokoneen kehittämistä ja myöhemmin yliopisto auttaisi heitä rakentamaan oman tietokoneensa. J. Lyons & Co. tietokoneen nimi oli LEO (Lyons Electronic Office) ja sitä käytettiin palkanlaskentaan, tilaustenkäsittelyyn, logistiikkaan ja teesekoitusten valmistukseen. LEO oli ensimmäinen kaupallinen tietojärjestelmä, ja tämän muistoksi Association for Information Systems (AIS) antaa LEO-palkinnon erinomaisille tutkijoille tietojärjestelmätieteen alalla.

von Neumannin arkkitehtuuri mahdollisti monenlaiset toteutukset, koska se määritteli vain tietokoneen yleisen mallin. Tietokoneita rakensivat aluksi sähköinsinöörit. Vaati kekseliäisyyttä ottaa muihin tarkoituksiin kehitettyjä keksintöjä ja soveltaa niiden ominaisuuksia tietokoneiden rakentamiseen. Ohjelmointi oli olennainen osa von Neumannin arkkitehtuuria. Voitiin valita, toteutettiinko ominaisuus laitteistolla vai ohjelmoimalla. Maurice Wilkes, joka oli yksi EDSAC-tietokoneen kehittäjistä, keksi mikroohjelmoinnin, jossa lyhyitä ohjelmia käytettiin korkeatason komentoina, joka helpotti ohjelmoijien työtä. Tietotekniikkaan erikoistuminen tuli mahdolliseksi yliopistojen sähkötekniikan laitoksilla tietokoneiden laajemman käytön ja tietotekniikan kehityksen myötä.

Moderniin tietotekniikkaan kuuluu suunnittelua, rakentamista, soveltamista, tietokonejärjestelmien ohjelmistojen ja laitteistokomponenttien ylläpitämistä, tietokoneohjattujen laitteiden kehittämistä, ja älykkäitä verkkoja. Tietotekniikka alkoi sähkötekniikan ja tietojenkäsittelytieteen yhdistelmänä. Se on kehittynyt neljän vuosikymmenen aikana itsenäiseksi tieteenalaksi, mutta sähkötekniikka ja tietojenkäsittelytiede ovat yhä osa tietotekniikkaa. Vaikka tietotekniikka sai alkunsa tietokoneiden keskeisten komponenttien kehittämisestä, hyvin harvat työskentelevät enää näiden parissa, koska tietokoneen komponenttien kehitystyö on nykyään teknisesti ja taloudellisesti niin vaativaa, että vain harva yritys pärjää kilpailussa. Tietotekniikan miniatyrisointi, luotettavuus ja kokonaisia järjestelmiä sisältävät mikrosirut ovat tehneet tietokoneista kaikkialla läsnä olevia ja korvanneet monia perinteisiä elektronisia laitteita. Tietotekniikan insinöörejä tarvitaan älypuhelinten, tablettien, langattomien verkkojen ja muiden digitaalisten tuotteiden valmistuksessa. Sulautettujen järjestelmien kautta he työskentelevät myös autojen, kodinkoneiden, kulutuselektroniikan ja viime aikoina esineiden internetin kanssa. ACM-järjestön kohderyhmänä on lähinnä teoreettiset tietojenkäsittelytieteilijät, mutta teknologiasta kiinnostuneille tutkijoille löytyy oma järjestö Institute of Electrical and Electronics Engineers (IEEE) Computer society.

Ohjelmat ohjaavat tietokoneen toimintaa ja ilman ohjelmia tietokoneet eivät toimisi. Tietokone luotiin laskimeksi, mutta rinnakkain laitteiston kehityksen kanssa, myös ohjelmat (nk. ohjelmistot, jotka koostuvat useasta yksinkertaisemmasta ohjelmasta) kehittyivät. Ohjelmistoille keksittiin uusia käyttötarkoituksia, ja nämä saivat ohjelmien koon kasvamaan. Matemaattiset menetelmät eivät enää riittäneet käsittelemään ohjelmoinnin monimutkaisuutta. Ongelmaa kutsuttiin ohjelmistokriisiksi. Nato järjesti ensimmäisen konferenssin ohjelmistojen monimutkaisuuden hallinnasta 1968 ja toisen 1969, joissa ehdotettiin teknisten tieteiden

menetelmien soveltamista. Tämä oli alku uudelle ohjelmistotekniikan alalle. IEEE määritelmän mukaan ohjelmistotekniikka on: ”Järjestelmällisten, kurinalaisten ja mitattavissa olevien menetelmien soveltamista ohjelmistojen kehittämiseen, käyttöön ja ylläpitoon; toisin sanoen tekniikan soveltamista ohjelmistoihin.” (IEEE, 2010).

Ohjelmistotekniikassa on kyse menetelmistä, työkaluista, ja kehitysprosesseista. Suurin haaste on monimutkaisuus: kuinka kehittää ohjelmaa niin, että prosessi olisi hallittava, mahdollistaisi useamman henkilön työskentelyn ohjelman kanssa samanaikaisesti ja näiden työskentelyn koordinoinnin. Proseduurit ja funktiot olivat ensimmäisiä menetelmiä ohjelmakoodin rakenteen kehittämiseksi, ensimmäinen muuttaa tietokoneen tilaa ja toinen palauttaa tuloksen; molempia voidaan kutsua ohjelman muista osista. Olio-ohjelmointi on tapa rakentaa ohjelma erillisistä komponenteista. Jokaisella oliolla voi olla useita metodeja, jotka ovat proseduurien ja funktioiden yhdistelmiä (voivat sekä muuttaa tilaa että palauttaa tuloksen). Rakennusarkkitehtuurista kopioitiin suunnittelumallien idea, joilla määriteltiin ratkaisu tietyn tyyppiseen ongelmaan olioiden välisenä vuorovaikutuksena. Arkkitehtuurin käsitettä käytetään myös kuvaamaan yleistä periaatetta, jolla ohjelmisto on rakennettu.

Tekemällä ohjelmistoista modulaarisempia (jakamalla ne komponentteihin) ohjelmankehitys voidaan jakaa usean henkilön kesken. On tärkeää, että komponentien toiminnasta sovitaan, koska komponentit voivat olla riippuvaisia toistensa ominaisuuksista. Riippuvaisuus aiheuttaa myös virheitä, kun ohjelman koodia muutetaan. Nykyaikaiset kehitysympäristöt mahdollistavat ohjelmiston eri osien välisten riippuvuuksien etsimisen. Ne tuovat esille myös ohjelmiston rakenteen ja millaisista toiminnallisuuksista ohjelma koostuu. Nykyaikaiset ohjelmistoprojektit ovat suuria, ja niiden tekemiseen tarvitaan useita ihmisiä. Versionhallintajärjestelmä auttaa yhdistämään kehitettävän ohjelman eri osia ja niiden versioita. Ohjelmistonkehitysprosessin hallitsemiseen on kaksi keskeistä mallia. Yhtä kutsutaan vesiputousmalliksi, jossa kehitys etenee vaatimusten, suunnittelun, toteutuksen, testaamisen ja ylläpidon vaiheiden kautta. Toista kutsutaan ketteräksi kehitykseksi, jossa kehitys etenee iteraatioissa ja iteraatiossa lisätään ominaisuuksia kokonaisuuteen. Vesiputousmallia suositellaan suuriin projekteihin ja ketterää kehitystä pieniin ja keskisuuriin. Ketterän kehityksen menetelmät ovat myös hyviä silloin, kun ohjelmiston lopullisesta ominaisuusjoukosta on epävarmuutta.

Tekniikan näkökulma koskee sekä tietokonelaitteiden että ohjelmistojen kehittämistä. Nykyinen digitaalinen yhteiskuntamme on seurausta tietokoneaikakauden alusta lähteneestä teknisestä kehityksestä. Kehitys on tapahtunut yliopistoissa, mutta myös monissa yrityksissä, joiden tuotteet ovat mahdollistaneet digitaalisen vallankumouksen. Teoreettinen työ on ollut tärkeä osa kehitystä. Varsinkin tietokoneaikauden alussa monet asiat eivät olisi olleet mahdollisia, jos ei olisi kehitetty teoreettisia menetelmiä, jotta tekniikasta saataisiin kaikki hyöty irti. Kun periaate on keksitty, on mahdollista kehittää sitä edelleen. Sekä teoreettinen että tekninen kehitys sai alkunsa ihmisen laskennan mallintamisesta. Mallintamisen myötä syntyi myös ajatus simuloida ihmisen älykkyyttä. Tässä tietojenkäsittelyä käytetään älykkyyden teoriana, joka on esimerkki tietojenkäsittelytieteen tieteellisestä näkökulmasta.

Tiede

Tieteen näkökulmasta tietojenkäsittelyä tulkitaan empiirisenä tutkimusalana kuten luonnontieteitä, joissa tutkimuksen kohteena ovat objektiiviset ilmiöt. Näkökulma eroaa matemaattisesta, joka koskee tietojenkäsittelyn teoriaa, ja teknisestä, joka koskee laitteiden ja ohjelmistojen rakentamista. Tiede voidaan määritellä kokeiden, havainnoinnin ja teorian luonnin menetelmien kautta (Okasha, 2016). Kaikki tieteenalat eivät voi käyttää kokeita, esim. tähtitiede, mutta teorioiden validoimiseksi on löydettävä tapa kerätä tietoa. Vuonna 1956 pidettiin Yhdysvaltojen Dartmouthissa kesäkoulu tietojenkäsittelytieteen tutkijoille, jotka olivat kiinnostuneita tekoälystä (engl. Artificial Intelligence). Tekoälyn tutkimus perustui ajatukseen, että: ”... jokainen oppimisen osa-alue tai älykkyyden piirre voidaan periaatteessa kuvata niin tarkasti, että voidaan rakentaa kone simuloimaan sitä.” (McCarthy et ai., 2006, s. 2). Tässä tietojenkäsittelyä käytetään tieteellisesti kuvaamaan sen ulkopuolista ilmiötä.

Tekoäly antoi tietojenkäsittelytieteelle mahdollisuuden osallistua poikkitieteelliseen aivojen, mielen ja älyn tutkimukseen osana kognitiotiedettä. Kognitiotiede on monitieteinen tutkimusala, joka yhdistää filosofiaa, psykologiaa, kielitiedettä, antropologiaa, neurotiedettä, ja tietojenkäsittelytiedettä. Laskennalliset mallit ja tietokone antoi muille tieteille työkalut aivojen tapahtumien mallintamiseen ja simulointiin. Voidaan sanoa, että kognitiotiede syntyi MIT-yliopiston seminaarissa samana vuonna kuin tekoälykin (Miller, 2003). Kuuluisa kielitieteilijä Noam Chomsky kuvaili silloin, millaista kapasiteettia tarvitaan kielen tuottamiseen. Hän vertasi Turing Konetta (TK) yksinkertaisempiin koneisiin, jonka perusteella hän päätteli, että tarvitaan vähintään TK:n ominaisuudet kielen tuottamiseksi. Georg Miller osoitti, että ihminen pystyy käsittelemään työmuistissa enintään seitsemän asiaa kerrallaan, mutta ryhmittelemällä tiedon määrää voidaan nostaa. Allen Newell ja Herbert Simon esittelivät Logic theorist tietokoneohjelman (sama ohjelma esitettiin myös Dartmouthissa), joka pystyi todistamaan matemaattisia teoreemoja. Ohjelma käytti heuristisia algoritmeja, jotka simuloivat ihmisen toimintaa puutteellisella tiedolla.

Toinen keskeinen tietojenkäsittelytieteen kontribuutio kognitiotieteelle oli Warren S. McCulloughin ja Walter Pittsin idea, että aivojen neuronien voidaan tulkita suorittavan loogisia operaatioita (Bermúdez, 2020). Aivoissa on 86 miljardia neuronia, jokaisella neuronilla voi olla yli 10 000 yhteyttä, jotka muodostavat monimutkaisen verkoston. Tiedämme, että neuronit kommunikoivat sähköisillä signaaleilla ja tätä toimintaa voidaan mitata. Neurotieteessä on pystytty kartoittamaan, miten eri aivoalueiden toiminta liittyy kognitiivisiin kykyihin. Mutta mitä toiminta tarkoittaa, eli millaista representaatioa (oletettavasti) aivot käyttävät, ei vielä tiedetä. McCullochin ja Pittsin idea kuvaa mekanismin, jossa keinotekoiset neuronit toimivat arvoilla 0 ja 1, kuten tietokone. Tämän mahdollistaa hermoverkon toiminnan simuloimisen tietokoneella. Keinotekoiset hermoverkot ovat mahdollistaneet aivojen toiminnan mallintamisen, kognitiivisten kykyjen simuloinnin ja älykkäiden ohjelmistojen kehittämisen. Jos hermoverkkojen simulaatiota käytetään ihmisen aivojen toiminnan selittämiseen on käyttötapa tieteellinen, mutta sitä voidaan myös käyttää teknisenä ratkaisuna, ero on tavoitteessa.

Tietojenkäsittelytieteen tutkimuskohteena voidaan sanoa olevan luonnolliset ja keinotekoiset informaatioprosessit (Denning, 2007). Osana tietojenkäsittelytiedettä tietojärjestelmätiede (Tjt) tutkii yritysten ja muiden organisaatioiden tietokoneiden hyödyntämistä, joten se sisältää molemmat prosessit. Organisaatio voidaan määritellä toisiinsa vuorovaikutuksessa olevien prosessien järjestelmäksi, jonka tarkoituksena on valmistaa tuotteita tai tarjota palveluita (Checkland & Holwell, 1998). Jokainen prosessi tarvitsee informaatiota toimiakseen ja tuottaa

puolestaan informaatiota, joita muut prosessit voivat tarvita. Lisäksi organisaation johtaminen ja prosessien koordinointi tarvitsee informaatiota. Tietojärjestelmätiede syntyi tavoitteesta suunnitella organisaation informaationvälitys tietotekniikan avulla. Tämä on tietojärjestelmätieteen tekninen puoli, mutta on sillä on myös yhteisölliseen käyttäytymiseen liittyvä puoli, jonka referenssitiede on sosiologia (Hevner et al., 2004). Tietojärjestelmätiede tutkii ihmisten informaation luontia ja käyttöä organisaation toiminnan mahdollistamiseksi. Myös se, miten tietotekniikka vaikuttaa organisaation toimintaan, on yksi tutkimusalue.

Sekä tekoälyssä että tietojärjestelmätieteessä tehdään tutkimusta, joka liittyy tietotekniikan ulkopuolisiin ilmiöihin ja jossa käytetään empiirisiä menetelmiä. Mutta tietojenkäsittelytieteen piirissä on myös syntynyt ajatus, että teoreettisten ja teknisten näkökulmien yhdistelmää voitaisiin pitää empiirisenä tieteenä. Tietojenkäsittelytieteen puitteissa tehtävän tutkimuksen tulisi silloin olla yhtä perusteellista ja käytettyjen menetelmien yhtä luotettavia, kuin luonnontieteissä. Kirjallisuudesta löytyy viisi keskeistä tietojenkäsittelytieteen kokeellisen tutkimuksen menetelmää (Tedre & Moisseinen, 2014): toteutettavuustutkimus, kokeellinen tutkimus, kenttätutkimus, vertaileva tutkimus ja kontrolloitu koe.

Toteutettavuustutkimus koskee uusia tekniikoita ja työkaluja. Tavoitteena on selvittää, onko tehtävän automatisointi kustannustehokasta, resurssitehokasta, toteutuskelpoista, luotettavaa, tai täyttääkö se jonkin muun yksinkertaisen kriteerin. Toteutettavuus, on osoitus siitä, että tekniikan käyttö on mahdollista. *Kokeellinen tutkimus* menee toteutettavuustutkimusta pidemmälle ja se pyrkii arvioimaan järjestelmän suorituskykyä tai vastaavuutta spesifikaatioon. Suorituskykyä arvioidaan ennalta määriteltyjen muuttujien mukaan. Tutkimus suoritetaan yleensä laboratorio-olosuhteissa, mutta se voidaan toteuttaa myös käytännössä, jos otetaan huomioon ympäristön vaikutus. Viimeinen vaihtoehto on myös lähtökohtana *kenttäkokeissa*, jossa tavoitteena on arvioida käytössä olevaa järjestelmää. Kenttäkokeissa todellisen ympäristön sattumanvaraisuus on osa koetta. Tavoitteena on kokeilla järjestelmän käytettävyyttä, luotettavuutta tai suorituskykyä. Tällaisia kokeita käytetään usein tietojärjestelmätieteessä, jossa onnistuminen mitataan käytännön hyötyinä.

Monella tietojenkäsittelytieteen osa-alueella etsitään ratkaisua, jolla on paras suorituskyky. Se voi koskea nopeutta, resurssien käyttöä, ratkaisun kattavuutta, palvelun laatua tai jotakin toista tekijää, jonka suhteen voidaan arvioida tuloksen paremmuutta. Nämä ovat esimerkkejä *vertailevasta tutkimuksesta*, jossa tavoitteena on arvioida, onko yksi ratkaisu parempi kuin toinen, kun kokeet tehdään samoissa olosuhteissa. *Kontrolloidun kokeen* voidaan sanoa olevan tieteellisen tutkimuksen korkeinta tasoa ja sitä voidaan myös käyttää tietojenkäsittelytieteessä. Siinä on huolehdittava kaikkien tutkimukseen vaikuttavien tekijöiden hallinnasta. Yksinkertaisimmillaan hallinnalla tarkoitetaan, että negatiiviset ja positiiviset tulokset saadaan odotetusti.

Informaatioteknologian levinneisyys ja sen kautta tietojenkäsittelytiede ovat myös muuttaneet tieteitä kuten muitakin yhteiskunnan osia. Nobel-palkinnon saanut Kenneth G. Wilson (1989) kirjoitti tietokoneiden käytöstä tieteissä, joita hän kutsui laskennallisiksi tieteiksi. Hän esitti, että perinteisen kokeellisen ja teoreettisen tutkimusmenetelmien rinnalla oli nyt kolmas menetelmä laskennallinen tutkimus. Myös empiiriset ja teoreettiset menetelmät ovat muuttuneet nykyaikaisen tietojenkäsittelyn mahdollisuuksien myötä (Denning, 2017a). Empiirinen tutkimus on tietojen keräämistä havaintojen ja kokeiden avulla hypoteesien vahvistamiseksi tai

hylkäämiseksi. Tietokoneet mahdollistavat suurten datamäärien käsittelyn ja analysoinnin. Koska tietokoneresurssien saatavuus on kehittynyt merkittävästi viime vuosina, tutkijan ei tarvitse enää tyytyä näytteiden tilastolliseen analyysiin, vaan hän voi analysoida koko aineiston. Teoreettisessa tutkimuksessa voidaan rakentaa laskennallisia malleja selittämään mitä tiedetään ja käyttää niitä hypoteesien tekemiseen mahdollisista ilmiöistä. Tietokoneet auttavat mallien yhtälöiden laskemisessa. Tietokoneiden käyttö ei ole kuitenkaan muuttanut sitä, että tieteen tekemiseen tarvitaan sekä dataa (empiriaa) että malleja (teoriaa).

