Co-funded by the Erasmus+ Programme of the European Union

8 modulis

Edukacinė informatinio mąstymo aplinka: projektavimas ir integravimo aspektai

Autoriai: Turku universitetas (Suomija)
Peter Larsson
Ashok Veerasamy

Receenzentai:
Panagiotis Kosmas (CARDET),
Mart Laanpere (TLN)
Vaida Masiulionytė-Dagienė (VU)

Išoriniai reecenzentai:
Andreas Mühling (Vokietija),
Filiz Kalelioğlu (Turkija)

Dizainas (piktogramos ir schemos):
Vaidotas Kinčius (Lietuva)

Pilotavimas:
CARDET (Kipras), Turku universitetas (Suomija), Vilniaus universitetas (Lietuva)

Vertė:
Vaida Masilionytė-Dagienė

Modulis sukurtas įgyvendinant „Erasmus+“ KA2 projektą „Būsimųjų mokytojų rengimas: Informatinis mąstymas ir STEAM“ (Future Teachers Education: Computational Thinking and STEAM TeaEdu4CT). Projektą koordinuoja prof. Valentina Dagienė, Vilniaus universitetas. Partneriai: Vienos technologijos universitetas (Austrija), CARDET (Kipras), Talino universitetas (Estija), Turku universitetas (Suomija), Paderborno universitetas (Vokietija), CESIE (Italija), Neimegeno Radboudo universitetas (Nyderlandai), KTH karališkasis technologijos institutas (Švedija), Ankaros universitetas (Turkija).

Turinys

Bendra apžvalga ir tikslas

Šiame modulyje aptariama, kaip sukurti edukacines aplinkas, kurios padėtų integruoti informatinį mąstymą su gamtos mokslais, technologijomis, inžinerija, menais ir matematika (STEAM). Edukacinės aplinkos pateikia technologijas, padedančias taikyti informatinį mąstymą. STEAM ugdymas sujungia menus su STEM (gamtos mokslais, technologijomis, inžinerija ir matematika), kad šių dalykų ansamblis taptų prieinamas platesniam mokinių ratui ir būtų skatinamas kūrybiškumas. Menai suteikia kontekstą ir kūrybiškumą STEM taikymui, o informatinis mąstymas daro įtaką šių dalykų praktinei daliai. Informatinis mąstymas suprantamas kaip bendras požiūris ir įgūdžių rinkinys, padedantis integruoti informatikos metodus į kitų disciplinų praktiką.

Šiame modulyje trys požiūriai (matematikos, inžinerijos ir gamtos mokslų) į informatiką kartu su informatinio mąstymo modeliu naudojami kaip sistema. Ši sistema padeda mokytojui projektuoti švietimo technologijų naudojimą, kad būtų lengviau integruoti informatinį mąstymą ir STEAM. Kiekvienas požiūris pateikia būdus kaip integruoti informatinį mąstymą su atitinkamu dalyku. Technologijų (T raidė trumpinyje STEM) dalykas čia pateikiamas kaip apimantis inžinerijos požiūrį. Studentai mokosi planuoti panaudoti ugdymo technologijas iš šių trijų požiūrių perspektyvos. Studentai taip pat pasiskirsto į komandas, kad sukurtų mokymosi intervencijos projektą, kuriame būtų naudojamas technologijų rinkinys, palengvinantis STEM dalykų integraciją. Menų (A raidė trumpinyje STEAM) vaidmuo, komandoms planuojant mokymosi intervenciją, yra suteikti kontekstą, įtraukti ir suteikti erdvės kūrybiškumui.

Šis modulis skirtas būsimiems STEAM dalykų mokytojams.

Tikslinės grupės ir išankstinės sąlygos

Tikslinę grupę sudaro būsimi STEM, STEAM arba vieno iš STEAM dalykų mokytojai.

Jokių išankstinių sąlygų nėra, išskyrus geras savo pagrindinio(-ių) dalyko(-ų) žinias ir reikalingos pedagoginės studijos.

Raktiniai žodžiai

mokymo projektavimas, edukacinės technologijos, informatinis mąstymas, STEM, STEAM

Susijusios kompetencijų sistemos

Šiame modulyje pateikiamos šios „DigCompEdu“ kompetencijų modelio kompetencijos (neaptartos kompetencijos neįtrauktos; išsamų modelio aprašymą žr. 1-o modulio 3-ame skyriuje (Valentina Dagienė)):

1. Profesinis įsitraukimas

1.3. Refleksijos praktika	Individualiai apmąstyti, kritiškai įvertinti ir aktyviai plėtoti savo ir savo švietimo bendruomenės skaitmeninę pedagoginę praktiką.
	Šis modulis suteikia aiškią sistemą, leidžiančią mąstyti apie skaitmeninių švietimo technologijų naudojimą mokant informatinio mąstymo ir STEAM.
2. Skaitmeniniai ištekliai	
2.1. Skaitmeninių išteklių pasirinkimas	Nustatyti, įvertinti ir pasirinkti skaitmeninius išteklius mokymui ir mokymuisi. Renkantis skaitmeninius išteklius ir planuojant jų naudojimą atsižvelgti į konkretų mokymosi tikslą, kontekstą, pedagoginį metodą ir besimokančiųjų grupę.
	Egzistuoja daugybė švietimo technologijų, todėl šiame modulyje daugiausia dėmesio skiriama mokymuisi atpažinti šablonus, kurie rodo galimybes panaudoti tam tikrą technologiją.
2.2. Skaitmeninių išteklių kūrimas ir redagavimas	Redaguoti esamus atvirai licencijuojamus išteklius ir kitus išteklius, kur tai yra leidžiama. Kurti naujus skaitmeninius švietimo išteklius arba būti jų kūrimo bendraautoriumi. Kuriant skaitmeninius išteklius ir planuojant jų naudojimą atsižvelgti į konkretų mokymosi tikslą, kontekstą, pedagoginį metodą ir besimokančiųjų grupę.
	Nors bazinės švietimo technologijos jau egzistuoja, paprastai jas reikia modifikuoti, kad atitiktų numatytus tikslus. Šiame modulyje studentai pirmiausia mokosi planuoti, kaip taikyti technologiją, o paskui – kaip ją įgyvendinti.
3. Mokymas ir mokymasis	
3.1. Mokymas	Planuoti ir diegti skaitmeninius prietaisus ir išteklius mokymo procese, kad būtų padidintas mokymo priemonių veiksmingumas. Tinkamai valdyti ir organizuoti skaitmenines mokymo veiklos intervencijas. Eksperimentuoti ir kurti naujus mokymo formatus ir pedagoginius metodus.
	Norint, kad švietimo technologijos būtų sėkmingai taikomos, jos turi būti pritaikytos temos kontekstui ir padėti mokymuisi reikalingoms veikloms. Praktinė šio modulio dalis skirta švietimo technologijų integravimui į mokymą ir mokymąsi.
3.4. Savarankiškas mokymasis	Naudoti skaitmenines technologijas savarankiško mokymosi procesuose, t. y. suteikti besimokantiesiems galimybę planuoti, stebėti ir apmąstyti savo mokymąsi, pateikti pasiektos pažangos rezultatus, dalytis įžvalgomis ir ieškoti kūrybiškų sprendimų.
	Šiame modulyje pateikiama teorija, kuria būsimasis mokytojas gali naudotis kurdamas inovatyvius mokymo planus, padedančius savarankiškam mokymuisi. Naudojamos švietimo technologijos turėtų padėti mokytis, bet ne pateikti paruoštus sprendimus, kurie slopina kūrybiškumą ir neleidžia patirti atradimo džiaugsmo.
4. Vertinimas	
4.1. Vertinimo strategijos	Naudoti skaitmenines technologijas formuojamajam ir apibendrinamajam vertinimui. Didinti vertinimo formų ir metodų įvairovę ir tinkamumą.

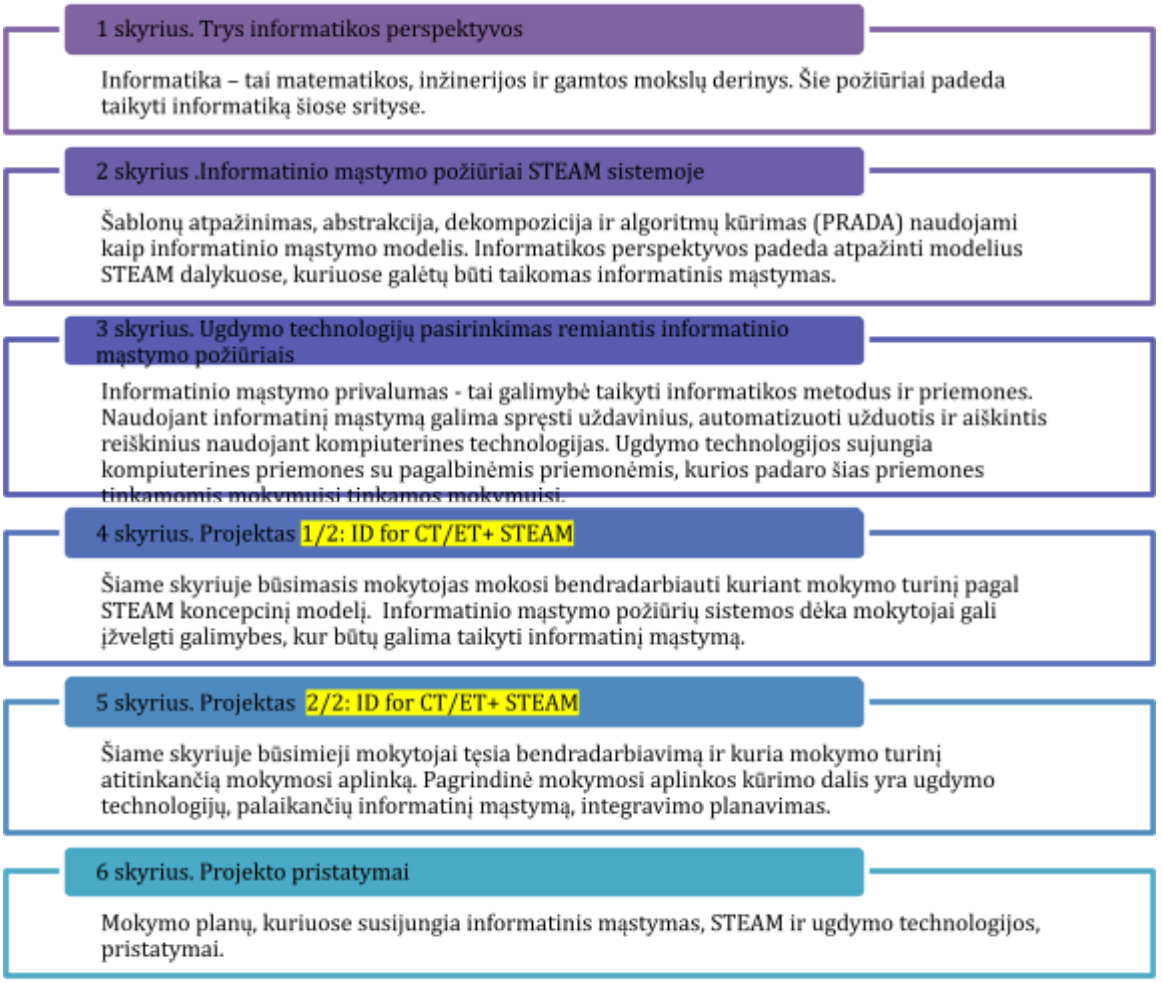
	Informatinio mąstymo ir STEAM edukacinėmis technologijomis siekiama padėti kurti ar atrasti sprendimus. Rezultatai pasiekiami palaipsniui, todėl galima taikyti formuojamąjį vertinimą. Paprastai gautas rezultatas yra daugiau nei jo dalių visuma, todėl jis tinka ir apibendrinamajam vertinimui.
5. Įgalinti besimokančiuosius	
5.3. Aktyvus besimokančiųjų įtraukimas	Naudoti skaitmenines technologijas, kad būtų skatinamas aktyvus ir kūrybiškas besimokančiųjų įsitraukimas į mokomąjį dalyką. Naudoti skaitmenines technologijas pedagoginėse strategijose, kurios skatina besimokančiųjų tarpdalykinius įgūdžius, atveria mokymąsi naujoms, realaus pasaulio situacijoms, įtraukia pačius besimokančiuosius į praktinę veiklą, mokslinius tyrimus ir sudėtingų problemų sprendimą arba kitais būdais didina besimokančiųjų aktyvumą ir kūrybinę raišką.
	Visa tai paskatino, kad prie STEM dalykų būtų pridėti liberalieji menai. Informatinio mąstymo įtraukimas į daugiau dalykų padeda taikyti skaitmenines technologijas.
6. Besimokančiųjų skaitmeninės kompetencijos gerinimas	
6.5. Skaitmeninių problemų sprendimas	Įtraukti mokymosi ir vertinimo veiklas, kurios reikalauja, kad besimokantieji identifikuotų ir spręstų technines problemas arba kūrybiškai perkeltų technologines žinias į naujas situacijas.
	Informatinis mąstymas iš esmės yra problemų sprendimo metodas. Jis atveria galimybę taikyti informacines technologijas sprendimui formuoti. Kurdami įdomias ir įtraukiančias užduotis, mokytojai perteikia savo žinias mokiniams.

Mokymosi rezultatai ir vertinimo metodai

Sėkmingai besimokantis galės...	Vertinimo metodai
atpažinti informatinio mąstymo taikymo galimybes STEAM dalykuose (analizė)	Plano (pagrįsto PRADA), kuriame aprašoma, kaip informatinis mąstymas galėtų būti taikomas STEAM dalyke(-uose), tarpusavio peržiūra.
sugebėti pasirinkti tinkamas ugdymo technologijas, padedančias mokytis STEAM (taikymas)	Plano (pagrįsto informatinio mąstymo perspektyvomis STEAM srityje), kaip pasirinkti tinkamą technologiją, padedančią mokytis STEAM dalykų, tarpusavio vertinimas.
gebėti kurti informatiniu mąstymu ir STEAM grindžiamą mokymosi priemonę, naudojant švietimo technologijas (kūrybiškumas)	Mokymo planavimo projekto ataskaita ir pristatymas

Modulio planas ir didaktiniai metodai

Šis modulis – tai profesinio tobulėjimo modelio programavimas, sujungimas ir kūrimas (angl. *Code, Connect and Create*, 3C) (Jocius ir kt., 2020), kuriuo siekiama padėti mokytojams integruoti disciplinos turinį ir informatinio mąstymo principus, tobulinimas. Programavimas čia yra apibendrintas iki ugdymo technologijų, skatinančių informatinį mąstymą, naudojimo, o disciplinos aiškinamos STEAM kontekste. 3C naudojamas informatinio mąstymo modelis PRADA (Dong ir kt., 2019), kurio esminiai principai apibrėžiami terminais, tinkamais mokytojams, neturintiems gerų informatikos srities žinių. Nors pažįstamas žodynas yra naudingas, jis gali neįtraukti tokių žinių, kurios yra informatinio mąstymo kontekstinio pagrindo dalis. Šiame modulyje PRADA modelio sąvokas išgryniname pristatydami matematikos, inžinerijos ir tiksliųjų mokslų požiūrius į informatiką, kurie padėtų jas lengviau taikyti.



Skyriai ir veiklos

1 skyrius. Trys informatikos perspektyvos

Vienas iš informatinio mąstymo iššūkių yra tai, kad jis labai gerai tinka STEM temoms. Atrodo, kad jis yra STEM dalis, bet iš tiesų jis tinka visur, kur spendžiamos problemos (Pears, 2019). Galima sakyti, kad informatinis mąstymas yra gebėjimas pastebėti informatikos metodų ir priemonių galimybes kartu turint žinių kaip visa tai galima pritaikyti. Norint taikyti informatinio mąstymo gebėjimus reikia būti įvaldžius informatikos pagrindus, kas atitinka Wing (2006) mintis. Vis dėlto nėra vieningos nuomonės, kas yra informatika, o galima sakyti, kad ją sudaro trys pagrindinės perspektyvos: matematika, inžinerija ir gamtos mokslai (Tedre, 2018). Tai su STEM filosofiniais pagrindais. Šiame skyriuje informatika pristatoma per šių trijų perspektyvų prizmę.

Informatika (anglų kalba dar vadinama *computer science* ir *computing*) yra jaunas mokslas. Pirmoji informatikos katedra buvo įkurta 1962 m. Purdue universitete. Padedant informatikai, kompiuteriai ir susijusios technologijos sparčiai kito – nuo pirmojo kompiuterio ENIAC 1945 m. (užėmusio 167 kvadratinių metrų plotą) iki milijardų tarpusavyje sujungtų mobiliųjų telefonų 2020 m., kurie telpa jų savininkų kišenėse. Iki 1990 m. šios srities mokslinius tyrimus buvo galima suskirstyti į kompiuterių inžineriją, kuri daugiausia dėmesio buvo skiriama techninei įrangai, teorinę informatikos dalį, kur tiriamas programavimas, ir informacines sistemas, kurios susijusios su kompiuterių naudojimu versle. Po 1990 m. atsirado dar dvi pakraipos. Programinės įrangos inžinerija tiria programinės įrangos kūrimą. Įmonių informacinės technologijos buvo skirtos technologiniams procesams valdyti ir palaikyti. Informatikos mokslinių tyrimų sričių skaičius auga: 1996 m. jų buvo 12, o 2005 m. – 26. Nors atrodo, kad sritys skiriasi, yra ir kai kas bendro visoms. Informatika yra unikalus matematikos, inžinerijos ir gamtos mokslų derinys.

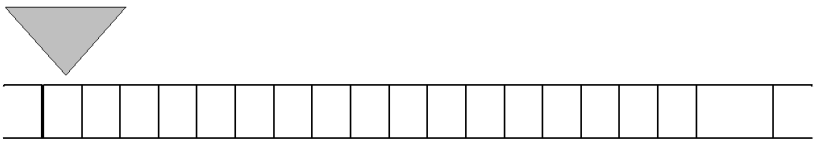
Matematika

Žodis *computer* iš pradžių reiškė profesiją, kai reikėjo atlikti skaičiavimus pagal tam tikrą planą. Paprastai vieno skaičiuotojo rezultatas būdavo derinamas su kitais, kad būtų gautas numatytas rezultatas. Nors norint teisingai atlikti skaičiavimus, iš skaičiuotojų buvo reikalaujama matematinių įgūdžių, kad darbas nebūtų vien mechaninis. Skaičiavimai buvo suskirstyti į aiškiai apibrėžtus skaičiavimus, kurie savo pobūdžiu buvo pasikartojantys. Tai labai skyrėsi nuo matematiko, kuris tiria matematinių sistemų savybes arba jų taikymą, darbo. Matematikai planavo, kaip skaičiuotojai turėtų atlikti savo darbą. Pirmoji skaičiavimo mašina buvo sukurta siekiant suteikti papildomų skaičiavimo pajėgumų. Kai kurie iš skaičiuotojų toliau dirbo mašinos programuotojais.

Matematika yra svarbiausias informatikos elementas ir, galima sakyti, sudaro informatikos teorijos branduolį. Įvairios matematikos sritys pateikia skaičiavimams reikalingas sąvokas ir metodus. Matematinis bendrosios paskirties skaičiavimo mašinos modelis buvo išrastas anksčiau nei jos fizinis atitikmuo. Įvairių rūšių matematinės pagalbinės priemonės ir skaičiavimo mašinos turi ilgą istoriją ir savo techninę raidą. Tik išradus šiuolaikinį kompiuterį (mašiną) galima sakyti, kad teorija ir praktika susiliejo. Teorinis kompiuterio modelis buvo sukurtas matematiniais tikslais, siekiant išsiaiškinti, ar galima išvesti, ar formulė duos rezultatą, ar ne. Šį klausimą iškėlė matematikos milžinas Deividas Hilbertas (David Hilbert), o į jį atsakė 24 metų britų studentas Alanas Tiuringas (Alan Turing).

