KLAIPĖDOS UNIVERSITETAS
GAMTOS IR MATEMATIKOS MOKSLŲ FAKULTETAS
INFORMATIKOS KATEDRA

VAIDAS LUKAUSKAS

GMVIN03 gr. studentas

ADAPTYVIOS E. MOKYMOSI APLINKOS INTELEKTUALAUS KOMPONENTO KŪRIMAS, NAUDOJANT KOHONENO SAVAIME SUSITVARKANČIO ŽEMĖLAPIO NEURONINĮ TINKLĄ

Baigiamasis magistro darbas

Darbo vadovas : prof. Antanas Andrius Bielskis

KLAIPĖDA, 2005

ANOTACIJA

Šiame darbe yra iašanalizuoti ir aprašyti adaptyvios e. mokymosi aplinkos intelektualaus komponento sukūrimo principai, naudojant Kohoneno savaime susitvarkančio žemėlapio neuroninius tinklus. Čia sukurtas intelektualus komponentas, kuris yra realizuotas panaudojant agentines technologijas, sprendimus, susijusius su e. mokymusi, priima atsižvelgiant į konkretaus sistemos naudotojo poreikius. Darbe taip pat yra aprašoma visa bendro pobūdžio adaptyvi e. mokymosi aplinka, pažymint intelektualaus komponento užimamą vietą joje, bei pastarojo komponento sąryšius su kitomis sistemos dalimis.

REIKŠMINGIEJI ŽODŽIAI: adaptyvi e. mokymosi aplinka, e. patarėjas, intelektualus komponentas, SOM tinklai e. mokyme.

ABSTRACT

In this master graduation work there are analyzed and described the principles of creation an intelligent component of adaptive e-learning system, by using Kohonen self organized map neural networks. There created intelligent component, which was realized by using agent technologies, distinguishes by abilities to make a decision in respect of concrete user needs. In this work there are also described common structure of adaptive e-learning system, by pointing the place, which takes intelligent component, and its relationships with other parts of such adaptive system.

KEYWORDS: adaptive e-learning environment, e-tutor, intelligent component, SOM's in e-learning.

PADĖKA

Noriu padėkoti prof. habil. dr. A. A. Bielskiui už pagalbą analizuojant adaptyvios e. mokymosi aplinkos veikimo principus, nagrinėjant, Kohoneno savaime susitvarkančio žemėlapio neuroninių tinklų panaudojimo e-mokyme galimybes, bei už pasiūlytas idėjas ir pateiktas pastabas.

TURINYS

ĮVADAS

Lietuvoje sparčiai didėjant interneto naudotojų skaičiui vis populiaresnis darosi e. mokymasis. Iš pirmo žvilgsnio tai išties labai patogus būdas įgyti naujų žinių: galima mokytis sau pageidaujamoje vietoje ir pageidaujamu laiku, be to, ir tokio mokymosi aplinkų yra sukurta pakankamai daug. Tačiau pradėjus mokintis naudotojui neretai tenka nusivilti tokių sistemų veikimu dėl jų intelektualumo trūkumo. Didžioji dalis e. mokymosi aplinkų ne tik Lietuvoje, bet ir visame pasaulyje yra paremtos principu "vienas dydis tinka visiems" (*angl.* "one size fits all"), t.y. mokymosi aplinka visiems naudotojams pateikia tą pačią mokymosi medžiagą tuo pačiu stiliumi, bei mokymosi priemonėmis ir visiškai neatsižvelgia į pačių naudotojų savybes, tokias kaip jų turimas žinias, gebėjimus įsisavinti informaciją ir siekiamus tikslus. Jau daugelį metų yra ieškoma būdų kaip tokioms sistemoms suteikti intelektualumo. Per visą šį ilgą laikotarpį buvo sukurta nemažai modelių skirtų e. mokymosi sistemų adaptyvumo prie naudotojų poreikių realizavimui, tačiau paskutiniųjų metų darbai, kaip intelektualios e. mokymosi aplinkos pagrindą, akcentuoja intelektualųjį komponetą (kartais dar vadinamą e-patarėju).

Prie šios problemos jau ne pirmus metus yra dirbama ir Klaipėdos Universitete. Ir čia jau pavyko pasiekti neblogų rezultatų e. mokymosi sistemų intelektualizavimui naudojant Q-mokymosi (*angl.* Q-learning) algoritmą. Naudodama šį algoritmą mokymo sistema naudotojui siūlo optimalų kurso išmokimo kelią, tačiau šis siūlymas yra paremtas atsižvelgiant į visų sistemos naudotojų mokymosi procesą, bet ne į pačio naudotojo asmenines savybes. Todėl šio darbo pagrindinis tikslas – sukurti adaptyvios e. moklymosi aplinkos intelektualų komponentą, priimantį sprendimus atsižvelgiant į asmenines sistemos naudotojo savybes. Šio tikslo įgyvendinimui reikia išspręsti tokius uždavinius:

1. Atlikti adaptyvios e. mokymosi aplinkos analizę.
2. Atlikti detalią adaptyvios e. mokymosi sistemos intelektualaus komponento analizę.
3. Aptarti duomenų klasifikavimo būdus, bei ištirti jų panaudojimo intelektualaus komponento kūrimui galimybes.
4. Išanalizuoti Kohoneno savaime susitvarkančio žemėlapio neuroninio tinklo veikimo principus.
5. Pasinaudojant agentinėmis technologijomis ir SOM neuroniniu tinklu, sukurti bei realizuoti praktiškai intelektualųjį komponetą.
6. Eksperimentiškai patikrinti intelektualaus komponento veikimą, tuo tikslu sukuriant demonstracinę/testinę, internetinėje terpėje veikiančią, vartotojo sąsają.

1. E-MOKYMOSI SISTEMŲ ADAPTACIJOS MODELIŲ ANALIZĖ

1.1 E-mokymosi ir jo aplinkos samprata

Šiandieniniame pasaulyje vis populiaresnis tampa mokymosi būdas ne sėdint bendroje klasėje, bet naudojant informacines technologijas mokytis virtualioje aplikoje (e-mokymasis). Tokių e-mokymosi aplinkų labai sparčiai daugėja, tačiau čia iškyla problema, kai pradedame vertinti jų kokybę ir duodamus rezultatus. Tada ir paaiškėja, kad toks individualus mokymasis (kuomet pats mokantysis nustato sau reikalavimus ir tikslus, bei laisvai pasirenka norimą medžiagą ir priemones), negali duoti gerų rezultatų naudojantis paprastomis e-mokymosi aplinkomis [36]. Pagrindinis tokios e-mokymosi aplinkos trūkumas yra tas, kad tos aplinkos dažniausiai būna vienodos visiems besimokantiesiems ir visai nėra atsižvelgiama į tai, kad besimokinantys asmenys gali turėti visai skirtingą pradinio pasiruošimo lygį, tam pačiam dalykui studijuoti, gali naudoti skirtingus resursus ir metodus. Tada ir atsiranda poreikis visą mokymosi apliką kiek galima labiau adaptuoti (pritaikyti) prie konkretaus asmens poreikių. Tokių aplinkų kūrimui yra naudojamos įvairiausios technikos, tačiau visa tai pamažu yra bandoma standartizuoti. Pagrindinis standartizavimo motyvas – sumažinti išlaidas, kurių reikia adaptyvių e-mokymosi sistemų (taip pat ir mokymosi kursų) sukūrimui [6]. Kitas tikslas – padidinti tokių aplinkų pakartotinio panaudojimo (reusable) galimybes (t.y. kuriant naują mokymosi aplinką, kai kurias reikiamas dalis būtų galima paimti iš jau sukurtų sistemų [4]).

Šiuo metu yra sukurta daugybė e-mokymosi sistemų. Visas jas mes galime suskirstyti į tris pagrindines grupes [19]:

1. *Kompiuterio asistuojamos instrukcijų sistemos* (*angl.* computer assisted instruction systems). Tokio pobūdžio sistemos naudotojui pateikia iš anksto nustatytą, statišką mokymosi medžiagą ir visai neatsižvelgia į jo poreikius. Čia dažniausiai temos yra pateikiamos viena po kitos tokia eilės tvarka, kokia buvo numatyta iš anksto.
2. *Hypermedia sistemos* (*angl.* hypermedia systems). Tokiose sistemose mokymosi proceso eiga visapusiškai priklauso nuo vartotojo. Čia besimokančiajam mokymosi medžiaga yra pateikiama hypermedia forma (daug nuorodų apie konkrečią temą), ir tik nuo vartotojo pasirinkimo priklauso tolimesnė mokymosi eiga.
3. *Intelektualaus patarėjo sistemos* (*angl.* intelligent tutoring systems). Tai sistemos kuriose mokymosi eiga yra pilnai valdoma ne vartotojo, bet pačios sistemos (t.y. sistema pati nusprendžia kokią megžiagą ir kada pateikti vartotojui). Tokios sistemos moka adaptuotis prie konkretaus vartotojo poreikių, sugeba atsižvelgti į vartotojo įgytas žinias.

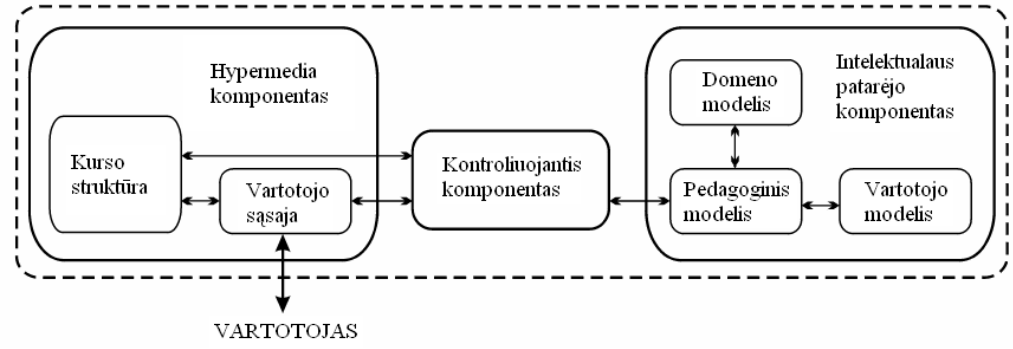
1.2 Adaptyvios e-mokymosi aplinkos savybės

Pats žodis "adaptyvus" e-mokyme labiausiai yra siejamas su įvairiausiomis mokymosi sistemos charakteristikomis ir galimybėmis. Sistema (e-mokymosi aplinka) yra laikoma adaptyvia jaigu ji moka [3]:

- Valdyti vartotojų veikas (ką vartotojai daro).
- Naudojant nustatytus modelius, teisingai tas veikas interpretuoti.
- Iš interpretuotų veikų teisingai nustatyti ko vartotojui reikia ir kam jis teikia prioritetus, bei visa tai pavaizduoti reikiamuose modeliuose.
- Mokymosi procesą vykdyti dinamiškai, atsižvelgiant į vartotojo turimas ir įgyjamas žinias.

1.3. Adaptyvios e-mokymosi aplinkos modelio analizė

Kadangi visos anksčiau paminėtos sistemos turi savo trūkumus (pvz. kompiuterio asistuojamos instrukcijų sistemos yra statiškos ir nepatogios; hypermedia sistemos neturėdamos gerų navigacijos įrankių gali nesunkiai "paklaidinti" vartotoją medžaigos gausybėje ir nutolinti jį nuo siekiamo tikslo [10], o intelektualaus patarėjo sistemos nesuteikia galimybių vartotojui tiesiogiai įtakoti mokymosi procesą [20]), tai yra bandoma jas apjungti sudarant naujus mokymosi aplinkos modelius. Kaip pavyzdį būtų galima pateikti adaptyvios hypermedia sistemos modelį [19]:



1 pav. ***Adaptyvios hypermedia sistemos modelis[19]***

Šiame modelyje **hypermedia komponentas** yra atsakingas už kurso medžiagos struktūrizavimą, jos vaizdavimą ir valdymą [20]. Jo sudedamosios dalys:

- *Kurso struktūra* – tai tinklinių puslapių, sudarančių visą mokymosi kursą, hierarchinė struktūra. Čia kursas yra suprantamas kaip atskirų mokymosi vienetų

visuma, kurių kiekvienas yra vaizduojamas atskiru tinkliniu puslapiu (*angl.* web page).

- *Vartotojo sąsaja* yra skirta sąveikos tarp vartotojo ir sistemos realizavimui. Ji atsakinga už tinklinių puslapių pavaizdavimą, navigacijos priemonių pateikimą.

Kontoliuojančio komponento paskirtis – tarpusavyje suderinti hypermedija ir intelektualaus patarėjo komponentus, tam kad sistema galėtų adekvačiai reaguoti į vartotojo veiksmus [10].