Tietokoneiden käyttö empiirisen ja teoreettisen tutkimuksen nopeuttamiseksi on itsessään vallankumous (Denning, 2017a). Tietojenkäsittelyn mahdollisuudet ovat kuitenkin laskentaa suuremmat. Staattisten mallien sijaan voidaan rakentaa dynaamisia simulaatioita, jotka toteuttavat kuvatut prosessit. Tämä mahdollistaa järjestelmien tutkimisen, joita ei voida fyysisesti havainnoida tai joiden seuraukset ovat liian monimutkaisia laskettaviksi. Simulaatiossa kuvataan järjestelmän komponentit ja miten ne ovat toistensa kanssa vuorovaikutuksessa. Järjestelmä muodostuu yksittäisten vuorovaikutusten aiheuttamista seurauksista kokonaisuuteen. Tekoälyssä laskentaa käytettiin simuloimaan ihmisen tietojenkäsittelyä. Näyttää siltä, että myös muita luonnollisia prosesseja voidaan tulkita informaatioprosesseiksi (Denning, 2017a). Informaatioprosessilla on syöte ja tuloste. Tietokone on erityisen sopiva työkalu näiden tulosten laskemiseen. Tietojenkäsittelytiede on informaatioteknologian tieteellistä tutkimusta; sitä voidaan käyttää ymmärtämään ihmisen älykkyyttä, organisaation toimintaa ja se on tehokas työkalu muissa tieteissä. Jotta tutkimusta tietojenkäsittelytieteessä voitaisiin kutsua tieteeksi, on siinä käytettävä tieteellistä menetelmää.

Yhteenveto

Tietojenkäsittelytieteen kolme näkökulmaa ovat selkeästi havaittavissa. Matemaattinen näkökulma antaa algoritmille täsmällisen määritelmän, joka on keskeinen osa tietojenkäsittelyn teoriaa (katso taulukko 1.1. toinen sarake vasemmalta). Määritelmän avulla voidaan tutkia, mitä ongelmia voidaan ratkaista laskemalla. Tietokoneet laajentavat matematiikan mahdollisuuksia sallimalla suurten datamäärien käsittelyn ja monimutkaiset laskutoimitukset, joita olisi hankala tai mahdotonta tehdä käsin. Digitaalinen yhteiskuntamme on seurausta informaatioteknologian kehityksestä tietokone- ja ohjelmistotekniikan avulla. Tietotekniikan näkökulmassa tietojenkäsittely etsii jatkuvasti uusia sovellusalueita tutkimalla, mitä tehtäviä voidaan suorittaa ohjaamalla informaatioteknologiaa (katso taulukko 1.1. kolmas sarake vasemmalta). Tietotekniikkaa käytetään tieteellisesti ihmisen älykkyyden simulointiin ja mallina tietojenkäsittelystä organisaatioissa. Monet tieteelliset ilmiöt ovat liian monimutkaisia, jotta niitä voisi käsitellä perinteisin keinoin. Kaikkien ilmiöiden mekanismit eivät myöskään ole havaittavissa, joten tietokoneita voidaan käyttää simuloimaan niitä muodostavia tekijöitä. Tieteellisiä ilmiöitä voidaan tutkia informaatioprosesseina, mikä ei olisi mahdollista ilman tietojenkäsittelytiedettä (katso taulukko 1.1. neljäs sarake vasemmalta).

Taulukko 1.1. Kolme näkökulmaa tietojenkäsittelytieteeseen

	Matematiikka	Tekniikka	Tiede
Kohde	laskennan mallit	tietokone- ja ohjelmistotekniikka	luonnolliset ja keinotekoiset informaatioprosessit

Menetelmä	laskutoimitukset ja todistukset	elektroniikka, tietotekniikka ja ohjelmointi	empiirinen tutkimus, laskennalliset mallit, simulaatio
Tieto	laskennan mahdollisuudet ja rajat	miten suunnitella ja valmistaa laitteita sekä ohjelmistoja	ilmiön tulkinta tietojenkäsittelynä
Kysymys	Mitä ongelmia voidaan ratkaista tietojenkäsittelyllä?	Mitä toimintoja voidaan toteuttaa ohjaamalla informaatioteknologioita?	Mitä ilmiöitä voidaan ymmärtää tietojenkäsittelyprosesseina?

Luento – Kolme näkökulmaa tietojenkäsittelytieteeseen

Hyödyntäen ylläolevaa tekstiä

- Matematiikka: mitä ongelmia voidaan ratkaista tietojenkäsittelyllä?
- Tekniikka: mitä toimintoja voidaan toteuttaa ohjaamalla informaatioteknologioita?
- Tiede: mitä ilmiöitä voidaan ymmärtää tietojenkäsittelyprosesseina?

Tehtävä 1.1 kotitehtävä - Oman aiheen tulkinta siihen sopivan tietojenkäsittelytieteen näkökulman kautta

Kirjoita internethaun ja/tai opettajan antaman lisäaineiston pohjalta lyhyt essee (200 sanaa), jossa kuvaat, miten omaan aiheeseen liittyvä esimerkki hyödyntää tietojenkäsittelytiedettä. Voit käyttää tietojenkäsittelytieteen näkökulmia apuna haussa (ks. yllä oleva taulukko 1.1).

Osio 2 – Ohjelmoinnillisen ajattelun näkökulmat STEAM-opetukseen viitekehys

Osiossa yhdistetään tietojenkäsittelytieteen näkökulmat ohjelmoinnilliseen ajatteluun STEAM-opetuksen tukemiseksi. Tietojenkäsittelytieteen näkökulmat matematiikka, tekniikka ja tiede vastasivat jokainen eri kysymykseen:

- Matematiikka: mitä ongelmia voidaan ratkaista tietojenkäsittelyllä?
- Tekniikka: mitä toimintoja voidaan toteuttaa ohjaamalla informaatioteknologioita?
- Tiede: mitä ilmiöitä voidaan ymmärtää tietojenkäsittelyprosesseina?

PRADA-malli luotiin käytännölliseksi tavaksi yhdistää ohjelmoinnillista ajattelua eri aineisiin alkaen esikoulusta aina toisen asteen opetukseen asti. Tavoitteena oli luoda helposti ymmärrettävä malli. PRADA lyhenne tulee englannin kielen sanoista Pattern Recognition (hahmontunnistus), Abstraction (abstraktio), Decomposition (analyysi) and Algorithms (algoritmit).

Huom! Käytämme tässä työssä PRADA-mallia, joka painottaa ohjelmoinnillisen ajattellun käsitteellistä puolta, tukemaan tulevien opettajien tietotekniikan valintaa aineensa opettamiseksi. Se poikkeaa TeaEdu4CT-hankkeen moduulissa 2 esitetystä mallista, joka painottaa ongelmanratkaisuprosessia.

PRADA-malli:

- **Hahmontunnistus** (**P**attern **R**ecognition) säännönmukaisuuksien havainnointi ja tunnistaminen; trendit ja säännönmukaisuudet datassa, prosesseissa ja ongelmissa
- **Abstraktio** (**A**bstraction) ongelman kannalta merkityksellisten ja keskeisten piirteiden sekä ominaisuuksien tunnistaminen
- **Analyysi** (**D**ecomposition) datan, prosessien tai ongelmien pilkkominen pienempiin ja hallittavampiin osiin
- **Algoritmit** (**A**lgorithms) ohjeet jotka määrittelevät ongelman tai ongelmatyypin ratkaisuprosessin selkeästi erottuvina vaiheina

Hahmontunnistus

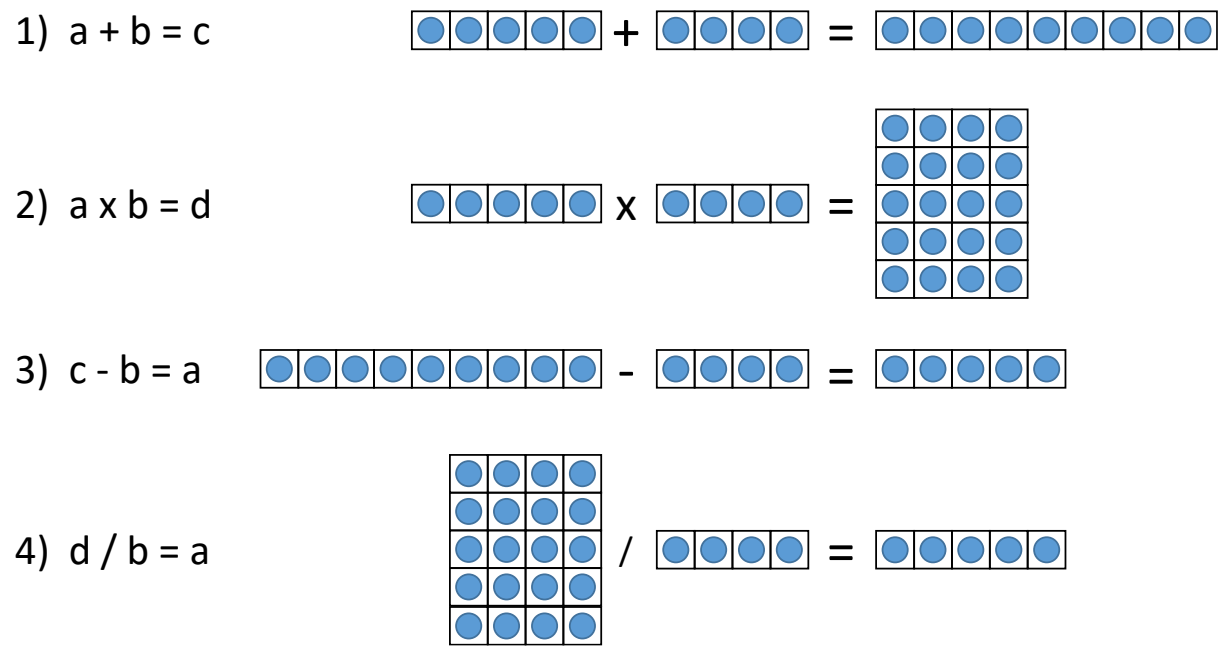
Säännönmukaisuudet tulkitaan tässä kahdella tavalla matemaattisella ja teknisellä. Matematiikan sanotaan olevan ”tieteiden kuningatar”, koska se tarjoaa käsitteet, joita käytetään teorioiden muodostamiseen muissa tieteissä. Säännönmukaisuksien tekninen tulkinta kattaa sekä tekniikan että tekniset artefaktit. Matematiikan osa-alueen säännönmukaisuudet (Devlin, 1994; Kvasz, 2019) voidaan määritellä pienellä joukolla tosiasioita ja sääntöjä. Näistä voidaan loogisesti johtaa kaikki muut osa-alueen käsitteet (Devlin, 2012). Tunnistamalla säännönmukaisuudet (hahmontunnistus) voimme tehdä tietojenkäsittelyssä käyttämämme abstraktit (matemaattiset) käsitteet konkreettisemmiksi. Ohjelmoinnillisen ajattelun teknisessä näkökulmasssa säännönmukaisuuksilla viitataan tietokoneiden, elektroniikan tai sulautettujen järjestelmien ohjaamiseen. Vaatimuksena on, että kohdejärjestelmän toimintaa voidaan kuvata erillisinä sähköisinä tiloina ja niiden välisinä siirtyminä. Sekä matemaattisessa että teknisessä näkökulmassa, ensimmäinen askel kohti ratkaisua on tunnistaa ongelmaan liittyvät säännönmukaisuudet.

Matematiikan oppiminen voidaan nähdä analogiana kielen oppimiselle (Sfard, 2007). Ilman kieltä emme voi käsitellä matemaattiikan abstrakteja olioita ja niiden ominaisuuksia. Matemaattisen keskustelussa käsitteet viittaavat näihin olioihin. Käytämme esimerkiksi numeroita edustamaan määrää ja muotoja kuvaamaan geometrisia olioita. Käsitteet kuvaavat myös olioiden välisiä suhteita, kuten yhtäläisyys, erisuuruus, samankaltaisuus, ekvivalenssi jne. Jokaisella matematiikan osa-alueella on omat käsitteensä, mutta yksi osa-alue voi käyttää toisen käsitteitä ja olioita, jos ne ovat osa sen määritelmää. Kun puhumme matemaattisista olioista, käytämme sanoja, mutta kirjallisessa viestinnässä käytämme symbolijärjestelmää, joka on paljon

pelkistetympi ja tehokkaampi. Symbolijärjestelmään kuuluu mm. numerot, algebran ja logiikan merkit ja kaavat. Symbolit toimivat myös matemaattisen ajattelun apuvälineinä. Voimme myös visualisoida matemaattisia rakenteita, joita oliot ja niiden väliset suhteet muodostavat, kuten geometriset piirustukset ja graafit. Kaavioilla voidaan esittää funktioiden ja kaavojen tuloksia eri syötteillä.

Jokaisen matematiikan osa-alueen ydin, on tosiasioiden ja sääntöjen muodostama järjestelmä, joka määrittelee sen oliot. Uusi käsite on oltava johdettavissa tästä järjestelmästä. Todistus on kertomus olioista ja niiden välisistä suhteista, jotka määrittelevät käsitteen. Osa-aluetta tutkivat matemaatikot hyväksyvät tai hylkäävät todistuksen riippuen siitä, kuinka hyvin se noudattaa vakiintunutta käsitystä kyseisestä järjestelmästä. Joskus olemassa olevaa matemaattista järjestelmää voidaan muuttaa, jos uusi määritelmä on perustavanlaatuinen, mutta nämä tilanteet ovat harvinaisia. Havaintojen todentamiseksi käytetään aksioomia, määritelmiä, väitteitä ja todistuksia. Matematiikan kieli on muodollinen ja perustuu rutinoituun tapaan puhua matemaattisista olioista. Yksi tärkeä rutiini on matemaattisten olioiden käsittely. Esimerkkejä näistä rutiineista ovat laskeminen, ongelmanratkaisu, validiointi, ja todistaminen.

Matemaattiset oliot ja niiden väliset suhteet ovat säännönmukaisia. Säännnönmukaisuus on laskutoimitusten perusta ja antaa meille sanaston keskustella matemaattiikan osa-alueesta. Yleensä havaittu säännönmukaisuus ei ole matemaattisen järjestelmän ydin, vaan sen seuraus. Meille on opetettu käsitteitä ja niiden suhteita matematiikan tunneilla tai kirjoissa ja harjoiteltu niitä laskemalla. Havaitessamme ongelman, voimme käyttää sen säännnönmukaisuuksia sopivan matematiikan osa-alueen valitsemiseksi, joka voisi sen ratkaista. Tässä käytetään matemaattisen osa-alueen säännönmukaisuuden esimerkkinä lukuteoriaa, joka tunnetaan paremmin aritmetiikkana (katso Kuva 2.1).



Kuva 2.1. Visualisoimalla yhteen-, kerto-, vähennys- ja jakolaskun operaatioita voidaan tuoda esille lukumäärien käsittelyyn liittyviä säännönmukaisuuksia.

American National Research Councilin Framework for K-12 Science Education (NRC, 2012) tiedeopetuksen viitekehys asettaa mallien kehittämisen, data-analyysin ja matemaattisen sekä ohjelmoinnillisen ajattelun kahdeksan keskeisen tieteellisen ja teknisen menetelmän joukkoon. Viitekehystä hyödynnetään American Next Generation Science Standards (NGSS Lead States, 2013) tiedeopetuksen suosituksissa, jossa matemaattisia malleja käytetään tukemaan tieteellisiä päätelmiä ja digitaalisia työkaluja (ohjelmoinnillinen ajattelu) suurten tietoaineistojen analysointiin. Myös tutkimuspohjaiset mallit ja viitekehykset, kuten ohjelmoinnillisen ajattelun integrointia oppiaineisiin kuvaava malli (Malyn-Smith et al., 2018) ja ohjelmoinnillinen ajattelu matematiikassa ja luonnontieteissä taksonomia (Winetrop et al., 2016) korostavat laskennallisten mallien ja tietoaineistojen merkitystä tieteissä. Mallinnusta ja data-analyysiä voidaan tulkita säännönmukaisuuksien tunnistamisena.

Malli voidaan yleisesti määritellä olevan tiettyyn tarkoitukseen luotu esitys ideasta, oliosta, tapahtumasta, prosessista tai järjestelmästä (Gilbert and Boulter, 1998). Keskeisiä mallin ominaisuuksia ovat:

- Se edustaa jotakin mallinnettavan järjestelmän osaa ja valitut piirteet heijastavat mallintajan mielenkiinnon kohdetta. Malli on keinotekoinen, se ei kuvaa luonnollista kohdetta.
- Samaa todellisuutta voidaan kuvata usealla eri mallilla, riippuen siitä, mitkä piirteet ovat mallintajalle mielenkiintoisia.

Tieteessä malli on järjestelmän esitys, joka koostuu olioista ja niiden ominaisuuksista tai muuttujista (Gutiérrez & Pintó, 2005). Malli kuvaa järjestelmän lakeja, jotka määrittelevät olioiden käyttäytymisen tai niiden muuttujien välisiä suhteita. Sen keskeinen tehtävä on selittää ja ennustaa. Tieteellisen mallin perusteella voidaan myös päätellä kohteena olevan järjestelmän ominaisuuksia tai käyttäytymistä (Justi & Gilbert, 2002). Mallit ovat usein matemaattisia, mutta niiden käsitteet ovat korkeammalla abstraktiotasolla. Matematiikan osa-alueen olioiden ja operaatioden sijaan mallit yhdistävät niitä kaavoiksi kuvaamaan kiinnostuksen kohteena olevia ilmiöitä. Matemaattista mallia voidaan ajatella kuvitteellisena ja yksinkertaistettuna versiona tutkittavan todellisuuden osasta, jossa tarkat laskutoimitukset ovat mahdollisia (Gowers, 2002). Newtonin mekaniikka on esimerkki tällaisesta mallista (katso Kuva 2.2).

$$1)\ \sum F = 0 \Leftrightarrow \frac{dv}{dt} \qquad 2)\ F = \frac{dp}{dt} = \frac{d(mv)}{dt} \qquad 3)\ F_A = -F_B$$

Kuva 2.2. Newtonin mekaniikka on toinen nimi fysiikan klassiselle mekaniikalle. Isaac Newton loi perustan klassiselle mekaniikalle määrittelemällä kolme liikkeen lakia: Ensimmäinen laki määrittelee, että kappale pysyy paikallaan tai jatkaa liikettä, ellei ulkopuolinen voima vaikuta siihen (1), Toinen laki määrittelee, että kappaleeseen vaikuttavien voimien summa on kappaleen massa kerrottuna sen kiihtyvyydellä (2) ja Kolmas laki määrittelee, että jos ensimmäinen kappale vaikuttaa toiseen kappaleeseen tietyllä voimalla, niin toinen vaikuttaa ensimmäiseen yhtä suurella, mutta vastakkaisuuntaisella voimalla (3).

Edellä annettu tulkinta tieteestä perustuu ajatukseen, että ilmiön kuvausta voidaan yksinkertaistaa riittävästi, jotta sen ominaisuudet voidaan laskea muuttujien arvoista. Kaikkia ilmiöitä ei kuitenkaan voida kuvata niin yksinkertaisesti, esimerkiksi ihmispopulaatioiden yhteisiä piirteitä, säätrendejä, talousennusteita tai tautien leviämistä kuvaavien muuttujien arvot sisältävät vaihteluja, jotka on otettava huomioon. Yksittäisten ja tarkkojen arvojen sijaan meillä on suuri määrä tietoa, joissa ilmiön yhtä muuttujaa edustavat useat arvot. Kaikki ilmiöt eivät myöskään ole staattisia, joten kerätyt arvot voivat vaihdella myös mittaushetken mukaan. Tietoa voidaan käyttää myös ennusteisiin ja itse ennusteen lisäksi on hyödyllistä arvioida, kuinka varma tai todennäköinen tulos on. Tilastollisia malleja käytetään datan (tietoa jota ei vielä ole tulkittu) vaihtelun ja epävarmuuden käsittelemiseen. Tavoitteena on saada käsitys siitä, mitä tietoa data sisältää.