Hilbertas iškėlė klausimą, kaip bendru atveju spręsti uždavinius (vok. *Entscheidungsproblem*). Reikėjo sukurti bendrą metodą, kuris leistų apskaičiuoti, ar pirmosios eilės logikos formulė galioja, kitaip tariant, ar ji įrodoma. Kad tai būtų įmanoma, reikėjo aiškiai apibrėžti skaičiavimo etapus, t. y. sukurti algoritmą. Tiuringas ėmėsi spręsti šią problemą ir naudodamasis rašikliu ir languotu popieriumi sukūrė kompiuterinį modelį. Vietoj kelių eilučių popieriaus lape skaičiavimus buvo galima atlikti vienoje begalinėje eilutėje. Užteko naudoti skaičius nulis ir vienas, nes bet kokią reikšmę buvo galima užkoduoti naudojant kelis iš šių skaičių. Viename langelyje buvo galima įrašyti tik vieną skaitmenį, o skaičiavimai buvo atliekami po vieną langelį. Skaičiuodamas žmogus gali pasikliauti savo protiniais gebėjimais, tačiau čia jie buvo pakeisti aiškiomis taisyklėmis. Taip buvo aprašyta mašina su begaline langeliais suskirstyta popieriaus juosta, skaitymo ir rašymo galvute ir taisyklėmis, kurios valdė jos vykdymą.

S_1	1	1	*right*	S_2
S_2	0	0	*right*	S_2
S_2	*end*	*uneven*	*stay*	–
S_3	1	1	*right*	S_2



1 pav. Tiuringo mašiną sudarė begalinė popieriaus juosta, skaitymo ir rašymo galvutė ir jos vykdymą valdančios taisyklės. Paveikslėlyje pateiktose taisyklėse aprašytas algoritmas, skirtas patikrinti, ar vienetų skaičius yra lyginis, ar nelyginis.

Taisyklės buvo svarbiausios Tiuringo mašinoje. Kiekviena taisyklė turėjo identifikatorių ir skirtingas versijas, kurios priklausė nuo langelio po skaitymo ir rašymo galvute turinio. Taisyklė galėjo valdyti skaitymo ir rašymo galvutę, kad ji parašytų simbolį, pasislinktų žingsnį į kairę, į dešinę arba nejudėtų iš vietos juostoje. Taisyklė taip pat apibrėždavo kitą taisyklę. Naudojant taisykles ir mašinas buvo galima vienareikšmiškai apibrėžti bet kokį skaičiavimą. Dar vienas išradimas buvo tai, kad taisyklių grupė galėjo turėti bendrą identifikatorių, o tai leido sukurti modulį, pagal kurį buvo galima pakartotinai naudoti jau apibrėžtus skaičiavimus. Šią savybę Tiuringas panaudojo apibrėždamas mašiną, kuri galėjo imituoti bet kurią kitą Tiuringo mašiną, jei jos aprašymas būtų užrašytas ant įsivaizduojamos popierinės juostos. Universalioji mašina leido Tiuringui įrodyti, kad mašina, galinti patikrinti, ar kita mašina davė rezultatą, yra neįmanoma. Tai reiškė, kad nėra bendros procedūros, pagal kurią būtų galima apskaičiuoti, ar formulė galioja.

Kai 1936 m. Tiuringas ketino paskelbti savo atradimus, sužinojo, kad amerikietis Alonzo Čerčas (Alonzo Church) jį aplenkė. Čerčas naudojo kitokį skaičiavimo modelį, pagrįstą funkcijomis ir supaprastinimu. Tiuringo straipsnis vis tiek buvo paskelbtas dėl jo metodo unikalumo, tačiau vėliau jis paskelbė papildymą, kuriame pripažino, kad Čerčas buvo pirmas. Papildyme jis taip pat aprašė Tiuringo mašiną, kurioje buvo įgyvendintas Čerčo skaičiavimo modelis. Abiejų modelių skaičiavimo pajėgumas buvo vienodas. Maždaug tuo pačiu metu buvo išrasti keli kiti skaičiavimo modeliai, kurie skyrėsi skaičiavimo atlikimo būdu. Visi jie buvo vienodo pajėgumo ir apibrėžė to, ką galima apskaičiuoti, ribas.

Skaičiavimo modeliai buvo sukurti anksčiau nei šiuolaikinis kompiuteris, tačiau galima teigti, kad daugelis šiandien naudojamų idėjų buvo sugalvotos siekiant formalizuoti skaičiavimus matematikoje (Davis, 2015). Pavyzdžiui, Tiuringo mašina apėmė programavimo kalbos (taisyklių), modalumo, programos ir duomenų toje pačioje atmintyje, virtualios mašinos ir interpretatoriaus idėją. Tiuringas taip pat pasiūlė paskirstytųjų skaičiavimų idėją, kai Tiuringo mašina gali gauti dalį sprendimo iš išorinio šaltinio. Tiuringo mašina buvo mechanizuotas žmogaus skaičiavimo būdas, todėl tai buvo ne tik teorinis, bet ir intuityvus skaičiavimo modelis. Tiuringo darbo svarbos ženklas yra Alano M. Tiuringo apdovanojimas, kurį skiria Kompiuterių mašinų asociacija (ACM), viena iš dviejų pagrindinių informatikos asociacijų pasaulyje. Tiuringo mašina vis dar naudojama formaliems įrodymams teorinėje informatikos srityje.

Be skaičiavimų, matematika suteikia informatikai matematines sistemas, kurios yra skaičiavimo objektai. Todėl informatiką galima laikyti matematikos šaka (Tedre, 2018). Kai kalbama apie formalių sistemų savybių tyrimą ir šių formalizmų naudojimą kitoms sistemoms tirti, tai neabejotinai yra matematika. Tačiau, kai kalbama apie skaičiavimo modelių įgyvendinimą kompiuterinių mašinų pavidalais, kyla ir fizinio įgyvendinimo klausimas. Tai, kas prasidėjo kaip elektros inžinerija, išsivystė į savarankiškas kompiuterių inžinerijos, kuri sukūrė technologiją, ir programinės įrangos inžinerijos, kuri nustatė, kaip ją valdyti, sritis. Kompiuterių mokslas (informatika) sujungia teorijos kūrimą ir praktinį taikymą.

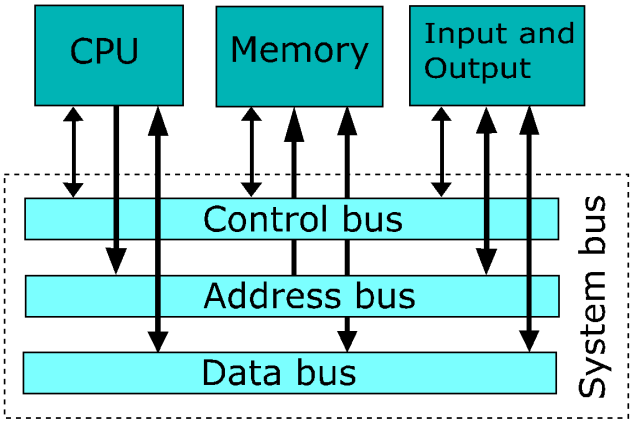
Inžinerija

Be matematikos, informatikoje labai svarbi yra inžinerija. Pirmieji kompiuteriai buvo sukurti universitetuose ir iš praktinio darbo susiformavo mokslas. Inžineriją apskritai galima apibrėžti taip: „Inžinerija – tai mokslo ir technikos žinių naudojimo disciplina, skirta įsivaizduoti, projektuoti, kurti, gaminti, eksploatuoti, prižiūrėti ir dekonstruoti sudėtingus prietaisus, mašinas, struktūras, sistemas ir procesus, padedančius žmogiškajai veiklai“ (Blockley, 2012, p. xi). Pirmasis kompiuteris buvo sukurtas Jungtinėse Amerikos Valstijose siekiant suteikti daugiau skaičiavimo pajėgumų šautuvų šaudymo lentelėms apskaičiuoti Antrojo pasaulinio karo metais. Kiekvienam naujam ginklui reikėjo maždaug 3000 reikšmių lentelės, o apskaičiuoti skaičius šimtui skaičiuotojų prireikdavo vieno mėnesio. Tiek pat laiko prireikė ir žmogaus valdomam mechaniniam diferencialiniam analizatoriui, kurių tuo metu visoje JAV buvo trys. Rengdama lenteles kariuomenė bendradarbiavo su Mūro elektrotechnikos mokykla (Moore School of Electrical Engineering) ir kartu turėjo du analizatorius ir du šimtus žmonių-skaičiuotojų. Vis dėlto jie nesugebėjo gaminti lentelių pakankamai greitai, kad neatsiliktų nuo ginklų tobulinimo.

Dėl skaičiavimo pajėgumų poreikio Mūro mokykloje kilo idėja sukurti visiškai elektroninę skaičiavimo mašiną. Kariuomenė sutiko finansuoti šią idėją ir 1943 m. pradėjo ENIAC (*Electronic Numerical Integrator and Computer*) kūrimo projektą. Projektuojant ENIAC buvo remiamasi tuo, kaip skaičiavimai buvo paskirstomi žmonėms-skaičiuotojams ir kaip diferencialinis analizatorius sujungdavo dalinius rezultatus. Siekiant išvengti veikimo sulėtėjimo, ENIAC turėjo būti visiškai elektroninis. Valdymo blokams, aritmetinėms grandinėms realizuoti ir ENIAC apdorojamiems skaičiams laikyti buvo naudojamos vakuuminės lempos. ENIAC, kaip ir dabartiniai kompiuteriai, buvo suprojektuotas kaip bendrosios paskirties (matematiniu požiūriu) kompiuteris, tačiau jis buvo paskirstytosios konstrukcijos su keliais skaičiavimo blokais ir naudojo dešimtainį skaičių vaizdavimą. Projektas baigtas 1945 m. gruodžio mėn. ENIAC buvo tūkstantį kartų greitesnis už geriausius to meto konkuruojančius skaičiavimo metodus.

ENIAC buvo sėkmingas tiek techniniu, tiek eksploataciniu požiūriu, tačiau jį buvo labai sunku programuoti. ENIAC ilgis buvo trisdešimt metrų, jį sudarė keturiasdešimt blokų. Kiekvieną įrenginį valdė komutatorius, be to, skirtingus įrenginius reikėjo sujungti jungiamaisiais laidais pagal suplanuotus skaičiavimo etapus. ENIAC komanda, bendradarbiaudama su matematiku Džonu Noimanu (*vengr.* Neumann János Lajos, *amer.* John von Neumann), sukūrė naujos rūšies kompiuterio projektą. 1945 m. buvo išplatinta preliminari ataskaita, kurioje buvo aprašytas naujojo kompiuterio projektas, čia buvo tik Noimano pavardė. Projektas buvo pavadintas Noimano architektūra, o dabartiniai kompiuteriai vis dar atitinka ją. Ši architektūra apibūdina pagrindinius kompiuterio komponentus ir jų tarpusavio sąveiką, o techninės detalės priklauso nuo įgyvendintojo (todėl ši architektūra taip ilgai naudojama).

Noimano architektūra apibūdino konstrukciją mašinos, kurią sudaro įvesties blokas, loginis ir aritmetinis blokas, valdymo blokas (šiuolaikiniuose kompiuteriuose loginis ir aritmetinis bei valdymo blokai yra viename centriniame procesoriuje), atmintis ir išvesties blokas. Skirtingi blokai buvo sujungti lygiagrečiąja magistrale. Kompiuteris valdomas komandomis, kurios saugomos atmintyje kartu su duomenimis. Komandos ir duomenys pateikiami dvejetainiais skaičiais, t. y., nuliais ir vienetais. Nuliai ir vienetai techniškai labai ekonomiški, palyginus su skaičių vaizdavimu dešimtaine sistema. Komandos vykdomos nuosekliai, o tai susiję ir su duomenų tvarkymu, ir su kompiuterio valdymu. Materialiuoju lygmeniu kompiuterio komandos yra tik kompiuterio komponentų grupė, turinti tam tikrą krūvį, kuris gali turėti įtakos kitų komponentų veikimui. Abstrakčiu požiūriu komandos sudaro numatyto skaičiavimo aprašymą, pagrįstą skaičiavimo modeliu. Tarp materialaus ir abstraktaus lygmens yra ribinis artefaktas – fizinis vykdytinos programos atvaizdas (Dasgupta, 2016).



2 pav. (CC BY-SA 3.0)[1] Noimano architektūra apibūdina pagrindinius šiuolaikinių kompiuterių komponentus ir jų tarpusavio sąveiką. Pagrindiniai komponentai yra įvesties blokas, centrinis procesorius (CPU), pagrindinė atmintis ir išvesties blokas.

Pirmieji Noimano architektūros sprendimai buvo įgyvendinti Jungtinėje Karalystėje. 1949 m. balandį Mančesterio universitete buvo parengtas „Mark I“, o vos po mėnesio Kembridže baigtas EDSAC (Electronic Delay Storage Automatic Calculator). EDSAC kūrė privati bendrovė ir universitetas. „J. Lyons & Company“ buvo didžiausia Didžiosios Britanijos viešojo maitinimo įmonė, kurios pelningumas priklausė nuo didžiulio finansų skyriaus. Jie nuolat ieškojo techninių priemonių, kurios padėtų pagerinti mechaninę finansinės veiklos dalį, ir buvo girdėję apie naująjį kompiuterį. Susitarta, kad jie finansuos EDSAC kūrimą, o vėliau universitetas padės jiems sukurti savo kompiuterį. „J. Lyons & Co.“ kompiuteris buvo pavadintas LEO (Lyons Electronic Office) ir jie jį naudojo darbo užmokesčiui, užsakymams ir siuntų valdymui bei arbatos mišinių gamybai valdyti. LEO buvo pirmasis kompiuterių pritaikymas versle, o šiam įvykiui paminėti Informacinių sistemų asociacija (AIS) skiria LEO apdovanojimą iškiliems informacinių sistemų srities mokslininkams.

Buvo galima realizuoti įvairias kompiuterio įgyvendinimo galimybes, nes Noimano architektūra apibūdino tik bendruosius kompiuterio principus. Tuo metu kompiuterius kūrė elektrotechnikos inžinieriai. Reikėjo išradingumo, kad sukurtus sprendimus būtų galima panaudoti kitiems tikslams ir pritaikyti juos kompiuteriams kurti. Programavimas buvo neatsiejama Noimano architektūros dalis. Buvo galima pasirinkti, ar funkcija bus įgyvendinta aparatine įranga, ar programuojant. Morisas Vilksas (Maurice Wilkes), vienas iš EDSAC konstruktorių, išrado mikroprogramavimą – programų rinkinį, kuris palengvino programuotojų darbą, nes programas buvo galima naudoti kaip aukštesnio lygio komandas kompiuteriui valdyti. Kompiuterių inžinerijos specializacija pradėjo atsirasti universitetų elektros inžinerijos fakultetuose, kartu buvo vis daugiau naudojami kompiuteriai.

Šiuolaikinė kompiuterių inžinerija – tai kompiuterinių sistemų, kompiuterių valdomos įrangos ir išmaniųjų įrenginių tinklų programinės ir techninės įrangos komponentų projektavimas, konstravimas, diegimas ir priežiūra. Pradžioje kompiuterių inžinerija buvo elektros inžinerijos ir informatikos derinys. Per pastaruosius keturis dešimtmečius ji vystėsi kaip savarankiška disciplina, tačiau elektros inžinerija ir informatika vis dar yra jos pamatinės disciplinos. Nors kompiuterių inžinerija prasidėjo nuo pagrindinių kompiuterių komponentų projektavimo, šiandien su jais susiję labai nedaug inžinierių. Dėl kompiuterių technologijų miniatiūrizacijos ši sritis tapo labai specializuota. Dėl silicio įtaisų mažo dydžio, didesnio patikimumo ir išbaigtų sistemų lustuose, kompiuteriai tapo visur paplitę ir pakeitė daugelį įprastinių elektroninių prietaisų. Kompiuterių inžinieriai reikalingi gaminant išmaniuosius telefonus, planšetinius kompiuterius, belaidžių tinklų įrangą ir kitus skaitmeninius produktus. Naudodami įtaisytąsias (angl. *embedded*) sistemas jie taip pat dirba su automobiliais, buitiniais prietaisais, buitine elektronika ir pastaruoju metu kuria daiktų internetą. ACM buvo labiau skirta kompiuterių

[1] W. Nowicki - darbas, paremtas diagrama, kuri, atrodo, savo ruožtu remiasi „Kompiuterių organizavimo ir architektūros pagrindų“ (The Essentials of Computer Organization and Architecture) 36 puslapiu 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

teoretikams, o IEEE (*Institute of Electrical and Electronics Engineers*) Kompiuterininkų skyrius vienija į technologijas orientuotus kompiuterijos tyrėjus.

Programavimas valdo kompiuterio veikimą, be programų kompiuteriai neveiktų. Kompiuteriai buvo sukurti matematiniams skaičiavimams, tačiau lygiagrečiai su jų fizine raida, techninė įranga, evoliucionavo ir kompiuteryje veikiančios programos, programinė įranga. Buvo sugalvotos kitos programinės įrangos panaudojimo sritys, dėl kurių programos plėtėsi. Matematinių metodų nepakako susidoroti su programavimo sudėtingumu, ir šis reiškinys imtas vadinti programinės įrangos krize. 1968 ir 1969 m. NATO surengė konferencijas kaip suvaldyti programinės įrangos sudėtingumą, kuriose buvo siūlomi inžinerijos mokslų metodai. Tai buvo pradžia naujai sričiai, pavadintai programinės įrangos inžinerija (angl. *software engineering*, SE). Šiuolaikinį programinės įrangos inžinerijos apibrėžimą pateikia IEEE: „Sistemingo, disciplinuoto, kiekybiškai įvertinamo požiūrio taikymas programinės įrangos kūrimui, eksploatavimui ir priežiūrai, t. y. inžinerijos taikymas programinei įrangai“ (IEEE, 2010).

Programinės įrangos inžinerija – tai programinės įrangos kūrimo metodai, priemonės ir jų valdymas. Pagrindinis iššūkis yra sudėtingumas: kaip struktūrizuoti programos tekstą (kodą), kad kūrimas būtų lengviau valdomas, kad prie produkto vienu metu galėtų dirbti keli žmonės ir kaip koordinuoti šią veiklą. Procedūros ir funkcijos buvo pirmieji programos struktūrizavimo metodai, idealiu atveju procedūra keičia kompiuterio būseną, o funkcija pateikia rezultatą; abi jos gali būti iškviestos iš kitų programos dalių. Objektinis programavimas – tai būdas kurti programą iš nepriklausomų komponentų. Kiekvienas objektas gali turėti keletą metodų, kurie yra procedūrų ir funkcijų abstrakcija, t. y., jie gali elgtis bet kaip. Projektavimo modeliai įkvėpimo sėmėsi iš architektūros, kad aprašytų bendrą sąveikaujančių objektų, kurie sprendžia tam tikrą problemą, modelį. Apskritai architektūra taip pat vartojama kaip terminas, apibūdinantis bendrą programinės įrangos struktūros principą.

Padarius programinę įrangą labiau modulinę, programavimo uždavinį galima padalyti keliems žmonėms. Svarbu, kad būtų susitarta dėl komponento elgsenos, nes nuo jo gali priklausyti kitos programinės įrangos dalys. Tai taip pat gali būti klaidų šaltinis, kai keičiama esama programos dalis. Šiuolaikinės programų kūrimo aplinkos leidžia ieškoti priklausomybių tarp skirtingų programinės įrangos dalių. Jos taip pat padeda parodyti bendrą programinės įrangos struktūrą ir išryškinti skirtingas komandų klases, iš kurių sudaryta programa. Šiuolaikiniai programinės įrangos projektai yra dideli, jiems kurti reikia daug žmonių. Versijų valdymo sistema padeda valdyti įvairias kuriamos programinės įrangos dalis ir versijas. Yra dvi pagrindinės programinės įrangos kūrimo projekto valdymo strategijos. Viena iš jų vadinama krioklio (angl. *waterfall*) metodu, kai programinė įranga kuriama reikalavimų, projektavimo, kūrimo, tikrinimo ir priežiūros etapais. Kitas kūrimo būdas vadinamas judriuoju (*agile*), kai kūrimas vyksta iteracijomis ir kiekvienoje iteracijoje visuma papildoma naujomis funkcijomis. Teigiama, kad krioklio metodas yra tinkamesnis didesniems projektams, o judrusis metodas – kai projektas yra mažas ar vidutinis arba kai esama neaiškumų dėl galutinio funkcijų rinkinio.