Pats svarbiausias komponentas siekiant sistemą padaryti adaptyvia yra **intelektualaus patarėjo komponentas**. Kadangi vienas iš šio darbo uždavinių yra intelektualaus komponento analizė ir jo sukūrimas, tai pastarąjį komponentą panagrinėsiu detaliau.

1.3.1. Intelektualaus patarėjo komponento analizė

Iš 1 pav. pateikto modelio galime pastebėti, kad norėdami padaryti hypermedia sistemą adaptyvia mes turime ją apjungti su intelektualaus patarėjo komponentu. Rezultatas - mes gauname hypermedia sistemą, pasižyminčią prisitaikymu prie vartotojo poreikių, lankstumu ir patogumu. Šis komponentas yra būdingas visoms e-mokymosi sistemoms, naudojančioms intelektualų patarėją [20, 7]. Tam kad sistema galėtų adaptuotis, jai yra būtina žinoti pateikiamo kurso apimtis ir struktūrą (t.y. domeno modelį), kiekvieno individualaus vartotojo turimas žinias ir pasiektus rezultatus, bei viso jų mokymosi proceso seką.

Pagrindinės intelektualaus patarėjo sudedamosios dalys yra šios:

- *Domeno modelis* (*angl.* The Domain model). Kadangi dauguma adaptyvių mokymosi aplinkų yra paremtos adaptyviu kurso medžiagos pateikimu, tai šis modelis yra skirtas būtent to siūlomo kurso medžiagos reprezentavimui (apibūdinimui) [6]. Jeigu kalbėtume apie bendresnio pobūdžio mokymosi aplinkas, tai domeno modelis galėtų apimti informaciją ne tik apie mokymosi kursą, bet ir informaciją apie mokymosi procese dalyvaujančius asmenis, roles bei informacijos paskirstymą. Pagrindinis šio modelio aspektas yra tas, kad čia siekiant mokymosi aplinką padaryti adaptyvia, tarp mokymosi kurso elementų yra identifikuojami atitinkami ryšiai, kurie leidžia mokymosi kursą pateikti atsižvelgiant į naudotojo poreikius. Būtent šio modelio dėka, sistema gali nusakyti kas jau buvo pateikta naudotojui (t.y. ką jis jau turėtų žinoti) ir ką dar reikės pateikti ateityje [19].
- *Vartotojo modelis*. Kiekviena mokymosi aplinka gali naudoti savo specifinius modelius bei charakteristikas, tačiau yra pastebėtas vienas bendras požymis, būdingas daugumai sistemų – tai yra, kad modelis gali būti papildomas (praplečiamas) mokymosi proceso metu, tam kad į jį būtų įtraukti visi reikiami elementai susiję su besimokančiojo ankstesnio mokymosi

duomenimis (kitaip tariant, įtraukiami naudotojo naudojimosi sistema istorijos ar valdymo elementai) [26]. Esminis tokio modelio požymis yra tas, kad jis apima ne tik bendro pobūdžio informaciją apie mokinį (tokia informacija gali būti laikoma informacija susijusi su besimokančiojo demografija ar ankstesniais pasiekimais mokymosi srityje), bet taip pat nuolatos kaupia informaciją apie besimokančiojo veiksmus sistemoje. Kartais vartotojo modelis yra siejamas su grupių modeliais.

- *Grupių modeliai* (*angl.* group models). Tokio tipo modeliai siekia išgauti charakteristikas iš grupių, kurias sudaro besimokantys asmenys [19,20]. Pagrindinis grupinių modelių skiriamasis bruožas yra tas, kad šie modeliai paprastai yra sudaromi dinamiškai (čia turima omenyje ne dinamišką modelių užpildymą, bet būtent sudarymą/sukūrimą) [26]. Kitas svarbus požymis – grupių modeliai yra paremti besimokančiųjų grupių identifikavimu/nustatymu pagal jų bendras charakteristikas, bendras mokinimosi aplinkas ar bendrus siekiamus tikslus. Šie grupių modeliai yra naudojami tiek nustatyti ir apibūdinti kuo besimokantieji yra panašūs ir kuo skirtingi, tiek nustatyti ar atitinkami naudotojai gali priklausyti konkrečiai vienai grupei, ar ne. Toks dinaminis naudotojų/besimokančiųjų grupių sudarymas ir nustatymas, kokiai konkrečiai grupei ar kelioms grupėms priklauso konkretus naudotojas, yra plačiai naudojamas produktų (prekių, paslaugų ir kt.) pasiūlos ar rekomendavimo modeliuose [36, 19].
- *Pedagoginis modelis*. Šis modelis yra atsakingas už sistemos priimamų sprendimų teisingumą, t.y. šis modelis pasinaudodamas domeno modelio ir vartotojo modelio teikiama informacija, turi nuspręsti kokią mokymosi medžiagą reiktų pateikti vartotojui tam tikru laiko momentu [19]. Šis modelis besimokančiajam pasiūlo patį palankiausią ir optimaliausią kurso mokymosi kelią.

1.3.2 E-mokymosi aplinkos adaptacijos technikos

Kaip pagrindines sistemų adaptacijos technikas butų galima išskirti:

- *Adaptyvi sąveika (angl.* adaptive interaction*).* Tai labiausiai yra susiję su sistemos sąsaja. Vartotojui turi būti suteikta galimybė pritaikyti sistemos sąsają savo poreikiams, tačiau jokiu būdu nepakeičiant mokymosi turinio ir esmės [6]. Kaip pavyzdį tokios adaptacijos galima paminėti vartotojo interfeiso keitimą: spalvų, šrifto dydžio, grafinių ir kitų objektų keitimas pagal vartotojo poreikius. Iš esmės tai keičia tik vizualų vaizdą, tačiau mokymosi turinį ir pateikimo formą išlaiko nepakitusius.
- *Adaptyvus kurso (mokymosi medžiagos) pristatymas* (*angl.* adaptive course delivery). Tai labiausiai paplitusi adaptacijos technika šiandieninėse e-mokymosi aplinkose. Jų pagrindinis

tikslas – pritaikyti (ar parinkti) tokį mokymosi kursą ar net keletą kursų, kurie labiausiai tenkintų naudotojo individualius poreikius [19]. Visa tai atliekama atsižvelgiant į tai, kad naudotojui būtų pateikta tik ta medžiaga, kuri atitinka jo reikalavimus ir žinojimo lygį. Čia yra siekiama optimalumo – pteikti tik tiek medžiagos, kad jos užtektų naudotojui išmokti ir, kad tas laikas, per kurį yra įsisavinama pateikiama medžiaga, būtų kuo mažesnis [4, 6]. Tokio tipo adaptacija naudotojui turi stengtis kuo labiau panaikinti mokytojo trūkumą (t.y. turi nurodyti kiek ir ką reikia mokytis, kokie yra pagrindiniai uždaviniai ir pan.), didinti naudotojo susidomėjimą naujomis žiniomis. Tokių adaptacijų pavyzdžiai yra dinaminis mokymosi kursų sudarymas, adaptyvus alternatyvių kursų medžiagos pasirinkimas ir kt.

- *Turinio suradimas ir apjungimas* (*angl.* content discovery and assembly). Tai yra siejama su potencialių mokymosi žinių atradimu ir pateikimu iš įvairiausių mokymosi šaltinių. Šiuo atveju adaptyvus mokymasis yra realizuojamas panaudojant adaptyvius modelius ir žinias apie besimokantį naudotoją, kurios yra surenkamos per to naudotojo mokinimosi/turimų žinių patikrinimo procesą [6].
- *Adaptyvaus bendradarbiavimo palaikymas* (*angl.* Adaptive Collaboration Support). Tokio pobūdžio mokymosi aplinka labiausiai akcentuoja adaptacijos palaikymą mokymosi procese bendraujant tarp daugelio atskirų mokinių, ar net adaptyvaus bendravimo sukūrimą siekaint bendrų tikslų (tarkim, išstudijuoti tam tikrą sritį, įgyti konkrečių žinių). Visa tai padeda išvengti besimokančiojo atskyrimą (izoliavimą) nuo jį supančios aplinkos [6, 46]. Tai ypač pabrėžia modernioji mokymosi teorija, kuri pagrindiniais sėkmingo mokymosi veiksniais laiko bendradarbiavimo svarbumą, bendrą (koperatinį) mokymąsi, besimokančiųjų tarpusavio bendravimą. Būtent adaptyvių technikų panaudojimas gali palengvinti komunikavimo/bendradarbiavimo procesus, bei užtikrinti reikiamą atitikimą tarp bendradarbiaujančiųjų.

1.4 Intelektualaus patarėjo komponentų modeliavimas laikantis standartų

1.4.1 Adaptyvaus domeno modeliavimas

Dabartiniai e-mokymosi standartai ir koncepcijos daugiausia dėmesio skiria mokymo medžiagos turiniui (tiksliau – turinio centralizavimui). Yra kuriami įvairiausi scenarijai, kurie nustato kaip tą mokymosi turinį reiktų pateikti naudotojui. Vienas iš galimų būdų – mokymosi medžiagos skaidymas į smulkesnes atskiras dalis – paketus [10]. Standartai ypač daug dėmesio skiria mokymosi medžiagos paieškos, apsikeitimo ir pakartotinio panaudojimo galimybių išplėtimui [26]. Mokymosi medžiaga šiuo atveju gali būti laikomi turinio elementai, mokymosi objektai ar net testavimo komponentai. Mokymosi objekto meta duomenų specifikacija (*angl.* The Learning Object

Metadata specification) aprašo kaip, būtent iš tokių atskirų mokymosi elementų ir paketų, turi būti sudaroma bendra, ištisinė mokymosi medžiaga [26]. Čia mokymosi medžiaga yra sudaroma dar atsižvelgiant į pageidaujamą jos sudėtingumą, tai yra, čia atsiranda toks komponentas, kaip pageidaujamo sunkumo lygiai. Visa tai jau yra realizuota, tačiau dabar yra siekiama, kad tokios medžiagos sudarymo darbas būtų perleistas agentams, kurie automatiškai, pagal konkretų naudotoją ir jo poreikius surinktų (sugeneruotų) jam tinkamiausią mokymosi medžiagą [8].

IMS mokymosi technologijų kūrimo specifikacijos (IMS Global Learning Consortium) mums siūlo koncepcinius modelius kurie suteikia galimybę analizuoti ne tik turinį, bet ir mokymosi metu vykstančius procesus ir kitas veikas [7]. MASIE centro ataskaita (MASSIE centre report, 2002) išskiria šiuos pagrindinius meta duomenų panaudojimo atvejus [26]:

- Turinio suskaidymui į tam tikras atskiras kategorijas.
- Sistematikų kūrimui (generavimui).
- Pakartotiniam (arba kitaip – daugkartiniam) panaudojimui.
- Dinaminiam surinkimui (t.y. kad mokymosi medžiagą būtų galima paruošti dinamiškai).

Visi aukščiau paminėti atvejai daugiau ar mažiau yra susiję su e-mokymosi aplinkos adaptacija ir pritaikymu asmeniniams poreikiams.

Galime pastebėti, kad bet kuri adaptyvi mokymosi aplinka, kuri siekia pateikti medžiagą atitinkančią vartotojo poreikius, reikalauja papildomos valdymo informacijos. Tai yra informacija apie tai, kas (kokie elementai, esybės) sudaro visą mokymosi kursą, kokiais ryšiais ir kaip jie tarpusavyje yra susiję [7]. Norėdami tai standartizuoti mes visų pirma turime turėti aiškius apibrėžimus, kurie tiksliai ir nedviprasmiškai apibrėžtų esybes ir ryšius tarp jų [26]. Visa tokių esybių ir ryšių procedūrinė interpretacija ir išplėtimas priklauso nuo konkrečios adaptyvios mokymosi aplinkos. Visa tai leis tų pačių standartų panaudojimą, kuriant skirtingas mokymosi aplinkas. Taip atsiranda galimybė mokymosi aplinkas laisvai papildyti naujomis savybėmis kai iškyla tam poreikis, bei pasiekti pageidaujamą adaptacijos lygį.

Naudodamiesi mokymosi objekto meta (MOM) duomenimis mes galime apibrėžti ryšius tarp atskirų elementų tik esant tam tikroms sąlygoms. Pagrindinės jų būtų [26]:

- Būtinas specifikacijų žodyno sukūrimas, kuris vienareikšmiškai apibrėžtų tiek ryšius tarp konceptų, tiek tų ryšių charakteristikas. Ryšiai turėtų būti apibrėžti taip, kad taikomoji programa (tiksliau, programa kurią naudos besimokantis asmuo) negalėtų skirtingai interpretuoti tų pačių dalykų.
- Kiekvienai mokymo esybei, kuri yra kaip individualus konceptas, yra priskiriamas MOM atitinkamas įrašas. Tai yra kiekvienos esybės meta duomenys.