Tilastollinen tutkimus on prosessi, joka sisältää viisi vaihetta: ongelma, suunnittelu, data, analyysi ja johtopäätös (Wolff et ai., 2016; Wild & Pfannkuch, 1999). Ensin tunnistetaan tutkimusongelma ja mitkä ovat siihen liittyvät kysymykset. Suunnittelu lähtee liikkeelle hypoteesin muodostamisesta, eli millaisia tuloksia oletetaan. Sitten tehdään suunnitelma, joka kuvaa, millainen data voisi vahvistaa tai kumota hypoteesin. Lisäksi luetellaan mahdolliset tietolähteet ja miten dataa voidaan kerätä. Data-vaiheessa tiedot kerätään tai hankitaan. Lisäksi on huomioitava datan laatu ja mahdolliset eettiset kysymykset, kuten suostumus, nimettömyys ja erilaiset luvat tiedon käyttämiseen. Yksi prosessin keskeisistä vaiheista on datan analyysi, jossa tulokset kerätään, ja niistä tehdään johtopäätöksiä. Lopuksi arvioidaan johtopäätösten paikkansapitävyyttä, ja mahdollistavatko tulokset uusien kysymysten esittämisen. Tilastollisen tutkimuksen perustana on useita menetelmiä ja työkaluja, joista aritmeettinen keskiarvoa ja keskihajonta ovat esimerkkejä (ks. Kuva 2.3).

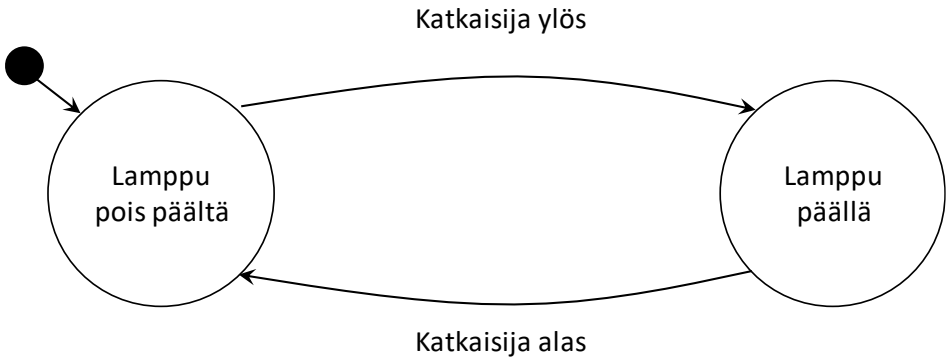
$$1)\ A = \frac{1}{n}\sum_{i=0}^{n} a_i = \frac{a_1+a_2+\cdots+a_n}{n} \qquad 2)\ \sigma = \sqrt{\frac{\sum(x_i-\mu)^2}{N}}$$

Kuva 2.3. Tilastolliset perusmenetelmät, kuten aritmeettinen keskiarvo ja keskihajonta, antavat yhteenvedon numeerisesta datasta löydetyistä säännönmukaisuuksista. Aritmeettinen keskiarvo edustaa muuttujan keskiarvoa aineistossa (1). Keskihajonta kuvaa, kuinka paljon muuttujien arvot poikkeavat aineiston keskiarvosta (2).

Eri tekniikan alat hyödyntävät matematiikan ja luonnontieteiden malleja. Ohjelmoinnillisen ajattelun kuuluu myös tekniset mallit, jotka liittyvät sähköisten, elektronisten ja tietokonelaitteiden ohjaukseen sekä automatisointiin. Sähkölaitteen ohjaamiseksi ohjelmallisesti on sen oltava tietokone tai siinä on oltava riittävä elektroniikka sellaisen kanssa kommunikoimiseen. Yksinkertaisissa tapauksissa laite voisi käyttää elektroniikkaa automaatioon ja ohjaukseen, mutta monimutkaisempiin tarpeisiin on helpompi lisätä tietojenkäsittelykapasiteettia kuin yrittää rakentaa sitä pelkillä elektronisilla komponenteilla. Tietokoneiden ja elektroniikan toiminta perustuu sähköisiin tiloihin, mikä tarkoittaa, että jokin sähkövaraus tai -virta on joko päällä tai pois päältä. Tämä ei tarkoita, että laitteen on oltava staattinen. Tilana voi olla, että moottori käy tai jopa kiihdyttää. Laitteen ohjaamiseksi tai automatisoimiseksi tarvittavista tiloista voidaan luoda malli.

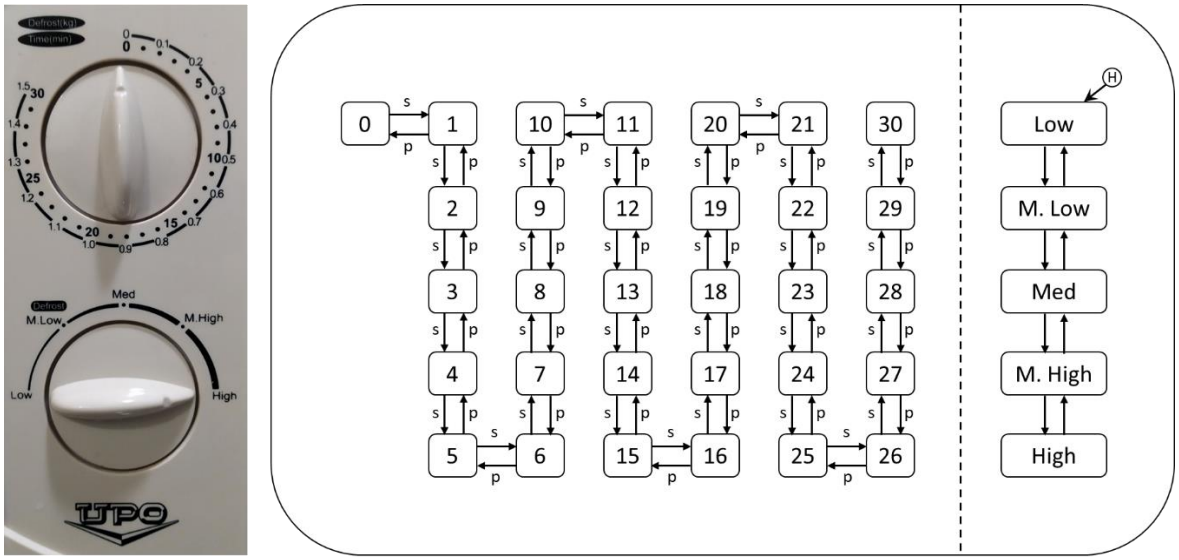
Laitteen ohjausmekanismin tai automatisoinnin malli tarvitsee aloitustilan, tilojen väliset siirtymät ja syötteet, jotka ohjaavat siirtymiä. Tällaista mallia kutsutaan äärelliseksi tilakoneeksi,

koska tilojen lukumäärä on rajallinen (katso kuva 2.4). Tilakone on yksinkertaisin laskennallinen malli, jolla on muisti. Yhdistelmälogiikka on yksinkertaisempi, koska se vain laskee tuloksen, mutta koska sillä ei ole muistia, se ei pysty käsittelemään tiloja. Tilakoneen tilan muuttuminen on seurausta saadusta syötteestä. Tietyssä tilassa voidaan yleensä siirtyä vain osaan muista tiloista. Tilakoneen määrittelee tilaluettelo, alkutila ja syötteet. Vaikka tilakone on toimintaperiaatteeltaan yksinkertainen voidaan sillä mallinta monenlaisia laitteita. Tilakone on abstraktio, se kuvaa laitteen toiminnan tuloksia, ei todellista mekaniikkaa.



Kuva 2.4. Yksinkertainen äärellinen tilakone, joka kuvaa hehkulampusta ja katkaisimesta koostuvaa järjestelmää. Tilakone kuvaa lampun tiloja, joita katkaisimen asento vaihtaa. Katkaisimen kääntäminen ylös sytyttää lampun ja sen kääntäminen alas sammuttaa lampun. Oletus- tai aloitustilassa on valo pois päältä ja katkaisin on alhaalla.

Tilakoneiden malleista voi tulla hyvin monimutkaisia, jos syötteitä ja tiloja on paljon, koska jokainen yhdistelmä on kuvattava. Harel (1987) kehitti tilakaaviot helpottamaan tilakoneen piirtämistä (ks. kuva 2.5). Tilakaavioissa käytetään erilaisia graafisia keinoja, joilla mallia voidaan yksinkertaistaa: ryhmittely, oletustila, ja-tila ja historia-nuoli (Thimbleby, 2007). Jos tietyssä tilassa syötteet aiheuttavat vaihtoehtoisia siirtymiä voidaan nämä visuaalisesti ryhmitellä yhdeksi tilaksi, joka sisältää useamman tilan. Tila, joka on aina ensimmäinen tilaryhmässä, voidaan merkitä oletustilaksi. Jos tilat jakavat osan piirteistä, mutta niissä on myös eroja, voidaan tämä mallintaa ja-tilalla. Visuaalisesti ja-tila on kahtia jaettu tilaryhmä, joka mahdollistaa osatilojen yhdistämisen. Joskus tilaryhmän viimeinen tila on muistettava, ja tämä voidaan merkitä historia-nuolella.



Kuva 2.5. Tilakaavio, jossa kuvataan yksinkertaisen mikroaaltouunin käyttöliittymän toimintaa. Käyttöliittymä koostuu ajastimen valitsimesta (0-30) ja tehotason asetuksesta (low-high). Tilakaaviossa s:llä merkityt nuolet kuvaavat ajan (sekuntteina) asettamista tilasiirtyminä ja p:llä merkityt nuolet tilojen muuttamista aikapulssilla. Tilakaavion käyttäminen merkintätapana yksinkertaistaa kahden tilan yhdistelmän kuvaamista sallimalla niiden esittämisen jaetttuna tilaryhmänä (ja-tila), joka mahdollistaa kummankin tilan muuttumisen erikseen. Tehotaso (tilakaavion oikea reuna) muistaa (H) viimeisimmän tilansa.

Abstraktio

Edellä kuvatut matematiikan, luonnontieteiden ja tekniikan mallit ovat esimerkkejä abstraktioista. Matematiikan olioita ei ole todellisuudessa, vaan ne ovat mielemme tuotteita. Filosofi Karl popper erotti ihmisen kokeman todellisuuden kolme tasoa: fyysinen maailma, ajatustemme maailma ja kulttuurimme aineettomat luomukset. Matematiikka on kulttuurin tuottamaa (Hersh, 2014), sitä on kehitetty vuosituhansien ajan ja sen tietoutta on välitetty matematiikan opetuksessa ja kirjallisuudessa. Tiede käyttää matemaattisia ja tilastollisia malleja kuvaamaan tutkittavia ilmiöitä. Mallit ovat matematiikan osa-alueita korkeammalla abstraktiotasolla, koska ne ovat yleensä matemaattisten muuttujien sekä operaatioden yhdistelmiä ja ne on nimetty ilmiön tai sen ominaisuuksia kuvaavilla käsitteillä. Insinööritieteissä käytetään tietojenkäsittelyn tilakoneita laitteiden toiminnan abstraktiona. Tilakoneet eivät kuvaa, miten laitteet toimivat, vaan ne mallintavat toiminnan tuloksia. Yleisesti ottaen jonkin asian abstrahointi tarkoittaa yksityiskohtien jättämistä pois tai toisiinsa liittyvien yksityiskohtien ryhmittelyä kuvaavan nimityksen alle.

Matematiikka, luonnontiede ja tekniikka käyttävät kaikki abstraktioita. Abstraktio on keskeinen käsite myös tietojenkäsittelytieteessä (Kramer, 2007) ja siksi myös ohjelmoinnnillisessa ajattelussa (Wing, 2006). Tietojenkäsittelytieteessä on ajateltava usealla abstraktiotasolla, monasti kahdella tasolla samanaikaisesti. Yksi esimerkki tietojenkäsittelytieteen opetuksen yhteydessä käytetystä abstraktiotasoista on Perrenet et al. (2005) esittämä nelitasoinen hierarkia: ongelma, olio, ohjelma ja suoritus (ks. luettelo jäljempänä). Hierarkian tasot kuvaavat erilaisia

tulkintoja algoritmista ongelman ratkaisemisessa ohjelmoimalla. Ongelman tasolla algoritmia edustaa syötteen ja tuloksen suhde. Oliotasolla suunnitellaan algoritmin askeleet, joita tarvitaan syötteen muuttamiseksi tulokseksi. Ohjelman tasolla algoritmi toteutetaan ohjelmointikielellä. Suorituksen tasolla algoritmia edustaa ohjelman määrittelemä tietokoneen fyysinen toiminta. Ohjelmoinnillista ajattelua voidaan pitää kykynä edetä hierarkiassa ongelman ratkaisemiseksi.

Algoritmin abstraktiotasot ohjelmoinnissa (Perrenet et al. 2005):

- Ongelman taso: muutos syötteistä tulokseksi
- Oliotasolla: algoritmin kuvaus
- Ohjelman tasolla: algoritmin toteutus ohjelmointikielellä
- Suorituksen tasolla: tietokone suorittaa ohjelman

Analyysi

Analyysissä ongelma pilkotaan pienempiin ja hallittavampiin osiin. Ohjelmoinnillisen ajattelun yhteydessä analyysin tavoitteena on löytää algoritmi, joka ratkaisisi ongelman. Ennen analyysiä ongelma on kuin musta laatikko, josta tiedetään vain syötteet, tulos ja mahdollisesti miten ongelma liittyy muihin ongelmiin. Musta laatikko edustaa ongelman ratkaisua, jota ei vielä tunneta. Ratkaisun selvittämiseksi ongelma jaetaan osaongelmiin. Jokaista osaongelmaa voidaan käsitellä omana ongelmana, joka on pilkottava osiin. Lopulta saavutetaan atomitaso, jolla ei ole enää osaongelmia. Ratkaisu voidaan suunnitella ongelman osien perusteella. Ongelmanratkaisua voidaan lähestyä ylhäältä alaspäin käsittelemällä ensin ylätason ongelmat tai alhaalta ylöspäin ratkaisemalla ensin atomiset ongelmat.

Analyysiprosessi sisältää kaksi vaihetta (Rich et al., 2019). Ensimmäinen, sisällön analyysi, on ongelman osien (osaongelmien) erottelemista ja luokittelua. Toisessa, suhteiden analysoinnissa etsitään osaongelmien välisiä yhteyksiä. Sisällön analysoinnissa on valittava periaate, jonka avulla ongelma jaetaan osiin. Periaate on riippuvainen asiayhteydestä ja ongelmanasettelusta. Osaongelmat tuovat uutta tietoa, joka ei ollut havaittavissa kokonaisuudesta. Kun osaongelmat on tunnistettu, voidaan myös tutkia niiden välisiä suhteita. Ennen ongelman pilkkomista suhteita ei voitu havaita. Osaongelmat ja niiden väliset suhteet muodostavat ongelman mallin. Esimerkki aritmetiikasta olisi kahden luonnollisen luvun yhteenlasku. Jos halutaan lisätä yksi luku toiseen, voidaan ottaa ensimmäisen luvun sijainti lukusuoralla lähtökohdaksi ja lasketaan siitä eteenpäin määrä, joka vastaa toisen luvun sijaintia nollasta. Uuden sijainnin luku on yhteenlaskun tulos.

Tekniikassa, tietojenkäsittelytieteessä, suunnittelussa ja muilla aloilla ongelmat jaetaan usein niiden sisältämien funktionaalisten suhteiden mukaan (Rich et al., 2019). Funktioanalyysi on seurausta sisällön ja suhteiden analysoinnin yhdistelmästä. Keskeistä funktioanalyysissä on osaongelmien välinen suhde. Jos suhde edustaa funktiota, tapahtuu kahden osaongelman välillä jokin operaatio. Aritmetiikassa funktionaalinen suhde voi olla yhteen-, vähennys-, kerto- tai jakolasku. Esimerkkejä aritmeettisten operaatioiden käytöstä muilla matematiikan osa-alueilla ovat toisenasteenyhtälö, suurin yhteinen tekijä tai Pythagoraan lause. Tietojenkäsittelytieteen algoritmeissa, jotka on ilmaistu imperatiivisella (tietokonetta ohjaavalla) ohjelmointikielellä, käytetään ohjausrakenteita, jotka muodostavat suhteet osaongelmien ratkaisujen välillä. Ongelman analysointi osaongelmien ja niiden suhteiden avulla tukee (algoritmisen) ratkaisun suunnittelua (Rich et al., 2019).

Algoritmit

Algoritmit ovat osa matematiikkaa. Joskus mitä tahansa vaiheittaista prosessia kuvataan algoritmiksi, mutta algoritmi ei ole prosessi vaan määritelmä. Määritelmän on kuvattava laskutoimituksen prosessi matemaattisen tarkasti. Tietotekniikka perustuu algoritmeihin, koska sen toimintaperiaate perustuu laskennalliseen malliin (ks. Osio 1/Matematiikka). Kaikki, mitä tietokone tekee on laskemista, mutta jotta se tietäisi, mitä tehdä, on jokainen laskutoimituksen askel määriteltävä. Tietojenkäsittelytieteen kontekstissa algoritmi voidaan määritellä seuraavasti: *"...äärellinen, abstrakti, tehokas, ohjausrakenteiden yhdistelmä, joka toteuttaa määrätyn päämäärän tietyin ehdoin."* (Hill, 2016, s. 47).

Robin K. Hillin (2016) algoritmin määritelmän (ks. edellä) tulkinta voidaan aloittaa ajattelemalla algoritmi erillisistä vaiheista muodostuvan prosessin määritelmänä. Algoritmi kuvaa äärellistä prosessia, jonka on päätyttävä. Se on abstrakti ja siinä käytetään vain niitä piirteitä, joita tarvitaan prosessin muutoksen kuvaamiseen. Äärellisyyden lisäksi algoritmin on oltava tehokas, mikä tarkoittaa, että resurssien ja ajan käyttö on oltava kohtuullista. Algoritmi on erilaisten kontrollirakenteiden yhdistelmä, jotka ohjaavat muuttujien arvojen käsittelyä, joka päättyy lopputulokseen. Ohjausrakenteet ja muuttujien käsittely esitetään käskyjen avulla imperatiivina. Algoritmin on täytettävä tehtävänsä eikä mitään muuta. Algoritmin tulisi käyttäytyä oikein millä tahansa annettujen rajojen sisällä olevalla syötteellä.

Ohjelmoinnillinen ajattelu voidaan määritellä henkisiksi kyvyiksi, joita tarvitaan algoritmin suorittamisen automatisoimiseksi (Denning, 2017b). Edellä määritellyt algoritmin ominaisuudet ovat edellytys sille, että se voidaan toteuttaa, se käyttäytyy odotetulla tavalla ja että tulos on oikea. Toteuttaminen tarkoittaa ohjelman kirjoittamista, joka sisältää algoritmin. Vaikka jokainen ohjelma on algoritminen, niin algoritmeiksi kutsutaan vain niitä ohjelman osia, jotka ovat riittävän yleisiä, jotta niitä voidaan käyttää uudelleen. Näille algoritmeille annetaan usein nimi ja niiden määritelmiä jaetaan tietojenkäsittelytieteilijöiden kesken esim. julkaisuissa ja konferensseissa. Suuri osa ohjelmakoodia liittyy tietokoneen hallintaan tietyssä kontekstissa, eikä koodia voi käyttää muuhun tarkoitukseen. Ohjelma kirjoitetaan suhteessa tietokoneen ominaisuuksiin. Tietokonetta voitaisiin ohjata suoraan, mutta se on hyvin vaikeaa ja virhealtista. Yleensä ohjelmoijat työskentelevät tietokonelaitteistoa korkeammalla abstraktiotasolla olevan mallin kanssa, jota ohjelmointikieli kuvaa.