Inžinerinis požiūris susijęs su technologiniu kompiuterių aparatinės ir programinės įrangos kūrimu. Dabartinė skaitmeninė visuomenė yra technologinės plėtros nuo kompiuterių eros pradžios rezultatas. Ši raida vyko universitetuose, taip pat daugelyje įmonių, kurių produktai padarė skaitmeninę revoliuciją įmanomą. Teorinis darbas buvo svarbi šios plėtros dalis. Ypač pradžioje daugelis dalykų nebūtų buvę įmanomi, jei ne teoriniai metodai, sukurti siekiant kuo geriau išnaudoti esamas technologijas. Kai principas buvo išrastas, tada buvo galima jį plėtoti

toliau. Ir teorinio, ir techninio kūrimo pradžia buvo ta pati – žmogaus skaičiavimo modeliavimas. Kartu su šiuo modeliavimu kilo mintis imituoti žmogaus intelektą. Čia informatika tarnauja intelekto teorijai ir tai yra informatikos mokslinės perspektyvos pavyzdys.

Gamtos mokslai

Mokslo požiūriu informatika yra empirinis tyrimas, kaip ir gamtos moksluose, kur tiriami objektyvūs reiškiniai. Tai skiriasi nuo matematinės perspektyvos, kuri yra susijusi su teoriniu informatikos aspektu, ir nuo inžinerinės perspektyvos, kuri susijusi su aparatinės ir programinės įrangos kūrimu. Gamtos mokslus galima apibūdinti eksperimentavimo, stebėjimo ir teorijos kūrimo metodais (Okasha, 2016). Ne visose srityse galima taikyti eksperimentavimą, pavyzdžiui, astronomijoje, tačiau turi būti būdas rinkti duomenis teorijoms patvirtinti. 1956 m. Dartmute (JAV) buvo surengta vasaros mokykla mokslininkams, besidomintiems nauja informatikos tema – dirbtiniu intelektu. Meninio intelekto tyrimai buvo grindžiami idėja, jog „Kiekvienas mokymosi aspektas ar intelekto ypatybė iš esmės gali būti taip tiksliai aprašyta, kad būtų galima sukurti mašiną, galinčią ją imituoti“ (McCarthy ir kt., 2006, p. 2). Čia informatika vartojama gamtos mokslo prasme, siekiant apibūdinti išorinį reiškinį.

Dirbtinis intelektas sudarė sąlygas informatikai dalyvauti tarpdisciplininiuose proto tyrimuose, pavadintuose kognityviniais mokslais. Protas yra sudėtinga tema, jungianti daugelio mokslų tyrimų sritis: filosofiją, psichologiją, lingvistiką, antropologiją, neuromokslus ir informatiką. Teorinis skaičiavimų ir kompiuterio modelis suteikė kitiems mokslams priemones modeliuoti ir simuliuoti tai, kas vyksta galvoje. Galima sakyti, kad kognityviniai mokslai gimė tuo pat metu kaip ir dirbtinis intelektas Mačiusetso technologijos instituto (MIT) seminare (Miller, 2003). Garsus lingvistas Noamas Chomskis (Noam Chomsky) apibūdino, kokių gebėjimų reikia kalbai kurti. Jis palygino Tiuringo mašiną su mažesnio pajėgumo mašinomis, kad parodytų, jog kalbai sukurti reikia sistemos, prilygstančios Tiuringo mašinai. Džordžas Mileris (George Miller) parodė, kad žmogus vienu metu savo darbinėje atmintyje gali tvarkyti ne daugiau kaip septynis dalykus, tačiau grupuojant informacijos kiekį galima padidinti. Alenas Niuelas (Allen Newell) ir Herbertas Saimonas (Herbert Simon) pristatė „Logic theorist“ (Logikos teoretiką) – programą, galinčią įrodyti matematines teoremas. „Logic theorist“ naudojo euristinius algoritmus, kurie simuliavo žmogaus samprotavimus, kai jis susiduria su neapibrėžtumu.

Kitas svarbus informatikos pritaikymas proto tyrimams buvo pagrįstas Vareno MakKaloko (Warren S. McCullough) ir Valterio Pitso (Walter Pitts) idėja, kad smegenų neuronus galima interpretuoti kaip atliekančius logines funkcijas (Bermúdez, 2020). Smegenyse yra 86 mlrd. neuronų, kurių kiekvienas gali turėti daugiau kaip 10 000 jungčių ir kartu jie sudaro didelį tinklą. Žinome, kad neuronai bendrauja elektriniais signalais ir šį aktyvumą galima stebėti. Neuromokslininkams pavyko atvaizduoti įvairių smegenų sričių aktyvumą pagal kognityvines funkcijas. Tačiau vis dar nežinoma, ką šis aktyvumas reiškia, t. y., kokia reprezentacija (kaip spėjama) yra naudojama. MakKaloko ir Pitso darbe pateikiamas mechanizmas, kai dirbtiniai neuronai veikia 0 ir 1 reikšmėmis kaip kompiuteris. Tai leidžia kompiuteriu simuliuoti neuronų tinklo operacijas. Dirbtiniai neuronų tinklai leido modeliuoti smegenų veikimą, simuliuoti pažintines funkcijas ir kurti protingą programinę įrangą. Būtent tikslinių reiškinių egzistavimas ir susiję tyrimų rezultatai, su kuriais galima palyginti, lemia, ar modeliavimas yra mokslas, ar inžinerija.

Informatiką galima apibūdinti kaip natūralių ir dirbtinių informacinių procesų tyrimą (Denning, 2007). Informacinės sistemos, kaip verslo ar apskritai organizaciniai kompiuterių naudojimo tyrimai, sujungia abu požiūrius. Organizaciją galima aiškinti kaip sąveikaujančių procesų sistemą, kuri gamina produktus ar teikia paslaugas, kad įgyvendintų savo paskirtį (Checkland and Holwell, 1998). Kiekvienam procesui reikia informacijos, kad jis galėtų veikti, ir jis kuria informaciją, kuria gali remtis kiti procesai. Be to, informacija reikalinga sistemai valdyti ir koordinuoti. Informacinės sistemos atsirado iš poreikio planuoti ir projektuoti informacinių technologijų naudojimą organizacijos informaciniams poreikiams tenkinti. Tai yra projektinė informacinės sistemos pusė, tačiau egzistuoja ir elgsenos pusė, kurios pamatinis mokslas yra sociologija (Hevner ir kt., 2004). Informacinė sistema mokslo požiūriu – tai tyrimai, kaip žmonės kuria ir naudoja informaciją, kad sudarytų sąlygas organizacijos veiklai. Dar viena tyrimų sritis – kaip informacinės technologijos veikia organizacijos funkcijas.

Ir dirbtinis intelektas, ir informacinės sistemos apima mokslinius tyrimus, susijusius su išoriniais reiškiniais, kuriuose taikomi empiriniai metodai. Tačiau taip pat kyla mintis, kad teorinių ir inžinerinių informatikos perspektyvų derinys galėtų būti laikomas empiriniu mokslu. Norint, kad tyrimai būtų kvalifikuojami, jie turėtų būti griežti, o taikomi metodai prilygti gamtos mokslų metodams. Literatūroje galima rasti penkis pagrindinius požiūrius į eksperimentinę informatiką (Tedre ir Moisseinen, 2014): galimybių, bandymų, lauko, palyginimo ir kontroliuojamą eksperimentą.

Galimybių eksperimentas susijęs su naujais metodais ir priemonėmis. Siekiama ištirti, ar užduotį galima automatizuoti ekonomiškai, efektyviai, įgyvendinamai, patikimai ar pagal kitus paprastus kriterijus. Eksperimento metodas –įrodymas, kad tai galima padaryti. *Bandomasis eksperimentas* yra platesnis nei įgyvendinamumas ir juo siekiama įvertinti, ar sistema patvirtina savo specifikaciją arba kaip gerai ji veikia. Jos veikimas vertinamas pagal iš anksto nustatytų kintamųjų rinkinį. Eksperimentas paprastai atliekamas laboratorijoje, tačiau gali būti atliekamas ir realiomis sąlygomis, jei atsižvelgiama į mažesnę aplinkos kontrolę. Pastarasis variantas taip pat yra *lauko eksperimento* aplinka, kai siekiama išbandyti naudojamą sistemą. Čia eksperimento dalis yra realios aplinkos nenumatytos aplinkybės. Siekiama išbandyti sistemos veikimą, tinkamumą naudoti arba patikimumą. Tokie eksperimentai dažnai naudojami informacinių sistemų srityje, kur sėkmė matuojama praktine nauda.

Daugelyje informatikos šakų ieškoma geriausiai veikiančio sprendimo. Tai gali būti susiję su išteklių naudojimu, greičiu, išsamumu, paslaugų kokybe ar kitu veiksniu, pagal kurį galima spręsti, ar rezultatas yra geresnis, ar blogesnis. Jie vadinami *lyginamaisiais eksperimentais,* kurių tikslas – įvertinti, ar vienas sprendimas yra geresnis už kitą, kai eksperimentai atliekami tomis pačiomis sąlygomis. Galima sakyti, kad *kontroliuojamasis eksperimentas* yra mokslo aukso standartas, jis taip pat gali būti naudojamas informatikos kontekste. Tai reiškia, kad stengiamasi kontroliuoti visus įtakos turinčius veiksnius. Paprasčiausias tipas turi teigiamą ir neigiamą kontrolę, kurių paprastai pakanka kintamųjų neapibrėžtumui pašalinti. Kontrolė reiškia, kad neigiami ir teigiami rezultatai gaunami tada, kai jų tikimasi.

Kompiuterinės technologijos ir jų paplitimas taip pat pakeitė mokslą, kaip ir kitas visuomenės veiklos sritis. Greta tradicinių eksperimentinių ir teorinių tyrimų dabar egzistuoja trečiasis metodas – kompiuteriniai tyrimai. Tačiau ir eksperimentiniai bei teoriniai metodai pasikeitė dėl šiuolaikinės kompiuterijos galimybių (Denning, 2017a). Eksperimentinis mokslas – tai duomenų rinkimas naudojantis stebėjimais ir eksperimentais, siekiant patvirtinti arba atmesti

hipotezę. Kompiuteriai leidžia tvarkyti ir analizuoti didelius duomenų rinkinius. Kadangi pastaraisiais metais kompiuterinių išteklių prieinamumas masiškai išaugo, mokslininkui nebereikia tenkintis statistine imčių analize, jis gali analizuoti visą duomenų rinkinį. Teoriniuose moksluose mokslininkas kuria matematinius modelius, kuriais paaiškina tai, kas žinoma, ir naudoja modelius hipotezėms apie galimus reiškinius kelti. Kompiuteriai padeda apskaičiuoti lygtis, kuriomis grindžiami modeliai. Kompiuterių naudojimas nepakeitė fakto, kad mokslui kurti reikia ir duomenų, ir modelių (teorijos).

Kompiuterių naudojimas eksperimentiniams ir teoriniams tyrimams spartinti yra revoliucija (Denning, 2017a). Tačiau skaičiavimai turi daugiau galimybių. Vietoj statinių modelių galima kurti dinamines simuliacijas, kurios įgyvendina aprašomus procesus. Tai leidžia tyrinėti dalykus, kurių negalima stebėti, arba leidžia pamatyti sudėtingų sąveikaujančių sistemų pasekmes. Šių sistemų nebūtų galima apskaičiuoti, nes yra per daug kintamųjų lygtims, tačiau galima sukurti komponentus ir leisti kompiuteriui pasirūpinti sąveika. Dirbtinio intelekto srityje kompiuteriniai skaičiavimai buvo naudojami simuliuoti, kaip žmogus apdoroja informaciją. Atrodo, kad ir kitus gamtos procesus galima interpretuoti kaip informacinius procesus (Denning, 2017a), nes jie turi pradinius duomenis (įvestis) ir sukuria rezultatus (išvestis). Kompiuteris yra ypač tinkamas įrankis tokio pobūdžio rezultatams apskaičiuoti. Informatika – tai moksliniai kompiuterių technologijų tyrimai; jie leidžia suprasti žmogaus intelekto ir organizacijų veikimą, be to, tai puiki priemonė kitiems mokslams. Tačiau tam, kad informatika būtų mokslas, reikia laikytis mokslinio metodo, net jei jis pritaikytas informatikos ypatybėms.

Santrauka

Yra aiškūs įrodymai, patvirtinantys tris informatikos perspektyvas. Matematika pateikia skaičiavimų teoriją, aiškiai ir griežtai apibrėždama algoritmą, kuris yra visų skaičiavimų pagrindas (žr. 1.1 lentelę, antrą stulpelį iš kairės). Tai leidžia ištirti, kokius uždavinius galima išspręsti skaičiavimais pasitelkus matematiką. Kompiuteriai išplečia matematikos galimybes, nes leidžia tvarkyti didelius duomenų kiekius ir labai sudėtingas sistemas, ko nebūtų įmanoma atlikti rankiniu būdu. Mūsų skaitmeninė visuomenė yra informacinių technologijų plėtros kompiuterių ir programinės įrangos inžinerijos srityje rezultatas. Inžinerinė kompiuterijos pusė nuolat randa naujų taikymo sričių, tyrinėdama, kokius uždavinius galima atlikti valdant informacines technologijas (žr. 1.1 lentelę. trečią stulpelį iš kairės). Informacinės technologijos moksliškai naudojamos žmogaus intelektui simuliuoti ir kaip informacijos apdorojimo modelis organizacijose. Daugelis mokslinių reiškinių yra per dideli, kad juos būtų galima suprasti tradicinėmis priemonėmis. Mechanizmai, lemiantys kai kuriuos reiškinius, ne visada yra stebimi, ir šiuo atveju kompiuteriai gali būti naudojami galimoms priežastims atsekti. Moksliniais skaičiavimo procesais galima suprasti reiškinius, kurie be skaičiavimų nebūtų prieinami (žr. 1.1 lentelę, ketvirtą stulpelį iš kairės).

1.1 lentelė. Trys informatikos perspektyvos

	Matematika	Inžinerija	Gamtos mokslai
Objektas	Skaičiavimo modeliai	Kompiuterių ir programinės įrangos technologijos	Natūralūs ir dirbtiniai informaciniai procesai

Metodas	Skaičiavimai ir įrodymai	Elektronika ir programavimas	Empiriniai tyrimai, kompiuteriniai modeliai, simuliacijos
Žinios	Skaičiavimo galimybės ir ribos	Kaip projektuoti ir gaminti aparatinę ir programinę įrangą	Kompiuterinis reiškinių aiškinimas
Klausimas	Kokius uždavinius galima išspręsti skaičiavimais?	Kokias užduotis galima atlikti valdant informacines technologijas?	Kokius reiškinius galima suprasti kaip skaičiavimo procesus?

Paskaita. Trijų informatikos perspektyvų pristatymas (remiantis ankščiau pateiktu tekstu)

- Matematika: kokius uždavinius galima išspręsti skaičiavimais
- Inžinerija: kokias užduotis galima atlikti valdant informacines technologijas
- Mokslas: kokius reiškinius galima suprasti kaip skaičiavimo procesus

Namų darbas. Žvilgsnis į savo temą iš vienos iš informatikos mokslo perspektyvų

1. Parašyti trumpą rašinį, remiantis internetine paieška ir (arba) mokytojo pateikta papildoma medžiaga. Rašinyje turi būti paaiškinta, kaip su jūsų dalyku susijusiame pavyzdyje panaudojama informatika. Paieškai galite pasinaudoti informatikos perspektyvomis (žr. 1.1 lentelę).

2 skyrius – Informatinio mąstymo perspektyvos STEAM sistemoje

Šiame skyriuje informatikos perspektyvos derinamos su informatiniu mąstymu, siekiant sukurti informatinę bazę STE(A)M ugdymui. Trys perspektyvos – matematikos, inžinerijos ir gamtos mokslų – atsako į skirtingo tipo klausimus.

- Matematika: Kokius uždavinius galima išspręsti kompiuteriu?
- Inžinerija: Kokius uždavinius galima išspręsti naudojant informacines technologijas?
- Gamtos mokslai: Kokius reiškinius galima suprasti kaip kompiuterinius procesus?

PRADA modelis (Dong et al., 2019), skirtas informatiniam mąstymui, buvo sukurtas kaip praktinis būdas integruoti informatinį mąstymą į 12 klasės moksleivių ugdymą. Buvo siekiama suprantamai pateikti, kas yra informatinis mąstymas. PRADA akronimą sudaro žodžiai „**P**attern **R**ecognition“ (šablonų atpažinimas), „**A**bstraction“ (abstrakcija), „**D**ecomposi–tion“ (dekompozicija) ir „**A**lgorithms“ (algoritmai). Šiame skyriuje išsamiau aptarsime PRADA

informatinio mąstymo sąvokas ir pažiūrėsime, kokią įtaką jų supratimui turi informatikos perspektyvos.

> Atkreipkite dėmesį! Mes naudojame PRADA modelį, kuriame pabrėžiama konceptualioji informatinio mąstymo dalis, kad padėtume būsimiems mokytojams pasirinkti tinkamą kompiuterinę priemonę jų dalykui mokyti. Šis modelis skiriasi nuo informatinio mąstymo modelio, pristatyto 2-o modulio 1-e skyriuje (Erik Barendsen), kuriame labiau akcentuojamas uždavinių sprendimas.

PRADA modelis:

- **Šablonų atpažinimas** stebint ir nustatant šablonus, tendencijas ir dėsningumus duomenyse, procesuose ar uždaviniuose.
- **Abstrakcija –** bendrųjų principų ir savybių, kurios yra svarbios ir susijusios su uždaviniu, nustatymas.
- **Dekompozicija –** duomenų, procesų ar uždavinių skaidymas į smulkesnes, suprantamas ir lengviau valdomas dalis.
- **Algoritmai,** kuriuose žingsnis po žingsnio sukuriamos instrukcijos uždaviniams spręsti.

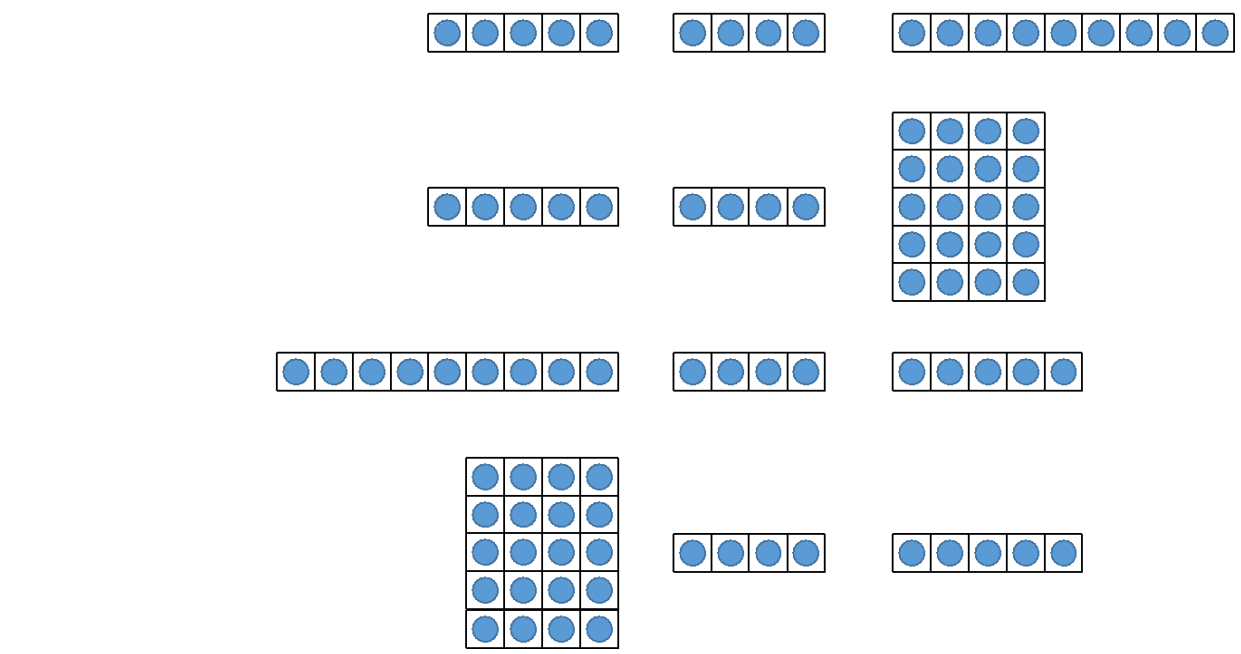
Šablonų atpažinimas

Šablonai čia aiškinami dvejopai – matematiniu ir techniniu požiūriu. Sakoma, kad matematika yra „mokslų karalienė“, nes joje pateikiamos sąvokos, kurios naudojamos formuojant mokslines teorijas. Techninis šablono aiškinimas apima ir inžineriją, ir technologiją. Skirtumas yra raidos nuo projekto iki produkto stadijoje. Matematiniai šablonai (Devlin, 1994; Kvasz, 2019) apibūdina matematikos srities dėsningumus, kuriuos galima apibrėžti nedideliu faktų ir taisyklių rinkiniu. Iš šių faktų ir taisyklių galima logiškai išvesti visas kitas tos srities sąvokas (Devlin, 2012). Šiame skyriuje naudojame šablonus, kad paaiškintume dažnai labai abstrakčias matematines sąvokas, kurias naudojame atlikdami skaičiavimus. Techniniu požiūriu informatinio mąstymo kontekste šablonai susiję su kompiuterių, elektronikos ir įstatytųjų sistemų valdymu. Keliamas reikalavimas, kad tikslinės sistemos veikimą būtų galima aprašyti diskrečiomis būsenomis ir perėjimais tarp jų. Tiek matematiniu, tiek techniniu požiūriu pirmas žingsnis sprendimo link yra atpažinti šablonus, kurie suformuoja uždavinį.