- Esybės meta duomenys naudodamiesi aukščiau aptartu žodynu ir esybių identifikatoriais, specifikuoja esybės ryšius su kitomis esybėmis.

MOM pagrindinis teikiamas privalumas – suderinamumas su dabartinėmis mokymosi aplinkomis ir sistemomis [6]. Čia norint sistemai suteikti adaptyvumo, reiktų sukurti specifikacijų žodyną aprašantį ryšius tarp mokymosi esybių, bei į mokymosi turinio aprašymą įterpti specifinius adaptacijos konstruktus. Tuomet mokymo sistema naudodama metaduomenis galėtų teisingai interpretuoti visą mokymo kurso medžiagą. Siekiant dar didesnio mokymosi aplinkos adaptyvumo, galima būtų pritaikyti standartus ir pačio mokymo kurso medžiagos sudarymui. Autoriai [6,7,26] pataria, naudojant MOM ir siekiant išvengti mokymo kurso valdymo problemų, visus duomenis susijusius su sistemos adaptacija (tai būtų konceptų, esybių meta duomenys, specifikacijų žodynas) saugoti atskirai nuo pačio mokymo kurso aprašymo.

Aukščiau aptartas adaptyvios e-mokymosi aplinkos kūrimo modelis labiausiai orentuotas į patį mokymosi kursą, tačiau dažnai pasitaiko atvejų, kur prireikia sudėtingesnių modelių, suteikiančių galimybę valdyti tiek kursą sudarančius konceptus ir ryšius tarp jų, tiek ir fizinę kurso medžiagos organizaciją (pavyzdžiui - failų išdėstymą diske, navigaciją tarp skirtingų kurso dalių). Čia atsiranda poreikis valdyti tiek konceptus, kurie labiau nusako loginę sistemos struktūrą, tiek jiems priklausančius resursus (fizinę sistemos realizaciją) [6].

Kaip pavyzdžius adaptyvių mokymosi aplinkų, kurios išplečia egzistuojančius standarus, būtų galima paminėti tokias sistemas: OPAL (Conlan at al. 2002 pasižymi mokymosi medžiagos parinkimu ir pateikimu pagal asmeninius poreikius, tam naudojant agregacinius modelius paremtus ADL SCORM - Advanced Distributed Learning: Shareable Courseware Object Reference Model), OLO (Rodriguez, Chen, Shi, Shang), KOD (Karagiannidis, Sampson, Cardinali) [26]. Pastarosios dvi sistemos akcentuoja standartinių meta duomenų išplėtimą tam, kad būtų galima realizuoti "paketinį" mokymą realizuojančių adaptyvių sistemų kūrimą. Galime pastebėti, kad įvairios sistemos adaptyvumui realizuoti naudoja įvairiausius skirtingus modelius, tačiau joms visoms yra būdingas siekis integruoti (apjungti) į bendrą e-mokymosi sistemą tradicinę mokymo medžiagą ir meta duomenis, įgalinančius tą medžiagą valdyti ir teisingai interpretuoti.

1.4.2 Vartotojo ir grupių modeliavimas

Dabar e-mokymo aplinkų kūrimui vis plačiau naudojami besimokančiojo ir grupės modelių standartai, kurie dar nėra galutinai nusistovėję ir nuolatos kinta. Šiai kategorijai priklauso IMS Learner Information Package specifikacijos [6], kurios stengiasi apjungti aukštesnio lygmens mokymosi veikas kartu su statine informacija apie besimokantį. Tokio pobūdžio informacija yra labai svarbi e-mokymosi aplinkose, tačiau dėl menko detalumo ir aiškumo lygio jų panaudojimas

kuriant adaptyvias mokymosi aplinkas (AMA) yra gana ribotas. Pagrindinė paroblema yra ta, kad AMA norėdama tiksliai identifikuoti naudotojo poreikius ir atsižvelgdama į juos pateikti naudotojui mokymo medžiagą, reikalauja to naudotojo "bendravimo" su sistema "istorijos". Tokia sistemos naudotojo "istorija" dažniausiai būna susijusi su pačiu domeno (kurso) modeliu, tačiau dar reikalauja ir papildomų ryšių atsiradimo. Siekiant kaupti naudotojo "istoriją", kuri turėtų prasmę sistemai, mes turime sukuri ryšius apibūdinančius naudotojo būseną ryšium su mokymosi esybėmis ir konceptais [4, 7]. Tokie ryšiai turėtų nusakyti tiek akivaizdžius besimokančiojo būseną nusakančius parametrus (pavyzdžiui – naudotojas perskaitė ar neperskaitė visas nuorodas pateiktas mokymosi medžiagoje), tiek ir sudėtingesnius, gaunamus vertinant skirtingais aspektais, tame tarpe ir atliekamų pratimų, užduočių rezultatais (šiuo atveju mes nustatinėjame ar naudotojas žino, ar jis yra pasirengęs tolimesniam kurso įsisavinimui ir t.t.). Tokio pobūdžio informacijos įterpimas į besimokančiojo modelį, savaime išplečia dabartinius standartus, kurie akcentuoja tik naudotojo statinės informacijos ir ir duomenų apie aukštesniųjų mokymosi lygių veikų (*angl.* top-level activities) saugojimą [26].

Dar vienas šių modelių skiriamas bruožas tas, kad čia naudojama besimokančiojo "istorija" nėra aprašoma iš anksto (kaip kad yra aprašinėjami mokymo kursai), o yra gaunama mokymosi proceso metu. Ir tai šiek tiek apsunkina standartų pritaikymą, nes mes ne visada iš anksto galime numatyti kas nutiks ateityje. Tačiau yra ir nemažai teikiamų privalumų:

- Istorijos apie naudotoją turėjimas, intelektualiems sistemos agentams organizuojantiems visą mokymosi procesą, padeda tiksliau parinkti tokią mokymosi medžiagą, kuri atitiktų besimokančiojo lygį ir poreikius. Čia "istorija" agentui pasako ką naudotojas jau išmoko ir ko dar ne [8].
- Realaus e-mokymosi proceso metu sukaupta informacija daug tiksliau gali apibūdinti mokymosi medžiagą (jos sudėtingumą ir tikslumą naudotojų atžvigiu), nei iš anksto, dažniausiai pačio kurso autoriaus subjektyvia nuomone apibrėžtas kursas, kas savo ruožtu gali klaidinti sistemos naudotojus, kurie pavyzdžiui, gali atlikti medžiagos filtravimą pagal pageidaujamą sudėtingumą. Tokiu atveju naudotojas gautų medžiagą, kuri neatitiktų jo asmeninių poreikių.

Detalios informacijos apie besimokantįjį kaupimas ir valdymas suteikia galimybę kurti ir modeliuoti grupes. Į jas būtų galima suskirstyti naudotojus pagal pasiektą išsimokslinimo lygį, kas padėtų jiems bendrauti tarpusavyje (bendravimas gali būti ir forumų forma).

1.4.3 Pedagoginio komponento modeliavimas

Bet kurios adaptyvios sistemos modeliavimas dažniausisi susideda iš dviejų atskirų, tačiau viena kitą papildančių dalių: adaptacijos logikos ir adaptacijos veiksmų specifikavimo [20]. Pirmoji dalis yra atsakinga už gaunamos (iš vieno ar kelių modelių) informacijos susiejimą ir nustatymą, kada iškyla poreikis adaptacijai, o antroji dalis aprašo kokie veiksmai turi būti atlikti norint pasiekti pageidaujamą rezultatą (naudotojo poreikiams pritaikytą sistemą).

Daugelyje sistemų, adaptacijos logika yra realizuojama naudojantis įvairiausiomis technikomis (taisyklėmis paremtomis technikomis, argumentais paremtais protavimais (angl. Case-based reasoners) ir kt.) todėl visa tai apjungti į vieną bendrą standartą būtų labai sunku ir netikslinga [19]. Beto, logika labai gali skirtis priklausomai kuo sistemos paskirties ir siekiamų tikslų. Lengviau standartizuojami ir labiau ištirti yra veiksmai, atliekami sistemos adaptacijai realizuoti. Šiuos veiksmus yra bandoma standartizuoti juos specifikuojant naudojantis XML kalba. Tokio pobūdžio specifikacijos yra labai lanksčios ir gali būti lengvai papildomos naujomis savybėmis, priklausomai nuo konkrečios kuriamos mokymosi aplinkos.

Viena iš aiškiausių šiuo metu egzistuojančių tokio pobūdžio specifikacijų yra IMS Learning Design specifikacija [26]. Tai specifikacija naudojama aprašyti tiek paprastų, tiek ir sudėtingų sistemų elgseną (funkcionalumą) dinaminiu požiūriu. Naudojant šias specifikacijas sistemos elgseną galime aprašyti naudojant programuojamus srautus (*angl.* programming flows), į kuriuos įeina ir sąlyginiai kintamieji, bei laikantis šių nuostatų:

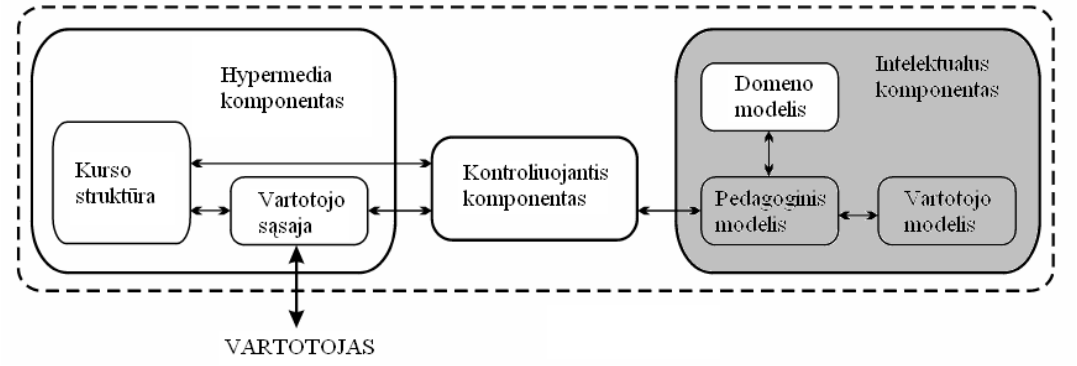
- Sistema apgalvojama žemame (nesudėtingame) lygmenyje, nes sudėtingos adaptyvios sistemos elgsenos specifikavimas yra labai sudėtingas ir neatsparus klaidų atsiradimui.
- Visos sąlygos (*angl.* conditionals) gali būti susijusios tik su tais kintamaisiais, ar su tomis būsenomis, kurios egzistuoja tik tame pačiame IMS Learning Design dokumento kontekste [4].
- Dinaminė elgsena negali būti aprašyta sisteminiame lygmenyje ir pritaikyta daugiau negu viename kontekste, ar daugiau negu vienoje mokymosi medžiagos aibėje [4].
- Ta pati elgsenos specifikacija negali būti pakartotinai panaudota (čia yra tvirtas ryšys tarp pačios elgsenos ir artifaktų su kuriais ji yra susieta)[26].

Visa tai riboja šių specifikacijų panaudojimą įvairiose sistemose, tačiau daugeliui neadaptyvių e-mokymosi aplinkų to pakanka, kad e-mokymosi aplinkai suteiktume galimybę prisitaikyti prie naudotojo poreikių.

2. ADAPTYVIOS E. MOKYMOSI APLINKOS INTELEKTUALAUS KOMPONENTO KŪRIMAS

2.1 Intelektualaus komponento apibrėžimas

Intelektualus komponentas yra pagrindinė adaptyvios e-mokymosi sistemos dalis, kurios dėka sistema geba analizuoti esamą situaciją ir adekvačiai į ją reaguoti t.y prisitaikyti prie sistemos naudotojo poreikių. 2 - ame pav. yra pateikiamas brėžinys, kuriame yra pavaizduotas kuriamasis intelektualus adaptyvios sistemos komponentas:



2 pav. ***Adaptyvios sistemos intelektualus komponentas ir jo sąryšis su kitomis sistemos dalimis[19]***

Aukščiau pateiktame brėžinyje, pilku fonu pažymėtos tos intelektualaus komponento sudedamosios dalys, kurių kūrimui ir realizaviui šiame darbe bus skirtas didžiausias dėmesys (t.y. pedagoginis modelis ir vartotojo modelis). Kaip matome iš 2 pav., domeno modelis taip pat priklauso intelektualiajam e-mokymosi sistemos komponentui, tačiau šio domeno realizavimui aš naudosiu prof. A. A. Bielskio siūlomą modelį [3], kuris bus aprašytas toliau šiame darbe.