Ohjelmointikielen kielioppia kutsutaan syntaksiksi. Semantiikka määrittelee puolestaan kielen lauseiden merkityksen. Merkitys voi viitata laskutoimituksiin tai tietokoneen toimintaan. Ohjelma on luettelo tekstirivejä, joita kutsutaan lauseiksi. Lauseet suoritetaan yksi kerrallaan tietyssä järjestyksessä. Lause koostuu komennosta ja arvoista, joita tarvitaan sen suorittamiseksi. Arvot voidaan antaa suoraan, mutta yleensä käytetään jotain kirjainsymbolia sijaisarvona kuten koulualgebrassa. Symbolia kutsutaan muuttujaksi, jos sitä voidaan muuttaa, ja vakioksi, jos arvo on pysyvä, kun se on kerran asetettu. Joskus lause sisältää matemaattisia tai loogisia lausekkeita, jotka laskevat uusia arvoja muuttujien/arvojen perusteella. Komennon ja lausekkeen ero on siinä, että ensimmäinen ohjaa tietokoneen toimintaa ja jälkimmäinen tuottaa arvoja. Lausekkeen tulos toimii komennon syötteenä. Ohjelman suoritus päättyy viimeiseen lauseeseen, jollei ole lausetta, joka määrittelisi muuta.

Ohjelma kirjoitetaan yksi lause jokaiselle riville. Ohjelman suoritus käy läpi jokaisen rivin. Ohjausrakenteet muuttavat suorituksen etenemistä. Ehdollisen lauseen avulla voidaan valita vaihtoehtoinen lauseiden ketju. On mahdollista, että suoritus palaa pääketjuun sen jälkeen, kun vaihtoehtoiset lauseet on suoritettu. Toistolause on rakenne, joka toistaa joukon lauseita, kunnes jokin ehto täyttyy. Kun toistolauseen suoritus on päättynyt, niin ohjelman suoritus jatkuu, jos lauseita on vielä jäljellä. Ohjelmat voivat kasvaa suuriksi, mikä vaikeuttaa ohjelmakoodin ymmärtämistä. Joskus ohjelman jonkin osan toiminnallisuus olisi hyödyllistä ohjelman toisessa osassa. Funktiot ja proseduurit mahdollistavat modulaarisuuden ja uudelleenkäytön. Ne ovat aliohjelmia, joita voidaan nimetä ja kutsua kuten kielen olemassa olevia komentoja. Funktio ja proseduuri eroavat toisistaan siinä, että ensimmäinen palauttaa arvon ja toinen muuttaa tietokoneen tilaa.

Python ja Racket ovat esimerkkejä opetuksessa käytettävistä tekstimuotoisista ohjelmointikielistä. On olemassa myös aloittelijoille suunnattuja visuaalisia ohjelmointikieliä, joissa ohjelmointi muistuttaa palapelin rakentamista. Palapelipohjaista lähestymistapaa visuaaliseen ohjelmointiin kutsutaan myös lohkopohjaiseksi ohjelmoinniksi (Weintrop, 2019). Tässä ohjelmointityypissä kielen komennot esitetään visuaalisina palikoina. Lohkojen muoto antaa vihjeitä siitä, miten ja missä niitä voidaan hyödyntää. Tällaisia ohjelmointikieliä voi jo käyttää viisivuotiaat, mutta useimmiten kielten kohderyhmänä ovat lapset kahdeksan ja kuudentoista välillä. Ohjelman kirjoittaminen on ohjelmalohkojen yhdistelemistä vetämällä ja pudottamalla. Yleensä ohjelmointiympäristö estää yhteensopimattomien lohkojen yhdistämisen. Tämä mahdollistaa ohjelman kirjoittamisen lause lauseelta kuten tekstimuotoisissa ohjelmointikielissä, mutta vältytään syntaksivirheiltä, joita tulee helposti ohjelmointikieltä opeteltaessa. Koska ohjelmointi tapahtuu vetäämällä ja pudottamalla, ohjelmointiympäristö voi tarjota lisätukea ryhmittelemällä lohkot toiminnon mukaan ja käyttämällä värejä tunnistamisen helpottamiseksi. Ohjelmointikielten mahdollisuuksien tunteminen ja käsitteiden muistaminen korvataan komentojen selaamisella.

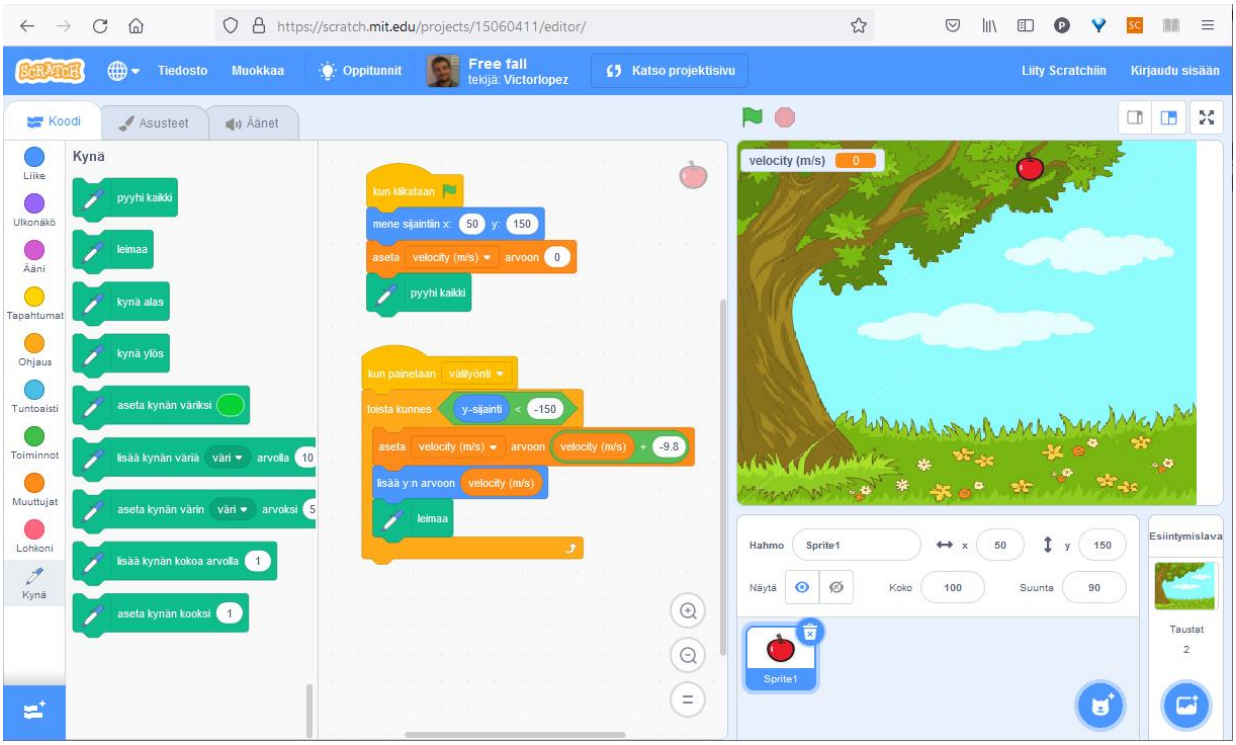
Lohkopohjaisen ohjelmoinnin sivuvaikutuksena käyttäjä vapautuu kirjoittamisesta. Käyttäjän ei tarvitse huolehtia kirjoitusvirheistä, oudoista välimerkeistä tai puuttuvista syntaksin osista. Komentojen graafinen esittäminen mahdollistaa pidempien kuvausten käytön, koska komentoja ei tarvitse kirjoittaa. Se antaa myös mahdollisuuden kuvailla merkitystä, koska tietokoneen tarvitsema tarkkuus voidaan piilottaa käyttäjältä. Lohkopohjaisen kielen lisäksi ohjelmointiympäristöön voidaan lisätä myös muita apuvälineitä. Scratchissa (Maloney et. al., 2010), joka on tällä hetkellä (vuonna 2020) tunnetuin lohkopohjainen ohjelmointikieli, on tehty erityisen helpoksi spriteiksi kutsuttujen näytön graafisten elementtien hallinta. Jokaisella spritellä on oma ohjelmansa, joka ohjaa sen käyttäytymistä. Yksi spriten tärkeimmistä ominaisuuksista on sen sijainti. Sijaintia muuttamalla on helppo saada asioita tapahtumaan ruudulla. Koska spritet ovat itsenäisiä kokonaisuuksia, kuten muiden kielten funktiot tai proseduurit, niitä voidaan helposti siirtää ohjelmasta toiseen.

Esimerkki: Painovoiman mallintaminen Scratchissä

Lopez ja Hernandez (2015) loivat esimerkin fysiikan projektista nimeltä "Vapaa putoaminen", joka voisi olla tehtävä ala- tai yläkoulun oppilaille. Projekti toteutettiin Scratch-ohjelmalla, ja

siinä mallinnettiin, miten painovoiman seurauksena esineen nopeus kiihtyy vapaassa pudotuksesa. Fysiikan ilmiö visualisoitiin animaatiolla omenapuusta, jonka yhdestä oksasta omena putoaa. Putoavan omenan nopeus näytetään arvona ja kiihtyvyys animoidaan tulostamalla omenan muuttuva sijainti kasvavin askelin. Tehtävän suorittamiseksi oppilaan on otettava selvää painovoimasta ja siitä, millainen kaava kuvaa sen vaikutuksia putoavaan kappaleeseen. Kaavan avulla olisi suoraviivaista laskea omenan nopeus oksan ja maan välisen matkan lopussa. Haasteena on käyttää Scratchin ominaisuuksia kiihtyvyydestä johtuvan nopeuden kasvamisen osoittamiseen.

Voisi ajatella, että alakoulun oppilaille ohjelman tulos näytettäisiin, mutta yläkoulun oppilaiden tehtävänä olisi itse keksiä, miten omenan putoamisen kasvava kiihtyvyys visualisoitaisiin. Lopezin ja Hernandezin esimerkin perusteella mallintamisen voisi aloittaa lisäämällä lavalle puun kuva, piirtämällä omenaa kuvaava sprite ja siirtämällä omena oikeaan sijaintiin kokeilemalla (ks. kuva 2.6 oikealla). Sitten on keksittävä, miten kiihtyvyysmalli yhdistetään omenan putoamiseen. Sijainnin muutos suhteessa aikaan (sekuntiin) olisi nopeus ja kiihtyvyys olisi omenan uuden sijainnin tulostaminen tasaisin väliajoin nopeuden kasvamisen huomioiden. Omenan lähtönopeus olisi nolla (ks. kuvan 2.6 keskellä oleva ensimmäinen lohkosarja).



Kuva 2.6. Kuvakaappaus Scratch-ympäristöstä, jossa on kuva puusta ja sen oksalla olevasta omena spritestä, jota käytetään visualisoimaan omenan putoaminen painovoiman vaikutuksesta.

Esimerkkisovelluksessa sekunti kuvataan yhtenä toistolauseen kierroksena (ks. aiemmin Kuva 2.6: keskellä toinen lohkosarja). Nopeus kuvataan omenan sijainnin muutoksena yhden kierroksen aikana. Ensin otetaan huomioon kiihtyvyyden vaikutus lisäämällä nopeuteen 9,8 (set velocity (m/s)...) ja muutetaan omenan sijaintia korkeussuunnassa (change y by velocity (m/s); ks. kuva 2.6: keskellä toinen lohkosarja). Leimaustoimintoa (stamp) käytetään toistolauseen

jokaisella kierroksella omenan tulostamiseen ruudulle (kynä ja leima; ks. aiemmin Kuva 2.6: keskellä toinen lohkosarja). Toistolause lopetetaan, kun omenan sijainti korkeussuunnassa on kuvan alareunassa. Kun malli on saatu toimimaan, on mahdollista kehittää mallia todentuntuisemmaksi. Oppilaat voisivat lisätä ilman kitkan tai mallintaa tuulen vaikutusta.

Lisätietoja esimerkkisovelluksesta: Scratch laskennallisen mallintamisen välineenä fysiikan opetuksessa (ks. viitteet Lopez & Hernandez, 2015).

Toteutus Scratchissa (vierailtu 30.8.2020): https://scratch.mit.edu/projects/15060411/.

Yhteenveto

Edellisessä esimerkissä on käytetty kaikkia ohjelmoinnillisen ajattelun PRADA-mallin taitoja: hahmontunnistus, abstrahointi, analyysi ja algoritmit. Nämä ovat taitoja, joita oppilaat tarvitsevat, jotta he osaavat hyödyntää tietojenkäsittelyn mahdollisuuksia LUMA-aineissa. Ohjelmoinnillisen ajattelun taidot on esiteltävä oppilaille ensin erikseen ja sitten yhdistettynä kokonaisuudeksi. On tärkeää, että ohjelmointia ei esitellä vain monimutkaisena laskimena, vaan kerrotaan tietojenkäsittelyn mahdollisuuksista. Yllä olevassa esimerkissä oppilaat tutustuivat painovoimaan fysiikan ilmiönä, rakensivat laskennallisen mallin ja käyttivät Scratch-ympäristön ominaisuuksia sen visualisointiin. Oppilaiden kypsyydestä riippuen voidaan esittää teoreettisia selityksiä ohjelmoinnnillisesta ajattelusta, mutta ilman käytännön harjoittelua sitä ei voida oppia. Myös opettajan on harjoiteltava ohjelmointia, jotta hän pystyy näyttämään miten ohjelmia luodaan.

Luento - Ohjelmoinnillisen ajattelun näkökulmat STEAM-opetukseen

Hyödyntäen ylläolevaa tekstiä

- Kolme tietojenkäsittelytieteen näkökulmaa
- Hahmontunnistus
- Abstraktio
- Analyysi
- Algoritmi suoritettuna tietokoneella

Tehtävä 2.1 kotitehtävä - oman aiheeseen liittyvän tehtävän kuvaus käyttäen Ohjelmoinnillisen ajattelun näkökulmia STEAM-opetukseen -viitekehystä

Tämän harjoituksen tavoitteena on soveltaa ohjelmoinnillisen ajattelua johonkin oppiaineesi aiheeseen. Painovoiman mallintaminen Scratchillä (ks. edellä) oli esimerkki, jossa fysiikan painovoimaa oli mallinnettu ja visualisoitu hyödyntäen tietotekniikkaa. Sinun tulisi tunnistaa

valitsemassasi aiheessa, miten ohjelmoinnillinen ajattelu voisi auttaa ongelman ratkaisemisessa, tietotekniikan hödyntämisessä tai ilmiön ymmärtämisenä tietojenkäsittelyprosessina.

1. Valitse mielenkiintoinen aihe oppiaineestasi.

2. Mikä tietojenkäsittelyn näkökulma sopii aiheeseesi?
   - Matematiikka: mitä ongelmia voidaan ratkaista tietojenkäsittelyllä?
   - Tekniikka: mitä toimintoja voidaan toteuttaa ohjaamalla informaatioteknologioita?
   - Tiede: mitä ilmiöitä voidaan ymmärtää tietojenkäsittelyprosesseina?

3. Sovella PRADA-mallia valitsemaasi aiheeseen valitun näkökulman mukaisesti
   - Hahmontunnistus
   - Abstraktio
   - Analyysi
   - Algoritmi suoritettuna tietokoneella

4. Millaisen tehtävän voisit keksiä aiheestasi ohjelmoinnillisen ajattelun soveltamisen perusteella?
   - Mikä on oppilaiden ikä tai kouluaste?
   - Kuvaile tehtävän ideaa (toteutuksen yksityiskohdat voi tässä ohittaa).

Tehtävä 2.2 Kotitehtävä - Vertaisarviointi

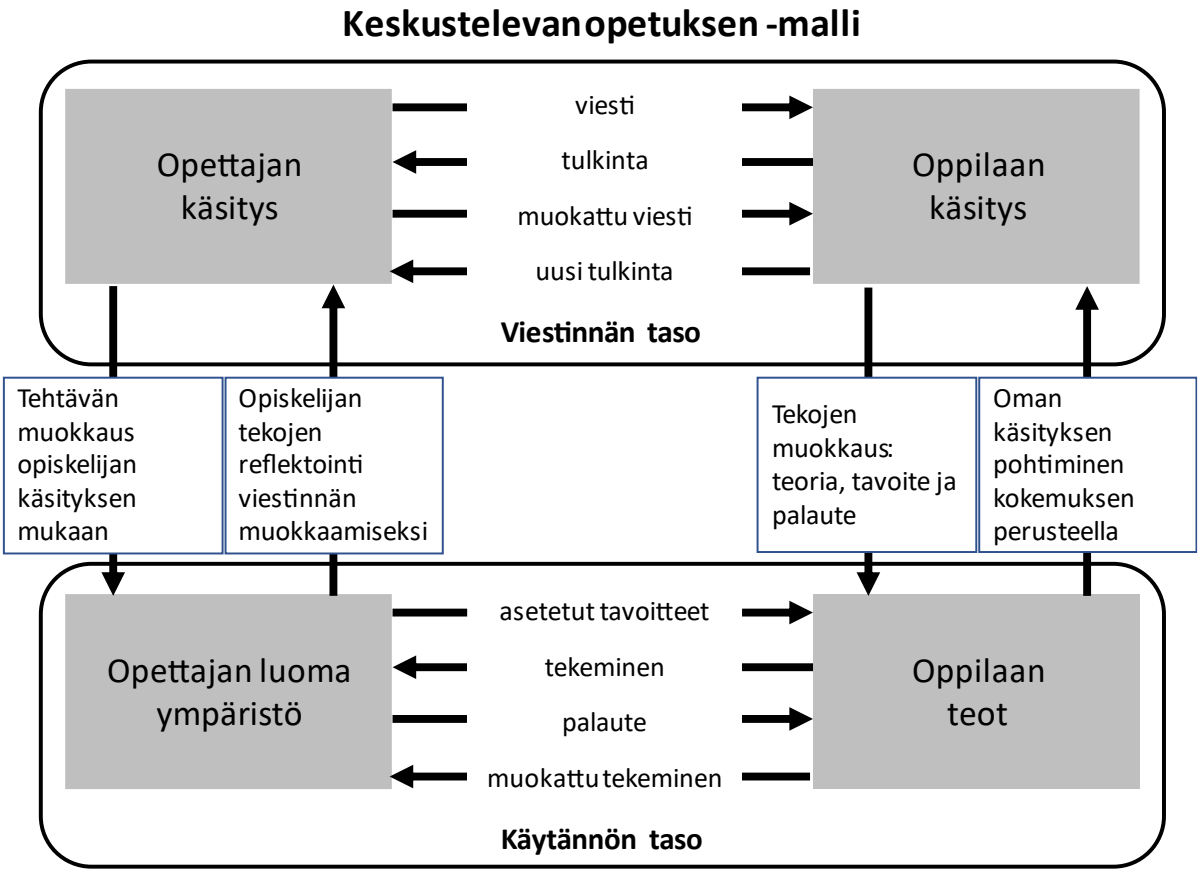
- Arvioi kolmen kolleegasi suunnittelemat tehtävät (Tehtävä 2.1)
- Arviointi on hyväksytty tai hylätty
- Jokainen alla oleva kohta on skaalalla 0-5 (hyväksytyssä suorituksessa keskiarvon on oltava 2 tai parempi)
  - tehtävän aiheen esitys
  - perustelut ja selkeys
  - luvun sisällön hyödyntäminen

Osio 3 – Opetusteknologian valinta Ohjelmoinnillisen ajattelun näkökulmien avulla

Nykyään oppilaat käyttävät sujuvasti digitaalisia teknologioita, mutta se ei välttämättä tarkoita, että he ymmärtäisivät niiden toimintaperiaatteita. Heidän on ensin harjoiteltava teknologioiden käyttöä organisoidummin, jotta he voivat myöhemmin soveltaa niitä vapaammin. STEAM-opetuksessa on kyse eri alojen tietojen ja taitojen yhdistämisestä ongelmien ratkaisemiseksi. Oppilaat voivat hyötyä sekä yleisten että tiettyyn aiheeseen suunnattujen opetusteknologioiden käytöstä työkaluina. Molemmissa tapauksissa opettaja voi käyttää Ohjelmoinnillisen ajattelun näkökulmat STEAM-opetukseen -viitekehystä tunnistaakseen opetusteknologioita, joissa

yhdistyy aiheen käsittely ja tietotekniikan mahdollisuudet. Tässä osiossa viitekehystä käytetään osana opetuksen mallia, joka auttaa valitsemaan valittuun pedagogiikkaan sopivat työkalut. Osiossa esitetään esimerkkejä viitekehyksen kolmen näkökulman mukaisista opetusteknologioista. Lopuksi pohditaan teknologian käyttöä ja miten se vaikuttaa opetukseen.