Matematikos mokymąsi galima laikyti analogišku kalbos mokymuisi (Sfard, 2007). Be kalbos negalėtume naudotis abstrakčiais matematikos objektais ir jų savybėmis. Matematiniame diskurse raktiniai žodžiai žymi objektus. Pavyzdžiui, skaičiais žymime kiekį, o figūromis – geometrinius objektus. Raktažodžiai taip pat apibūdina santykius tarp objektų, pavyzdžiui, lygybę, nelygybę, panašumą, atitikmenį ir kt. Kiekviena matematikos šaka turi savo raktažodžius, tačiau viena sritis gali naudoti kitos srities raktažodžius ir objektus, jei jie yra jos sudedamoji dalis. Kai kalbame apie matematinius objektus, vartojame žodžius, tačiau vaizdiniam informacijos perdavimui naudojame simbolių sistemą, kuri yra kur kas glaustesnė ir efektyvesnė komunikacijos priemonė. Simbolių sistemą sudaro skaitmenys, algebriniai ir loginiai užrašai bei formulės. Simboliai taip pat veikia kaip pažinimo priemonės. Taip pat galime atvaizduoti struktūras, kurias sudaro objektai ir jų ryšiai, pavyzdžiui, geometrinius brėžinius ir grafikus. Diagramos leidžia vizualiai parodyti formulių ir funkcijų rezultatus su skirtingais įvesties duomenimis.

Kiekviena matematikos šaka apibūdinama faktų ir taisyklių sistema, apibrėžiančia jos objektus. Naujas matematinis apibrėžimas turi būti įrodytas, kad jį būtų galima kildinti iš šios sistemos. Įrodymas – tai apibrėžimą sudarančių objektų ir jų tarpusavio ryšių aprašymas. Tokį aprašymą priima arba atmeta tos srities matematikai, priklausomai nuo to, kaip gerai jis atitinka nusistovėjusį atitinkamos matematinės sistemos supratimą. Kartais esama sistema gali būti pakeista, jei naujasis apibrėžimas yra esminis, tačiau tokie atvejai reti. Aprašymuose naujoms išvadoms pagrįsti naudojamos aksiomos, apibrėžtys, teiginiai ir įrodymai. Matematikos kalba yra formali ir paremta rutininiu kalbėjimo apie dominančius objektus būdu. Viena iš svarbių rutinos rūšių yra veiksmai su matematiniais objektais. Šių rutininių veiksmų pavyzdžiai yra skaičiavimai, uždavinių sprendimas, patvirtinimas ir įrodymas.

Galima sakyti, kad matematiniai objektai ir jų ryšiai sudaro šabloną. Šis šablonas yra mūsų skaičiavimų pagrindas ir suteikia mums žinių apie matematikos sritį. Paprastai šablonas yra ne matematinės sistemos branduolys, o jos pasekmė. Sąvokų ir jų santykių mokėmės matematikos pamokose arba iš knygų ir praktiškai juos išbandėme atlikdami skaičiavimus. Gavę uždavinį, galime pasinaudoti šablonais, kad atpažintume, ar kuri nors matematikos sritis padėtų jį išspręsti. Pateikiame matematinę skaičių teoriją, geriau žinomą kaip aritmetiką (žr. 2.1 pav.) matematiniam dėsningumui iliustruoti.



2.1 pav. Sudėties, daugybos, atimties ir dalybos veiksmus galima vizualizuoti ir pavyzdžiais parodyti kiekybinius dėsningumus, susijusius su simboliniais veiksmais

JAV Nacionalinės mokslinių tyrimų tarybos parengtose bendrosiose gamtos mokslų ugdymo gairėse (NRC, 2012 m.) tarp aštuonių pagrindinių mokslo ir inžinerinės praktikos sričių įvardijamas modelių kūrimas, duomenų analizė, matematinis ir informatinis mąstymas. Ši sistema pakartota ir „Amerikos ateities kartos gamtos mokslų standartuose“ (NGSS Lead States, 2013), kuriuose šios praktikos derinamos taip, kad mokslinėms išvadoms pagrįsti naudojami matematiniai atvaizdavimai (modeliai), o dideliems duomenų rinkiniams analizuoti

– skaitmeninės priemonės (informatinis mąstymas). Be to, tokiose sistemose kaip „Informatinio mąstymo matematikoje ir gamtos moksluose taksonomija“ (Weintrop et al., 2016) ir „Informatinio mąstymo integravimo elementai iš disciplinos perspektyvos“ (Malyn-Smith et al., 2018) pabrėžiama, kad modelių ir duomenų naudojimas yra svarbi mokslinės praktikos dalis. Modeliavimas ir duomenų analizė gali būti interpretuojami kaip šablonų atpažinimas.

Dažniausiai modelis reiškia idėjos, objekto, įvykio, proceso ar sistemos atvaizdavimą, sukurtą konkrečiam tikslui (Gilbert ir Boulter, 1998). Pagrindinės modelio savybės yra šios:

- Jis vaizduoja tam tikrą modeliuojamos sistemos dalį, o pasirinkti aspektai atspindi modeliuotojo interesus. Modelis yra žmogaus kūrinys, jis neegzistuoja gamtos pasaulyje.
- Ta pati tikrovė gali būti vaizduojama keliais modeliais, atsižvelgiant į tai, kokie aspektai yra įdomūs tam, kas kuria modelį.

Moksle modelis gali būti laikomas sistemos, kurią sudaro objektų ir jų savybių arba kintamųjų rinkinys, atvaizdu (Gutiérrez ir Pintó, 2005). Modelyje vaizduojami sistemos dėsniai, apibrėžiantys objektų elgseną arba objektų kintamųjų tarpusavio ryšius. Esminė modelio funkcija yra paaiškinti ir numatyti. Moksliniai modeliai taip pat yra atvaizdai, kuriais galima remtis samprotavimo tikslais (Justi ir Gilbert, 2002). Modeliai dažnai yra matematiniai, tačiau jie yra aukštesnio abstrakcijos lygmens nei paprastos matematinės sąvokos. Vietoj pagrindinių matematinės srities objektų ir operacijų modeliai juos sujungia į formules, kad aprašytų dominančius reiškinius. Matematinį modelį galima suvokti kaip menamą ir supaprastintą tiriamos tikrovės dalies versiją, kurioje galimi tikslūs skaičiavimai (Gowers, 2002). Niutono mechanika yra tokio modelio pavyzdys (žr. 2.2 pav.).

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

2.2 pav. Niutono mechanika yra kitas klasikinės mechanikos pavadinimas fizikoje. Izaokas Niutonas, apibrėždamas tris judėjimo dėsnius, padėjo klasikinės mechanikos pagrindus: Pirmasis dėsnis apibrėžia, kad objektas nejuda arba toliau juda, jei jo neveikia išorinė jėga (1), Antrasis dėsnis apibrėžia, kad objektą veikiančių jėgų suma yra lygi objekto masei, padaugintai iš jo pagreičio (2), o Trečiasis dėsnis apibrėžia, kad kai vienas objektas veikia jėga kitą, antrasis objektas veikia lygiaverte jėga pirmąjį. Šie dėsniai yra mokslinių fizikos šablonų pavyzdžiai.

Ankstesnis mokslo vaizdinys grindžiamas idėja, kad reiškinio aprašymą galima supaprastinti iki tokio lygio, kad iš nustatytų kintamųjų būtų galima išgauti informaciją naudojant tikslius skaičiavimus. Ne visi reiškiniai gali būti aprašyti taip paprastai, pavyzdžiui, kai kurių žmonių populiacijų bendrieji bruožai, orų tendencijos, ekonominės prognozės ar ligų plitimas apima svyravimus, į kuriuos reikia atsižvelgti. Vietoj diskrečių ir tikslių reikšmių turime didelius duomenų kiekius, kuriuose kategorijos objektai gali turėti skirtingas tų pačių kintamųjų reikšmes. Visi reiškiniai nėra statiški, todėl gautos reikšmės gali kisti priklausomai nuo matavimo momento. Duomenys taip pat gali būti naudojami prognozėms, o be pačios prognozės, svarbu įvertinti, kiek tikras ar tikėtinas yra rezultatas. Duomenų kintamumui ir neapibrėžtumui valdyti naudojami statistiniai modeliai. Jais siekiama suprasti, ką duomenys reiškia ar numato.

Galima teigti, kad statistinis tyrimas yra procesas, susidedantis iš penkių etapų: uždavinio pateikimo, planavimo, duomenų, analizės ir išvadų (Wolff ir kt., 2016; Wild ir Pfannkuch, 1999). Pirmiausia nustatomas tyrimo tikslas ir su juo susiję svarbūs klausimai. Planavimas pradedamas nuo hipotezės formulavimo, t. y., kokių rezultatų tikimasi. Tada sudaromas planas, kuriame aprašoma, kokie duomenys galėtų patvirtinti arba atmesti hipotezę. Taip pat išvardijami galimi duomenų šaltiniai ir būdai, kaip juos galima gauti. Duomenų etape yra renkami arba įsigyjami duomenys. Čia reikia atsižvelgti į duomenų kokybę ir galimas etines problemas, pavyzdžiui, sutikimą, anonimiškumą ir įvairius leidimus. Vienas iš svarbiausių proceso etapų yra duomenų analizė, kai gaunami rezultatai ir pagal juos pateikiami paaiškinimai. Galiausiai įvertinamas paaiškinimo pagrįstumas ir pagal gautus rezultatus kuriami papildomi klausimai. Egzistuoja daugybė metodų ir priemonių, kuriomis galima pagrįsti duomenų analizę, šiuo atveju kaip pavyzdį naudosime įprastus statistinius matus, pavyzdžiui, aritmetinį vidurkį ir standartinį nuokrypį (žr. 2.3 pav.).

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

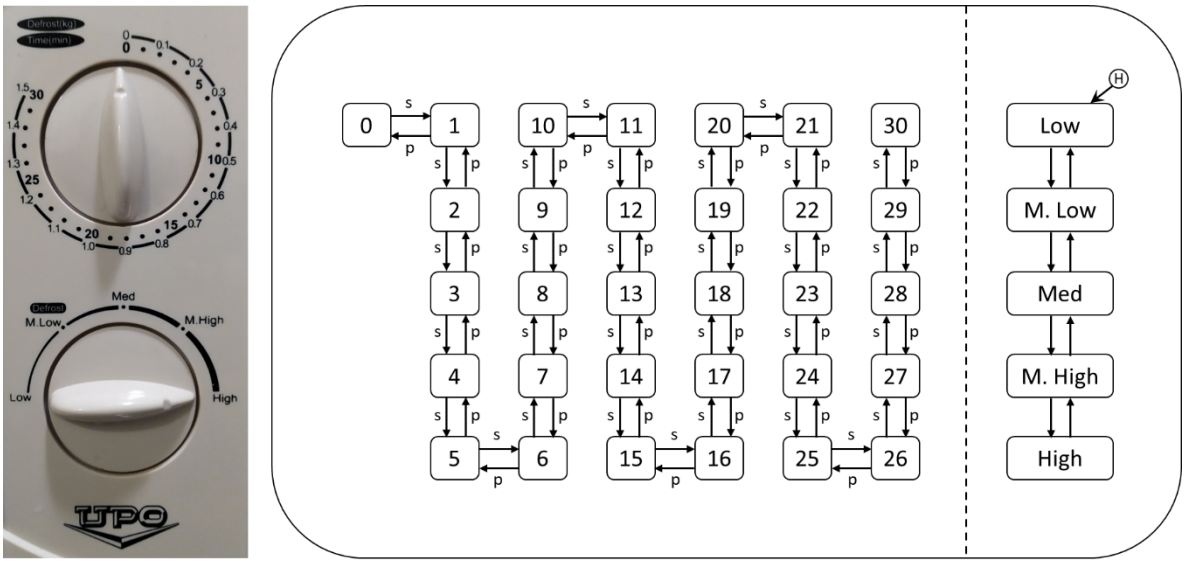
2.3 pav. Pagrindiniai statistiniai matai (aritmetinis vidurkis ir standartinis nuokrypis) apibendrina skaitiniuose duomenyse esančius dėsningumus. Aritmetinis vidurkis apibūdina vidutinę duomenų rinkinio rodiklio vertę (1). Standartinis nuokrypis apibūdina, kiek atributų reikšmės skiriasi nuo duomenų rinkinio vidurkio (2).

Įvairios inžinerijos šakos savo darbe naudoja matematikos ir gamtos mokslų modelius. Informatinio mąstymo kontekste taip pat yra techninių šablonų, susijusių su elektros, elektronikos ir skaičiavimo prietaisų valdymu ir automatizavimu. Norint valdyti elektroninį prietaisą kompiuteriniu būdu, jis turi būti kompiuterinis arba turėti elektroniką, kad būtų galima su tokiu prietaisu palaikyti ryšį. Paprastais atvejais prietaisui automatizuoti ir valdyti galima naudoti elektroniką, tačiau sudėtingesniems poreikiams tenkinti paprasčiau pridėti kompiuterinių pajėgumų, nei bandyti įrenginį sukurti naudojant tik elektroninius komponentus. Kompiuteriai ir elektronika veikia diskrečiomis būsenomis, o tai reiškia, kad tam tikras elektros krūvis arba srovė yra arba įjungta, arba išjungta. Tai nereiškia, kad prietaisas turi būti statiškas. Būklė gali būti tokia, kad variklis veikia arba net greitėja. Galima sukurti modelį, kuris apibūdintų būsenas, valdančias arba automatizuojančias įrenginį.

Įrenginio valdymo ar automatizavimo mechanizmo modeliui reikia pradinės būsenos, perėjimų tarp būsenų, perėjimus valdančių įėjimų. Toks modelis vadinamas būsenų mašina arba baigtiniu automatu (angl. *finite state machine*, FSM), nes būsenų skaičius yra ribotas (žr. 2.4 pav.). Baigtinis automatas yra paprasčiausias skaičiavimo modelis, turintis atmintį. Kombinacinė logika yra paprastesnė, nes ji gali tik apskaičiuoti rezultatą, tačiau dėl to, kad neturi atminties, negali tvarkyti būsenų. Baigtinis automatas keičia būsenas pereidamas į kitą būseną, kai gauna įvesties duomenis. Paprastai, priklausomai nuo mašinos būsenos, galimų perėjimų skaičius yra ribotas, o skirtingos būsenos gali turėti skirtingus perėjimus. Baigtinis automatas apibrėžiamas būsenų sąrašu, pradine būsena ir perėjimus inicijuojančiais įėjimais. Nors šis modeliavimas formaliai yra paprastas, tačiau juo galima modeliuoti įvairius įrenginius. Baigtinis automatas yra abstrakcija, jis aprašo prietaiso operacijų rezultatus, o ne tikrąją mechaniką.

2.4 pav. Paprastas baigtinis automatas, apibūdinantis sistemą, kurią sudaro jungiklis ir lemputė. „FSM“ aprašo lemputės būsenas, kai jungiklis keičia būsenas. Perjungus jungiklį į viršų, lemputė įsijungia, o perjungus jungiklį žemyn, lemputė išsijungia. Standartinė arba pradinė būsena yra išjungta šviesa, o jungiklis yra nuleistas.

Baigtinių automatų modeliai gali būti labai sudėtingi, jei yra daug įėjimų ir būsenų, nes reikia modeliuoti kiekvieną jų kombinaciją. Harel (1987) sukūrė būsenų diagramas, kad būtų lengviau modeliuoti baigtinius automatus (žr. 2.5 pav.). Būsenų diagramose naudojamos įvairios grafinės priemonės, kad būsenų mašinos modelį būtų lengviau interpretuoti: klasteriai, numatytosios būsenos, „ir“ būsenos ir istorijos sąsaja (Thimbleby, 2007). Jei įvestis sukelia būsenos perėjimą į grupę būsenų, kurios viena kitą papildo, tuomet jas galima vizualiai sugrupuoti į mazgą. Būklę, kuri būsenų grupėje visada yra pirmoji, galima pažymėti kaip numatytąją būseną. Jei kelios būsenos turi kažką bendro, bet skiriasi tam tikru aspektu, tai galima sumodeliuoti naudojant „ir“ būseną. Vizualiai „ir“ būsena yra klasteris, padalytas į dvi dalis, leidžiantis derinti pakaitines būsenas. Kartais reikia įsiminti paskutinę būsenų klasterio būseną, tai galima pažymėti istorijos sąsaja. Jei prie būsenų klasterio pridedama istorijos sąsaja, tada po klasterio pavadinimo papildomai įrašyta žvaigždute galima pažymėti, kad turima omenyje viena iš klasterio viduje esančių būsenų.



2.5 pav. Būsenų diagrama, apibūdinanti paprastos mikrobangų krosnelės vartotojo sąsajos veikimą. Sąsają sudaro laikmačio ratukas (0-30) ir galios lygio (žemas-aukštas) nustatymas. Paveikslo viduryje „s“ pažymėtos rodyklės apibūdina laiko nustatymą kaip būsenų keitimą, o „p“ pažymėtos rodyklės apibūdina laiko keitimą laiko impulsu. Naudojant būsenų diagramų formalumą supaprastinamas dviejų

būsenų hierarchijų derinio aprašymas, leidžiant jas keisti atskirai. Galios lygis (H) atsimena savo paskutinę būseną hierarchijoje.

Abstrakcija

Aprašyti matematikos, gamtos mokslų, inžinerijos ir technologijų modeliai yra abstrakcijų pavyzdžiai. Matematiniai objektai neegzistuoja tikrovėje, bet yra mūsų proto produktas. Filosofas Karlas Popperis išskyrė tris žmogaus patiriamos tikrovės lygmenis: fizinį pasaulį, mūsų minčių pasaulį ir nematerialius mūsų kultūros kūrinius. Matematika yra kultūros kūrinys (Hersh, 2014), ji buvo kuriama tūkstantmečius ir paskleista per mokymą ir rašymą. Mokslas naudoja matematinius ir statistinius modelius, kad tiksliai aprašytų dominančius reiškinius. Kadangi modeliai paprastai yra pagrindinių operacijų derinys ir yra pavadinti pagal vaizduojamus reiškinius, galima sakyti, kad jie yra aukštesnio abstrakcijos lygio nei paprasta matematika. Inžinerija ir technologijos kaip abstrakciją prietaiso veikimui aprašyti naudojo iš informatikos paimtas būsenų mašinas (baigtinius automatus). Būsenų mašinos neaprašo, kaip įrenginiai veikia; jos modeliuoja operacijų rezultatus. Apskritai ką nors abstrahuoti – tai praleisti detales arba sugrupuoti susijusias detales su bendra aprašomąja žyma.