2.2 Intelektualaus komponento kūrimas naudojant Q-learning algoritmą

Šis algoritmas leidžia mokytis tiesiai iš patirties (klaidos ir paskatinimo metodu) su kitais studentais su panašiom mokymosi charakteristikom [1]. Jis yra pagrįstas vertės – veiksmo funkcija Q(s,a), kad aprašyti veiksmų taisykles [13, 17]. Viena iš pagrindinių šio algoritmo charakteristikų - jis nereikalauja, kad sistema vykdytų optimalią konvergencijos seką. Šis algoritmas turi galimybę mokyti Q funkciją pagal optimalias taisykles [17]. Tai labai svarbu e-mokymosi sistemai, turint omenyje, kad mokymasis yra pagrįstas interaktyviai bendrauti su vartotojais.

$$Q(s,a) = (1-\alpha) * Q(s,a) + \alpha * (r + \gamma * \max_{a'} Q(s',a')),$$

čia Q(s,a) – yra veiksmo-įvertinimo funkcija;

s – būsena (state);

a – veiksmas (actions);

α – mokymosi greičio koeficientas;

γ – pasitikėjimo būsimais įvertinimais koeficientas;

s' – ateities būsena;

a' – veiksmas ateities būsenoje, turintis didžiausią Q įvertinimą.

r – paskatinimas (reward) - tai nulinė arba vienetinė reikšmė. Jei veiksmas (action) buvo sėkmingas r = 1, jei nesėkmingas r = 0. E. patarėjo atveju žingsnis sėkmingas skaitomas tada kai tema įsisavinta (testas išspręstas teisingai) ir studentas gali pereiti prie sekančios temos [1].

$max_{a'}$ – maksimali sekančios būsenos bet kurio veiksmo reikšmė.

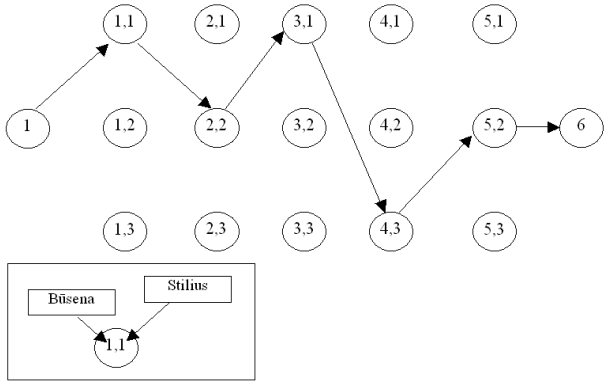
Q(s', a') – ateities būsenos Q reikšmė prie geriausio ateities veiksmo.

Vykstant sistemos apsimokymui visi duomenys, reikalingi Q-learning algoritmui yra saugomi tokios lentelės pavidale:

1 lentelė. ***Q reikšmių lentelė***

	Stilius 1	**Stilius 2**	**Stilius 3**	**…**
Tema1	0,87	0,45	0,63	...
Tema2	0,71	0,92	0,69	...
Tema3	0,835	0	0,743	...
Tema4	0,579	0,68	0,812	...
Tema5	0,45	0,943	0,87	...
...	...	...	...	...

Pagal 1-os lentelės turimus Q-reikšmių duomenis, optimalų mokymosi kelią naujam besimokančiajam būtų galima pavaizduoti taip:



3 pav. ***Optimalaus mokymosi kelio atvaizdavimas pagal Q-lentelės duomenis[1]***

Iš 3. pav. galime pastebėti kaip yra vykdomas optimalaus kelio radimas: kiekvienos temos mokymuisi yra siūlomas tas mokymosi stilius, kurio Q reikšmė tai temai yra didžiausia. Jeigu siūlomu stiliumi besimokinančiam nepavyksta išmokti temos, tai yra siūlomas kitas mokymosi stilius, tuo pat metu atitinkamai tai pažymint Q-reikšmių lentelėje. Šio algoritmo pagrindiniai privalumai yra tai, kad jis gana greitai apsimoko (t.y. greitai konverguoja) ir nereikalauja daug resursų skaičiavimams atlikti (duomenims saugoti užtenka TEMŲ SK. x STILIŲ SK. matmenų lentelės), tačiau yra ir trūkumų – jis apie tai, kokiu stiliumi pateikti mokymosi medžiagą sprendžia iš to kaip sekėsi mokytis atitinkamą temą visiems mokiniams (t.y. sprendžiama bandrai pagal visus o ne pagal individualius rezultatus), ko pasekoje yra didesnė tikimybė, kad naujam mokiniui tema bus pateikta ne tuo stiliumi, kuris jam labiau tiktų. Mano manymu, sistema galėtų priimti žymiai tikslesnius sprendimus vartotojo atžvilgiu, jeigu ji gebėtų klasifikuoti duomenis – t.y. besimokančius vartotojus pagal panašumo požymius grupuotų į tam tikras grupes ir sektų kaip konkrečiai grupei sekasi mokytis atitinkamą temą. Tokiu atveju mums gali padėti grupių modeliavimas [26].

Tam kad geriau suprasti grupių modeliavimą toliau aš panagrinėsiu keletą dažniausiai praktikoje naudojamų duomenų klasifikavimo (grupavimo) metodų.

2.3 Dažniausiai naudojami duomenų klasifikavimo metodai

2.3.1 Skaidymas į klases (*angl.* clustering)

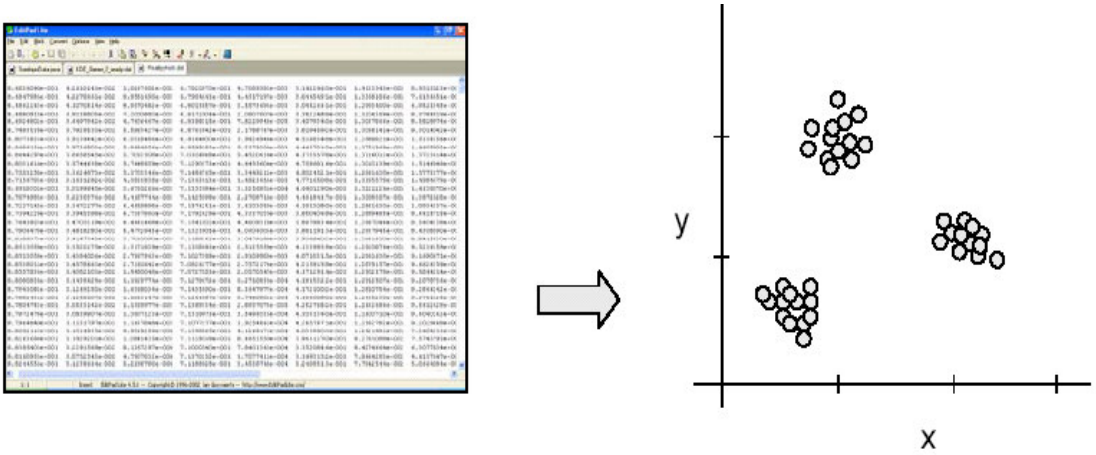
Daugelyje praktikoje pasitaikančių situacijų mes turime daugybę daugiadimensinių duomenų aibių, kurios iš pirmo žvilgsnio mums nieko naudingo nepasako. Peržiūrėti kiekvieną tokių duomenų vektorių ir jame ižvelgti tam tikras savybes, praktiškai yra neįmanoma. Tačiau toks didelis duomenų kiekis gali įgauti prasmę kuomet mes tuos duomenis pradedame skirsyti į grupes [5, 9]. Skaidymas į klases yra vienas iš būdų iš nesutvarkytų duomenų išgauti naudingą informaciją, juos suskaidant į grupes pagal jų panašumą vienas kitam. Dažniausiai prieš pradėdami tokius duomenis skaidyti į klases mes net nežinome kokios klasės bus gautos. Jeigu tokiems suskaidytiems duomenims mes pritaikytume vizulaizavimo priemones, tai galėtume nesunkiai susidaryti pirmą įspūdį apie turimus duomenis ir jų pasiskirstymą vienas kito atžvilgiu (4 pav.).

Yra skiriami du pagrindiniai skirstymo į klases metodai [34]:

- padalijimas
- hierachinis skirstymas

Pirmojo metodo metu yra siekiama vieną didelį duomenų masyvą, susidedantį iš n objektų, suskaidyti į k mažų klasterių aibę. Paprastai yra užduodamas padalijimo kriterijus bei klasteriai k, o mūsų tikslas yra rasti tuos klasterius iš k, kurie optimizuoja užduotą kriterijų. Čia padalijimo

kriterijus yra yra suprantamas kaip tikslo funkcija (kitaip, “kainos funkcija”) kuri turėtų būti kiek įmanoma mažesnė [35]. Skaidymo į klasterius padalijimo metodui priklauso K-vidurkių (K-Means) metodas.



4 pav. ***Duomenų suskirstymas į klases ir jų vizualizavimas***

Hierarchinio skirstymo metu, duomenų klasteriai gali būti gaunami skilimo (“iš viršaus - žemyn”) arba sukaupimo (“iš apačios - viršun”) būdu [34]. Pastaruoju būdu mes pradedame nuo to, kad kiekvienas net mažiausias duomenų elementas yra traktuojamas kaip klasteris. Kiekvieno žingsnio metu yra ieškoma panašiausių tokių klasterių (tiksliau tariant ieškoma dviejų panašiausių klasterių) ir radus juos jie tarpusavyje yra apjungiami. Tokiu būdu kas kart yra suformuojami vis didesni ir didesni klasteriai. Visa procedūra yra baigiama tuomet kai lieka tik vienas klasteris. Skilimo metodas priešingai – pradeda nuo to kad pradinis klasteris (yra tik vienas) apima visus duomanų elementus ir kiekvieno žingsnio metu, mažiausiai su pradiniu klasteriu susijęs elementas yra atskiriamas nuo klasterio (pradinis klasteris suskaidomas į du klasterius) [34]. Pastaroji procedūra gali būti kartojama tol, kol klasterių bus tiek kiek elemetų yra duomenyse. Kaip sukaupimo metodo realizaciją būtų galima paminėti AGNES (Agglomerative Nesting) algoritmą.

2.3.2 Skaidymas K-vidurkių metodu

K-vidurkių algoritmą sukūrė MacQueen, 1967 m. Tai bene plačiausiai paplitęs algoritmas skirtas duomenų suskirstymui į klases. Šio algoritmo veikimo principas yra paremtas tuo, kad klasteris yra formuojamas apie tam tikrą centrinį tašką [29]. Pastarasis taškas yra tas, kurio atstumas iki kitų to paties klasterio elementų yra mažiausias. Dažnai centrinis klasterio taškas yra apibrėžiamas kaip klasterio elemetų aritmetinis vidurkis. Šiame algoritme tikslo funkcija yra apibrėžiama taip [32]:

$$E = \sum_{k} \left\| x_k - m_{c(x_k)} \right\|^2$$

t.y. atstumų tarp kiekvieno klasterio elemto x_k ir tą klasterį "atstovaujančio" elemento $m_{c(x_k)}$.

Geriausias skaidymas į klasterius yra pasiekiamas kada ši klaida yra kuo mažesnė [29].

Pati algoritmo seka susideda iš tokių etapų: pasirenkamas pageidaujamas kiekis klasterių *k,* ir kiekvienam jų iš duomenų aibės atsitiktinai yra parenkami pradiniai taškai (tai yra tie patys centriniai taškai). Toliau kiekvieną likusių duomenų elementą mes priskiriame tam klasteriui, kuriam jis yra tinkamiausias (paskaičiuojama pagal atstumą iki kiekvieno klasterio centrinio taško). Po kiekvieno tokio paskirstymo, kiekvieno klasterio centriniai taškai yra perskaičiuojami skaičiuojant atitinkamo klasterio elementų vidurkį. Toks duomenų priskyrimas atitinkamam klasteriui ir po to sekantis centrinių taškų perskaičiavimas yra tęsiamas tol, kol nebeįvyksta jokių pokyčių duomenų skirstyme. Kitaip tariant, algoritmas baigiamas, kai visi duomenų masyvo elementai yra priskirti tam klastriui, kurio centrinis taškas yra jam artimiausias [32]. Kuomet algoritmas pasiekia būseną kada centrinių taškų perskaičiavimas jau nebeduoda geresnių rezultatų negu tie kuriuos jau pasiekėme, tuomet yra laikoma kad duomenų skirstymas konvergavo.

Kalbant apie šio algoritmo privalumus reikia pastebėti, kad k-vidurkių algoritmas yra labai veiksmingas. Jo efektyvumas pgrinde priklauso nuo šių parametrų: duomenų masyvo elementų skaičiaus (*n*) , pasirinklų klasterių skaičiaus (*k*) ir iteracijų skaičiaus (*t*) (t.y. kiek kartų mes perskirstėme visus masyvo elemtus) [35]. Priklausomybė tarp šių parametrų yra tokia, kad kuomet mes turime didelį diekį duomenų *n,* kiti du parametrai paprastai būna maži.