Keskustelevan opetuksen -mallissa (The Conversational Framework; ks. Kuva 3.1) oppimista kuvataan opettajan ja oppilaiden välisenä vuoropuheluna (Laurillard, 2002; Laurillard, 2012). Mallissa on kaksi tasoa: viestintätaso, jossa keskustellaan teorioista ja käsitteistä, ja käytännön taso, jossa oppilaan teot ja opettajan palaute antavat oppilaalle uusia kokemuksia. Viestintätasolla opettaja jakaa ajatuksiaan teorioista, ja oppilaat vastaavat omilla tulkinnoillaan. Jos viesti ymmärretään väärin, opettaja voi muokata sitä, ja oppilaat voivat vastata uusilla tulkinnoilla. Monessa aiheessa on myös käytännön ulottuvuus. Käytännön tasolla opettaja luo ympäristön, jossa oppilaat voivat soveltaa teorioita tekemällä tehtäviä. Opettaja voi muokata tehtävät sopivalle vaatimustasolle käydyn keskustelun perusteella. Opettaja asettaa tavoitteet tehtäville, joihin oppilaiden tulisi pyrkiä. Tehtävien tekemisestä saadun palautteen perusteella oppilaat voivat pohtia osaamistaan ja tarvittaessa korjata ratkaisutapojaan. Opettaja kehittää saadun kokemuksen perusteella aiheen opetusta.



Kuva 3.1. Keskustelevan opetuksen -malli kuvaa opettajan ja oppilaan vuorovaikutusta sekä viestinnän tasolla että opettajan tarjoamien tehtävien kautta. Sekä opettaja että oppilas muokkaavat toimintaansa viestinnän ja tehtävistä saatujen kokemusten perusteella. Opettaja luo

oppilaalle ympäristön, jossa hän voi soveltaa tietojaan, ja opiskelija muokkaa toimintaansa palautteen perusteella, joka vaikuttaa myös hänen käsitykseensä opiskeltavasta aiheesta.

Keskustelevan opetuksen -malli antaa yksinkertaistetun kuvan opetuksesta, mutta siinä on riittävästi yksityiskohtia, jotta se kattaa yleisimmät oppimistavat (Laurillard, 2012). Se kattaa oppimisen tiedonhankintana, tutkimuksena, harjoitteluna, konstruktiona, keskusteluna ja yhteistyönä. Mallin tavoitteena on auttaa opettajaa opetuksen suunnittelussa. Tässä osiossa olemme erityisen kiinnostuneita opetusteknologioiden hyödyntämisestä osana oppimisympäristön suunnittelua (ks. edellä Kuva 3.1, alempi puoli). Opettaja voi hyödyntää opetusteknologioiden mahdollisuuksia suunnitellassaan tehtäviä, joiden avulla oppilaat voivat harjoitella opittua asiaa käytännössä. Harjoittelun toimivuuden kannalta opettajan palaute on keskeinen. Teknologia voi myös tukea palautteen antamista tai jopa automatisoida sen. Edesmenneen Seymour Papertin keksimät mikromaailmat (Papert, 1980; 1996) -konsepti on esimerkki siitä, millaisia oppimisympäristöjä voidaan luoda digitaalisilla teknologioilla.

Tietojenkäsittelyn näkökulmasta eri LUMA-aineet hyödyntävät tietotekniikkaa eri tarkoituksiin. Matematiikka tutkii, millaisia ongelmia voidaan ratkaista laskemalla. Tietokoneet tehtiin laskemista varten, ja vaikka tavoite ei olisikaan matemaattinen, ongelma on silti ratkaistava laskennan ohjaamina askelina (algoritmeina). Tekniikassa kyse on sähkölaitteiden ohjaamisesta. Monipuolisempaa ohjausta varten laitteessa on oltava tietotekniikkaa (nk. sulautetutjärjestelmät). Tieteiden ilmiöitä voidaan myös mallintaa tietojenkäsittelyprosesseina. Jos ilmiön käyttäytymistä voidaan kuvata matemaattisesti, niin tulos voidaan laskea tietokoneella. Monimutkaisia ilmiöitä voidaan kuvata tilastotieteen keinoin. Tilastotieteellä voidaan käsitellä vaihtelua, epävarmuutta ja suuria datamääriä. Vaikka LUMA-aineet käyttävät tietotekniikkaa eri tarkoituksiin, voidaan niissä kaikissa hyödyntää ohjelmoinnillista ajattelua ongelmanratkaisussa.

Ohjelmoinnillisen ajattelun näkökulmat STEAM-opetukseen -viitekehys tarjoaa hahmontunnistuksen, abstrahoinnin, analyysin ja algoritmit ongelmanratkaisukeinoina LUMA-aineisiin. Valitun opetusteknologian on tarjottava ominaisuuksia, jotka tukevat opiskeltavaa aihetta ja ohjelmoinnillista ajattelua. Opetusteknologian ominaisuuksien tulisi tarjota ratkaisuja osaongelmiin, joita yhdistelemällä oppilaat ratkaisevat heille annetun tehtävän (ongelma). Ongelman ratkaisun kuvaaminen syötteiden ja tuloksen suhteena on ongelmanratkaisua korkeimmalla abstraktiotasolla. Objektitasolla oppilaat määrittelevät algoritmin, joka kuvaa ongelman ratkaisuun tarvittavaa muutosta. Ohjelmatasolla oppilaat toteuttavat määrittelemänsä algoritmin ohjelmointikielellä tai käyttäen jotain muuta tietojenkäsittelyn esitystapaa. Ohjelman suorituksen tulos on ongelman ratkaisu. Opetusteknologian tulisi antaa oppilaille ratkaisusta palautetta, jotta he voivat arvioida tulosta ja tehdä tarvittaessa korjauksia.

Keskustelevan opetuksen -mallin tarkoituksena (tässä moduulissa) on auttaa opettajaa oppimisympäristön kehittämisessä. Uusista oppimisympäristöistä saatujen kokemusten jakamiseksi voidaan hyödyntää Pedagogisia suunnittelumalleja (Pedagogical Patterns) (Laurillard, 2012). Pedagoginen suunnittelumalli on tapa esittää, kokeilla ja jakaa digitaalisen teknologian tukeman opetuksen periaatteita ja käytäntöjä (Laurillard & McAndrew, 2003). Suunnittelumallin avulla voidaan kuvata oppitunnin tai sen osan kulkua. Lisäksi Pedagogiset suunnittelumallit luokitellaan, jotta opettajat voivat helpommin valita ja vertailla erilaisia opetusmenetelmiä. Luokittelu sisältää seuraavat luokat (ks. Taulukko 3.1): lähde (jos perustuu julkaistuun teokseen), yhteenveto, aihe, oppimistulokset, oppimistapa, kesto, oppilaan

pohjatiedot, oppimisympäristö ja ryhmäkoko. Pedagogista suunnittelumallia käytetään tässä apuvälineenä oppimisympäristön opetusteknologian ja sen hyödyntämisen kuvaamisessa.

Taulukko 3.1. Pedagogisen suunnittelumallin luokittelu (Laurillard, 2012).

Luokka	**Kuvaus**
Lähde	alkuperäinen lähde ja myöhemmät julkaisut aiheesta
Tiivistelmä	lyhyt kuvaus opetettavasta aiheesta ja opetusmenetelmästä
Aihe	avainsanat, jotka auttavat opettajaa mallin valinnassa
Oppimistulokset	mitä oppilas tietää tai osaa oppitunnin/opetustuokion jälkeen
Oppimistapa	oppimistapa tai pedagogisen toiminnan suunnitelma
Kesto	kokonaisaika, voi jakautua useammalle oppitunnille
Oppilaan pohjatiedot	tarvittavat ennakkotiedot, kokemus ja mielenkiinnon kohteet
Oppimisympäristö	lähiopetus, etäopetus, tai näiden yhdistelmä
Ryhmäkoko	oppilaiden minimi ja maksimi määrät

Opetusteknologioiden hyödyntämisen esimerkkejä LUMA-aineissa

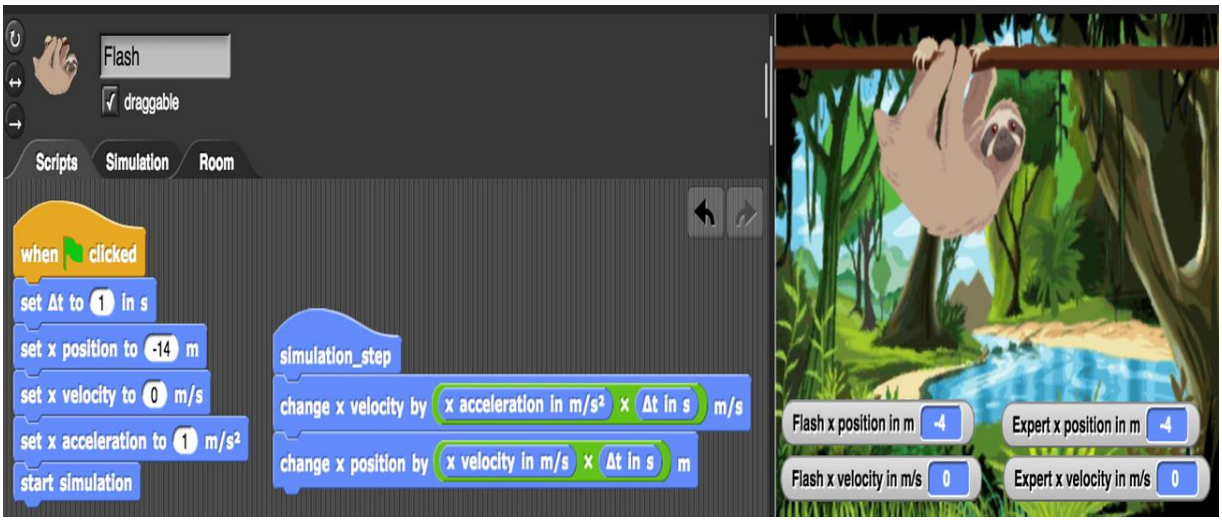
Tässä luvussa esitellään esimerkkejä LUMA-aineisiin liittyvistä opetusteknologioista. Kukin esimerkki esitetään Pedagogisena suunnittelumallina, jossa teknologia on osa opetuksen ja oppimisen sykliä. Opetusteknologia ja sen käytön kuvaus muodostavat kokonaisuuden. Voidaan keksiä uusi tapoja käyttää teknologiaa, mutta silloin on kyse myös uudesta mallista. Lisäksi esitetään luokittelu, joka kertoo suunnittelumallin pääpiirteet ja auttaa lukijaa päättämään, onko malli tarkemman läpikäynnin arvoinen. Seuraavaksi esitetään esimerkkejä tieteen, tekniikan ja matematiikan Pedagogisista suunnittelumalleista.

C2STEM – laskennallinen mallintaminen LUMA-aineissa (tiede)

Luokka	**Kuvaus**
Lähde	Hutchins, N. M., Biswas, G., Maróti, M., Lédeczi, Á., Grover, S., Wolf, R., ... & McElhaney, K. (2020). C2STEM: A System for Synergistic Learning of Physics and Computational Thinking. Journal of Science Education and Technology, 29(1), 83-100, https://doi.org/10.1007/s10956-019-09804-9.
Tiivistelmä	Mallinnus- ja simulointiympäristöä C2STEM käytetään tässä osana fysiikan opetusta. Aiheena on kinematiikka, johon kuuluu käsitteet sijainti, nopeus ja kiihtyvyys. Luodut mallit ovat laskennallisia, jossa yhdistyy fysiikka ja ohjelmoinnillinen ajattelu. Mallinnusympäristö tukee oppimista ohjatuilla tehtävillä ja integroiduilla formatiivisella arvioinnilla.

Aihe	LUMA ja ohjelmoinnillinen ajattelu, oppiminen mallintamalla, ohjelmoinnillinen ajattelu, avoin oppimisympäristö
Oppimistulokset	Opiskelija oppii fysiikan käsitteet ja käytännöt. Oppiminen mallintamalla käyttäen tietoteknistä ympäristöä opettaa myös ohjelmoinnillisen ajattelun käytäntöjä.
Oppimistapa	oppiminen mallintamalla
Kesto	-
Oppilaan pohjatiedot	peruskoulu
Oppimisympäristö	läsnä- ja etäopiskelu
Ryhmäkoko	henkilökohtainen työ

C2STEM on avoin ympäristö LUMA-aineiden oppimista varten (ks. Kuva 3.3). Se on suunniteltu tukemaan lukion luonnontieteiden opetusta ja yhteisöllistä oppimista. Oppiminen perustuu laskennalliseen mallinttamiseen. Ympäristö tukee keskeisiä tieteellisiä käytäntöjä, kuten ilmiöiden mallintamista, havaintojen perustelemista ja tulosten tarkistamista. Laskennalliset mallit auttavat ymmärtämään monimutkaisia ilmiöitä matematiikan, fysiikan ja tietojenkäsittelytieteen avulla. C2STEM antaa oppilaiden tutkia mallien käyttäytymistä, mahdollistamalla niiden kokeilun erilaisilla arvoilla.



Kuva 3.3. Laiskiaisen liikkumisen mallintaminen C2STEM-ympäristössä.

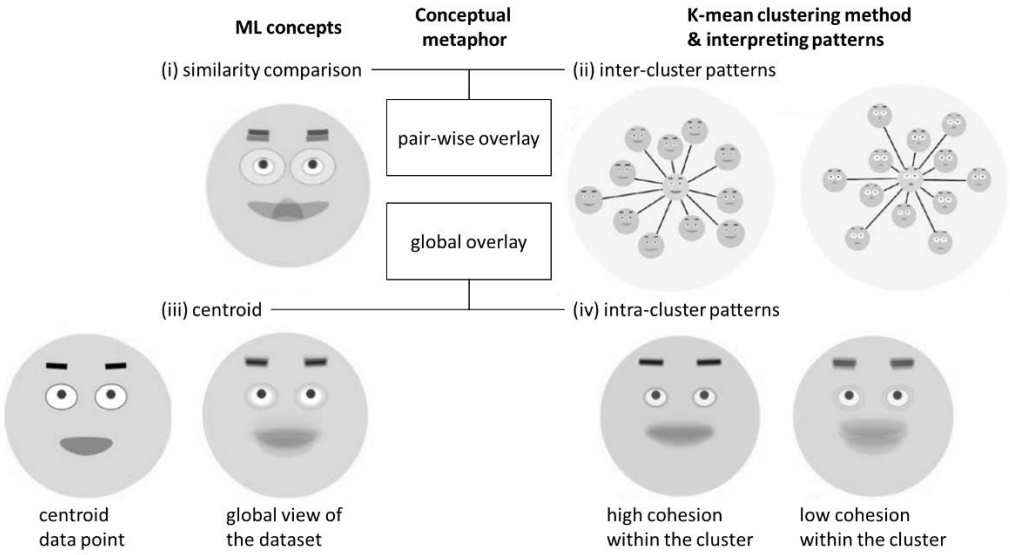
Lisätietoa C2STEM-mallinnusympäristöstä

- artikkeli: C2STEM: A System for Synergistic Learning of Physics and Computational Thinking, https://doi.org/10.1007/s10956-019-09804-9
- nettisivut ja mallinnusympäristö: https://www.c2stem.org

SmileyCluster – koneoppiminen tieteellisen tietojenkäsittelyn mallina (tiede)

Luokka	**Description**
Lähde	Wan, X., Zhou, X., Ye, Z., Mortensen, C. K., & Bai, Z. (2020, June). SmileyCluster: supporting accessible machine learning in K-12 scientific discovery. In Proceedings of the Interaction Design and Children Conference (pp. 23-35), https://doi.org/10.1145/3392063.3394440.
Tiivistelmä	Koska tekoälyn käyttö yleistyy, tulisi nuortenkin oppilaiden ymmärtää sen luonnetta, jotta he voivat arvioida tekoälyn merkitystä yleisesti ja työelämässä. Koneoppiminen on tällä hetkellä yksi tekoälyn tärkeimmistä osa-alueista, mutta sen vaatiman edistyneen matematiikan ja tietojenkäsittelyn tietouden vuoksi se ei ole nuorten oppilaiden ulottuvilla. SmileyCluster on ympäristö, jossa koneoppimisen hahmojen tunnistus ja luokittelu on visualisoitu, jotta oppilaat voivat käyttää omia vastaavia kykyjään ymmärtääkseen teknologian toimintaa.
Aihe	tiedon visualisointi; tekemällä oppiminen; tekoälyn lukutaito; scientific discovery; STEM education
Oppimistulokset	Hahmontunnistuksen ymmärtäminen ja miten koneoppimisen käsitteet ja menetelmät liittyvät siihen.
Oppimistapa	konstruktivismi, tutkiva oppiminen, yhteistyö
Kesto	-
Oppilaan pohjatiedot	yläkoulu
Oppimisympäristö	lähiopetus
Ryhmäkoko	parityö

SmileyCluster on verkkopohjainen yhteisöllinen ympäristö koneoppimisen käsitteiden ja menetelmien oppimiseen. Koneoppiminen on yksi tekoälyn keskeisistä tekniikoista, johon perustuvat nykyaikana laajalti huomiota saaneeet hahmontunnistuksen ja luokittelun sovellukset, kuten tiedonhaku, kasvojen tunnistus, kielten kääntäminen, ja pelien pelaamisen automatisointi. Oppilaille aihe on kuitenkin haastava johtuen sen sisältämän matematiikan vaativuudesta. SmileyCluster-ympäristössä ongelma on ratkaistu visualisoimalla koneoppiminen käyttäen piirrettyjä kasvoja vertauskuvana. Kasvojen piirteet yhdistetään analysoitavan dataan luokitteluun ja piirteiden eri muotoja käytetään kuvaamaan arvojen eroja (ks. Kuva 3.4). Luomalla päällekkäiskuvia kasvoista voidaan visualisoida millaisia luokkien arvojen väliä koneoppiminen tunnistaa datasta. Harjoitus tukee myös LUMA-aineissa tarvittavaa tilastollisen tiedon käsittelyn ymmärtämistä.