Abstrakcijos naudojamos matematikoje, gamtos moksluose, inžinerijoje ir technologijose. Abstrakcija taip pat yra pagrindinė informatikos (Kramer, 2007), taigi ir informatinio mąstymo (Wing, 2006) sąvoka. Tačiau informatika ir informatinis mąstymas reikalauja mąstyti keliais abstrakcijos lygiais, dažnai dviem lygiais vienu metu. Vienas iš informatikos mokymo kontekste naudojamų abstrakcijos lygmenų pavyzdžių yra Perrenet ir kt. (2005) keturių lygmenų hierarchija: uždavinys, objektas, programa ir vykdymas (žr. toliau pateiktą sąrašą). Hierarchijos lygmenys apibūdina skirtingas algoritmo interpretacijas programavimo srityje. Uždavinio lygmeniu algoritmas gali būti suvokiamas kaip pokytis, reikalingas nuo įvesties iki išvesties. Įvestys grindžiamos šablonais, įrėminančiais uždavinį. Objekto lygmeniu galvojama apie bendrus algoritmo veiksmus, kurie reikalingi įvestims paversti išvestimis. Programos lygmeniu algoritmas įgyvendinamas programavimo kalba. Programavimo kalba yra aukšto lygio kompiuterio veikimo aprašymas arba kompiuterinis modelis. Vykdymo lygmuo yra fizinis kompiuterio veikimas. Čia žengėme žingsnius nuo šablono, kuris yra uždavinio erdvės aplinkoje, iki uždavinio sprendimo algoritmo apibrėžimo ir jo sprendimo kaip programos, vykdytinos kompiuteriu. Informatinis mąstymas yra gebėjimas atlikti šį procesą.

Algoritmų abstrakcijos lygiai pagal Perrenet et al. (2005):

- Uždavinio lygis: perėjimas nuo įvesties prie išvesties;
- Objekto lygmuo: (aukšto lygio) algoritmo aprašymas;
- Programos lygmuo: algoritmo įgyvendinimas programavimo kalba;
- Vykdymo lygmuo: kompiuterio darbas.

Dekompozicija

Dekompozicija – tai uždavinio skaidymas į lengvai išsprendžiamas dalis. Infomatinio mąstymo kontekste siekiama rasti algoritmą, kuris išspręstų uždavinį. Prieš dekompoziciją apie uždavinį žinome tik kaip apie juodąją dėžę su įėjimais, išėjimais ir galimais ryšiais su kitais uždaviniais. Juodoji dėžė vaizduoja dar nežinomą vidinę uždavinio struktūrą. Norint išsiaiškinti vidaus veikimą, uždavinys išskaidomas į daugybę smulkesnių uždavinių. Kiekvienas smulkesnis

uždavinys gali būti traktuojama kaip uždavinys, kurį reikia išskaidyti į sudedamąsias dalis. Galiausiai pasiekiamas elementarus (žemiausias) lygmuo, kuriame nebelieka smulkesnių uždavinių. Sprendimą galima sukurti remiantis nustatytomis uždavinio sudedamosiomis dalimis. Sprendimą galima rasti iš viršaus į apačią, pirmiausia sprendžiant aukščiausio lygio uždavinius, arba iš apačios į viršų, pirmiausia sprendžiant žemiausio lygio uždavinius.

Dekompozicijos procesas apima dviejų rūšių veiksmus (Rich ir kt., 2019). Pirmasis, turinio dekompozicija, yra uždavinio dalių (smulkesnių uždavinių) išskyrimas ir suskirstymas į kategorijas. Antrasis, santykinis skaidymas, skirtas ryšiams tarp smulkesnių uždavinių surasti. Atliekant turinio dekompoziciją reikia pasirinkti principą, kuriuo remiantis uždavinys suskaidomas į dalis. Principas priklauso nuo uždavinio formuluotės ir konteksto. Iš atskirų smulkesnių uždavinių gaunama informacija, kuri nebuvo prieinama, kai jie buvo sujungti. Nustačius smulkesnius uždavinius, galima pradėti jiems priskirti ryšius. Iki šio etapo tarp uždavinių nebuvo jokio ryšio. Daliniai uždaviniai ir jų ryšiai sudaro uždavinio modelį. Pavyzdys iš aritmetikos būtų dviejų natūraliųjų skaičių sudėtis. Ryšį tarp skaičių galima vertinti kaip skirtumą tarp pozicijų skaičių eilutėje. Norint pridėti vieną skaičių prie kito, reikėtų imti pirmojo skaičiaus vietą kaip pradžią ir apskaičiuoti sumą, lygią antrojo skaičiaus vietai priekyje. Naujosios pozicijos skaičius yra sudėties rezultatas.

Inžinerijos, informatikos, projektavimo ir kitose srityse uždaviniai dažnai skirstomi pagal jų funkcijas (Rich ir kt., 2019). Funkcinis skaidymas yra esminių ir santykinių procesų sujungimo rezultatas. Pagrindinė sąlyga, kad dekompozicija būtų funkcinė, yra ryšiai tarp smulkesnių uždavinių. Jei ryšys yra funkcinis, tai tarp dviejų smulkesnių uždavinių vyksta tam tikra veikla. Funkcinis santykis aritmetikoje gali būti sudėtis, atimtis, daugyba arba dalyba. Aritmetinių funkcijų naudojimo kitose matematikos srityse pavyzdžiai yra kvadratinė lygtis, didžiausias bendrasis daliklis ir Pitagoro teorema. Informatikoje algoritmas, išreikštas imperatyvaus tipo (komandų kompiuteriui) programavimo kalba, naudotų valdymo struktūras, kurios formuoja ryšį tarp smulkesnių uždavinių sprendinių. Uždavinio skaidymas naudojant esminį ir reliacinį skaidymą padeda gauti naujos reikšmingos informacijos (algoritminiam) sprendimui parengti (Rich ir kt., 2019).

Algoritmai

Algoritmai yra matematikos dalis. Kartais bet koks žingsnis po žingsnio vykdomas procesas vadinamas algoritmu, tačiau algoritmas yra ne pats procesas, o jo apibrėžimas (aprašas). Jis turi aprašyti procesą matematiškai tiksliai. Informatika (kompiuterių mokslas) naudoja daug algoritmų, nes jo veikimo principas pagrįstas skaičiavimo modeliu (žr. 1 skyriaus dalį „Matematika“). Viskas, ką daro kompiuteris, yra skaičiavimas (vadinamas kompiuteriniu skaičiavimu), bet kad mašina žinotų, ką daryti, kiekvienas veiksmas turi būti aiškus. Informatikos kontekste algoritmas gali būti apibrėžiamas taip: „*... baigtinė, abstrakti, efektyvi, sudėtinė valdymo struktūra, imperatyviai pateikta, siekianti tam tikro tikslo pagal pateiktas nuostatas*“ (Hill, 2016, p. 47).

Norint išskaidyti Robin K. Hill (2016) algoritmo apibrėžimą (žr. ankščiau), galima pradėti nuo to, kad algoritmą galima įsivaizduoti kaip procesą, susidedantį iš diskrečių žingsnių. Algoritmas apibūdina baigtinį procesą. Aprašymas yra abstraktus ir jame naudojamos tik tos savybės, kurių reikia numatytam pokyčiui pavaizduoti. Be baigtinumo, algoritmas turi būti veiksmingas, o tai reiškia, kad naudojami ištekliai ir laikas turi būti pagrįsti. Algoritmas – tai

įvairių valdymo struktūrų derinys, kuriuo vadovaujamasi keičiant jo reikšmes ir kuris galiausiai baigiasi rezultatu. Valdymo struktūros ir kintamųjų pokyčiai pateikiami kaip imperatyvai naudojant komandas. Algoritmas turi atitikti numatytą tikslą ir nieko daugiau. Algoritmas turėtų teisingai elgtis su bet kokia įvestimi iš nurodyto intervalo.

Informatinis mąstymas gali būti apibrėžiamas kaip protiniai įgūdžiai, reikalingi algoritmui automatiškai vykdyti (Denning, 2017b). Pirmiau apibrėžtos algoritmo savybės yra būtina sąlyga, kad jį būtų galima įgyvendinti, jis elgsis taip, kaip tikimasi, ir rezultatas bus teisingas. Įgyvendinti – tai parašyti programą, kurios dalis yra algoritmas. Nors kiekviena programa yra algoritminė, algoritmais vadinamos tik tos programos dalys, kurios yra pakankamai bendros, kad jas būtų galima pakartotinai panaudoti kokiame nors kitame kontekste. Šie bendrieji algoritmai turi pavadinimus, o jų apibrėžimais dalijasi kompiuterių mokslininkai. Daug programos teksto yra susiję su kompiuterio valdymu tam tikrame kontekste, o kitame kontekste tas tekstas (kodas) negali būti naudojamas. Programa rašoma atsižvelgiant į kompiuterio savybes. Galima būtų valdyti kompiuterį tiesiogiai, tačiau tai labai sudėtinga ir didina klaidų tikimybę. Apskritai programuotojai dirba su aukštesnio lygio programos modeliu, kuris aprašomas programavimo kalba.

Programa rašoma kokia nors programavimo kalba. Programavimo kalbos gramatika vadinama sintakse. Semantika apibrėžia kalbos sakinių (komandų) prasmę. Reikšmė gali būti susijusi su skaičiavimais arba kompiuterio veikimu. Programa yra sakinių, kurie vadinami teiginiais, sąrašas. Teiginiai vykdomi po vieną tam tikra tvarka. Teiginį sudaro komanda ir reikšmės, kurių reikia komandai įvykdyti. Reikšmės gali būti apibrėžtos tiesiogiai, bet paprastai kaip pakaitalas naudojamas koks nors raidinis simbolis, kaip mokyklinėje algebroje. Simbolis vadinamas kintamuoju, jei jį galima keisti, ir konstanta, jei kartą nustatytas jis yra pastovus. Kartais į teiginį įtraukiamos matematiniai ar loginiai reiškiniai, kurie apskaičiuoja naujas reikšmes priklausomai nuo kintamųjų ir konstantų reikšmių. Komanda nuo reiškinio skiriasi tuo, kad komanda valdo kompiuterio veikimą, o reiškinys sukuria reikšmes. Reiškinio rezultatas yra įvesties duomenys komandai. Reiškinius sudaro operacijų ženklai (pavyzdžiui, sudėties ar atimties matematikoje ir AND ar OR logikoje) ir jų veikiamos kintamųjų bei konstantų reikšmės. Programos vykdymas baigiasi ties paskutiniuoju teiginiu.

Programos rašomos po vieną sakinį (komandą). Programa vykdo kiekvieną sakinį, kol pasiekia pabaigą. Valdymo sakiniai keičia vykdymo eigą. Sąlyginiai sakiniai (pasirinkimo komandos) leidžia pasirinkti alternatyvų vykdymo kelią. Gali būti, kad įvykdžius alternatyvias komandas vykdymas grįžta į pagrindinį kelią. Ciklo teiginys – tai konstrukcija, kuri kartoja komandų rinkinį tol, kol įvykdoma sąlyga. Pasibaigus ciklo vykdymui, programos vykdymas tęsiamas toliau, jei dar liko sakinių (komandų). Programos gali būti gana didelės, todėl nelengva suprasti jų tekstą. Kartais vienos programos dalies funkcionalumas praverstų kitoje. Funkcijos ir procedūros užtikrina moduliaciją ir pakartotinį naudojimą. Jos yra paprogramės, kurias galima pavadinti ir iškviesti kaip esamas kalbos komandas. Funkcija nuo procedūros skiriasi tuo, kad pirmoji pateikia reikšmę (rezultatą), o antroji keičia kompiuterio būseną. „Python“ ir „Racket“ yra tekstinių programavimo kalbų pavyzdžiai. Taip pat yra vaizdinių programavimo kalbų, skirtų pradedantiesiems, kuriose programavimas primena dėlionės konstravimą.

Dėlionėmis grindžiamas vizualinio programavimo metodas dar vadinamas blokais grindžiamu programavimu (Weintrop, 2019). Šio tipo programavimo kalbos komandos pateikiamos kaip vizualiniai blokai. Blokų forma suteikia užuominų, kaip ir kur juos galima naudoti. Šios rūšies

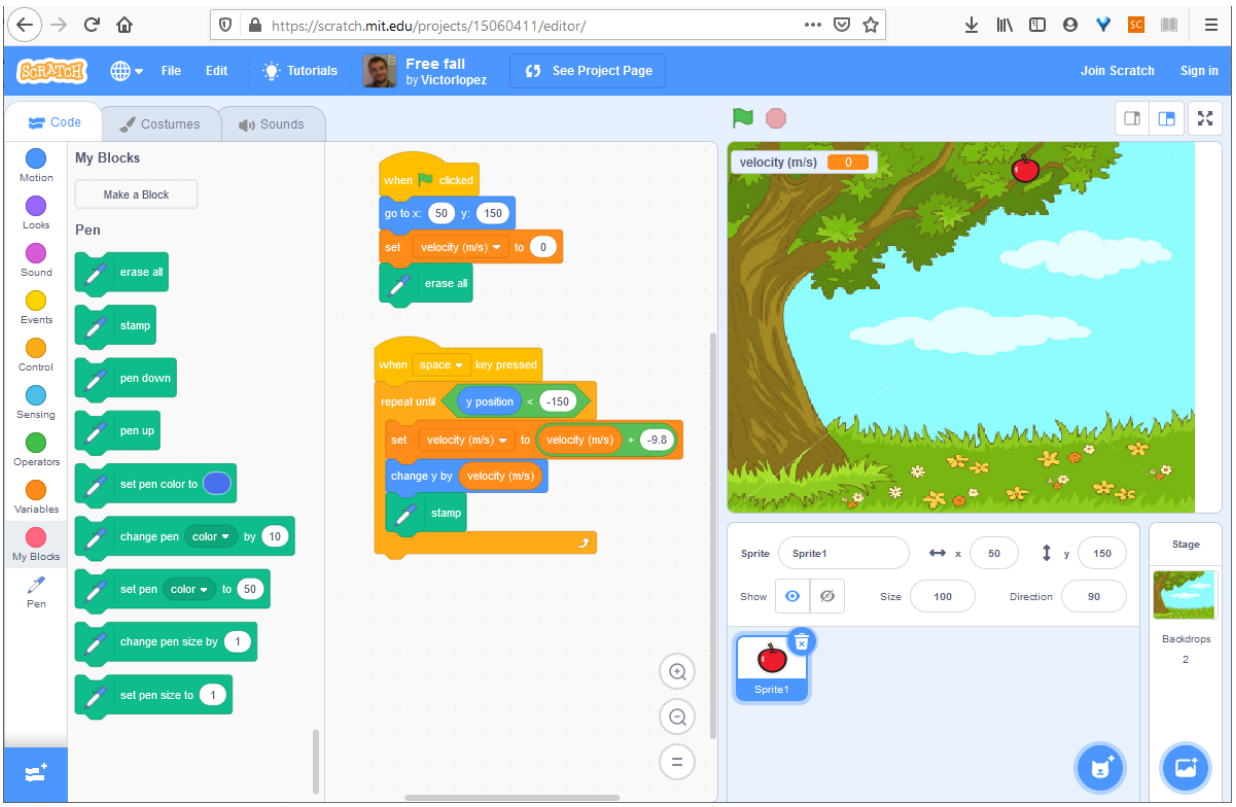
programavimo kalbų tikslinė grupė gali būti vaikai nuo penkerių metų, tačiau dažniausiai šios kalbos skirtos aštuonerių–šešiolikos metų vaikams. Parašyti programą – tai sujungti programos blokus juos velkant. Paprastai programavimo aplinka neleidžia sujungti tarpusavyje nederančių blokų. Tai leidžia rašyti programos komandą po komandos, kaip tekstinėse programavimo kalbose, kartu užkertant kelią sintaksės klaidoms. Kadangi programavimas atliekamas vilkimo būdu, programavimo aplinka gali suteikti papildomą pagalbą grupuojant blokus pagal funkcijas ir naudojant spalvas, kad būtų lengviau juos atpažinti. Nebereikia atsiminti, kas galima, vietoj to užtenka peržiūrėti komandų (blokų) sąrašą.

Šalutinis blokais pagrįsto programavimo poveikis yra tas, kad nereikia rašyti. Vartotojui nereikia jaudintis dėl rašybos klaidų, keistų skyrybos ženklų ar trūkstamų sintaksės dalių. Grafinis teiginių vaizdavimas leidžia naudoti ilgesnius aprašymus, nes komandų nereikia rašyti. Tai taip pat suteikia galimybę aprašyti prasmę, nes kompiuteriui reikalingą tikslumą galima paslėpti nuo naudotojo. Be blokais pagrįstos kalbos, į programavimo aplinką galima įtraukti ir kitų pagalbinių priemonių. „Scratch“ (Maloney et. al., 2010), kuri šiuo metu (2020 m.) yra žinomiausia blokų pagrindu sukurta programavimo kalba, ypač palengvintas grafinių ekrano elementų, vadinamų veikėjais (angl. *sprite*), valdymas. Kiekvienas veikėjas turi savo programą, kuri valdo jo elgesį. Viena iš svarbiausių veikėjo savybių yra jo vieta. Dėl to lengva padaryti, kad ekrane kas nors įvyktų. Kadangi veikėjai yra nepriklausomi, kaip ir kitų kalbų funkcijos ar procedūros, juos galima lengvai perkelti iš vienos programos į kitą.

Atvejo analizė: gravitacijos modeliavimas programa „Scratch“

Lopez‘as ir Hernandez‘as (2015) sukūrė fizikos projekto „Laisvasis kritimas“ pavyzdį, kuris galėtų būti užduotis pradinių arba vidurinių mokyklų mokiniams. Projektas buvo įgyvendintas „Scratch“ programa ir jame buvo modeliuojama, kaip gravitacija sukelia pagreitį, kai daiktas laisvai krenta. Fizikinis reiškinys buvo vizualizuotas medžio ir nuo vienos jo šakos krintančio obuolio paveikslėliu. Krintančio obuolio greitis rodomas kaip reikšmė, o pagreitis animuojamas spausdinant obuolio padėtį didėjančiais žingsniais. Norėdamas atlikti užduotį, mokinys turi išsiaiškinti, kas yra gravitacija ir kokia formulė apibūdina jos poveikį krintančiam daiktui. Naudojant formulę būtų paprasta apskaičiuoti greitį atstumo nuo įsivaizduojamos šakos iki žemės pabaigoje. Uždavinys – pasinaudoti „Scratch“ programos savybėmis ir parodyti greičio didėjimą dėl pagreičio.

Galima būtų manyti, kad pradinių klasių mokiniams bus rodomas programos rezultatas, o vidurinių klasių mokiniams dalis užduoties būtų patiems sugalvoti vizualizavimo principą. Lopez‘o ir Hernandez‘o modelio įgyvendinimą galima būtų pradėti nuo to, kad pradžioje reikia pridėti sceną su medžiu, nupiešti obuolį vaizduojantį veikėją ir tada bandyti perkelti obuolį į reikiamą padėtį (žr. 2.6 pav. dešinėje pusėje). Tuomet reikia sugalvoti, kaip sujungti pagreičio modelį su obuolio padėties pokyčiu. Padėties pokytis laiko atžvilgiu (sekundėmis) būtų greitis, o pagreitis – tai naujos obuolio padėties spausdinimas vienodais laiko intervalais. Pradinis obuolio greitis būtų lygus nuliui (žr. 2.6 paveikslo vidury pirmąjį blokų rinkinį).



2.6 pav. „Scratch“ aplinkos ekrano kopija, kurioje pavaizduoti grafiniai obuolio medyje elementai ir susijusi programa, skirta vizualizuoti obuolio kritimą dėl gravitacijos.