Tačiau šis algoritmas turi ir savų trūkumų. Visų pirma reikia pastebėti, kad geriausiai tinkama klasterių reikšmė *k* iš anksto nėra žinoma (mes nežinome iš pradžių į kiek klasterių geriausia būtų skirstyti duomenis). Siekiant tai sužinoti yra būtina atlikti "bandymas-klaida" (*angl.* trial and error) veiksmus. Antras trūkumas – algoritmas yra labai jautrus pradiniam centrinių taškų pasirinkimui. Blogas šių taškų pasirinkimas gali neleisti mums pasiekti norimo rezultato [29]. Kitas trūkumas yra tas, kad šis algoritmas nėra pakankamai stiprus aplinkinių taškų (esančių toliau nuo klasterio centrinio taško) atžvilgiu. Tokie užribio taškai gali priversti klasterio centrinį tašką išeiti už pačio pagrininio klasterio ribų, o jeigu tas užribio taškas būtų pasirinktas kaip pradinis taškas, tai privestų prie to, kad daugiau nei vienas elementas nebūtų priskirtas pastarąjam klasteriui (būtų sukurtas izoliuotas klasteris [39]).

2.3.3 Hierarchinis skaidymas daudojant AGNES metodą

Šis metodas aprašo duomenų hierarchinį skirstymą "iš apačios į viršų" [5]. Šio algoritmo seka susideda iš tokių veiksmų: tariame kad duomenų masyvas susideda iš *n* elementų. Tada mes

inicijuojame tiek pat *n* klasterių (t.y. kiekvienam elementui po atskirą klasterį). Toliau seka trikampės atstumų matricos, parodančios atstumus tarp klasterių, skaičiavimas (5 pav.).

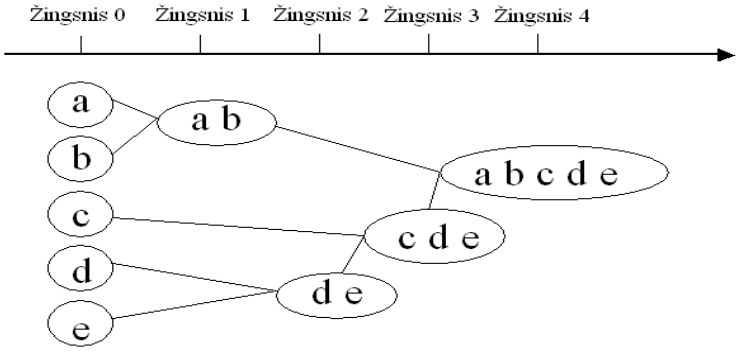
x_2	$d(x_1, x_2)$		
x_3	$d(x_1, x_3)$	$d(x_2, x_3)$	
x_4	$d(x_1, x_4)$	$d(x_2, x_4)$	$d(x_3, x_4)$
	x_1	x_2	x_3

5 pav. ***Atstumų matrica duomenų aibei iš keturių elementų***

Sekančiame žingsnyje nustatome ir pasirenkame tuos klasterius x_i ir x_j, tarp kurių atstumas $d(x_i, x_j)$ būtų mažiausias. Pasirinkti klasteriai x_i ir x_j yra apjungiami tarpusavyje (t.y. gaunamas naujas klasteris x_{ij}), o atstumų matrica yra atnaujinama iš jos išmetant x_i ir x_j stulpelius ir įterpiant naują eilutę klasteriui x_{ij}. Naujai suformuoto klasterio atstumų įverits gali būti paskaičiuotas pagal tokias formules [34]:

- $d_{vieneinis}(x_{ij}, x_k) = \min(d(x_i, x_k), d(x_j, x_k))$
- $d_{pilnutinis}(x_{ij}, x_k) = \max(d(x_i, x_k), d(x_j, x_k))$
- $d_{vidutinis}(x_{ij}, x_k) = \frac{1}{2}(d(x_i, x_k), d(x_j, x_k))$

Algoritmas yra baigiamas kuomet visi duomenų masyvo elementai yra apjungti į vieną bendrą klasterį. Rezultate mes gauname hierarchinį sujungimų medį (3 pav.), kurio kiekviena šaka atvaizduoja sujungimo operacijas (kokie klasteriau buvo sujungti). Tam kad gautume *k* klasterių, mes turime "nukirpti" medį ties k-tąja šaka žemiau šakninio elemento.



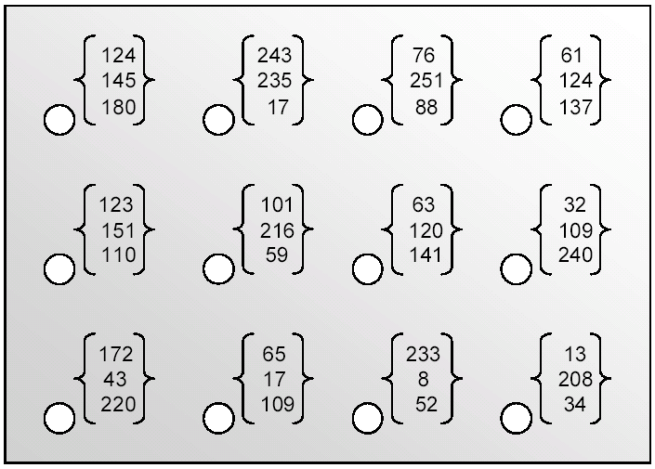
5 pav. ***Duomenų skirstymas į klasterius naudojant AGNES metodą***

Pagrindinis šio algoritmo trūkumas yra tas, kad atstumų matricos perskaičiavimas reikalauja nemažai resursų ir tai, kad mes nebegalime atšaukti prieš tai padarytos apjungimo operacijos. Kaip privalumą būtų galima paminėti kad šis algoritmas yra gana stabilus (t.y. toleruoja nedidelius duomenų masyvo pakeitimus)[5].

2.4 Savaime susitvarkančio žemėlapio (SOM) panaudojimo analizė

Savaime susitvarkantis žemėlapis buvo išrastas vieno Suomijos adademijos profesoriaus - Teuvo Kohonen, kaip priemonė padedanti daugiadimensinius duomenis pavaizduoti mažesnių dimensijų (paprastai vienos ar dviejų) erdvėse [21]. Šis procesas, sumažinantis vektoriaus dimensiją, yra duomenų suspaudimo metodika, žinoma kaip vektoriaus kvantavimas [2, 33]. Šalia to, Kohoneno sukurta technika, papildomai sukuria tinklą, kuris informaciją saugo tokiu būdu, kad yra išlaikomi visi topologiniai ryšiai tarp turimų (analizuojamų) duomenų.

Savaime susitvarkantis žemėlapis priklauso nevadovaujamo (*angl.* unsupervised) ir konkurentingo mokymosi algoritmų klasei [34]. Pats žemėlapis yra suprantamas kaip neuroninis tinklas susidedantis iš daugybės viršūnių (*angl.* node's) kurios yra išsidėsčiusios taip, kad sudaro dviejų dimensijų tinklelį. Kalbant apie įprastinius neuronimius tinklus, tinklo viršūnių sujungimai yra asocijuojami su svorių vektoriumi, o SOM atveju, svorių vektoriais būtų galima laikyti kiekvieną iš tinklą sudarančių viršūnių. 6 pav. yra pateiktas supaprastintas 4 x 3 dydžio SOM tinklo, susidedančio iš vektorių, kurių dimensija lygi 3, pavyzdys:

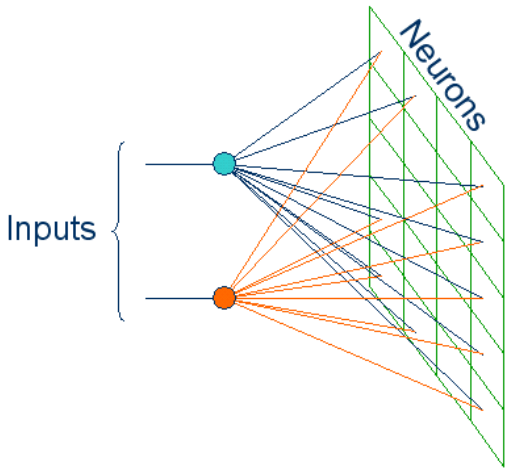


6 pav. ***4 x 3 Som tinklo pavyzdys [34]***

Visi įvedimo duomenys, kurie yra perduodami SOM žemėlapiui, yra pateikiami vektoriaus pavidalu. Tai reiškia, kad jeigu įėjimo duomenų dimensija yra *n* dydžio, tai kiekviena SOM tinklo viršūnė saugos n-dimensijos dydžio svorių vektorių [21]:

$$m_i = \left[m_{i1}, m_{i2,} m_{i3}, ..., m_{in}\right]$$

Kiekvienas įėjimo vektorius vektorius yra tiesiogiai sujungtas su kiekviena SOM žemėlapio viršūne (7 pav.), o pačio žemėlapio neuronai tarpusavyje yra sujungti kaimyniniais ryšiais.



7 pav. ***SOM tinklas su įėjimo vektoriais***

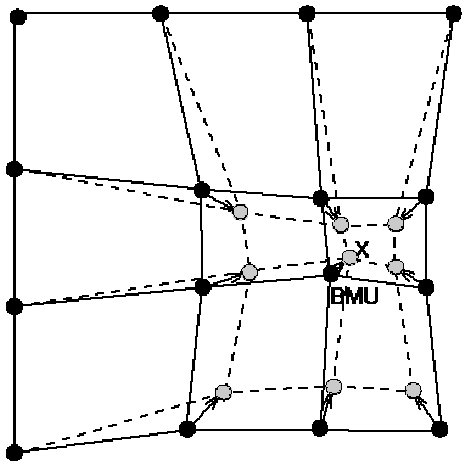
Pagrindinis savaime susitvarkančio žemėlapio veikimo principas yra susijęs su žemėlapį sudarančių svorių vektorių pritaikymu (pakoregavimu) prie įvedimo duomenų vektoriaus [11, 12, 15]. Toks žemėlapis mums tarsi pateikia įvedimo duomenų "paveikslą". Kadangi žemėlapį sudarančių viršūnių skaičius dažniausiai būna mažesnis už įvedimo duomenų masyvo elementų skaičių, tai praktiškai tampa neįmanoma atvaizduoti kiekvieną atskirą įvedimo vektorių ant žemėlapio. Tačiau tai ir nėra šio algoritmo tikslas. Pagrindinis siekis yra taip pakoreguoti žemėlapio viršūnių vektorius, kad žemėlapis atspindėtų įvedimo duomenų struktūrą (pasiskirstymą) ir išsaugotų svarbiausius ryšius tarp įvedimo duomenų [23,34]. Čia mus labiausiai domina sužinoti koreliaciją tarp įvedimo duomenų ir atstumus tarp labiausiai nepanašių jų vienetų (t.y. labiausiai panašūs ivedimo vektoriai turėtų būti lokalizuoti ant savaime susitvarkančio žemėlapio arti vienas kito).

2.4.1 Savaime susitvarkančio žemėlapio algoritmas

Savaime susitvarkančio žemėlapio apmokymas yra iteratyvus procesas [21]. Kiekvienos iteracijos metu iš mokymosi duomenų masyvo yra pasirenkamas vienas elementas x (n dimensijos vektorius). Toliau mokymas yra vykdomas konkuravimo būdu, t.y. pagal pasirinktą mokymosi duomenų elementą mes ant žemėlapio nustatome laimėjusią viršūnę c, dažnai dar vadinamą labiausiai tinkančiu elementu (*angl.* Best Matching Unit BMU). Laimėjusia žemėlapio viršūne yra laikoma ta, kurios saugomas vektorius m_c yra panašiausias į mokymosi duomenų pasirinktą vektorių x.

$$\|x - m_c\| = \min_i \|x - m_i\|$$

Nustačius laimėjusią viršūnę, jos svorių vektoriaus elementų reikšmės yra pritaikomos prie įėjimo vektoriaus elementų reikšmių (8 pav.). Tokiu būdu laimėjusi viršūnė m_c yra padaroma panašesnė į įėjimo vektorių x.