Kuva 3.4. Datan piirteitä kuvaavien kasvojen ja niiden päällekkäisyyden metafora sekä miten ne liittyvät koneoppimisen tekniikoihin.

Lisätietoa Smiley Cluster koneoppimisen visualisoinnista

- artikkeli: SmileyCluster: supporting accessible machine learning in K-12 scientific discovery, https://doi.org/10.1145/3392063.3394440
- video: https://www.youtube.com/watch?v=ZiSR-5zpabw

Malli opetusrobotiikan ja ohjelmoinnillisen ajattelun yhdistämiseksi (tekniikka)

Luokka	**Kuvaus**
Lähde	Chevalier, M., Giang, C., Piatti, A., & Mondada, F. (2020). Fostering computational thinking through educational robotics: a model for creative computational problem solving. International Journal of STEM Education, 7(1), 1-18, https://doi.org/10.1186/s40594-020-00238-z.
Tiivistelmä	Robotiikkaa käytetään yhä useammin kouluissa (nk. opetusrobotiikka) ohjelmoinnillisen ajattelun opettamiseen. Creative Computational Problem Solving -malli tukee opettajia opetusrobotiikan hyödyntämisen suunnittelussa, toteutuksessa ja arvioinnissa. Malli auttaa opettajaa osoittamaan oppilaille ohjelmoinnillisen ajattelun hyötyjä käyttämällä sen keskeisiä käsitteitä robotin ohjaamisen eri vaiheissa. Oppilas saa näin tasapainoisemman kuvan ongelmanratkaisusta kuin pelkkään ohjelmointiin keskittymällä.

Aihe	ohjelmoinnillinen ajattelu, opetusrobotiikka, ongelmanratkaisu
Oppimistulokset	Oppilaat osaavat ohjata robottia ja ratkaista ongelmia sen avulla. He oppivat käyttämään ongelmien analysointia, ideoiden keksimistä ja ratkaisujen muotoilua ohjelmoinnillisen ajattelun puitteissa.
Oppimistapa	tutkiva oppiminen
Kesto	-
Oppilaan pohjatiedot	alakoulun oppilaat (9 ja 10 vuotiaita)
Oppimisympäristö	lähiopetus
Ryhmäkoko	2-3 oppilasta

Opetusrobotiikka on yksi ohjelmoinnillisen ajattelun sovelluskohde. Usein oppilaat keskittyvät liikaa itse robotin ohjelmointiin ja pyrkivät ratkaisemaan ongelman yritys ja erehdys menetelmällä, jolloin ohjelmoinnillisen ajattelun mahdollisuudet jäävät käyttämättä. Oppilaiden käyttäytymisen muuttamiseksi on kehitetty Creative Computational Problem Solving -malli, jossa ongelmanratkaisu koostuu viidestä vaiheesta: ongelman analysointi, ratkaisun ideointi, robotin käyttäytymisen määrittely, robotin ohjelmointi ja robotin suoriutumisen arviointi. Ongelmanratkaisun eri vaiheissa oppilaat joutuvat käyttämään kaikkia ohjelmoinnillisen ajattelun keinoja. Creative Computational Problem Solving -mallia on kokeiltu Thymio-robotilla (ks. Kuva 3.5), mutta malli ei rajoita robotin tyyppiä. LUMA-aineissa robotiikka liittyy erityisesti teknologiaan, mutta robotin toiminnan suunnittelussa tarvitaan usein myös matematiikkaa ja fysiikkaa.

Kuva 3.5. (Chevalier et al. 2020; CC BY 4.0)[2] Thymio-robotin käyttö ruohonleikkurin toiminnan simuloimiseen.

[2] Kuvaa on hieman muunneltu poistamalla alueita reunoista. Alkuperäinen: Chevalier, M., Giang, C., Piatti, A. et al. Fostering computational thinking through educational robotics: a model for creative computational problem solving. IJ STEM Ed 7, 39 (2020). https://doi.org/10.1186/s40594-020-00238-z. is Open Access licensed by CC BY 4.0.

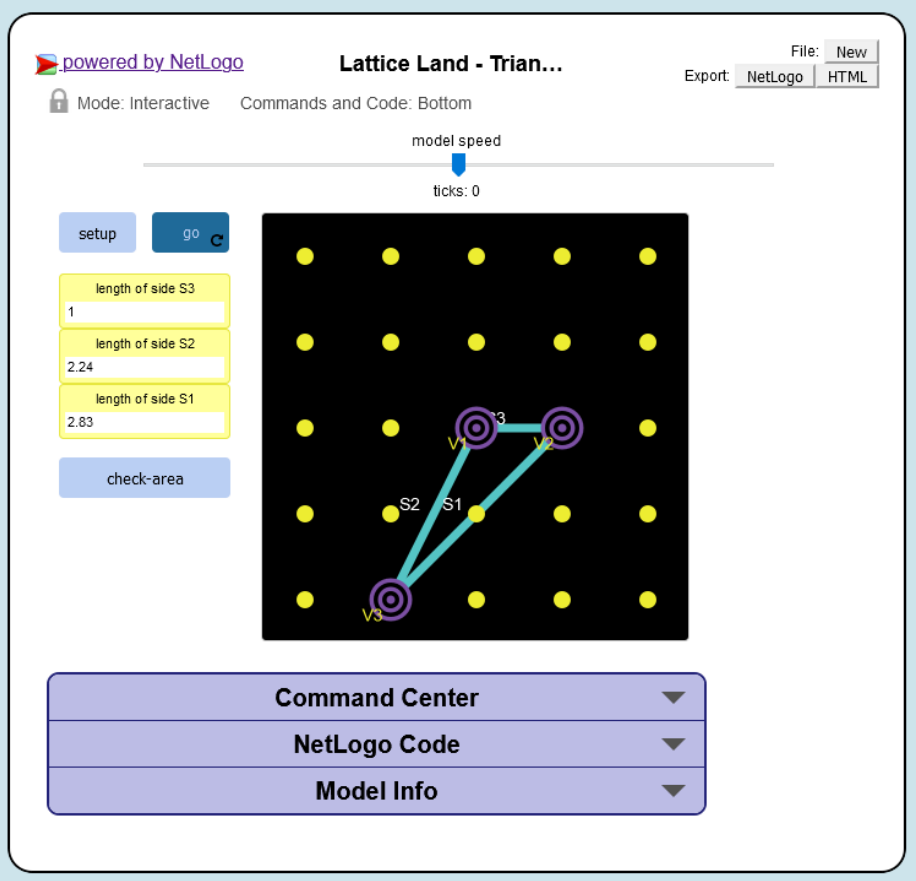
Lisätietoa opetusrobotiikasta ja Creative Computational Problem Solving -mallista

- artikkeli: Fostering computational thinking through educational robotics: a model for creative computational problem solving, https://doi.org/10.1186/s40594-020-00238-z
- robotti: https://www.thymio.org

Lattice Land - mikromaailmoja geometrian tutkimiseen (matematiikka)

Luokka	**Kuvaus**
Lähde	Pei, C., Weintrop, D., & Wilensky, U. (2018). Cultivating computational thinking practices and mathematical habits of mind in lattice land. Mathematical Thinking and Learning, 20(1), 75-89, https://doi.org/10.1080/10986065.2018.1403543.
Tiivistelmä	Lattice Land on lukion geometrian oppimiseen tarkoitettu mikromaailma. Tavoitteena on harjoitella sekä matemaattista että ohjelmoinnillista ajattelua. Oppilas oppii kokeilemista, tutkimista, hahmontunnistusta ja hypoteesien esittämistä matemaattisesti.
Aihe	geometria, tietotekniset oppimisympäristöt, matemaattinen ajattelu, ohjelmoinnillinen ajattelu
Oppimistulokset	Oppilaat oppivat tunnistamaan geometrisia kuvioita ja päättelemään niiden ominaisuuksia. Lisäksi he oppivat ohjelmoinnillista ajattelua, jota käytetään oppimismenetelmänä.
Oppimistapa	konstruktionistinen, tekemällä oppiminen
Kesto	-
Oppilaan pohjatiedot	peruskoulu
Oppimisympäristö	etäopetus
Ryhmäkoko	henkilökohtainen työ

Lattice Land on kokoelma geometrian mikromaailmoja (Pei ym., 2018), jotka perustuvat NetLogo-ympäristöön (Wilensky, 1999). Mikromaailmojen tarjoamat rakenteet tukevat geometrian oppimisen aloittamista, mutta vapaan kokeilun salliminen mahdollistaa etenemisen valmiita malleja pidemmälle. Geometrian mikromaailmojen lähtökohtana on pistematriisin muodostama ristikko, jossa jokaisella pisteellä on kokonaislukukoordinaatit (x, y). Monikulmioita voidaan piirtää ristikkoon yhdistämällä pisteitä kaarilla. Yhdistetyt pisteet muodostavat monikulmion kärjet. Oppilas voi muokata monikulmioita siirtämällä niiden kärkipisteitä tai lisäämällä uusia kaaria. Koska kaarien piirtäminen ja kärkipisteiden siirtäminen voi tapahtua vain pisteiden välillä, tarjoaa ristikko monikulmioiden muodostamista ja muokkaamista tukevan rakenteen (ks. Kuva 3.6). Eri mikromaailmat voivat tuoda lisää ominaisuuksia tai rajoituksia.



Kuva 3.6. Lattice Triangles Explore mikromaailmassa tutkitaan, kuinka monta erikokoista kolmiota voidaan muodostaa ristikossa.

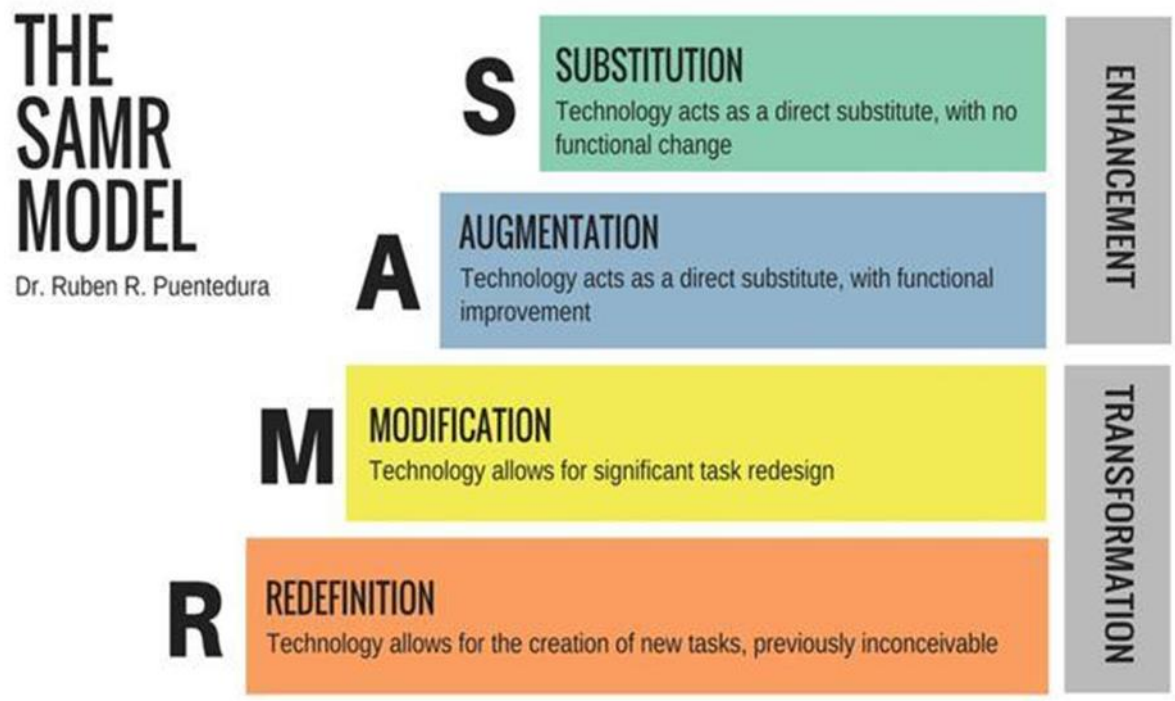
Lisätietoja Lattice Land -mikromaailmasta:

- artikkeli: https://doi.org/10.1080/10986065.2018.1403543
- ladattava ympäristö (valitse Lattice Land eri versioista): https://ccl.northwestern.edu/netlogo/download.shtml
- ympäristö verkossa (valitse Lattice Land eri versioista): http://ccl.northwestern.edu/netlogo/models/index.cgi

Teknologian hyödyntämisasteen taksonomia

Perustuen Moduuli 1:n osioon 5: Project Based Learning (PBL) (Valentina Dagiene)

Opetusteknologian mahdollisuuksia voidaan hyödyntää enemmän tai vähemmän. Kun otetaan uusi teknologia käyttöön, voidaan sitä ensin hyödyntää olemassa olevan teknologian korvikkeena. Myöhemmin, kun kokemusta on kertynyt, teknologian mahdollisuuksia voidaan hyödyntää täysimääräisemmin. Teknologian hyödyntämisen astetta voidaan tarkastella neliportaisella taksonomialla korvaaminen, täydentäminen, muuttaminen ja uudistaminen, jota kuvaa SAMR-malli (Substitution, Augmentation, Modification and Redefinition; Puentedura, 2006; ks. Kuv 3.7). Malli tukee teknologian valintaa, käyttöä ja arviointia oppimisen tukemiseksi.



Kuva 3.7. SAMR-malli auttaa opettajia pohtimaan teknologian roolia oppimisen tukemisessa (koulutustutkija Ruben Puentedurán kehittämä 2010, Creative Commons lisenssi)

Korvaaminen
"Korvaamisella" tarkoitetaan perinteisten toimintojen ja materiaalien - kuten luentojen tai kurssimonisteiden - korvaamista digitaalisilla versioilla. Sisältö ei muutu olennaisesti, ainoastaan tapa, jolla se toimitetaan. Tavoitteena on pitää asiat yksinkertaisina, joten pyörää ei tarvitse keksiä uudelleen. Skannaa oppimateriaalit ja tehtäväpaperit, muunna ne PDF-tiedostoiksi ja siirrä ne verkkoon Microsoft One Driven, Google Driven tai vastaavan tiedostonjakopalvelun avulla. Mieti luokkahuoneen seinillä olevaa tietoa, kuten luokkahuoneen sääntöjä, päiväohjelmaa tai sanastoja, ja muunna ne digitaaliseen muotoon, johon oppilaat voivat helposti viitata. Voit tehdä luennoistasi sekä synkronisia että asynkronisia versioita käyttämällä videoneuvottelupalveluita, kuten Zoom tai Skype, ja niiden taltiointiominaisuuksia. Voit myös luoda omia opetusvideoita, joita opiskelijat voivat katsoa omaan tahtiinsa.

Täydentäminen
Tälle tasolle kuuluu digitaalisten sisältöjen täydentäminen kommenteilla, hyperlinkeillä tai multimedialla. Sisältö säilyy ennallaan, mutta oppilaat voivat nyt hyödyntää digitaalisia ominaisuuksia oppimisen tehostamiseksi. Oppilaat voivat luoda digitaalisia portfolioita tekemistään töistä. Paperisten monivalintatehtävien sijaan voidaan hyödyntää digitaalisia palveluita, kuten Socrative ja Kahoot. Opettajat voivat myös hyödyntää sähköisiä ilmoitustauluja, kuten Padlet-verkkopalvelu, keräämään oppilaiden kysymyksiä, vastauksia ja mielipiteitä.

Muuttaminen

Opetusta voidaan muuttaa siirtymällä verkko-opetukseen hyödyntämällä digitaalisia oppimisjärjestelmiä kuten Google Classroom, Moodle, Schoologyn tai Canvas. Oppimisjärjestelmien avulla voidaan jakaa oppimateriaaleja, luoda yhteisiä kalentereita, palauttaa tehtäviä, arvostella niitä ja antaa arvosanoja. Ne tarjoavat myös vaihtoehtoisia viestintäkanavia, joka voi tukea niitä oppilaita, jotka eivät tavallisesti uskalla luokassa vastata, kysyä tai kommentoida opittavia aiheita. Videoneuvottelujen, kuten Zoom, mahdollisuus keskustella kirjallisesti chatissä, voi madaltaa kysymysten esittämisen kynnystä, kun siinä ei keskeytä luennoitsijaa tai joudu kilpailemaan huomiosta. Asynkroniset viestintämuodot tukevat niitä oppilaita, jotka haluavat koota ajatuksiaan, ennen kuin he osallistuvat keskusteluun.

Uudistaminen

Oppiminen muuttuu perusteellisesti "uudistamisen" tasolla, jolloin teknologia mahdollistaa toiminnan, joka oli aiemmin mahdotonta luokkahuoneessa. Digitaaliset kirjekaverit voivat yhdistää opiskelijat muuhun maailmaan, olipa kyse sitten toisista opiskelijoista tai alan asiantuntijoista. Kun luokka on lukenut kirjan, voidaan kutsua kirjailija virtuaalisesti keskustelemaan teoksestaan ja vastaamaan kysymyksiin. Virtuaaliset opintomatkat mahdollistavat tutustumisen esimerkiksi Amazonin sademetsään, Louvreen tai Egyptin pyramideihin. Teknologian avulla luokkahuoneeseen voidaan osallistua ympärimaailmaa. Quadbloggingin kaltaiset alustat yhdistävät kaukaiset luokkahuoneet toisiinsa ja oppilaat voivat esittää kysymyksiä toiselle luokalle ja vastata heiltä saapuneisiin kysymyksiin. Käyttäen wiki- tai blogi-alustoja oppilaat voivat julkaista virtuaalisesti omia sisältöjään.

Luento - Opetusteknologian valinta Ohjelmoinnillisen ajattelun näkökulmien avulla

Hyödyntäen ylläolevaa tekstiä

- Ohjelmoinnillisen ajattelun näkökulmat STEAM-opetukseen -viitekehyksen soveltaminen
- Tiedettä tukevat opetusteknologiat
- Tekniikkaa tukevat opetusteknologiat
- Matematiikkaa tukevat opetusteknologiat
- SAMR-malli: opetusteknologian hyödyntämisen aste

Tehtävä 3.1 kotitehtävä - Opetusteknologian tuen suunnittelu omalle aiheelle

1. Valitse aihe oppiaineestasi ja suunnittele tehtävä käyttäen "Ohjelmoinnillisen ajattelun näkökulmat STEAM-opetukseen" -viitekehystä (katso lisätietoja osiosta 2).

2. Valitse sopiva opetusteknologia, joka tukee oppilaita tehtävän suorittamisessa (käytä luennolla kuvattuja opetusteknologioita mallina etsiessäsi kirjallisuudessa/Internetistä sopivaa teknologiaa).

3. Mille SAMR-mallin teknologian hyödyntämisen tasolle sijoittaisit opetusteknologian käytön tehtävässä (vähimmäistasona tulisi olla täydentäminen). Muuta tehtävän suunnittelmaa, jos opetusteknologia ei näytä tuovan mitään lisää oppimiseen.