Modelio programoje sekundė vaizduojama kaip vienas ciklo ratas (*repeat until* – kartoti, kol tenkinama sąlyga; žr. 2.6 paveikslo vidury antrąjį blokų rinkinį). Dabar greitis yra obuolio padėties pokytis per vieną ciklo ratą. Pirmiausia atsižvelgiama į pagreičio poveikį prie greičio pridedant 9,8 (*set velocity (m/s)...*) ir pakeičiama vertikali obuolio padėtis (*change y by velocity (m/s)*; žr. 2.6 pav.) Antspaudo (*stamp*) komanda naudojama kiekviename ciklo rate obuoliui spausdinti (žr. 2.6 pav.). Ciklas baigiamas, kai obuolio padėtis *y* yra ties apatiniu scenos kraštu. Sukūrus modelį, galima jį tobulinti, kad būtų tikroviškesnis. Mokiniai galėtų pridėti oro trintį arba sumodeliuoti vėjo poveikį.

Daugiau informacijos apie šį atvejo tyrimą:
Scratch as a computational modelling tool for teaching physics (Lopez & Hernandez, 2015).

Programa „Scratch“ kalba: https://scratch.mit.edu/projects/15060411/ (žiūrėta 2020-08-30)

Santrauka

Nagrinėtame pavyzdyje naudojami informatinio mąstymo modelių atpažinimo, abstrakcijos, dekompozicijos ir algoritmų kaip kompiuterinių programų komponentai. Tai įgūdžiai, kuriuos mokiniai turi turėti, kad žinotų, kaip panaudoti kompiuterijos galimybes STEM srityje. Mokiniai turi būti supažindinami su skirtingais informatinio mąstymo komponentais: iš pradžių atskirai, o paskui sujungti į visumą. Svarbu, kad programavimas nebūtų pristatomas kaip naudojimasis sudėtingu skaičiuotuvu, bet būtų aiškinamos realios kompiuterių galimybės. Studijuodami šį pavyzdį mokiniai susipažino su gravitacijos tema, sukūrė kompiuterinį modelį

ir jam vizualizuoti naudojo „Scratch“ aplinkos savybes. Atsižvelgiant į mokinių brandą, galima pateikti teorinius informatinio mąstymo paaiškinimus, tačiau be praktikos informatinio mąstymo neišmokstama. Be to, mokytojas turi pats praktiškai gebėti programuoti, kad galėtų perteikti įžvalgas, reikalingas programai sukurti.

Paskaita. Informatinio mąstymo perspektyvos STEAM sistemai

- Trys informatikos perspektyvos
- Šablonų atpažinimas
- Abstrakcija
- Dekompozicija
- Kompiuterio vykdomas algoritmas

Namų darbas. Savo temos aprašymas naudojant perskaitytą modulio tekstą apie informatinio mąstymo perspektyvas

Šios užduoties tikslas - praktiškai pritaikyti informatinį mąstymą savo dalyko temai. Gravitacijos modeliavimas naudojant „Scratch“ programą yra pavyzdys, kai gravitacija fizikoje modeliuojama ir vizualizuojama kompiuteriniu būdu. Turite nustatyti, kaip informatinis mąstymas galėtų padėti išspręsti uždavinį, sukurti technologiją ar suprasti reiškinį.

1. Pasirinkite įdomią kurio nors dalyko temą.
2. Kuri informatinio mąstymo perspektyva susijusi su Jūsų tema?
   - Matematika: Kokius uždavinius galima išspręsti skaičiavimais?
   - Inžinerija: Kaip galima spręsti uždavinius, susijusius su skaičiavimais? Kokie uždaviniai gali būti sprendžiami taikant informacines technologijas?
   - Gamtos mokslai: Kokie reiškiniai gali būti modeliuojami naudojant skaičiavimo procesus?
3. Pritaikykite PRADA modelį savo temai pagal pasirinktą perspektyvą.
   - Šablonų atpažinimas
   - Abstrakcija
   - Dekompozicija
   - Kompiuterio vykdomas algoritmas
4. Kokią užduotį būtų galima sukurti iš informatinio mąstymo temos?
   - Nurodykite mokinių amžių arba klasę.
   - Apibūdinkite užduoties idėją (įgyvendinimo detalių galima nepateikti).

Namų darbas. Tarpusavio vertinimas

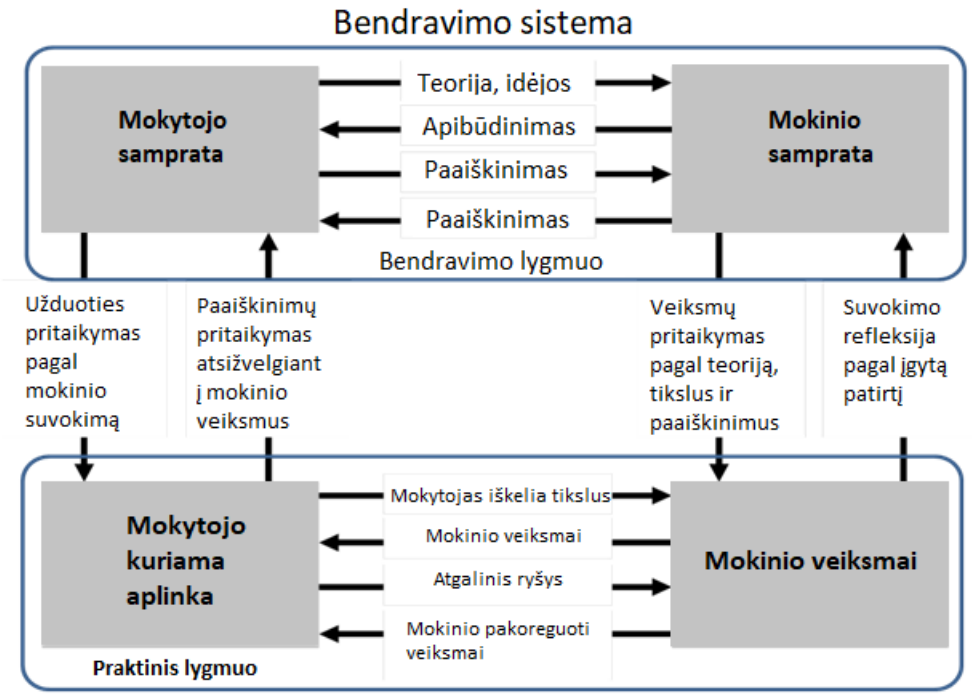
- Peržiūrėkite trijų bendramokslių 2.1 namų darbo atlikimą
- Vertinimas: kiekvienas punktas gali būti vertinamas pagal 0–5 balų skalę kreipiant dėmesį į
  a) temos pristatymą,

b) argumentaciją ir aiškumą.
c) skyriaus turinio panaudojimą.

3 skyrius. Švietimo technologijų pasirinkimas remiantis informatinio mąstymo perspektyvomis

Šiuolaikiniai mokiniai laisvai naudojasi skaitmeninėmis technologijomis, tačiau tai nebūtinai reiškia, kad jie supranta jų principus. Kad vėliau galėtų laisvai naudotis skaitmeninėmis technologijomis, jie pirmiausia turi pradėti jas naudoti labiau organizuotoje aplinkoje. STEAM ugdymas – tai skirtingų sričių žinių ir įgūdžių derinimas siekiant išspręsti gautą uždavinį. Mokiniams naudinga patirtis, įgyta naudojant ir bendrąsias, ir konkrečiam dalykui skirtas priemones. Abiem atvejais mokytojas gali naudotis informatinio mąstymo perspektyvų sistema, kad atpažintų ugdymo technologijas, kurios sujungia konkrečios temos modelių tvarkymą su kompiuterinėmis galimybėmis. Šiame skyriuje bendravimo sistema yra įdėta į mokymo modelį, padedantį pasirinkti priemonę, atitinkančią numatytą pedagogiką. Pateikiami trijų perspektyvų edukacinių technologijų pavyzdžiai. Galiausiai mokytojas gali nuspręsti, kiek technologinės galimybės gali veikti mokymą.

Bendravimo sistema (angl. *conversational framework*) (žr. 3.1 pav.) apibūdina mokymąsi kaip pasikartojantį mokytojo ir mokinių dialogą (Laurillard, 2002; Laurillard, 2012). Skiriami du lygmenys: bendravimo lygmuo, kuriame keičiamasi teorijomis ir sąvokomis, ir praktinis lygmuo, kuriame veiksmais įgyjama patirtis. Šiuos lygmenis sieja mokytojo ir mokinių įsitraukimas į teorijų pritaikymą praktikoje ir teorijų apmąstymas remiantis įgyta patirtimi. Diskurso metu mokytojas pateikia teorijas ir idėjas, o mokiniai atsako konceptualizuodami tai, ką išgirdo. Jei mokiniai neteisingai supranta, mokytojas gali pakeisti pateikiamą informaciją, o mokiniai gali reaguoti pateikdami naujas interpretacijas. Daugelyje temų yra ir praktinis aspektas. Mokytojas sukuria aplinką, kurioje mokiniai, atlikdami užduotis, gali praktiškai pritaikyti teoriją. Atsižvelgiant į mokinių atsakymus diskurso metu, užduotys gali būti pritaikytos atitinkamam žinių lygiui. Mokytojas nustato tikslus, kokias užduotis mokiniai turėtų atlikti. Mokiniai pakeičia savo veiksmus atsižvelgdami į teorinį supratimą ir pateiktus tikslus. Remdamiesi grįžtamuoju ryšiu, gautu atliekant užduotis, mokiniai apmąsto savo supratimą ir pritaiko savo veiksmus, kad galėtų bandyti dar kartą. Remdamasis mokinių atliktais veiksmais, mokytojas kitam kartui pakeičia mokymo metodą.



3.1 pav. Bendravimo sistema apibūdina mokytojo ir mokinio sąveiką tiek koncepciniu lygmeniu, tiek per mokytojo siūlomas užduotis. Abu pritaiko savo veiksmus, remdamiesi bendravimu ir savo patirtimi. Mokytojas sukuria aplinką, kurioje mokinys gali praktiškai pritaikyti savo žinias. Mokinys, remdamasis grįžtamuoju ryšiu, gali tikslinti savo veiksmus, kurie taip pat turi įtakos jo temos sampratai.

Ši bendravimo sistema yra modelis ir pateikia supaprastintą mokymo vaizdą, tačiau ji yra pakankamai išsami, kad apimtų dažniausiai pasitaikančius mokymosi būdus (Laurillard, 2012). Ji apima mokymąsi per žinių įgijimą, tyrimą, praktiką, gamybą, diskusijas ir bendradarbiavimą. Sistemos tikslas – padėti mokytojui kurti mokymosi aplinką. Projektuojant galima naudoti įvairias technologijas. Šiame skyriuje mus ypač domina technologijos, padedančios įgyvendinti sistemos praktinį ir modeliavimo aspektą (žr. 3.1 pav. apatinę dalį). Edukacinės technologijos padeda mokytojui kurti užduotis mokiniams, kurios leistų jiems praktiškai išbandyti išmoktas sąvokas. Praktikos dimensija remiasi mokytojo grįžtamuoju ryšiu apie mokinių pasiekimus. Technologijos taip pat gali būti atsakymų, grįžtamojo ryšio ir vertinimo platforma. Modeliavimo dimensijoje mokytojas sukuria technologinę aplinką, kuri suteikia grįžtamąjį ryšį. Kalbant apie informatinį mąstymą, puikus modeliavimo aspekto pavyzdys yra vėlyvasis Seymouro Paperto darbas apie konstruktyvizmą (Harel ir Papert, 1991) ir mikropasaulius (Papert, 1980; 1996).

Žvelgiant iš informatikos perspektyvos, skirtingi STEM dalykai atsako į skirtingus klausimus. Matematika pasako, kokias problemas galima išspręsti skaičiavimais. Kompiuteriai buvo sukurti skaičiavimams, todėl, nors tikslas nėra matematinis, uždavinys turi būti išreikštas matematiškai tiksliai. Inžinerija ir technologijos yra susijusios su elektroninių prietaisų valdymu. Sudėtingesniam valdymui įrenginyje turi būti įmontuota elektronikos arba kompiuterinės technologijos. Su kompiuteriais nesusiję reiškiniai taip pat gali būti modeliuojami kaip skaičiavimo procesai. Visi reiškiniai, kurių elgesį galima aprašyti matematiškai, gali būti apskaičiuoti kompiuteriu. Reiškinius, kurių negalima aprašyti tiksliomis formulėmis, galima aprašyti naudojant statistiką. Statistika gali apdoroti svyravimus,

neapibrėžtumą ir didelius duomenų kiekius. Perspektyvos atsako į skirtingus klausimus, nors skaičiavimo mechanizmas yra tas pats.

Informatinio mąstymo perspektyvų sistemą sudaro šablonų atpažinimas, abstrahavimas, dekompozicija ir algoritmai. Pasirinkta technologija turi gebėti atvaizduoti uždavinį įrėminantį modelį. Paprastai modelis atspindi vieną iš STEM perspektyvų. Galimybė naudoti sąvokas ir ryšius, kurie įvardijami pagal modelio žodyną, palengvina mokiniams mąstyti apie pokytį, kuris išspręstų uždavinį. Pokytis nuo įvesties prie išvesties, kaip uždavinio sprendimas, yra aukščiausio abstrakcijos lygmens (uždvinio lygmens) aprašymas. Objekto lygmeniu mokiniai apibrėžia algoritmą, kuris veikia remdamasis modelio sąvokomis ir ryšiais. Programos lygmuo gali būti artimesnis algoritmui (objekto lygmuo) arba kompiuteriui (vykdymo lygmuo). Mokiniams vertinga naudoti programavimo kalbą ar kitokias informatikos priemones, leidžiančias tiesiogiai dekomponuoti uždavinį, neatliekant transformacijų tarp objekto ir programos lygmens. Mokomoji technologija turėtų teikti mokiniams grįžtamąjį ryšį, kad jie galėtų įvertinti rezultatus ir prireikus atlikti taisymus.

Bendravimo sistema skirta padėti mokytojui kurti mokymosi aplinką. Naujos mokymosi aplinkos patirčiai skleisti galima naudoti pedagoginius šablonus (Laurillard, 2012). Pedagoginis šablonas – tai būdas suformuluoti, išbandyti ir dalytis mokymo, naudojant skaitmenines technologijas, principais ir praktika (Laurillard ir McAndrew, 2003). Šablonas gali būti naudojamas mokymo ir mokymosi veiklų sekai apibūdinti. Be to, pedagoginiame šablone pateikiama informacija, apibūdinanti pačio šablono kontekstą (žr. 3.1 lentelę), kilmę, santrauką, temas, mokymosi rezultatus, pagrindimą, trukmę, besimokančiųjų charakteristikas, aplinką ir grupės dydį. Pedagoginis šablonas čia naudojamas kaip pagalbinė priemonė STEAM mokymo technologijoms aprašyti. Vėliau studentai, būdami mokytojais, taip pat gali naudotis šablonais dalydamiesi savo mokymo planais, kuriuose naudojamos edukacinės technologijos.

3.1 lentelė. Pedagoginio Šablono konteksto aprašai (Laurillard, 2012).

Kategorija	**Aprašymas**
Kilmė	pirminis šaltinis ir vėlesni autoriai
Santrauka	trumpas aprašymas, ko ir kaip mokoma
Temos	raktiniai žodžiai, kurie padės kitiems mokytojams nuspręsti, ar jiems tai aktualu
Mokymosi rezultatai	ką besimokantysis žinos arba gebės padaryti mokymo pabaigoje
Pagrindimas	mokymosi metodas arba pedagoginio dizaino principas
Trukmė	bendras mokymosi valandų skaičius, nebūtinai nepertraukiamas
Besimokančiojo savybės	išsilavinimo reikalavimai, patirtis, interesai
Mokymosi būdas	akis į akį, mišriai arba internetu
Grupės dydis	besimokančiųjų skaičiaus intervalas nuo mažiausio iki didžiausio

STE(A)M švietimo technologijų pavyzdžiai

Šiame skyrelyje pateikiami STEM temoms skirtų švietimo technologijų pavyzdžiai. Kiekvienas pavyzdys pateikiamas kaip pedagoginis šablonas, apibūdinantis pedagoginį projektą, kuriame technologija yra mokymo ir mokymosi ciklo dalis. Technologijos derinys ir jos naudojimo aprašymas sudaro visumą. Galima sugalvoti naują technologijos naudojimo būdą, tačiau tai taip pat būtų naujas šablonas. Be to, pateikiamas bendras aprašymas, kuris nusako pagrindines šablono charakteristikas ir padeda skaitytojui nuspręsti, ar verta į šabloną pažvelgti atidžiau. Toliau pateikiami matematikos, technologijų bei inžinerijos ir gamtos mokslų modulio temų šablonų pavyzdžiai.

C2STEM (gamtos mokslai)

Kategorija	**Aprašymas**
Kilmė	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
Santrauka	Fizikos mokymui naudojama modeliavimo ir simuliavimo aplinka C2STEM. Tema – kinematika, įskaitant padėties, greičio ir pagreičio sąvokas. Sukurti modeliai yra kompiuteriniai, o tai padeda mokytis ir fizikos, ir informatinio mąstymo praktikos. Modeliavimo aplinka padeda mokytis naudojant palengvintas užduotis ir integruotus formuojamuosius vertinimus.
Temos	STEM+CT, sinergetinis mokymasis, mokymasis modeliuojant, informatinis mąstymas, įrodymais grįstas dizainas, atvira mokymosi aplinka.
Mokymosi rezultatai	Mokinys žinos sąvokas ir supras fizikos praktinę dalį. Mokantis modeliuoti naudojant kompiuterinę aplinką taip pat mokomasi informatinio mąstymo, pavyzdžiui, skaidymo, tobulinimo ir klaidų ieškojimo.
Pagrindimas	mokymasis modeliuojant.
Trukmė	-
Besimokančiojo savybės	vidurinės mokyklos mokiniai
Mokymosi būdas	mišrus
Grupės dydis	1

C2STEM (Collaborative, Computational STEM) – tai fizikos mokymosi aplinka. Ji apjungia vizualinių modelių kūrimą ir konkrečiai sričiai būdingą modeliavimo kalbą. Mokiniai mokosi modeliuodami objektų judėjimą. Mokytis padeda palengvintos užduotys ir integruotas formuojamasis vertinimas. Aplinkoje vietoj lygčių naudojami intuityvūs skaičiavimo pateikimai, remiamasi pagrindinėmis mokslinėmis praktikomis (modeliavimu, tikrinimu ir aiškinimu) ir sudaromos sąlygos mokytis programavimo kontekstualiai. Aplinka yra laisvai prieinama internete (https://www.c2stem.org/), ji taip pat naudojama ir kitiems STEM dalykams, pavyzdžiui, jūrinei biologijai ir žemės mokslams (geografijai, geologijai ...).

C2STEM remiasi „NetsBlox“ – vizualine debesijos programavimo aplinka, leidžiančia kurti tinklines programas. „NetsBlox“ remiasi „Snap!“, kuri yra „Scratch“ išvestinė programa. Šį perimamumą geriausiai atskleidžia vaizdinė naršykle pagrįsta programų rengyklė su velkamais blokais, scena, kurioje kaupiami vaizdiniai elementai, ir lengvu veikėjų valdymu (žr. 3.2 pav.). Tinklinės savybės išryškėja leidžiant programoms naudotis saityno paslaugomis, kad jas vienu metu galėtų išbandyti keli naudotojai ir kad jos būtų saugomos centralizuotai. Skirtingi naudotojai gali turėti skirtingą požiūrį į bendrą projektą. Mokytojas taip pat gali paslėpti programos tekstą (kodą), kad programavimo aplinkai pridėtų papildomų savybių, mokiniui nežinant, kaip tai įgyvendinta. Tai leidžia pateikti programavimo užduotis su tam tikromis jau numatytomis funkcijomis, tačiau nedarant įtakos mokinių kuriamos programos tekstui.