8 pav. ***BMU taško ir jo kaimyninių vektorių priderinimas prie įėjimo vektoriaus***

Šiame algoritme yra praplečiamos standartinės konkurencinio mokymosi procedūros papildomai keičiant (adaptuojant prie įėjimo vektoriaus) ir kaimyninių žemėlapio neuronų, esančių šalia laimėjusio neurono (BMU), vektorių reikšmes [23, 37, 41]. Pastarosios reikšmės yra adaptuojamos prie įėjimo vektoriaus naudojant skirtingą adaptacijos lygį, kurį būtų galima aprašyti tokia lygybe [21]:

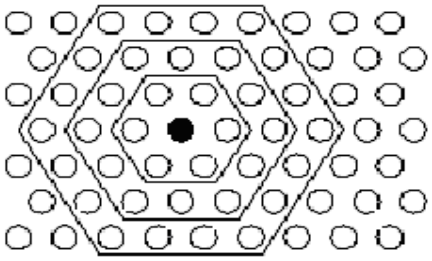
$$m_i(t+1) := m_i(t) + h_{ci}(t) * (x - m_i(t))$$

Šioje formulėje esantis parametras $h_{ci}(t)$ yra vadinamas "kaimynine funkcija". Tai yra Gaussian kreivė, kuri mažėja einant nuo kaimyninių viršūnių centro link tolimesnių kaimyninių viršūnių taškų. Kaimyninė funkcija yra skaičiuojama naudojantis formule [21]:

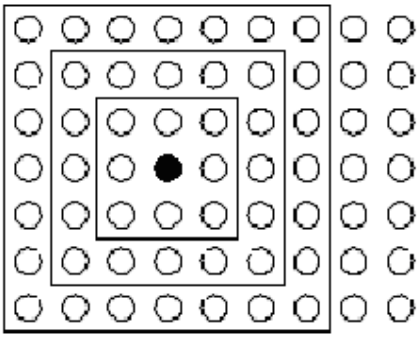
$$h_{ci}(t) = \alpha(t) * \exp\left(-\frac{\|rc - ri\|^2}{2o(t)^2}\right)$$

Kaimyninėje formulėje esantis parametras $\alpha(t)$ yra vadinamas mokymosi lygiu (*angl.* learning rate). Jis gali būti traktuojamas kaip kaimynų branduolio viršūnė (pagrindinis taškas) arba aukščiausias taškas. $o(t)$ - tai yra kaimynų branduoluo plotis (radiusas). Būtent jis nurodo plotą,

kuriame esančių žemėlapio viršūnių svorių vektoriai bus įtakojami. Abu šie parametrai, tiek kaimyninės funkcijos aukštis ir plotis (radiusas), laikui bėgant (vykstant iteracijoms) mažėja monotoniškai. Kad būtų aiškiau, 9 ir 10 pav. yra pateikta kaip aplinkiniai BMU neuronai gali būti traktuojami kaimyniniais. Abiejuose paveikslėliuose yra pavaizduoti tik trys kaimynų lygiai (kiekvienas jų išskirtas atskiru rėmeliu).



9 pav. ***Kaimyninių neuronų parinkimas naudojant šešiakampį rėmelį***



10 pav. ***Kaimyninių neuronų parinkimas naudojant keturkampį rėmelį***

Iš $h_{ci}(t)$ lygybės (žr. aukščiau) galime pastebėti tai, kad tos žemėlapio viršūnės, kurios yra arčiau laimėjusios, bus įtakojamos labiau, o tos kurios yra toliau – įtaka bus mažesnė. Mokymosi proceso pradžioje, BMU vektorius bus modifikuojamas labai stipriai, o kaimyninių viršūnių radiusas bus labai didelis. Tačiau mokymosi procesui einant į pabaigą, įtaka laimėjusiai viršunei ir šalia jos esančiom kaimyninėm viršūnėm, bus labai nedidelė, o kaimynų radiusas savo ruoštu apims praktiškai tik pačią BMU viršūnę. Toks algoritmo veikimas atitiktų "grubų rūšiavimą" žemėlapio mokymosi pradžioje ir "tikslų sureguliavimą" mokymosi pabaigoje [22].

2.4.2 SOM savybė - topologijos išsaugojimas

Kadangi apmokant SOM prie įėjimo vektoriaus yra pritaikomi ne tik BMU elementai, bet ir šalia jo esančios žemėlapio viršūnės, tai atsiranda didesnė tikimybė, kad eteityje į žemėlapio įėjimą paduotas panašus vektorius pateks būtent į šią sritį. Viso to pasekoje mes gausime tai, kad labiausiai

panašūs įėjimo duomenys bus sugrupuoti arti vienas kito [34, 41]. Mokymosi proceso metu tokiu būdu yra gaunamas vaizdas (pasinaudojant specialiomis priemonėmis tokią informaciją nesunkiai galime vizualizuoti) apie turimus įvedimo duomenis. Savaime susitvarkantys žemėlapiai pasižymi dar ir tuo, kad jie gali parodyti ne tik įvedimo duomenų panašumus ar skirtumus, bet ir klasterių, kuriems šie elementai priklauso, bendrumo ryšius, t.y. ant žemėlapio viena šalia kitos išryškėjusios įvedimo duomenų grupės tarpusavyje gali būti labiau panašios negu tos, kurios ant žemėlapio yra nutolusios toliau viena nuo kitos.

Nors galutiniame rezultate mes gauname žemėlapį, kuriame panašūs įėjimo duomenys lokalizuojasi atitinkamose žemėlapio srityse, vis dėlto savaime susitvarkančio žemėlapio algoritmas nėra laikomas skaidymo į klasterius (*angl.* clustering) algoritmu [21, 22]. Visų pirma, jis mums pasitarnauja supaprastinant daugiadimensinius duomenis (juos mes atvaizduojame dviejų dimensijų žemėlapyje), bei vizualizuojant informaciją (t.y. vizualizuojant panašių duomenų grupes). Tačiau šio metodo pagalba ms negalime turimus duomenis tiksliai suskirstyti į pageidaujamą klasterių kiekį. Toks žemėlapis neparodo tikslių klasterio ribų, neturi akivaizdaus centrinio taško (t.y. centroido).

2.4.3 Savaime susitvarkančio žemėlapio parametrų pasirinkimas

SOM algoritmo vykdymo metu gautas galutinis žemėlapis labai priklauso nuo pasirinktų jo mokymo parametrų. Pagrindiniai SOM parametrai yra šie [21]:

- Žemėlapio plotis ir aukštis;
- Mokymosi iteracijų skaičius;
- Pradinio kaimynų radiuso dydis (kiek kaimynų apie BMU mes įtakosime pradėję žemėlapio apmokymą);
- Mokymosi lygio pradinė reikšmė.

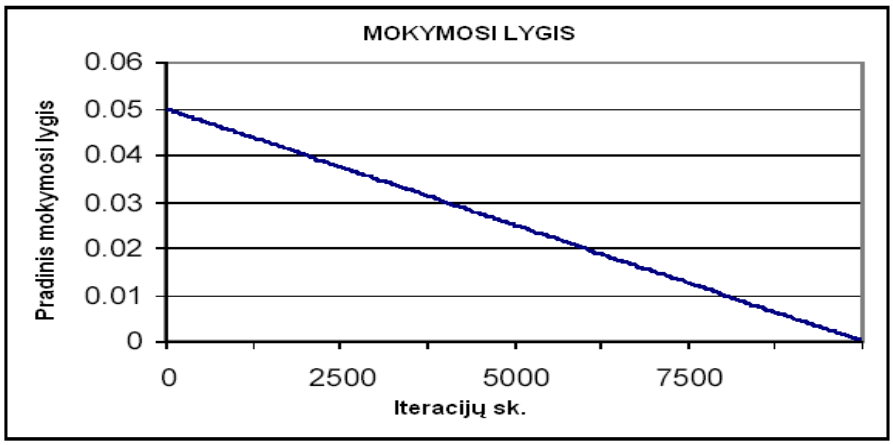
Literatūroje nėra griežtų nurodymų pasirenkant šiuos parametrus. Dažniausiai šias reikšmes nustatome bandymų metu (pasirenkame tam tikras reikšmes ir tikriname kaip mūsų žemėlapis mokosi). Pats metodo autorius, Kohonen, siūlo naudoti stačiakampio (bet ne kvadrato) formos žemėlapius, kurių dydis galėtų būti tarkim 15 x 10, o pradinį kaimynų radiusą rekomenduojama imti žemėlapio aukštį (šiuo atveju 10). Daugumoje darbų [14, 23], pradinis radiusas yra skaičiuojamas pagal formulę:

$$radius = \frac{\max(plotis, aukštis)}{2};$$

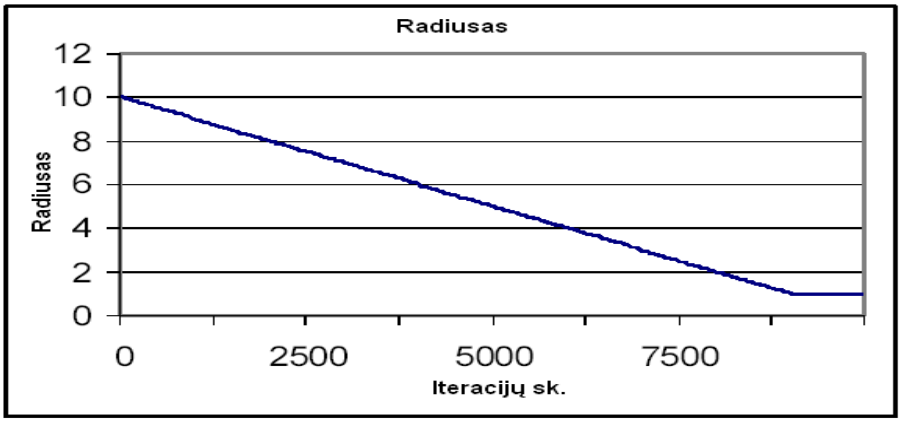
Pasirenkant pradinį mokymosi lygį, Kohonen pataria imti 0,05.

Toliau yra pateikiami mokymosi lygio ir kaimynų radiuso kitimo grafikai, pasirinkus tokias parametrų reikšmes:

- Iteraciju skaičius = 10000
- Pradinis radiusas = 10
- Pradinis mokymosi lygis = 0,05



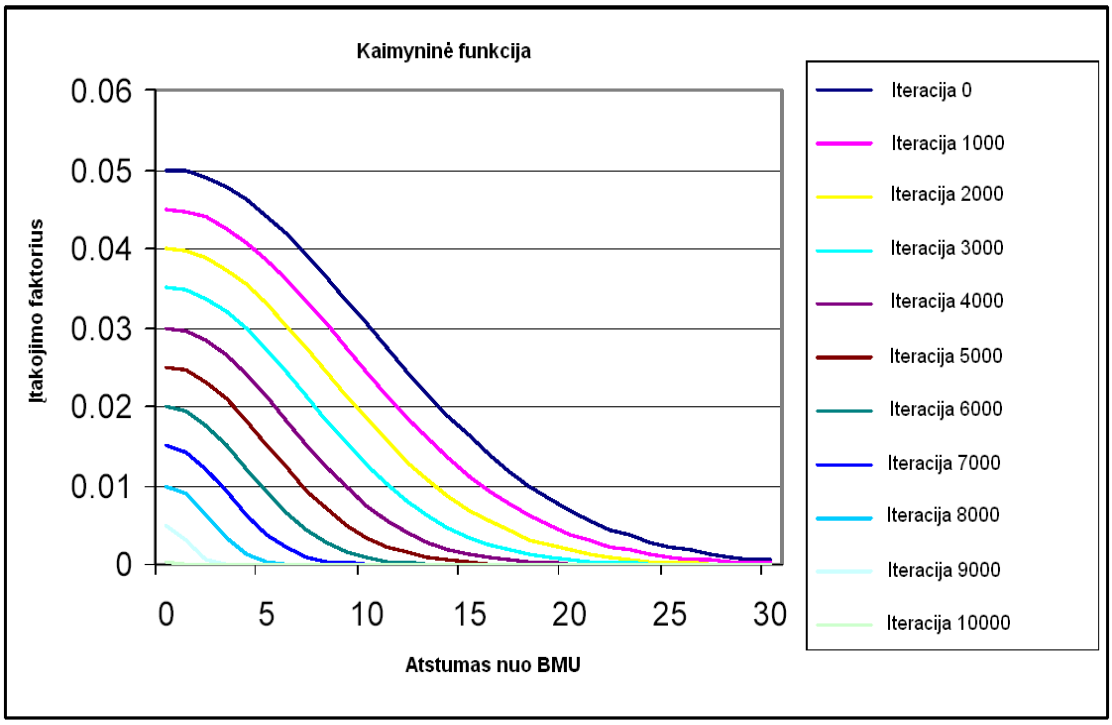
11 pav. ***Mokymosi lygio kitimas didėjant iteracijų skaičiui***



12 pav. ***Kaimynų radiuso kitimas didėjant iteracijų skaičiui***

Iš aukščiau pateiktų grafikų galima pamatyti, kad mokymosi lygis monotoniškai mažėja iki nulio (tuomet laikoma, kad žemėlapis jau apsimokino ir tolau nebesimokins), o kaimynų radiusas mažėja tik iki 1 (jis tampa lygus pašiam BMU, t.y. ne viena kaimyninė viršūnė nebėra įtakojama BMU).