4. Täytä taulukko kuvaamaan suunnittelemasi tehtävän pedagogista suunnitelumallia.

Pedagoginen suunnittelumalli	
Aihe	
Lähde	
Tiivistelmä	
Aihe	
Oppimistulokset	
Oppimistapa	

Tehtävä 3.2 Kotitehtävä - Vertaisarviointi

- Arvioi kolmen kolleegasi suunnittelemat tehtävät (Tehtävä 3.1)
- Arviointi on hyväksytty tai hylätty
- Jokainen alla oleva kohta on skaalalla 0-5 (hyväksytyssä suorituksessa keskiarvon on oltava 2 tai parempi)
    - tehtävän aiheen esitys
    - perustelut ja selkeys
    - luvun sisällön hyödyntäminen

Osio 4 – Opetussisällön luominen STEAM-opetukseen hyödyntäen ohjelmoinnillista ajattelua

Tässä osiossa esitellään STEAM-opetuksen suunnittelun malli (Quigley & Herro, 2017). Painopiste on opetussisällön luomisessa. Työkalujen valinnan tulisi perustua sisältöön, mutta opetusteknologian tuki mahdollistaa myös sellaisia sisältöjä, jotka muuten olisivat oppilaille liian vaikeita. Osiossa alkaa myös 3-4 oppilaan ryhmissä tehtävä projektityö, joka kokoaa yhteen koko moduulin sisällön. Projektin ensimmäisessä osassa valitaan opetussisältö, etsitään mahdollisuuksia ohjelmoinnnillisen ajattelun soveltamiseen ja suunnitellaan millainen tehtävä toisi opetussisällön esille. STEAM-opetuksessa käytetään ongelmaperustaista pedagogiikkaa. Ongelmat ovat avoimia, jotta oppilaat voivat vapaasti kehittää omia ratkaisujaan eri LUMA-aineita (STE(A)M) hyödyntäen. STEAM-lyhenteessä A viittaa ihmis-ja yhteiskuntatiteteisiin,

joita hyödyntäen voidaan luoda tehtävänantoja, jotka ovat kiinnostavia laajemmalle oppilasjoukolle ja liittyvät ajankohtaisin ongelmiin.

Sekä opetussisältö että oppimisympäristö ovat tärkeitä STEAM-opetuksen onnistumisen kannalta. Ne ovat riippuvaisia toisistaan, sisällön tulisi hyödyntää oppimisympäristön mahdollisuuksia, ja oppimisympäristöä suunniteltaessa tulisi ottaa huomioon sisällön tarpeet. Tässä osiossa keskitytään oppilaille tarjottavaan sisältöön. STEAM-opetuksessa ongelmat on otettu tosielämästä ja niiden luonteeseen kuuluu, että niihin ei useinkaan ole yhtä ainoaa oikeaa vastausta. Opettajan tulisi luoda realistisia skenaarioita ongelman esittämiseksi oppilaille. Erilaisten vastausten mahdollisuuden tarkoituksena on innostaa luovuuteen ja tehtävän laajuuden tulisi tehdä yhteistyöstä luontevaa.

Opettaja järjestää, valmistelee ja välittää sisällön oppilailleen. STEAM-sisältöä suunniteltaessa olisi otettava huomioon kolme vaatimusta: ongelmalähtöinen oppiminen, oppiaineiden integrointi ja ongelmanratkaisutaidot. Ongelmalähtöisyys on keskeistä STEAM-pedagogiikassa. Sisältö esitetään suhteessa ongelman eri näkökulmiin. Näkökulmien tulisi edustaa eri oppiaineita. Opettaja ei voi olla jokaisen oppiaineen asiantuntija, mutta hän voi viitata eri lähteisiin ja muihin asiantuntijoihin, jotka voivat antaa lisätietoja. Ongelmaa ei pitäisi esittää kysymyksenä, johon on vain yksi oikea vastaus, vaan ongelmallisena tilanteena, jota voi ratkaista usealla tavalla. Tilanne antaa ongelmalle kontekstin, ja sen pitäisi tuntua oppilaista aidolta ja merkitykselliseltä. Aitous on tärkeää, sillä jos ongelma ja sen konteksti tuntuvat vääriltä, oppilaiden motivaatio heikkenee.

Oppiaineita integroimalla opetetaan yhdistämään eri alojen tietoja ja menetelmiä käsillä olevan ongelman ratkaisemiseksi. Vaikka STEAM-opetuksen ideana on yhdistää lyhenteessä mainittuja oppiaineita, niin jokainen ongelma ei vaadi niitä kaikkia. Tarvittavien tietojen ja menetelmien valinnan pitäisi olla oppilaiden päätettävissä. Oppiaineiden integroinnin aste oppilaiden vastauksissa voi vaihdella. Siihen vaikuttaa ongelman esittely ja oppilaiden aikaisemmat kokemukset oppiaineiden yhdistämisestä. Ongelmanratkaisu voi olla monialaista, alojen välistä tai poikkialaista. Monialainen ongelmanratkaisu on vähiten integroitu, jolloin ongelman eri näkökulmiin vastataan erikseen. Alojen välisessä ongelmanratkaisussa yhdistetään eri alojen menetelmiä. Poikkialaisessa ongelmanratkaisussa yhden alan ongelmaa tarkastellaan eri alojen näkökulmista.

Ongelmanratkaisutaidot kuvaavat oppilaiden ominaisuuksia. Taitoja on kolmenlaisia: kognitiivisia, vuorovaikutuksellisia ja luovia. Ongelmia ratkaisemalla taidot kehittyvät. Taidot ovat yleisiä, mutta tietyntyyppiset ongelmat ja menetelmät voivat kehittää niitä enemmän. Abstrahoiminen, analysoiminen, soveltaminen, luokitteleminen, muotoileminen, tulkitseminen, hahmottaminen, mallintaminen, yhdisteleminen ja kyseenalaistaminen ovat esimerkkejä kognitiivisista taidoista. Näitä voidaan kehittää havainnointia, pohdintaa, päättelyä ja kokemusten kertymistä tukevilla opetusmenetelmillä. Vuorovaikutustaidoissa on kyse viestinnästä ja yhteistyöstä. Yksittäisen oppilaan näkökulmasta vuorovaikutukseen kuuluu kyky esitellä, ideoida, kuvata, perustella, selittää, vastata, viestiä ja väitellä. Yhteistyöllä tarkoitetaan yhdessä oppimista jakamalla ongelmaan liittyvät tehtävät. Optimaalisesti yhteistyö johtaisi tietojen yhdistämiseen, todisteiden keräämiseen ja jaettuun kokemukseen.

Luovaa oppimista tarvitaan ideoiden, innovaatioiden, ratkaisujen ja sisältöjen kehittämiseksi. Tarvittavia taitoja ovat muun muassa suunnittelu, hahmontunnistus, leikkiminen, esittäminen,

mallintaminen ja ideoiden yhdistäminen. STEAM-opetus edistää luovuutta sallimalla useita tapoja osoittaa osaamista ja ratkaista ongelmia. Opettajan on tarjottava oppilaille tietoja ja työkaluja, joiden avulla he voivat ratkaista ongelmia eri tilanteissa ja saada niistä kokemuksia. Sekä ihmis- ja yhteiskuntatieteet että tekniset aineet STEAM-opetukseesa mahdollistavat luovuuden. Yhdistämällä ne muihin oppiaineisiin voidaan löytää uusia ilmaisumuotoja, jotka tukevat sekä taitojen kehittymistä että sisällön ymmärtämistä. Muotoilu on taiteen ja teknologian yhdistämistä ongelmanratkaisemiseksi tuotteita luomalla. Joskus teknologian huono käytettävyys voidaan ratkaista muotoilulla.

Ohjelmoinnillinen ajattelu liittyy moniin STEAM-opetuksen piirteisiin. Se tukee eri oppiaineiden näkökulmia ja tarjoaa välineitä niiden tutkimiseen. Tietokoneiden avulla voidaan simuloida erilaisia ilmiöitä tai teknologioita, jos niitä ei voida kokea käytännössä. Ilmiöitä kuvaavaa dataa ja laskutoimitusten tuloksia voidaan myös visualisoida. Tietokoneella luodut esitykset eivät ole vain tekstiä ja kuvaa, vaan myös animaatioita, ääniä ja videoita. Erilaisten tietoteknisten ominaisuuksien valinta ja yhdistely vaatii luovuutta. Kuten STEAM-opetuksen eri oppiaineita, niin oppilaita ei myöskään tulisi pakottaa hyödyntämään ohjelmoinnillista ajattelua ja tietotekniikkaa, vaan niiden pitäisi olla käytettävissä tarvittaessa. Oppilailla on oltava aiempaa kokemusta tietotekniikasta ja tietoteknisistä työkaluista, jotta he voivat hyödyntää niitä osana monialaista (STEAM) oppimista.

Esimerkki STEAM-tehtävästä 1

"On arvioitu, että vain yksi tuhannesta merikilpikonnan poikasesta selviää aikuiseksi. (http://www.seeturtles.org/baby-turtles/). Paikallinen merikilpikonnasairaala on yksi taho, joka pelastaa ja kuntouttaa merikilpikonnia. Miksi vain harva merikilpikonna selviää aikuiseksi? Mitä vaaroja merikilpikonnat kohtaavat? Voivatko ihmiset vahingoittaa merikilpikonnia tahattomasti? Tehtävänä on 5-henkilön ryhmissä suunnitella, miten merikilpikonnia voidaan auttaa. Tutkikaa, mikä vahingoittaa ja tappaa merikilpikonnia. Voitte ottaa yhteyttä merikilpikonnasairaalaan ja haastatella heitä. Miten esittelisitte merikilpikonnien kuolleisuuden kehitystä? Tehkää tutkimuksesta kaksiosainen esitys: esitelkää ensimmäisessä osassa merikilpikonnien selviytymiseen vaikuttavat tekijät ja toisessa suunnitelma miten merikilpikonnia voidaan auttaa. Voitte käyttää ChatterPix Kids, Toontastic tai muita sovelluksia haastattelujen tallentamiseen ja esityksen luomiseen. Voitte itse päättää, miten esittelette havaintonne muulle luokalle. Luokka päättää yhdessä, mitä se voi tehdä merikilpikonnien selviytymisen edistämiseksi." (Quiqley ym. 2020, Sähköinen lisämateriaali 2).

Esimerkki STEAM-tehtävästä 2

"Seuraavan sukupolven tietotekniikan käyttöliittymät (esim. katseenseuranta, kasvojentunnistus, iWatch) sulauttavat teknologian vaatteisiin eli sähköisiin e-tekstiileihin. Kehitys voisi tuoda tietokoneet lähemmäksi ihmistä. Uudet laitteet voisivat myös muuttaa vaatteiden suunnittelua, koska teknologian mahdollisuudet ja rajoitukset on otettava huomioon. Huippu-urheilijat ovat kiinnostuneita e-tekstiileistä, koska ne pystyvät havaitsemaan lihasväsymystä. Terveydenhuoltoala toivoo, että tekniikka auttaisi potilaita, joilla on esimerkiksi epilepsia, diabetes tai astma. E-tekstiilien valmistajat haluaisivat kuitenkin laajentaa markkinoita valitsemalla kohderyhmäksi teinit! He ovat ottaneet yhteyttä kouluun kuullakseen ideoitanne sähköisten tekstiilien käyttömahdollisuuksista. Keksikää ryhmässä elämänalue, johon teidän

mielestä e-tekstiilit toisivat kiinnostavan lisän. Suunnitelkaa esitys, joka sisältää e-tekstiilin prototyypin, miten sitä käytettäisiin ja lopuksi keskustelua teknologian tulevista mahdollisuuksista ja haasteista. " (Quiqley et al. 2020, Sähköinen lisämateriaali 2).

Esimerkki STEAM-tehtävästä 3

"Kun yhä useammat ihmiset muuttavat X-alueelle ja yritykset jatkavat kasvuaan, niin liikenne lisääntyy ja aiheuttaa viivytyksiä erityisesti lukioon johtavan tien varrella. Tämä tarkoittaa esimerkiksi sitä, että henkilö, joka asuu 16 kilometria lukiosta, saattaa käyttää 45 minuuttia matkaan, vaikka siihen pitäisi kulua alle 20 minuuttia. Tästä koituu suuria kustannuksia perheille, jotka viettävät vähemmän aikaa yhdessä, ja ympäristölle, joka saastuu liikenteen päästöjen lisääntyessä. Kaupunkisuunnittelija on antanut ryhmällesi tehtäväksi kehittää ratkaisu alueen liikenneongelmiin. Käytätte tieteellisen menetelmän ja teknisen suunnitteluprosessin tuntemustanne ratkaisun löytämiseen ja sen testaamiseen. Olette päättäneet, että on hyödyllistä miettiä alueen historiaa, jotta saatte paremman käsityksen maankäytön tarpeista ja liikenteen kehityksestä. Lisäksi voitte lukea artikkeleita X-alueen maankäytöstä, liikenteestä sekä ennustuksista asukas- ja yritysmäärän kehityksestä. Teidän on luotava malli liikenteen kehityksestä ja esitettävä ratkaisu ruuhkien lieventämiseksi. Asiakkaana on kaupunki, mutta myös paikallinen yhteisö on huomioitava ratkaisun kehittämisessä.

Ryhmätyö: Suunnitelkaa ratkaisu liikenneongelmiin alueen historian ja liikennemallien avulla?" (Quiqley et al. 2020, Sähköinen lisämateriaali 2).

Luento - Opetussisällön luominen STEAM-opetukseen hyödyntäen ohjelmoinnillista ajattelua

Ylläolevan tekstin ja Ohjelmoinnillisen ajattelun näkökulmat STEAM-opetukseen viitekehys lukuun (Osio 2) perustuen

- Ongelmalähtöinen oppiminen
- Oppiaineiden integrointi
- Ongelmanratkaisutaidot
- Ohjelmoinnillisen ajattelun näkökulmien soveltaminen
- STEAM-tehtävien esimerkit

 STEAM-opetuksen käsitteellinen malli – Opetussisältö

Quigley, C. F., Herro, D., & Jamil, F. M. (2017). Developing a conceptual model of STEAM teaching practices. School Science and Mathematics, 117(1-2), 1-12. https://doi.org/10.1111/ssm.12201

Tehtävä 4.1 projekti 1/2

Tämä on kurssin projektin alku. Tavoitteena on suunnitella opetusinterventio. Projekti on jaettu kahteen osaan. Tässä osassa kokoatte projektitiimin ja suunnittelette opetettavan sisällön. Projektin laajuus voi vaihdella opettajan antamien ohjeiden mukaan.

Teidän tulisi muodostaa 3-4 hengen ryhmä. Olisi hyödyllistä, jos ryhmässä on eri oppiaineiden edustajia. Valittavan aiheen tulisi olla ajankohtainen, yhdistää eri oppiaineita, liittyä tosielämään ja olla merkityksellinen oppilaille (valitkaa oppiaste). Laatikaa tehtävän kuvaus ja suunnittelkaa, millaista tietoa oppilaat voisivat tarvita. Tehtävän tulisi antaa tilaa luovuudelle ja vaihtoehtoisille ratkaisuille. Muistakaa, että aiheessa pitäisi olla mahdollisuus hyödyntää ohjelmoinnnillista ajattelua.

Yhteenveto

- 3-4 oppilaan ryhmissä
- aiheen valinta
- tehtävän kuvaus
- opetussisällön suunnittelu
- ohjelmoinnillisen ajattelun käyttäminen

Tehtävä 4.2 ryhmä keskustelee ohjaajan kanssa projektin suunnitelmasta

- Kun projektin suunnitelma on valmis.
- Ryhmä esittelee suuunnitelman ohjaajalle.

Osio 5 – STEAM-opetusta ja ohjelmoinnillista ajattelua tukevien oppimisympäristöjen suunnittelu

Osio jatkaa STEAM-opetuksen suunnittelun käsitteellisen mallin (Quigley & Herro, 2017; Quigley & Herro, 2019) hyödyntämistä. Painopiste on oppimisympäristön suunnittelussa. Tämän moduulin muissa osioissa on lisäksi annettu perusteet opetusteknologioiden valinnalle, jotka tukevat ohjelmoinnillisen ajattelun integrointia STEAM-opetukseen. Keskeistä ei kuitenkaan ole teknologian valinta vaan sellaisen ympäristön luominen, jossa tehtävä voidaan suorittaa. Koska lähestymistapa ongelmanratkaisuun on oppilaiden päätettävissä, pitäisi myös tietyn teknologian käyttäminen olla heidän valintansa. Valitsemalla ongelmia, joissa algoritmin automatisointi olisi luonnollinen osa ratkaisua, kannustetaan oppilaita ohjelmoinnilliseen ajatteluun. Osio aloittaa myös projektityön toisen osan, jossa suunnitellaan oppimisympäristö ja valitaan

opetusteknologiat edellisessä osiossa luotua tehtävää varten. STEAM-oppimisympäristön suunnittelussa keskitytään opetusmenetelmiin, teknologian integrointiin, arviointikäytäntöihin ja tasapuolisen osallistumisen mahdollistamiseen.

STEAM-opetus perustuu oppilaslähtöiseen opetukseen, joka eroaa perinteisistä opettajajohtoisista oppitunneista. Oppilaita tulisi kannustaa ottamaan vastuuta omasta oppimisestaan. Tämä voi olla aluksi vaikeaa, ja oppilaat voivat edelleen odottaa vastauksia opettajalta. Käyttäytymisen muuttumista voidaan tukea säännöllä, että oppilaiden tulisi kysyä tai etsiä vastauksia kolmesta lähteestä, ennen kuin he kysyvät opettajalta. Oppilaiden tulisi oppia luottamaan toistensa tietoihin ja siihen, mitä he itse voivat selvittää. Lisäksi, ennen kuin opettaja vastaa oppilaille, voisi hän pyytää heitä kertomaan, mitä he ovat löytäneet. Opettajan ei pitäisi antaa oikeaa vastausta, vaan hyödyntää oppilaiden kertomia havaintoja ja kannustaa heitä tutkimaan lisää. Tosielämän ongelmiin on harvoin yksiselitteisiä vastauksia ja oppilaiden tulisi tottua siihen, että tarkkaa vastausta ei aina ole olemassa. Vastaukset voivat kuitenkin olla huonommin tai paremmin valmisteltuja ja siksi oppilaiden kyky perustella ratkaisun valinta on tärkeää.

Oppitunneilla kehitetään sellaisia taitoja kuten abstrahoiminen, argumentoiminen, analysoiminen, muotoileminen, soveltaminen, yhteistyön tekeminen ja todisteiden esittäminen. Opettajan tulisi luoda oppilaille tilaisuuksia taitojen käyttämiseen. STEAM-opetuksen mallin mukaan oppilaiden tulisi oppia kokeilemaan useita keinoja ongelman ratkaisemiseksi. Vapaus valita ratkaisutapa mahdollistaa kekseliäisyyden ja luovuuden. Opettajan tulisi täydentää annettua tehtävää erilaisilla tiedoilla, työkaluilla ja mahdollisuuksilla erilaisiin kokemuksiin. Tehtävän kuvauksen on tuettava oppilaslähtöistä oppimista, jossa vertaistuki ja yhteistyö ovat luontevia. Annetun ongelman tulisi mahdollistaa erilaiset oppilaiden roolit ratkaisun kehittämisessä. Kun ongelmanratkaisu edellyttää erilaisia tietojen ja taitojen yhdistelmiä, niin vertaisilta avun pyytäminen ja yhteistyö eivät tunnu pakotetulta. Opettajan olisi suunniteltava tehtävänanto niin, että se kannustaa ikätasolle sopivalla tavalla emotionaalisen ja sosiaalisen sitoutumisen oppimiseen.