„SmileyCluster“ (gamtos mokslai)

Kategorija	**Aprašymas**
Kilmė	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
Santrauka	Kadangi vis dažniau naudojamas dirbtinis intelektas, besimokantys jaunuoliai turi suprasti šios technologijos prigimtį, kad galėtų įvertinti jos reikšmę asmeniniam ir profesiniam gyvenimui. Šiuo metu mašininis mokymasis (angl. *machine learning*) yra viena iš pagrindinių dirbtinio intelekto sričių, tačiau dėl savo sudėtingo matematinio ir informatinio pobūdžio jis nėra prieinamas jauniems mokiniams. „SmileyCluster“ – tai aplinka, kurioje mašininio mokymosi gebėjimas atpažinti modelius ir klasifikuoti yra vizualizuojamas taip, kad besimokantysis galėtų pasinaudoti savo paties gebėjimais ir suprasti technologijos funkcionalumą.
Temos	duomenų vizualizavimas; praktinis mokymasis; dirbtinio intelekto raštingumas; moksliniai atradimai; STEM švietimas
Mokymosi rezultatai	apskritai suprasti šablonų atpažinimą ir kaip mašininio mokymosi sąvokos ir metodai susiję su šablonų atpažinimu
Pagrindimas	konstruktyvizmas, tyrinėjimu grįstas mokymasis, bendradarbiavimas
Trukmė	-
Besimokančiojo savybės	vidurinė mokykla
Mokymosi būdas	gyvai
Grupės dydis	mokinių poros

Mokomoji robotika ir kūrybiškas kompiuterinių problemų sprendimas (technologijos ir inžinerija)

Kategorija	Aprašymas
Kilmė	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
Santrauka	Edukacinė robotika vis dažniau naudojama klasėse mokant informatinio mąstymo. Kūrybiškas kompiuterinių uždavinių sprendimas – tai modelis, padedantis mokytojams kurti, įgyvendinti ir vertinti robotikos veiklas. Modelis padeda mokytojui planuoti mokymo intervenciją, kuri supažindina su atitinkamomis informatinio mąstymo sąvokomis įvairiuose robotikos veiklų etapuose. Jis suteikia mokiniui labiau subalansuotą požiūrį į uždavinių sprendimą, o ne tik sutelkia dėmesį į programavimą.
Temos	informatinis mąstymas, mokomoji robotika, mokomoji intervencija, uždavinių sprendimas, bandymų ir klaidų metodas.
Mokymosi rezultatai	Mokiniai gebės valdyti robotą ir spręsti su tuo susijusias uždavinius. Jie išmoks taikyti bandymų ir klaidų metodą, analizuoti uždavinius, generuoti idėjas ir formuluoti sprendimus informatinio mąstymo kontekste.
Pagrindimas	tyrinėjimais grįstas mokymasis
Trukmė	-
Besimokančiojo savybės	pradinių klasių mokiniai (9–10 metų)
Mokymosi būdas	gyvai
Grupės dydis	2–3 mokiniai

„Lattice Land“ – geometriją tyrinėjantys mikropasauliai (matematika)

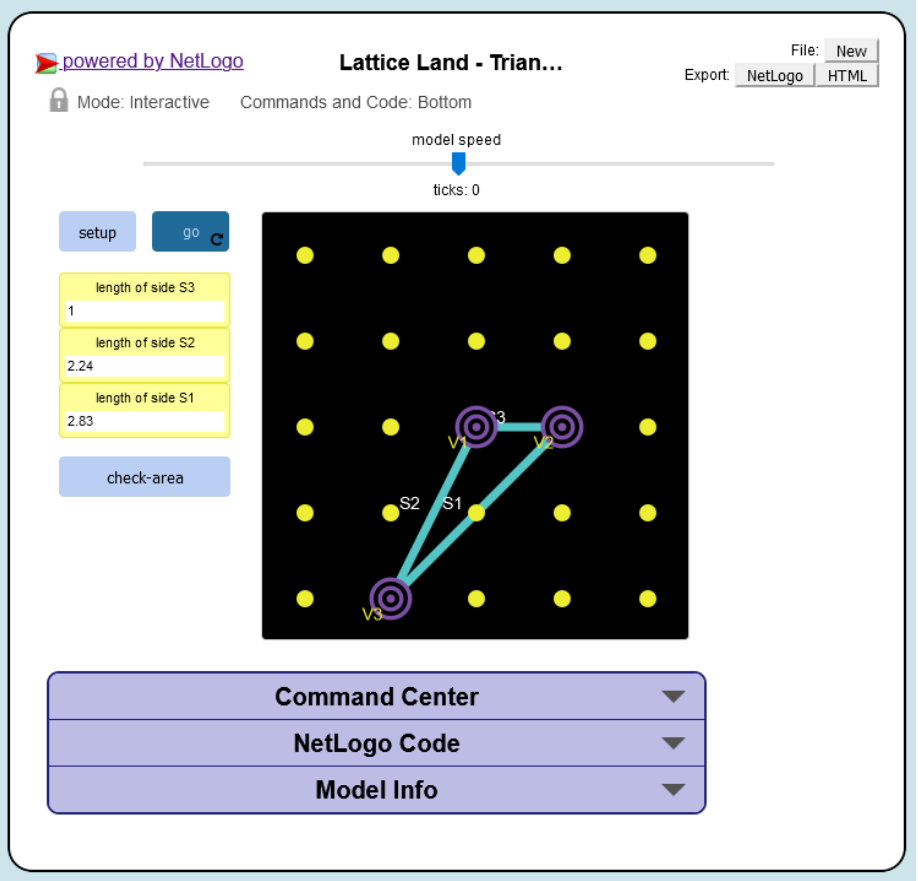
Kategorija	Aprašymas
Kilmė	Pei, C., Weintrop, D., & Wilensky, U. (2018). Cultivating computational thinking practices and mathematical habits of mind in lattice land. Mathematical Thinking and Learning, 20(1), 75-89, https://doi.org/10.1080/10986065.2018.1403543.
Santrauka	Mokomasi vidurinės mokyklos geometrijos, naudojant matematinį mikropasaulį, vadinamą „Tinklelių šalimi“. Tikslas – lavinti matematinio mąstymo ir informatinio mąstymo įpročius. Mokinys mokosi meistrauti, eksperimentuoti, atpažinti modelius ir pateikti hipotezes formalia matematine notacija.
Temos	tinklelio geometrija, kompiuterinė mokymosi aplinka, matematinis mąstymas, informatinis mąstymas

Mokymosi rezultatai	Mokiniai mokosi atpažinti ir pagrįsti sudėtingus geometrinius modelius. Be to, jie mokosi skaičiavimo metodų, naudojamų kaip mokymosi dalis.
Pagrindimas	konstruktyvistinis; mokymasis veikiant
Trukmė	-
Besimokančiojo savybės	vidurinės mokyklos moksleiviai
Mokymosi būdas	internete
Grupės dydis	asmeninis

„Lattice Land“ – tai matematinių mikropasaulių rinkinys, paremtas „NetLogo“ (Wilensky, 1999) aplinka. Jis skirtas įvadui į tinklelių geometriją, nes siūlo struktūrą ir leidžia laisvai tyrinėti. Aplinka gali būti naudojama laisvam eksperimentavimui, tačiau ji taip pat tinka ir tyrimais grindžiamam mokymuisi, jei tyrinėjant vadovaujamasi užduotimi ar klausimu.

Mikropasauliai sudaryti iš taškų matricos suformuoto tinklelio. Kiekvienas taškas turi koordinatę (x, y), čia x ir y yra sveikieji skaičiai. Tinklelyje galima braižyti daugiakampius, sujungiant taškus atkarpomis. Sujungti taškai sudaro daugiakampio viršūnes. Besimokantysis gali sąveikauti su mikropasauliu brėždamas segmentus tarp taškų, kad sudarytų daugiakampius, ir keisti daugiakampius perkeldamas jų viršūnes arba pridėdamas papildomų segmentų. Kadangi pridėti segmentus ir perkelti viršūnes galima tik tarp taškų, tinklelis suteikia tyrinėjimui struktūrą. Skirtinguose mikropasauliuose galima įvesti papildomų funkcijų arba apribojimų. Pavyzdžiui, mikropasaulis gali pateikti tokią informaciją kaip segmento ilgis ir daugiakampio plotas. Apribojimas galėtų būti, kai leidžiama manipuliuoti tik pateikto daugiakampio viršūnėmis.

„Lattice Land“ mikropasaulį galima tyrinėti „NetLogo“ saityne arba aplinką ir mikropasaulį galima atsisiųsti (https://netlogoweb.org). Straipsnyje buvo aprašyti trys mikropasauliai ir gretimi klausimai, kuriuose buvo nagrinėjama geometrinė ploto sąvoka. Vienas iš pavyzdžių buvo mikropasaulis „Lattice Triangles Explore“ (žr. 3.2 pav.), kuriame buvo keturi su keturiais dydžio tinklelis ir vienas trikampis. Klausimas buvo, kiek skirtingų dydžių plotų, kuriuos sudaro trikampis, galima rasti šiame tinklelyje. Mokinys turėtų ištirti galimybes perkeldamas trikampio viršūnes, kad sudarytų skirtingus trikampius. Teisingas atsakymas yra šešiolika, tačiau mokiniai gali atrasti ir kitų faktų. Mokinius reikėtų paskatinti nepamiršti tokių atradimų. Viena iš tokių išvadų galėtų būti ta, kad, vieną viršūnę perkėlus pirmyn ir atgal išilgai tos pačios eilutės ar stulpelio, plotas nesikeičia. Kitas – kad jei viena iš trikampio kraštinių yra lygiagreti tinklelio x arba y ašiai, tuomet neįmanoma gauti trikampio, kurio plotas būtų 2,5.



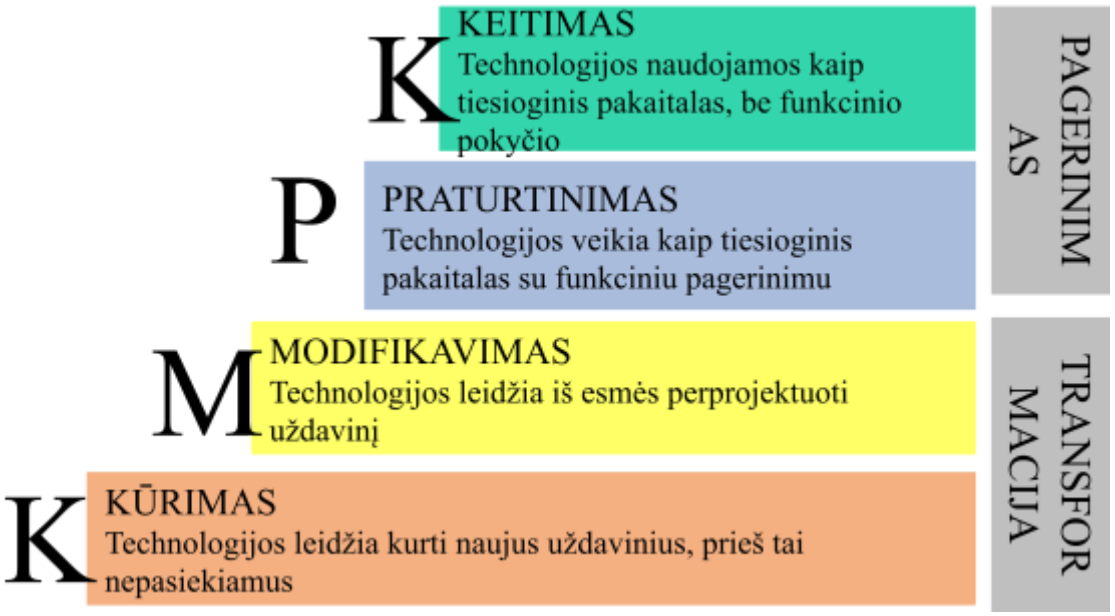
3.2 pav. „Lattice Triangles Explore“ mikropasaulis: tyrinėkite ir sužinokite, kiek skirtingų dydžių sričių, kurias sudaro trikampis, galima rasti šiame tinklelyje.

„Lattice Triangles Explore“ mikropasaulyje mokiniai gali tyrinėti ir geriau suprasti trikampio sąvoką. Ypač išryškėja ryšys tarp formos, kraštinių ilgio ir ploto. Tai modeliavimo ir simuliavimo praktika, susijusi su informatiniu mąstymu (Weintrop ir kt., 2016). „Lattice Land“ naudojamas interaktyvumas ir pagalba, kad būtų suteikta tyrinėjimo laisvė ir gauti prasmingi rezultatai. Tai nebūtų įmanoma ar prieinama kitoje terpėje.

Taksonomija technologijų naudojimo lygiui parinkti

Remiamasi 1-o modulio 5-u skyriumi: projektinis mokymasis (Valentina Dagienė)

Švietimo technologijų galimybes galima panaudoti didesniu ar mažesniu mastu. Pradedant naudoti naują technologiją, pirmiausia ją galima naudoti kaip esamos technologijos pakaitalą. Vėliau, kai įgyjama patirties, technologines galimybes galima išnaudoti plačiau. Keitimo, praturtinimo, modifikavimo ir kūrimo (angl. *Substitution, Augmentation, Modification and Redefinition*, SAMR) modelyje (Puentedura, 2006) pateikiama keturių lygių technologijų pasirinkimo, naudojimo ir vertinimo taksonomija. Šis modelis padeda mąstyti apie technologijų vaidmenį kaip pagalbą mokymuisi.



5.3 pav. KPMK (angl. SAMR) modelis gali padėti pedagogams galvoti apie technologijų vaidmenį mokymesi (Rubenas Puentedura, 2010)

Keitimas (*Substitution*)

Keitimas reiškia, kad tradicinė veikla ir medžiaga, pavyzdžiui, paskaitos klasėje ar popieriniai darbo lapai, pakeičiami skaitmeninėmis versijomis. Turinys iš esmės nesikeičia, tik keičiamas jo pateikimo būdas. Šiuo atveju siekiama, kad viskas būtų paprasta: nereikia išradinėti dviračio iš naujo. Nuskenuokite savo pamokas ir darbo lapus, paverskite juos PDF formato dokumentais ir paskelbkite juos internete naudodamiesi „Microsoft OneDrive“, „Google Drive“ ar panašia dalijimosi failais paslauga. Pagalvokite apie informaciją, kurią laikote ant sienų, pavyzdžiui, klasės normatyvus, dienotvarkę ar žodžių sąrašus, ir paverskite juos skaitmeniniais formatais, kuriais mokiniai galėtų lengvai naudotis. Taip pat gali būti naudinga pateikti tiek sinchronines, tiek asinchronines paskaitų versijas. Jei klasės susirinkimus organizuojate naudodamiesi vaizdo konferencijų paslauga, pavyzdžiui, „Zoom“ ar „Skype“, pateikite įrašą mokiniams, kurie negali dalyvauti. Taip pat galite sukurti savo mokomuosius vaizdo įrašus, kuriuos mokiniai galės peržiūrėti savo tempu.

Praturtinimas (*Augmentation*)

Šiame lygyje naudojami interaktyvūs skaitmeniniai patobulinimai ir elementai, pavyzdžiui, komentarai, hipersaitai ar daugialypė terpė. Turinys išlieka nepakitęs, tačiau mokiniai gali naudotis skaitmeninėmis funkcijomis, kad patobulintų pamoką.

Pavyzdžiui, mokiniai gali kurti skaitmeninius portfelius ir multimedijos pateiktis, taip suteikdami daugiau galimybių parodyti, kaip supranta temą. Užuot daliję popierines viktorinas, galite jas paversti žaidimais naudodami tokias priemones kaip „Socrative“ ir „Kahoot“.
Mokytojai taip pat gali sukurti virtualias skelbimų lentas, naudodamiesi tokiomis programėlėmis kaip „Padlet“, kuriose mokiniai gali skelbti klausimus, nuorodas ir nuotraukas.

Modifikavimas (*Modification*)

Šiame lygyje mokytojai gali galvoti apie mokymosi valdymo sistemą, pavyzdžiui, „Google Classroom“, „Moodle“, „Schoology“ arba „Canvas“, kad galėtų tvarkyti logistinius klasės valdymo aspektus, pavyzdžiui, stebėti pažymius, siųsti mokiniams žinutes, kurti kalendorių ir skelbti užduotis. Mokymas internetu atveria naujus bendravimo kanalus, kurių daugelis gali padėti mokiniams, turintiems problemų tradicinių pamokų metu. Tyrimai rodo, kad, pavyzdžiui, mergaitės mažiau linkusios kalbėti per pamokas, todėl joms gali būti naudingi alternatyvūs pokalbiai, kurie gali vykti kartu su mokymu ir kurie skatina dalyvauti.

Tuo tarpu „Zoom“ tekstinių pokalbių funkcija suteikia mokiniams galimybę parašyti savo klausimus, o tai gali būti mažiau įkyru, jei pokalbyje dalyvauja kelios dešimtys mokinių. Be to, mokiniams, kurie mieliau renkasi kaupti mintis, gali būti naudingos lėtesnio tempo asinchroninės diskusijos interneto forume ar el. paštu.

Kūrimas (*Redefinition*)

Mokymasis iš esmės keičiasi perėjus į kūrimo (pertvarkymo) lygmenį, kuomet galima vykdyti veiklą, kuri anksčiau buvo neįmanoma klasėje, pavyzdžiui, virtualūs susirašinėjimai su draugais gali sujungti mokinius su kitais pasaulio kraštais, nesvarbu, ar tai būtų kiti mokiniai, ar tam tikros srities ekspertai. Virtualios ekskursijos leidžia mokiniams aplankyti Amazonės atogrąžų miškus, Luvrą ar Egipto piramides. Klasėje perskaitę knygą, galite pakviesti autorių pasikalbėti apie jo kūrinį ir atsakyti į klausimus.

Technologijos taip pat suteikia galimybę į virtualią klasę pritraukti autentišką auditoriją ir iš mokinių padaryti leidėjus. Vaikai gali rašyti savo „vikius“ ar tinklaraščius, kad juos galėtų viešai naudoti ir gauti grįžtamąjį ryšį, o tokios platformos kaip „Quadblogging“ gali sujungti nutolusias klases, kad mokiniai ir rašytų, ir atsakytų. Mokiniai gali spręsti vietos problemas, pavyzdžiui, tirti netoliese esančios upės vandens kokybę, ir pakviesti bendruomenės narius įvertinti jų skaitmeninius pasiūlymus.

Paskaita – švietimo technologijos informatinio mąstymo perspektyvoms

- Informatinio mąstymo perspektyvų taikymas
- Švietimo technologijos, padedančios matematikai
- Technologijas ir inžineriją remiančios švietimo technologijos
- Švietimo technologijos, padedančios gamtos mokslams
- Keitimo, praturtinimo, modifikavimo ir kūrimo modelis

STEAM dalykams skirtų švietimo technologijų taikomųjų programų aprašų biblioteka (pagal Veerasamy et al., rengiama).

Namų darbas. Švietimo technologijų pagalbos savo temai planavimas

1. Pasirinkite savo dalyko temą ir suplanuokite užduotį naudodamiesi informatinio mąstymo perspektyvų STEAM sistema (daugiau informacijos 2-e skyriuje).
2. Pasirinkite tinkamą mokomąją technologiją, kuri padėtų mokiniams atlikti užduotį (naudokite paskaitoje aprašytas edukacines technologijas kaip pavyzdį, kai ieškote literatūros ar šaltinių internete).
3. Kuriame keitimo, praturtinimo, modifikavimo ir kūrimo modelio lygyje panaudotumėte edukacines technologijas (vien keitimo nepakanka)? Persvarstykite užduoties planą, jei atrodo, kad mokymo technologijos neduoda jokios naudos mokymuisi.
4. Aprašykite savo sukurtą užduoties mokymo pedagoginį planą.

Namų darbas. Tarpusavio vertinimai

- Peržiūrėkite trijų bendramokslių 3.1 namų darbo atlikimą
- Vertinimas: kiekvienas punktas gali būti vertinamas pagal 0–5 balų skalę kreipiant dėmesį į
  a) temos pristatymą,
  b) argumentaciją ir aiškumą.
  c) skyriaus turinio panaudojimą.