Kaip matome iš žemiau pateikto 13 pav. kaimyninės funkcijos kitimas tolstant nuo BMU nėra tiesinis, bet monotoniškai mažėja pagal Gaussian kreivę (kuo toliau žemėlapio viršunė nutolusi nuo BMU, tuo mažiau ji bus įtakojama įvedimo vektoriaus). Tame pačiame grafike pavaizduota kaip įtakos faktorius ir atstumas nuo BMU priklauso nuo iteracijų skaičiaus: iteracijų skaičiui didėjant abu šie parametrai mažėja (tačiau kinta pagal tą pačią kreivę):



13 pav. ***Kaimyninės funkcijos priklausomybė nuo iteracijų skaičiaus ir atstumo iki BMU***

Pasirenkant mokymosi iteracijų skaičių yra labai svarbu atsižvelgti į turimų mokymosi duomenų masyvo dydį [2, 15, 21]. Norėdami pasiekti žemėlapio statistinį teisingumą, iteracijų skaičių turėtume paimti žymiai didesnį už turimų duomenų kiekį.

Pasirinkus visus anksčiau aprašytus parametrus, dar yra labai svarbu pasirinkti kokiu metodu bus įvertinamas įvedimo vektoriaus panašumas į žemėlapio viršunių vektorius (t.y. kaip mes ieškosime BMU). Dauguma autorių savo darbuose naudoja Manhattano arba Euklidinio atstumo skaičiavimo formules. Pirmoji jų yra užrašoma tokia lygybe [12]:

$$d(x_i, x_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{in} - x_{jn}|$$

Euklidinis atstumas yra skaičiuojamas pagal sekančią formulę:

$$d(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{in} - x_{jn})^2}$$

Kiek rečiau pasitaiko atvejų, kuomet yra naudojamas šių abiejų metodų apibendinimas – Minkowsky atstumas [34]: $d(x_i, x_j) = \left(\sum_k (x_{ik} - x_{jk})^r \right)^{\frac{1}{r}}$

2.4.4 SOM kokybės įvertinimas

Paprastai duomenų skaidymo į klasterius rezultato gerumas gali būti įvertinamas šiais matais:

- Kokybiniais – įvertinama kokiu mąstu pavyko atskleisti paslėptas duomenų struktūras. Tai labiausiai priklauso nuo klasterių apibrėžimo ir jų atvaizdavimo, o taip pat ir nuo įvertintojo įgūdžių, bei patirties. Tokie įvertinimai praktiškai negali būti atlikti neturint pradinės informacijos apie tai, kokio pobūdžio informacija buvo pateikta apmokant žemėlapį [18]. Beto, kokybiniai įvertinimai nėra tokie tikslūs ir objektyvūs kaip kiekybiniai įvertinimai.
- Kiekybiniais – įvertinama kuomet mes nuturime jokios pradinės informacijos apie turimus duomenis (t.y. visai nežinime ką jie galėtų reikšti). Yra laikoma, kad klasteriavimo metodas gavo gerus rezultatus tuo atveju, jeigu gautus klasterius sudarantys elementai tarpusavyje yra labai panašūs, o lyginant tarp skirtingų klasterių – labai skirtingi [18].

Kiekybinį klasteriavimo įvertinimą smulkiau nagrinėjo Davies ir Bouldin (1979) [34]. Jie kiekybinį duomenų klasteriavimo įvertinimą siūlė skaičiuoti taip: tarkime C – duomenų klasterių skaičius; $S_C(Q_k)$ - klasterio Q_k vidutinis atstumas, apibrėžiamas kaip suma visų atstumų tarp visų klasterio elementų x_i ir klasterio centroido c_k padalinta iš klasterio elementų skaičiaus:

$$S_C = \frac{\sum_i \|x_i - c_k\|}{N_k};$$

Išorinis klasterio atstumas $d_{ce}(Q_k, Q_l)$ tarp klasteriu Q_k ir Q_l yra apibrėžiamas kaip atstumas tarp šių klastrių centroidų:

$$d_{ce} = d_{centroid} = \|c_k - c_l\|$$

Dabar galime užrašyti David-Bouldin indeksą, kuris ir nusako gauto klasteriavimo rezultato kiekybinį įvertinimą [34]:

$$\frac{1}{C}\sum_{k=1}^{c} \max_{t \neq k}\left(\frac{S_C(Q_k) + S_C(Q_t)}{d_{ce}(Q_k, Q_t)}\right)$$

Norint paskaičiuoti kiekybinį įvertinimą, klasterio vidinių atstumų skaičiavimui galima naudoti ir kitus metodus (nebūtinai tą, kuris buvo pateiktas aukščiau).

Tačiau, kadangi SOM nėra klasteriavimo algoritmas, tai pastarasis metodas žemėlapio gerumo kiekybiniam įvertinimui nelabai tinka (pagrinde dėl to, kad iš savaime susitvarkančio žemėlapio mes sunkiai, arba visai netikslai, galime išskirti atskirus "klasterius". Vieno vertintojo požiūriu klasterio ribos galėtų būti vienoje vietoje, kito vertintojo – kitoje, ko pasekoje gauti rezultatai nebūtų objektyvūs). Siekiant išspręsti šią problemą, Kohonen (1995) ir kiti autoriai (Kiviluoto (1996)) [45] pasiūlė įvesti vidutinę kiekybinę klaidą (*angl.* Average Quantization Error AQE) ir topografinę klaidą.

Vidutinė kiekybinė klaida yra apibrėžiama kaip vidutinis atstumas tarp kiekvieno duomenų elemento iš duomenų masyvo ir BMU elemento ant žemėlapio. Tai gali būti traktuojama kaip žemėlapio susiskaidymo (*angl.* resolution) įvertis:

$$AQE = \frac{1}{N}\sum_{N}\left\|x - m_{c(x)}\right\|$$

Pačio žemėlapio "gerumo" įvertį būtų galima apskaičiuoti pagal tokią formulę:

$$\frac{1}{1+AQE}$$

Norėdami įvertinti savaime susitvrakančio žemėlapio topologijos išsaugojimą, mes galime naudoti topologijos klaidą, kuri yra apibrėžiama, kaip procentinė dalis tokių duomenų elementų, kuriems pirmasis BMU ir antrasis BMU elementai nėra gretutiniai (*angl.* adjacent) žemėlapio elementai (t.y. nėra vienas šalia kito).

Yra būtina paminėti tai, kad žemi šių dviejų klaidų rodikliai dar negarantuoja gero (atskleidžiančio tikrąsias duomenų savybes) žemėlapio. Beto, šias klaidas galima minimizuoti atitinkmai didinant žemėlapio dydį, tačiau tai ne visada yra veiksminga.

2.4.5 SOM tikslo funkcija ir jos panašumas į K-metodo tikslo funkciją

Daugumoje neuroninių tinklų algoritmuose yra aprašoma tikslo funkcija, tačiau SOM atveju šioje srityje atsiranda savi niuansai. Visų pirma reiktų pastebėti, kad SOM tikslo funkcija negali būti tapatinama su SOM vidutine kiekybine klaida AQE. Tuo galima įsitikinti išanalizavus tokią situaciją: tarkim žemėlapio mokymosi pradžioje žemėlapio viršūnės (jų saugomi svorių vektoriai) yra inicializuojami atsitiktinai parenkant svorių vektorius iš mokymosi duomenų. Tokiu atveju AQE bus labai maža, kadangi mažiausiai *M* mokymosi duomenų elementų, kur *M* reiškia savaime susitverkančio žemėlapio viršūnių skaičių, ant žemėlapio turės savo BMU viršunę nutolusią nuo jų per atstumą lygų 0. Visos šios viršūnės turės klaidą lygią 0, ko pasekoje smarkiai bus sumažinta ir bendra viso žemėlapio vidutinė kiekybinė klaida [45]. Toliau įvykdžius bent vieną žemėlapio

mokymosi iteraciją, dalis žemėlapio viršūnių bus adaptuotos prie mokymui pasirinkto elemento iš mokymosi duomenų masyvo. Šio etapo metu, pasirinkto elemento kiekybinė klaida dar labiau bus sumažinta (o tiksliau – ji kaip ir anksčiau liks 0), o kitų, kaimyninių viršūnių svoriai bus pakeisti ir tokiu būdu jų kiekybinė klaida padidės (nes jų reikšmės nebeatitiks mokymosi duomenų masyve esančių elementų reikšmėms). Visa tai jau po pirmos mokymosi iteracijos smarkiai padidina *AQE* reikšmę.

Cottrell 1995 metais atlikęs nuoseklią savaime susitvarkančio tinklo matematinę analizę preiškė, kad "nepaisant plataus Kohoneno tinklų naudojimo, šis algoritmas nepasiduoda pilnam matematinės analizės atlikimui" [18]. Jo teigimu, bendram SOM atvejui nėra tiksaus konvergavimo įrodymo ar tikslo funkcijos apibrėžimo. Šiuo metu tik atskiram SOM atvejui, kuomet mokymosi duomenys yra vienos dimensijos ir žemėlapis yra vienos dimensijos masyvas, egzsituoja pilnas matematinis pagrindimas. Tariant kad mokymosi duomenys yra diskretūs ir kaimynų branduolys griežtai apibrėžtas (tiksliai fiksuotas) galime parašyti lokalią tikslo finkciją [34]:

$$E = \sum_k \sum_i h_{ci} \left\| x_k - m_i \right\|^2 ;$$

čia: h_{ci} - jau anksčiau aprašyta kaimyninė funkcija;

x_k - k – tasis elementas iš mokymosi duomenų aibės;

m_i - i – tasis kaimyninis elementas, esnatis šalia x_k elemento BMU viršūnės.

Nepaisant to, kad SOM yra neuroninis tinklas, ir jo samprotavimo koncepcija labai skiriasi nuo k-metodo veikimo [32], tačiau tarp šių dviejų metodų yra ir tam tikrų panašumų. Pagrindinis panašumas matomas tarp K-metodo klasterio centroido ir SOM tinkle naudojamo BMU taško apibrėžimų. Sutapatinę šiuos dalykus, mes SOM žemėlapį susidedantį iš *k* viršūnių, galime traktuoti kaip *k* klasterių aibę. Kiekviena žemėlapio viršūnė gali būti laikoma BMU tašku grupei elementų iš mokymosi duomenų aibės, kitaip tariant – tie elementai yra mokymosi duomenų centrodai. Taip pat panašumų galime įžvelgti ir kalbant apie tikslo funkciją: k-metodas atsižvelgia į kiekvieno duomenų elemento atstumą iki jo BMU t.y. centroido, kas yra aktualu ir SOM algoritmui. Tačiau pastarasis dar papildomai atsižvelgia ir į savo kaimyninių elementų atstumus iki centroido, kurie paprastai yra įvertinami naudojant kaimyninę funkciją. K-metodo ir SOM tikslo funkcijos sąryšis gali būti aprašytas tokia lygybe:

$$E = \sum_k \left\| x_k - n_c \right\|^2 + \sum_i \sum_j h_{ij} N_i \left\| n_i - m_j \right\|^2 ;$$

čia: $\sum_k \left\| x_k - n_c \right\|^2$ - K-metodo tikslo funkcija

n_i - i-tojo elemento BMU taškas.

N_i - kaimyninių elementų (viršūnių) skaičius apie n_i BMU tašką.

Iš pastarosios lygybės galime pastebėti, kad tuo atveju, kada kaimynų radiusas lygus 0 ir apima tiktai patį BMU tašką, tiek K-metodo, tiek ir SOM – tiklso funkcijos yra vienodos.