Teknologia[3] on yksi STEAM-lyhenteen oppiaineista. Oppimisympäristössä oppilailla tulisi olla käytettävissään erilaisia teknologisia työkaluja ongelmien ratkaisemiseksi. Heidän ei pitäisi olla teknisten ratkaisujen käyttäjiä vaan soveltajia. Jotta oppilaat voivat soveltaa teknologiaa ongelmaan, he tarvitsevat aiempaa kokemusta työkaluista. Vasta kun teknologia on tuttu, voi se tukea oppimista. Tämän moduulin yhteydessä painotetaan erityisesti työkaluja, jotka tukevat ohjelmoinnillista ajattelua. Ohjelmoinnillisen ajattelun ja tietotekniikan hyödyntämiseksi on ongelmassa oltava näkökulmia, jotka hyötyvät algoritmin suorittamisen automatisoinnista. On tehtävä ero valmiiden ratkaisujen ja ratkaisun luomisen välillä. Esimerkiksi historiallisten tosiasioiden esittäminen verkkosivulla, ei kehitä ohjelmoinnillista ajattelua, mutta tosiasioiden löytäminen analysoimalla ohjelmallisesti historiallista dataa, puolestaan kehittää. Oppilaat voivat hyötyä siitä, että heillä on sekä tiettyihin tarkoituksiin tarkoitettuja työkaluja että yleisiä työkaluja, joita voidaan muokata (ohjelmoida) tarpeen mukaan. Yleiset työkalut voivat tukea

[3] Teknologia on monessa maassa erillinen oppiaine. Siihen voi kuulua teoriaa teknologian kehityksestä ja ominaisuuksista sekä käytännön harjoituksia. Suomessa ei ole vastaavaa oppiainetta, mutta käsityöaineet ovat lähinnä käytännön harjoituksia.

paremmin eri oppiaineiden menetelmien yhdistelmiä, koska niitä ei ole tehty rajattua käyttötarkoitusta varten.

Arviointi on keskeinen osa kaikkia opetusmalleja ja siten myös STEAM-opetusta. Opetus, oppiminen ja arviointi on sovitettava yhteen opetuksen tavoitteiden saavuttamiseksi. STEAM-opetuksen oppilaslähtöisyys tekee perinteisistä arviointimenetelmistä, kuten monivalintakokeista, vääränlaisia. Arviointimenetelmän tulisi heijastaa STEAM-mallia, jossa tavoitteena on, että oppilaat tutustuvat useaan tiedonhankintatapaan, osaavat hyödyntää monenlaisia taitoja ja oppivat tekemään yhteistyötä ratkaisun löytämiseksi. Jotta arviointi olisi autenttista, on oppilaita pyydettävä soveltamaan oppimiaan tietoja ja taitoja ratkaistavan ongelman kontekstissa. Prosessin päättyminen ratkaisuun on osoitus opituista tiedoista ja taidoista. Pelkkä lopputuloksen arviointi ei kuitenkaan mahdollista suorituksen parantamista oppimisprosessin aikana. Jotta voidaan varmistaa, että arviointi liittyy STEAM-opetuskäytäntöihin, tulisi sen olla osa oppimisprosessia. Opettajan on annettava palautetta riittävän usein ongelmanratkaisun aikana. Näin opettaja voi varmistaa, että oppilaat ymmärtävät sisällön oppimistavoitteiden mukaisesti. Palaute voi auttaa oppilaita oppimisen tiedostamisessa ja pohtimaan eri aiheita syvällisemmin.

Tasapuolisen osallistumisen tavoitteena on oikeudenmukaisuus ja kohtuullisuus aikaisemman osaamisen vaatimuksissa. Kulttuurinormit ja perheiden perinteet vaikuttavat oppilaiden taustatietoihin. Ne on huomioitava, jotta kaikilla olisi yhtäläiset lähtökohdat. Mahdollistamalla yhteydet asiantuntijoihin ja pääsyn mentorointiin opettaja voi tukea oppilaita, jotka eivät välttämättä saisi vastaavaa tietoa koulun ulkopuolella. Oppilaat voivat ammattilaisten kautta tutustua erilaisiin näkemyksiin ja uramahdollisuuksiin. Myös kulttuurinen herkkyys on tärkeää, jotta ei edistetä stereotypioita, vaan osoitetaan, että tulevaisuus voi olla sellainen kuin siitä haluaa tehdä, taustasta riippumatta. Tekemällä tilaa itseilmaisulle opettaja antaa oppilaille mahdollisuuden tuoda esille vahvuuksiaan. STEAM-opetuksen oppilaslähtöisyys ja arviointi osana oppimisprosessia, mahdollistaa tasa-arvoisemman osallistumisen, koska oppilaita arvioidaan yksilöllisemmin.

Luento - STEAM-opetusta ja ohjelmoinnillista ajattelua tukevien oppimisympäristöjen suunnittelu

Yllä olevaan tekstiin ja Osion 3: Opetusteknologian valinta Ohjelmoinnillisen ajattelun näkökulmien avulla -lukuun perustuen

- Opetusmenetelmät
- Arviointikäytännöt
- Tasa-arvoinen osallistuminen
- Ohjelmoinnillista ajattelua tukevien opetusteknologioiden valinta

STEAM-opetuksen käsitteellinen malli – Oppimisympäristö

Quigley, C. F., Herro, D., & Jamil, F. M. (2017). Developing a conceptual model of STEAM teaching practices. School Science and Mathematics, 117(1-2), 1-12. https://doi.org/10.1111/ssm.12201

Quigley, C. F., Herro, D., Shekell, C., Cian, H., & Jacques, L. (2020). Connected learning in STEAM classrooms: Opportunities for engaging youth in science and math classrooms. International Journal of Science and Mathematics Education, 18(8), 1441-1463. https://doi.org/10.1007/s10763-019-10034-z

Tehtävä 5.1 projekti 2/2

Osiossa 4 aloitettu projekti jatkuu tässä. Kun opetuksen sisältö on valmis, niin voidaan aloittaa toteutuksen suunnittelu. Oppilaat tarvitsevat tehtävän suorittamiseen tietoa, ja sen antamiseen on erilaisia tapoja. Voit suunnitella luentoja ja oppimateriaaleja sekä ehdottaa Internet-sivustoja tai avainsanoja tiedonhakuun. Asiantuntijoiden käyttäminen vahvistaisi tehtävän aitoutta. Toinen osa toteutusta on tarvittava teknologian tuki. Aiheeseen voi liittyä räätälöityä opetusteknologiaa tai joskus riittää yleiskäyttöisempikin. Ohjelmoinnillisen ajattelun hyödyntämiseksi, olisi kiinnitettävä erityistä huomiota siihen, että käytettävät ohjelmat ja muu tietotekniikka soveltuvat aiottuun tarkoitukseen.

Yhteenveto

- Osion 4 projekti jatkuu…
- Oppimisympäristön suunnittelu
- Opetusteknologian tuen suunnittelu
    - erityisesti tuki ohjelmoinnilliselle ajattelulle
- Raportin kirjoittaminen
- Esityksen valmistelu

Osio 6 – Projektien esittely

STEAM-opetuksen malliin perustuvien opetusinterventioiden suunnitelmien (projektityöt) esitttely.

Tehtävä 6.1 Esitykset ja keskustelu

Oman projektin esittely

15 min varattu aikaa jokaiselle esitykselle, josta….


10 min STEAM-opetuksen suunnitelman esitys.


5 min yleisön kysymyksille ja kommentteille.

Arviointi ja sen toteutus

Kaikki arviointitehtävät tulisi olla tehtynä ennen määraikaa.

Arviointitehtävä arvionnin tulisi mitata moduulin oppimistavoitteiden saavuttamista	**Arviointikriteerit ja -menetelmät** kirjallisissa tehtävissä esim. pituus (sanoina), rakenne (johdanto, pääteksti, yhteenveto), käsitteiden oikeanlainen käyttö.
Arviointi suunnitelmasta ohjelmoinnillisen ajattelun soveltamiseksi valitussa STEAM-aiheessa (osio 2).	Hyväksytty/Hylätty, vertaisarviointi. Kriteerit: aiheen esitys, perustelut ja selkeys, osion sisällön hyödyntäminen.
Arviointi suunnitelmasta hyödyntää opetusteknologiaa valitussa STEAM-aiheessa (osio 3).	Hyväksytty/Hylätty, vertaisarviointi. Kriteerit: aiheen esitys, perustelut ja selkeys, osion sisällön hyödyntäminen.
STEAM-opetuksen suunnitteluprojektin raportin ja esityksen arviointi (osiot 4,5 ja 6)	Luennoitsija antaa kurssin arvosanan projektityön perusteella. Projektiraportti arvioidaan "STEAM Classroom assessment for Student Learning Experience" -kriteerien perusteella (ks. jäljempänä). Se, miten hyvin projektin esitys tiivistää pääajatukset, voi antaa ylimääräisen +-pisteen lopulliseen arvosanaan (suomalaisessa arviointijärjestelmässä asteikolla 0-5 plussa olisi 0,25 pistettä).

Vertaisarvioinnit

Kotitehtävät osioissa 2 & 3 vertaisarvioidaan arvosanoilla Hyväksytty/Hylätty. Oppilaan (tuleva opettaja) on suoritettava kummankin osion kotitehtävät hyväksytysti.

- Kotitehtävästä arvioidaan alla olevat piirteet skaalalla 0-5 (hyväksytyssä suorituksessa keskiarvon on oltava 2 tai parempi)
    - tehtävän aiheen esitys
    - perustelut ja selkeys
    - luvun sisällön hyödyntäminen

Oppimiskokemuksen arvionti STEAM luokkahuoneessa (lisätty ohjelmoinnillisen ajattelun arviointikriteerit)

Oppilaiden (tulevat opettajat) osioissa 4 & 5 projektityönä suunnitteleman opetusintervention arvioinnissa käytetään explisiittistä kirteeristöä. Arvosana on 0-5 (0 hylätty). Projektin tulosten esitys osiossa 6 voi antaa yleimääräisen +-pisteen (0,25).

Lue *TeaEdu4CT_moduuli_8_arviointi.xlsx* asiakirja (alkuperäinen ks. viite alla Arviointi perustuu…)

Projektityön arviointi (0-5 kriteeriä kohden):

- Noudattaa oppiaineen/oppiaineiden opetussuunitelmaa
- Eri alojen integrointi
- Ongelmanratkaisutaitojen kehitttäminen
- Ohjelmoinnillinen ajattelu
- Opetusmenetelmät
- Arviointitapa
- Tasa-arvoinen osallistuminen

Arviointi perustuu:

Quigley, C. F., Herro, D., Shekell, C., Cian, H., & Jacques, L. (2020). Connected learning in STEAM classrooms: Opportunities for engaging youth in science and math classrooms. International Journal of Science and Mathematics Education, 18(8), 1441-1463. https://doi.org/10.1007/s10763-019-10034-z

ESM 1 (Sähköinen lisämateriaali; https://doi.org/10.1007/s10763-019-10034-z)

Projektiryhmä voi saada ylimääräisen +-pisteen (suomalaisessa arvostelussa skaalalla 0-5 tämä vastaa 0,25 pistettä) projektin keskeisten ideoiden esittämisestä (osio 6).

Moduulin räätälöinti

Osioiden 4 ja 5 projektityön laajuutta muuttamalla voidaan helposti lisätä tai vähentää moduulin työmäärää. Projektia voidaan laajentaa valitsemalla kohteeksi opetusinterventioiden sarjan tai kokonaisen kurssin suunnittelun. Moduulin työmäärää voidaan myös vähentää tarjoamalla valmiita oppimateriaaleja ja mallipohjia. Ääritapauksessa projektityö voisi olla henkilökohtainen tehtävä, jossa kehitetään oppimisintervention idea, joka esitetään osiossa 6.

Lähteet

Bermúdez, J. (2020). Cognitive Science: An Introduction to the Science of the Mind (3rd ed.). Cambridge: Cambridge University Press.

Blockley, D. (2012). Engineering: a very short introduction. OUP Oxford.

Checkland, P., & Holwell, S. (1998). Information, systems, and information systems. Chichester: John Wiley & Sons.

Chevalier, M., Giang, C., Piatti, A., & Mondada, F. (2020). Fostering computational thinking through educational robotics: a model for creative computational problem solving. International Journal of STEM Education, 7(1), 1-18.

Dasgupta, S. (2016). Computer science: a very short introduction (Vol. 466). Oxford University Press.

Davis, M., Sigal, R., & Weyuker, E. J. (2015). Computability, complexity, and languages: fundamentals of theoretical computer science. 2. edition. Elsevier.

Denning, P. J. (2007). Computing is a natural science. Communications of the ACM, 50(7), 13-18.

Denning, P. J. (2017a). Computational Thinking in Science. American Scientist, 105(1), 13-17.

Denning, P. J. (2017b). Remaining trouble spots with computational thinking. *Communications of the ACM*, 60(6), 33-39

Devlin, K. (1994). *Mathematics: The science of patterns: The search for order in life, mind and the universe*. Macmillan.

Devlin, K. J. (2012). *Introduction to mathematical thinking*. Palo Alto, CA: Keith Devlin.

Dong, Y., Catete, V., Jocius, R., Lytle, N., Barnes, T., Albert, J., ... & Andrews, A. (2019). PRADA: A Practical Model for Integrating Computational Thinking in K-12 Education. *In Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 906-912).

Gilbert, J. and Boulter, C. (1998). Learning Science Through Models and Modelling. International Handbook of Science Education vol 1 ed B J Fraser and K G Tobin (Dordrecht: Kluwer Academic) pp 53–66

Gowers, T. (2002). Mathematics: A very short introduction (Vol. 66). Oxford Paperbacks.

Gutiérrez, R. and Pintó, R. (2005). Teachers' conceptions of scientific model. Results from a preliminary study ESERA 2005 Conf. Proc (Noordwijkerhout, Netherlands)

Harel, D. (1987). Statecharts: A visual formalism for complex systems. Science of computer programming, 8(3), 231-274.

Harel, I., & Papert, S. (Eds.). (1991). Constructionism. Ablex Publishing.

Hersh, R. (2014). Experiencing Mathematics: What do we do, when we do mathematics? (Vol. 83). American Mathematical Soc.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS quarterly, 75-105.

Hill, R. K. (2016). What an algorithm is. Philosophy & Technology, 29(1), 35-59.

Hutchins, N. M., Biswas, G., Maróti, M., Lédeczi, Á., Grover, S., Wolf, R., ... & McElhaney, K. (2020). C2STEM: A System for Synergistic Learning of Physics and Computational Thinking. Journal of Science Education and Technology, 29(1), 83-100.

IEEE (2010) Iso/iec/ieee 24765:2010 systems and software engineering - vocabulary

Jocius, R., Joshi, D., Dong, Y., Robinson, R., Cateté, V., Barnes, T., ... & Lytle, N. (2020). Code, Connect, Create: The 3C Professional Development Model to Support Computational Thinking Infusion. *In Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 971-977).

Justi, R. S. and Gilbert, J. (2002). Modelling, teachers' views of the nature of modelling, and implications for the education of modellers Int. J. Sci. Educ. 24 369–87

Kramer, J. (2007). Is abstraction the key to computing? Communications of the ACM, 50(4), 36-42.

Kvasz, L. (2019). How Can Abstract Objects of Mathematics Be Known?. *Philosophia Mathematica*, 27(3), 316-334.

Laurillard, D. (2002). Rethinking University Teaching in the Digital Age.

Laurillard, D. (2012). Teaching as a design science: Building pedagogical patterns for learning and technology. Routledge.

Laurillard, D., & McAndrew, P. (2003). Reusable educational software: A basis for generic learning activities. Reusing online resources: A sustainable approach to e-learning, 81-93.

Lopez, V., & Hernandez, M. I. (2015). Scratch as a computational modelling tool for teaching physics. Physics Education, 50(3), 310.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. ACM Transactions on Computing Education (TOCE), 10(4), 1-15.

Malyn-Smith, J., Lee, I. A., Martin, F., Grover, S., Evans, M. A., & Pillai, S. (2018). Developing a framework for computational thinking from a disciplinary perspective. In Proceedings of the International Conference on Computational Thinking Education (p. 5).

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. AI magazine, 27(4), 12-12.

Miller, G. A. (2003). The cognitive revolution: a historical perspective. Trends in cognitive sciences, 7(3), 141-144.

NRC (National Research Council). 2012. A framework for K–12 science education: Practices, crosscutting concepts, and core ideas. Washington, DC: National Academies Press.

NGSS Lead States. 2013. Next Generation Science Standards: For states, by states. Washington, DC: National Academies Press. www.nextgenscience.org/next-generation-science-standards.

Okasha, S. (2016). Philosophy of Science: Very Short Introduction. Oxford University Press.

Papert, S. (1980). Mindstorms: Computers, children, and powerful ideas. NY: Basic Books, 255.

Papert, S. (1996). An exploration in the space of mathematics educations. International Journal of Computers for Mathematical Learning, 1(1), 95-123.

Pears, A. (2019). Developing Computational Thinking," Fad" or" Fundamental"?. *Constructivist Foundations*, 14(3).

Pei, C., Weintrop, D., & Wilensky, U. (2018). Cultivating computational thinking practices and mathematical habits of mind in lattice land. Mathematical Thinking and Learning, 20(1), 75-89.

Perrenet, J., Groote, J. F., & Kaasenbrood, E. (2005). Exploring students' understanding of the concept of algorithm: levels of abstraction. ACM SIGCSE Bulletin, 37(3), 64-68.

Puentedura, R. (2006). Transformation, technology, and education [Blog

post]. Retrieved from http://hippasus.com/resources/tte/.

Quigley, C. F., Herro, D., & Jamil, F. M. (2017). Developing a conceptual model of STEAM teaching practices. *School Science and Mathematics,* 117(1-2), 1-12.

Quigley, C. F., & Herro, D. (2019). An Educator's Guide to STEAM: Engaging Students Using Real-World Problems. Teachers College Press.

Quigley, C. F., Herro, D., Shekell, C., Cian, H., & Jacques, L. (2020). Connected learning in STEAM classrooms: Opportunities for engaging youth in science and math classrooms. International Journal of Science and Mathematics Education, 18(8), 1441-1463. https://doi.org/10.1007/s10763-019-10034-z

Rich, P. J., Egan, G., & Ellsworth, J. (2019, July). A Framework for Decomposition in Computational Thinking. In Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (pp. 416-421).

Sfard, A. (2007). When the rules of discourse change, but nobody tells you: Making sense of mathematics learning from a commognitive standpoint. The journal of the learning sciences, 16(4), 565-613.

Tedre, M. (2018). The nature of computing as a discipline. *Computer science education: Perspectives on teaching and learning in school*, 2.

Tedre, M., & Moisseinen, N. (2014). Experiments in computing: A survey. The Scientific World Journal, 2014.

Thimbleby, H. (2007). Press on: Principles of Interaction Programming.

Wan, X., Zhou, X., Ye, Z., Mortensen, C. K., & Bai, Z. (2020, June). SmileyCluster: supporting accessible machine learning in K-12 scientific discovery. In Proceedings of the Interaction Design and Children Conference (pp. 23-35)

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127-147.

Weintrop, D. (2019). Block-based programming in computer science education. Communications of the ACM, 62(8), 22-25.

Wild, C. J., & Pfannkuch, M. (1999). Statistical thinking in empirical enquiry. International statistical review, 67(3), 223-248.

Wilson, K. G. (1989). Grand challenges to computational science. Future Generation Computer Systems, 5(2-3), 171-189.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.

Wolff, A., Gooch, D., Montaner, J. J. C., Rashid, U., & Kortuem, G. (2016). Creating an understanding of data literacy for a data-driven society. The Journal of Community Informatics, 12(3).