4 skyrius. Informatinio mąstymo, integruoto su STEAM, mokymo turinio kūrimas

Šiame skyriuje pristatomas STEAM mokymo planavimo koncepcinis modelis (Quigley ir Herro, 2017). Daugiausia dėmesio skiriama mokymo turinio kūrimui. Priemonės turėtų būti pasirenkamos atsižvelgiant į turinį, tačiau jos turi leisti pristatyti turinį, kuris kitu atveju mokiniams būtų per sudėtingas. Taip pat pradedamas projektinis darbas, atliekamas 3–4 studentų komandose, kuris apjungia viską, ko buvo mokoma šiame modulyje. Tai pirmoji projekto dalis, kurioje kuriamas mokomasis turinys ir ieškoma informatinio mąstymo taikymo galimybių. STEAM mokymui taikoma uždavinių (problemų) sprendimo pedagogika. Uždaviniai yra atviri, kad mokiniai galėtų laisvai kurti savo sprendimus naudodamiesi įvairiomis STE(A)M sritimis. Šioje santrumpoje A reiškia laisvuosius menus, kurie leidžia platesnei mokinių auditorijai įsitraukti į uždavinių sprendimą ir padeda jiems pamatyti, kaip uždavinys susijęs su realiu pasauliu.

Sėkmingam STEAM mokymui svarbus ir mokymo turinys, ir mokymosi aplinka. Jie yra tarpusavyje susiję: mokymo turinys turėtų išnaudoti mokymosi aplinkos galimybes, o kuriant mokymosi aplinką reikėtų atsižvelgti į mokymo turinio poreikius. Čia daugiausia dėmesio skiriame turiniui, tačiau galima pasinaudoti ankstesnėmis modulio dalimis ir žiniomis apie mokymosi aplinką. Siekiama, kad dėmesys būtų sutelktas į tai, kokių žinių mokiniai turėtų įgyti. STEAM uždaviniai paimti iš realaus pasaulio, kur jie dažnai neturi vienintelio teisingo

atsakymo. Mokytojas turėtų sukurti realius scenarijus, kuriais remiantis mokiniams būtų pateikiami uždaviniai. Galimybė pateikti įvairių rūšių atsakymus ir užduoties dydis turėtų skatinti kūrybiškumą ir bendradarbiavimo poreikį.

Mokymas – tai, kaip mokytojas organizuoja, parengia ir perteikia turinį. STEAM turinys turėtų būti kuriamas atsižvelgiant į tris reikalavimus: uždavinių sprendimą, disciplinos integravimą ir uždavinių sprendimo įgūdžius. STEAM pedagogikoje svarbiausia yra probleminis pateikimas. Turinys pateikiamas atsižvelgiant į įvairius uždavinio aspektus. Šie aspektai turėtų būti iš įvairių disciplinų ar turinio sričių. Mokytojas negali būti kiekvienos disciplinos ekspertas, tačiau gali nurodyti šaltinius ir kitus ekspertus, kurie gali suteikti daugiau informacijos. Pats uždavinys turėtų būti ne klausimas su vienu teisingu atsakymu, o reali situacija, kurioje yra keli uždavinio sprendimo būdai. Situacija turėtų pateikti uždavinio kontekstą ir mokiniams turėtų atrodyti aktuali. Svarbu autentiškumas, jei uždavinys ir jo kontekstas atrodys netikri, mokinys bus mažiau motyvuotas mokytis.

Disciplinų integracija – tai mokinių mokymas derinti įvairių sričių turinį ir metodus sprendžiant iškilusią problemą. Nors STEAM idėja – naudoti visus šioje santrumpoje įvardytus dalykus, ne visiems uždaviniams ar jų sprendimo būdams jų visų prireiks. Turinio sričių ir metodų pasirinkimas turėtų priklausyti nuo mokinių. Mokinių atsakymuose disciplinų integracijos laipsnis gali būti įvairus. Jis priklauso nuo to, kaip pateikiamas uždavinys, ir nuo mokinių ankstesnės patirties derinant temas. Disciplinų integracija gali būti daugiadisciplininė, tarpdisciplininė arba transdisciplininė. Daugiadisciplininis uždavinių sprendimas yra mažiausiai integruotas, į skirtingus uždavinio aspektus atsakoma atskirai. Tarpdisciplininis uždavinių sprendimas sujungia skirtingų disciplinų metodus. Transdisciplinišumo atveju vienos disciplinos turinys tampa aktualesnis, kai į jį žvelgiama iš kelių disciplinų konteksto.

Uždavinių sprendimo įgūdžiai apibūdina mokinių savybes. Skiriami trijų tipų įgūdžiai: pažintiniai, sąveikos ir kūrybiniai. Sprendžiant uždaviniu šie įgūdžiai yra ugdomi. Gebėjimai yra bendrieji, tačiau tam tikri uždaviniai ir jų sprendimo būdai gali pagerinti tam tikrą gebėjimą. Abstrahavimas, analizė, taikymas, klasifikavimas, formulavimas, interpretavimas, suvokimas, modeliavimas, sintezė ir klausinėjimas yra pažintinių įgūdžių pavyzdžiai. Juos galima tobulinti taikant mokymo metodus, kurie remiasi stebėjimu, patirtimi, apmąstymu ir samprotavimu. Sąveikos įgūdžiai susiję su bendravimu ir bendradarbiavimu. Žvelgiant iš vieno mokinio perspektyvos, šie įgūdžiai apima gebėjimą mąstyti, perduoti informaciją, konstruoti paaiškinimus, dalyvauti argumentacijoje, pateikti įrodymus, pristatyti, atsakyti ir paaiškinti. Bendradarbiauti – tai mokytis kartu pasiskirsčius su uždaviniu susijusias užduotis. Optimaliu atveju bendradarbiaujant pavyktų sujungti individualias žinias, surinkti įrodymus ir bendrą patirtį.

Kūrybiškas mokymasis reikalingas inovacijoms, idėjoms, sprendimams ir jų realizavimui ugdyti. Reikalingi tokie įgūdžiai kaip projektavimas, šablonų atpažinimas, žaidimas, atlikimas, modeliavimas ir idėjų jungimas. STEAM mokymas yra pritaikytas kūrybiškumui, nes leidžia įvairiais būdais spręsti uždavinius ir parodyti supratimą. Mokytojas turi pasiūlyti mokiniams sąvokas ir priemones, kad jie galėtų įsitraukti į uždavinių sprendimo scenarijus ir įgytų patirties. Tiek meno, tiek technologijų dalykai, įtraukti į STEAM akronimą, suteikia galimybę kūrybiškumui. Derinant juos su kitomis temomis, galima rasti naujų išraiškos formų, kurios padeda ugdyti ir įgūdžius, ir turinio supratimą. Dizainas – tai meno ir technologijų derinimas

kuriant produktus kaip uždavinių sprendimus. Kartais technologijos pritaikomumas yra problema, kurią gali išspręsti dizainas.

Informatinis mąstymas apima daugelį STEAM pedagogikos aspektų. Jis apima įvairių dalykų požiūrius ir suteikia priemones jiems tyrinėti. Nors kompiuterio valdymas nėra menas (yra skirtingų nuomonių), jis gali būti naudojamas kaip meninė priemonė. Kompiuteriais galima modeliuoti įvairius reiškinius ar technologijas, jei jų nėra. Taip pat galima vizualizuoti įvairius duomenų rinkinius ir rezultatus. Kompiuterinis pateikimas – tai ne tik tekstas ir paveikslėliai, bet ir animacija, garsai ir vaizdo įrašai. Pasirinkti ir derinti įvairias kompiuterių galimybes leidžia kūrybiškas jų naudojimas. Kaip ir įvairūs STEAM dalykai, kompiuterių ir susijusių priemonių galimybės neturėtų būti primetamos mokiniams, bet turėtų būti prieinamos, kai jų prireikia. Mokiniams reikia turėti išankstinės informatinio mąstymo ir darbo su kompiuterinėmis priemonėmis patirties, kad galėtų tai panaudoti STEAM projektuose.

Paskaita. STEAM mokymo modelis (1)

- Problemomis paremtas pateikimas
- Disciplinos integracija
- Problemų sprendimo įgūdžiai
- Informatinio mąstymo perspektyvų taikymas

STEAM mokymo koncepcinis modelis – mokymo turinys

Namų darbai komandose. Pradedamas kurso projektas

- komandomis po 3–4 studentus
- temos pasirinkimas
- mokomojo turinio kūrimas
- informatinio mąstymo taikymo galimybių atpažinimas.

Komanda, parengusi planą, aptaria jį su projekto vadovu.

5 skyrius. Mokymosi aplinkos kūrimas informatiniam mąstymui, integruotam į STEAM

Šis skyrius tęsia STEAM mokymo planavimo koncepcinį modelį (Quigley & Herro, 2017; Quigley & Herro, 2019). Daugiausia dėmesio skiriama mokymosi aplinkos projektavimui. Kiti šio modulio skyriai suteikė pagrindus pasirinkti ugdymo technologijas, kurios padeda integruoti

informatinį mąstymą į STEAM temas. Kaip ir planuojant turinį, dėmesys turėtų būti sutelktas ne į technologiją, o į aplinkos, kurioje galima įgyvendinti scenarijų, sukūrimą. Kadangi uždavinių sprendimo būdas priklauso nuo mokinių, lygiai taip pat ir kokias technologijas naudoti turėtų lemti jų sprendimas. Įtraukiant uždavinius, kurių natūralus sprendimas būtų automatizuoti algoritmai, mokiniai skatinami mąstyti informatiniu būdu. Šis skyrius taip pat yra antrosios projektinio darbo dalies pradžia, kai kuriama mokymosi aplinka ir pasirenkamos ankstesniame skyriuje sukurtam(-iems) scenarijui(-ams) pritaikytos mokomosios technologijos. Projektuojant STEAM mokymosi aplinką daugiausia dėmesio reikėtų skirti mokymo metodams, technologijų integravimui, vertinimo praktikai ir lygiaverčiam dalyvavimui.

STEAM mokymas grindžiamas mokytojo pagalba mokiniams, kas skiriasi nuo tradicinių mokytojo vedamų pamokų. Mokiniai turėtų būti skatinami prisiimti atsakomybę už savo mokymąsi. Iš pradžių tai gali būti sudėtinga ir mokiniai vis dar gali tikėtis atsakymų iš mokytojo. Vienas iš metodų, kuriais galima pakeisti tokį elgesį, yra toks: prieš kreipdamiesi į mokytoją, mokiniai turėtų paklausti arba pasidomėti trijuose šaltiniuose. Mokiniai turėtų išmokti pasikliauti bendraamžių žiniomis arba tuo, ką patys gali sužinoti. Be to, prieš mokytojui atsakant mokiniams, jų galima paprašyti papasakoti, ką jie iki šiol rado. Mokytojas neturėtų pateikti teisingo atsakymo, bet remtis pateiktais rezultatais ir skatinti toliau tyrinėti. Realūs uždaviniai ir problemos neturi vienareikšmio atsakymo, todėl mokiniai turėtų išmokti jaustis patogiai nežinodami tikslaus atsakymo. Tačiau atsakymai gali būti daugiau ar mažiau gerai paruošti, tad šiuo atveju svarbus mokinių gebėjimas pagrįsti pasirinktą sprendimą.

Pamokose ugdomi abstrahavimo, analizės, taikymo, formulavimo, bendradarbiavimo, argumentavimo, įrodymų pateikimo ir pristatymo įgūdžiai. Mokytojas turėtų sudaryti mokiniams galimybes nuolat naudoti šiuos įgūdžius įvairiomis aplinkybėmis. Pagal STEAM mokymo modelį mokiniai turėtų išmokti ieškoti kelių uždavinio sprendimo būdų. Laisvė rinktis sprendimo būdą skatina išradingumą ir kūrybinių įgūdžių panaudojimą. Siekdamas tai paremti, mokytojas turėtų papildyti uždavinio sprendimo scenarijų sąvokomis, priemonėmis ir galimybėmis, kad mokiniai galėtų įgyti įvairios patirties. Problemos aprašymas turėtų padėti mokiniams mokytis savarankiškai, kai bendraamžių pagalba ir bendradarbiavimas atsiranda kaip natūralūs poreikiai. Tam reikia, kad uždavinio apibūdinimas leistų mokiniams prisiimti skirtingus vaidmenis sprendžiant tą uždavinį. Kai sprendžiant uždavinį reikia derinti skirtingas žinias ir įgūdžius, tuomet prašymas bendraamžių pagalbos ir bendradarbiavimas neatrodo priverstinai skatinami. Mokytojas turėtų parengti uždavinio scenarijų taip, kad skatintų pagal amžių atitinkantį socialinį ir emocinį įsitraukimą į mokymąsi.

Technologijos yra vienas iš STEAM dalykų. Kuriant mokymosi aplinką, mokiniai turėtų turėti galimybę naudotis įvairiomis technologinėmis priemonėmis uždaviniams spręsti. Jie turėtų būti ne naudotojai, o techninių sprendimų gamintojai. Kad galėtų taikyti technologijas uždaviniui spręsti, mokiniai turi turėti išankstinės patirties su priemonėmis, kurios yra susiję su to uždavinio scenarijumi. Tik gerai susipažinę su technologijomis, jie gali jas integruoti į mokymąsi. Šio modulio kontekste ypač daug dėmesio skiriama priemonėms, padedančioms integruoti informatinį mąstymą. Sprendime turi būti aspektų, kuriems naudingas algoritmo vykdymo automatizavimas. Reikia skirti paruoštų sprendimų naudojimą ir sprendimo kūrimą. Pavyzdžiui, istorinių faktų pateikimas tinklalapyje nėra informatinis mąstymas, tačiau tų faktų atradimas analizuojant duomenis kompiuteriu yra informatinis mąstymas. Mokiniams gali būti naudinga turėti įrankius, skirtus konkretiems tikslams, ir bendruosius įrankius, kuriuos galima

modifikuoti (suprogramuoti) pagal poreikį. Bendrosios priemonės gali labiau padėti derinti įvairių dalykų metodus, nes jos nėra skirtos tik vienam kontekstui.

Vertinimas yra pagrindinė bet kurio švietimo modelio, taigi ir STEAM mokymo, dalis. Mokymas, mokymasis ir vertinimas turi būti suderinti, kad būtų pasiekti ugdymo tikslai. Dėl į mokinį orientuoto STEAM mokymosi aspekto tradiciniai vertinimo metodai, pavyzdžiui, testai su keliais atsakymų variantais, yra netinkami. Metodo pasirinkimas turėtų atspindėti STEAM modelį, kuriame siekiama, kad mokiniai įsitrauktų į įvairius tyrimo būdus, išmoktų naudotis įvairiais įgūdžiais ir bendradarbiautų ieškodami sprendimo. Kad sprendimas būtų autentiškas, mokinių turėtų būti prašoma pritaikyti įgytas žinias ir įgūdžius sprendžiamo uždavinio kontekste. Uždavinio sprendimo procesas, kuris baigiasi sprendinio radimu, rodo įgytas žinias. Tačiau vertinant tik rezultatą lieka galimybė tobulinti įgūdžius, kuriais naudojamasi vykstant procesui. Siekiant užtikrinti, kad vertinimas būtų susietas su STEAM mokymo praktika, jis turėtų būti įtrauktas į mokymosi procesą. Mokytojas turėtų dažnai ir kokybiškai teikti mokiniams grįžtamąjį ryšį uždavinių sprendimo metu. Taip mokytojas gali užtikrinti, kad mokinių turinio supratimas atitiktų mokymosi tikslus. Grįžtamasis ryšys taip pat gali paskatinti mokinius atidžiau stebėti savo pažangą ir giliau mąstyti įvairiomis temomis.

Lygiaverčio dalyvavimo tikslas – būti sąžiningam, atsižvelgiant į mokinio turimus įgūdžius ir žinias. Kultūrinės normos ir šeimos tradicijos turi įtakos mokinių turimoms žinioms. Į tai reikėtų atsižvelgti. Mokytojas, pakviesdamas mentorius ir ekspertus, gali padėti tiems mokiniams, kurie galbūt neturėjo galimybės gauti tokių žinių už mokyklos ribų. Taip galima supažindinti mokinius su kitokiomis pažiūromis ir būsimų profesijų galimybėmis. Čia taip pat svarbus kultūrinis jautrumas, kad nebūtų skatinami stereotipiniai požiūriai, bet būtų parodyta, kad ateitis gali būti tokia, kokią ją susikursi, nepriklausomai nuo savo kilmės. Sukurdamas erdvę saviraiškai, mokytojas leidžia mokiniui parodyti savo stipriąsias puses. Kadangi STEAM mokymas yra orientuotas į mokinį, o vertinimas yra įtrauktas į mokymosi procesą, tai sudaro sąlygas lygiateisiškiau dalyvauti, nes kiekvienas mokinys nėra verčiamas taikyti tą patį vertinimo modelį.

Paskaita. STEAM mokymo modelis (2)

- Mokymo metodai
- Vertinimo praktika
- Lygiavertis dalyvavimas
- Švietimo technologijų pasirinkimas informatiniam mąstymui

STEAM mokymo koncepcinis modelis. Mokymosi aplinka

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

Namų darbai komandose. Tęsiamas kurso projektas

- Tęsiama iš 4 skyriaus
- Mokymosi aplinkos kūrimas
- Švietimo technologijų pagalbos planavimas
- Ataskaitos rašymas
- Pristatymų rengimas

6 skyrius. Projektų pristatymai

Būsimųjų mokytojų komandų mokomųjų projektų, skirtų derinti informatinį mąstymą, STEAM ir edukacines technologijas, pristatymas.

Komandų pristatymai ir diskusijos

kiekvienai komandai

10 min pristatymui

5 min. klausytojų klausimams ir komentarams

Vertinimo reikalavimai ir vertinimo strategija

Visos vertinimo užduotys turi būti pateiktos iki nustatyto termino.

Vertinimo užduotis Turėtų įvertinti ir pateikti įrodymus apie modulio mokymosi rezultatų pasiekimą.	**Vertinimo kriterijai ir metodas** Rašto darbų vertinimas, pvz., apimtis (žodžiais), struktūra (įvadas, pagrindinė dalis, išvados), tinkamas terminų ir sąvokų vartojimas.
Informatinio mąstymo taikymo STEAM temai(-oms) plano (pagrįsto PRADA) peržiūra.	Priimtas ar nepriimtas, tarpusavio vertinimas.

	Kriterijai: temos pristatymas, argumentacija ir aiškumas, skyriaus turinio panaudojimas.
Plano (remiantis informatinio mąstymo perspektyvomis STEAM sistemai), skirto pasirinkti tinkamą technologiją, padedančią mokytis STEAM temos(-ų), peržiūra.	Priimtas ar nepriimtas, tarpusavio vertinimas. Kriterijai: temos pristatymas, argumentacija ir aiškumas, skyriaus turinio panaudojimas.
Mokomojo plano (dizaino) projekto ataskaita ir pristatymas.	Dėstytojas kursą įvertina remdamasis projektiniu darbu. Projekto ataskaita vertinama remiantis „STEAM klasės mokinių mokymosi patirties vertinimo“ kriterijais (žr. toliau). Už tai, kaip gerai projekto pristatyme apibendrintos pagrindinės idėjos, galima gauti papildomą įvertinimą (vertinant 10 balų sistemą tai būtų 0,5).

Įgyvendinimo idėjos

4 ir 5 skyrius leidžia lengvai pratęsti arba sutrumpinti modulį, patobulinant arba supaprastinant projektinį darbą. Projektinį darbą galima pratęsti rengiant mokymosi plano rinkinį arba visą kursą. Taip pat lengva sutrumpinti modulį pasiūlant mokiniams paruoštą medžiagą ir šablonus. Kraštutiniu atveju projektinis darbas galėtų būti individuali užduotis, skirta mokymosi planui sumanyti ir idėjoms pristatyti 6 skyriuje.

Informacijos šaltiniai

Aho, A. V. (2011). Ubiquity symposium: Computation and computational thinking. *Ubiquity*, 2011(January).

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.

Candela, L. (2019). Encoding and Decoding Mathematics for the Natural Sciences: an inferential account of the application of mathematics and its reasonableness. The Vassar College Journal of Philosophy, 8.

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.

Larsson, P., Apiola, M. V., & Laakso, M. J. (2019, May). The uniqueness of Computational thinking. *In 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 687-692). IEEE.

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.

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., & Apiola, M. (2013). Three computing traditions in school computing education. *In Improving computer science education* (pp. 108-124). Routledge.

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.

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).