2.4.6 SOM žemėlapio inicijavimas

Žemėlapio inicijavimui dažniausiai naudojimas vienas iš trijų pagridinių metodų [11, 33]:

- *Naudojant atsitiktines reikšmes*, visiškai nepriklausomas nuo turimų mokymosi duomenų. Tai atitinka žemėlapį, kuris "nieko nežino" apie turimus mokymosi duomenis. Tai yra pats paprasčiausias inicijavimo metodas, tačiau jis reikalauja daugiau iteracijų, tam kad žemėlapis galėtų prisitaikyti prie turimų duomenų. Šis metodas nepasižymi efektyvumu.
- *Atsitiktinai parenkant elementus iš mokymosi duomenų masyvo.* Šio metodo privalumas tas, kad čia jau pradiniai žemėlapio turimi duomenys yra iš tos pačios duomenų erdvės kaip ir mokymosi duomenys (nebereikia eikvoti papildomų iteracijų tam, kad žemėlapis prisitaikytų prie duomenų). Kuomet žemėlapio mokymas prasideda, žemėlapis yra tokioje būsenoje, kad jau pavaizduoja bent dalį turimų mokymosi duomenų. Kadangi žemėlapio apsimokymui reikia mažiau iteracijų, tai savo ruožtu reikia atlikti mažiau sudėtingų skaičiavimo operacijų ko pasekoje sutaupoma resursų. Tačiau ir šis metodas negarantuoja optimalaus žemėlapio inicijavimo. Nors paprastai inicijuojamo žemėlapio viršūnių skaičius yra žymiai mažesnis už mokymosi duomenų skaičių, tačiau kadangi inicijavimo elementai perenkami atsitiktinai, tai gali įvykti tokia situacija, kad labai dažnai inicijavimui bus paimtas tas elementas kuris iš esmės labai skiriasi nuo daugumos mokymosi duomenų (kurie yra esminiai), ko pasekoje mes gausime žemėlapį, kuris labai mažai atitiks tikrąją situaciją. Taip pat galima ir priešinga situacija – kai mažiau pasitaikantys elementai iš vis nebus paimti inicializuojant žemėlapį.
- *Elementus iš mokymosi duomenų masyvo parenkant taip, kad jie labiau atspindėtų turimų duomenų pasiskirstymą.* Tai nėra lengvas uždavinys, nes paprastai mūsų inicijuojamas žemėlapis yra dviejų dimensijų, o mokymosi duomenys – daugiadimensiniai. Iškyla klausimas – kaip daugiadimensiniai duomenys galėtų būti pakankamai tiksliai charakterizuojami naudojant dviejų ar net vienos dimensijos duomenis? Galima bandyti pasirinkti vieną dimensiją (duomenų požymį), kuri mūsų supratimu, geriausiai charakterizuoja turimus duomenis. Tačiau kaip nustatyti tą požimį? Jeigu mūsų pasirinktas požymis dažniausiai saugo reikšmę artimą to požymio iš visų turimų duomenų vidutinei reikšmei, tai mes nesunkiai galėsime parinkti žemėlapio inicijavimo reikšmes (nes reikšmės išsibarstymas apie vidurkį labai mažas). Tačiau tokiu atveju, žemėlapio informatyvumas taip pat bus labai mažas. Informatyvus žemėlapis būtų tuo atveju, jeigu pasirinktas požymis parodytų dižiausią savo variaciją (parodytų didžiausius

nukrypimus nuo vidutinės reikšmės). Šioje situacijoje į pagalbą būtų galima pasitelkti specialią matematinę procedūrą – svarbiausio (pagrindinio) komponento analizę (*angl.* Principal Component Analysis (PCA)).

Pagrindinis PCA tikslas – sumažinti daugiadimensinių duomenų dimensiją, tuo pačiu metu išsaugant didžiąją dalį duomenų variantiškumo [38]. Čia pirmasis svarbiausias komponentas siekia išgauti kaip galima daugiau duomenų variantiškumo, o sekantys komponentai savo ruožtu siekia išgauti kiek galima daugiau iš to kas liko. Šis metodas yra paremtas tikrinių vektorių (*angl.* Eigenvectors) ir kovariacijos principu.

Pabandysime detaliau aptarti šio metodo veikimą.

Tarkime, kad A yra kvadratinė matrica. Jeigu egzistuoja vektorius $x \neq 0$ (ir $x \in R^n$) toks kad $Ax = \lambda x$ kokiam tai skaliariniam dydžiui λ, tai λ yra vadinama matricos A su tikriniu vektoriumi x, tikrine reikšme (*angl.* Eigenvalue) [42]. Čia mes galime laikyti matricą A (kurios matmenys yra $k*k$) kaip tiesinę transformaciją k-dimensinėje Euklidinėje erdvėje. Tada matricos A tikrinis vektorius mums parodo tos transformacijos kryptį, o tikrnė reikšmė – jos dydį [38].

Kovariacija tarp dviejų savybių yra suprantama kaip matas nusakantis jų sugebėjimą kisti drauge. Mūsų analizuojamame kontekste, kovariacija yra naudojama nusakyti abipusius ryšius (priklausomybes) tarp mokymosi duomenų dimensijų [42]. Kalbant matematiškai, dviejų savybių x_i ir x_j kovariacija yra apibrėžiama kaip tų savybių nuokrypio, nuo jų vidurkių, vidutinė reikšmė [42]:

$$\operatorname{cov}(x_i, x_j) = o_{ij} = E[(x_i - u_i)(x_j - u_j)] = \frac{1}{n} * \{[x_{i1} - \mu_i][x_{j1} - \mu_j] + [x_{i2} - \mu_i][x_{j2} - \mu_j] + \ldots + [x_{i2} - \mu_i][x_{jn} - \mu_j]\}$$

Jeigu savybės x_i ir x_j yra linkusios didėti kartu, tai o_{ij} bus teigiamas, o jeigu linkusios mažėti karu - o_{ij} bus neigiamas. Tuo atveju, kai x_i ir x_j yra nepriklausomi vienas nuo kito - $o_{ij} = 0$.

Toliau apibrėšime kovariacijos matricą V:

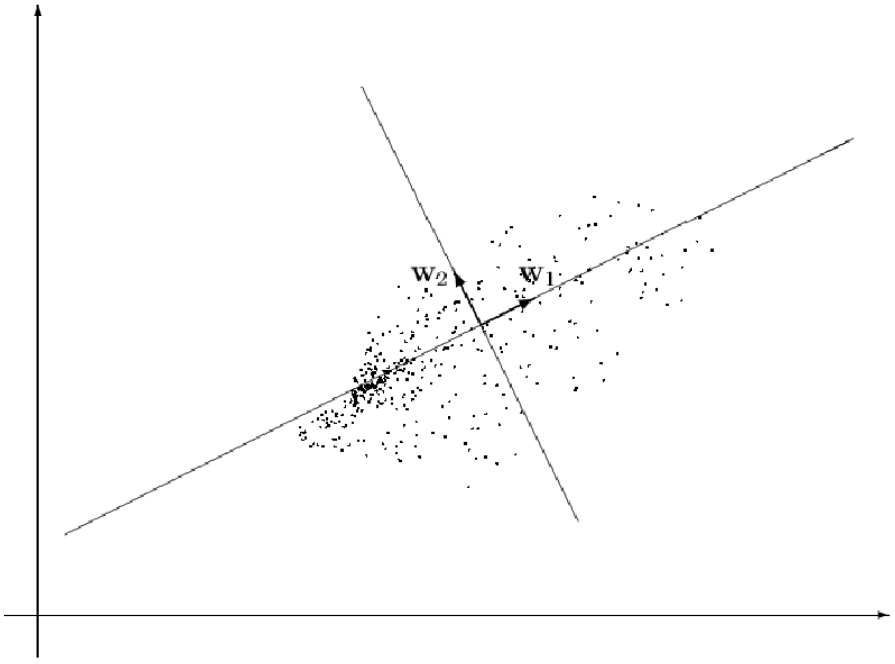
$$V_{ij} = \operatorname{cov}(x_i, x_j)$$

Jeigu mes turime k skaičių savybių, tai kovariacijos matricos dydis bus $k*k$:

$$V = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1k} \\ \sigma_{21} & \sigma_{22} & \ldots & \sigma_{2k} \\ \ldots & \ldots & \ldots & \ldots \\ \sigma_{k1} & \sigma_{k1} & & \sigma_{kk} \end{bmatrix}$$

Iš pateiktos kovariacijos matricos matyti, kad ji yra kvadratinė matrica, o kadangi $o_{ij} = o_{ji}$ tai ši matrica dar yra ir diagonalinė. Visa tai garantuoja tikrinių reikšmių egzistavimą ir tai, kad jos yra realios.

Toliau yra pateikiamas brėžinys (14 pav.), vaizduojantis duomenų charakterizavimą naudojant tikrinius vektorius. Jame pavaizduoti du tikriniai vektoriai w_1 ir w_2 parodo kryptis kuriomis yra didžiausias duomenų išsibarstymas. Kuo didesnė tikrinė reikšmė, tuo duomenų variacija apie tą tikrinį vektorių yra didesnė [38]. Čia pavaizduotas tikrinis vektorius w_1 atitinka vektorių su didesne tikrine reikšme nei vektorius w_2. Pirmasis vektorius yra vadinamas dominuojančiu tikriniu vektoriumi (*angl.* dominant eigenvector) arba pirmuoju svarbiausiu komponentu (*angl.* first principal component). Atitinkamai vektorius w_2 atitinką kitą tikrinį vektorių su didžiausia tikrine reikšme ir yra vadinamas antruoju svarbiausiu komponentu.



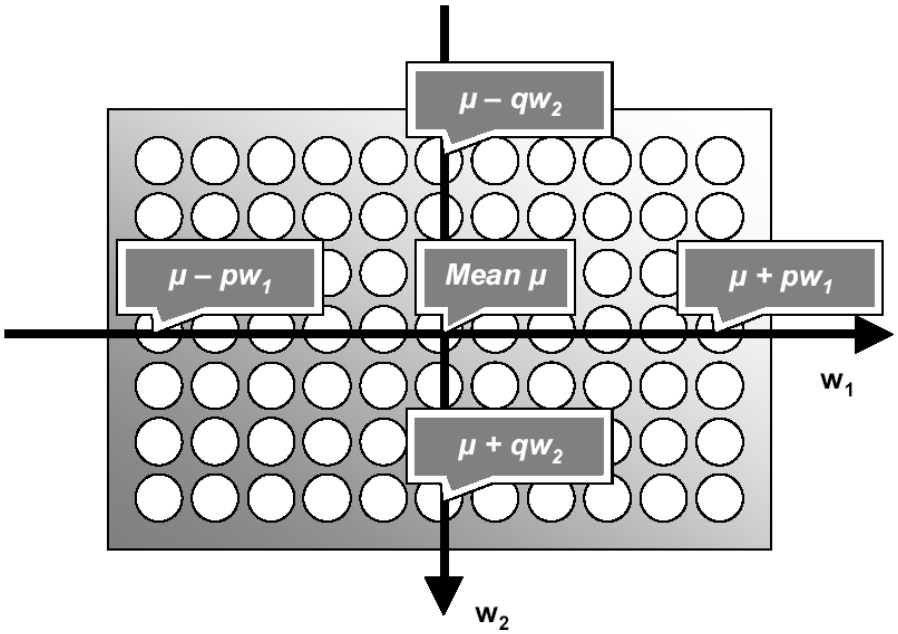
14 pav. ***Tikrinių vektorių panaudojimas duomenų charakterizavimui***

Dabar galima visa tai kas buvo aptarta aukščiau pritaikyti savaime susitvarkančio žemėlapio inicijavimui. Čia kiekvienas įėjimo vektorius (mokymosi duomenų elementas) yra daugiadimensinis dydis (*k*-matis vektorius $m_i = [m_{i1}, m_{i2}, ..., m_{ik}]$), kurio kiekviena dimensija atitinka vieną iš atributų sudarančių visą mokymosi elementą. Tokiu atveju, visa mokymosi duomenų aibė *D,* susidedanti iš *n* elementų, gali būti pavaizduota kaip matrica su *n* eilučių ir *k* stulpelių skaičiumi:

$$D = \begin{bmatrix} m_{11} & m_{11} & \dots & m_{1k} \\ m_{21} & m_{22} & \dots & m_{2k} \\ \dots & \dots & \dots & \dots \\ m_{n1} & m_{n2} & \dots & m_{nk} \end{bmatrix}$$

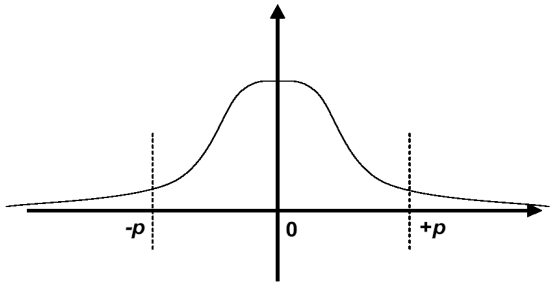
Sekantis žingsnis būtų mokymosi duomenų vidurkio $\mu = [\mu_1, \mu_2, ..., \mu_k]$, kur μ_i yra *i*-tojo stulpelio vidutinė reikšmė, skaičiavimas. Apskaičiavę vidurkį mes galime surasti kovariacijos matricą, parodančią sąryšius tarp kiekvienos iš *k* dimensijų (po šio žingsnio mes gausime *k*k* matmenų kvadratinę matricą). Turėdami kovariacinę matricą mes galime surasti *k* tikrinių reikšmių $\lambda_1, \lambda_2, ..., \lambda_k$. Gautas tikrines reikšmes išrikiuojame mažėjimo tvarka, kad būtų matomas savybių svarbumas [38]: $\lambda_1 \geq \lambda_1 \geq ... \geq \lambda_k$.

Inicializuojant žemėlapį, gauta vidutinė reikšmė μ yra patalpinama žemėlapio centre (žr. 15 pav.). Ašių atidėjimui mes naudojame du labiausiai dominuojančius tikrinius vektorius w_1 ir w_2, kurių tikrinės reikšmės yra atitinkamai λ_1 ir λ_2. Žemėlapio viršūnės, išsidėsčiusios dešinėje vidutinės reikmės μ pusėje, yra inicializuojamos prie μ pridedant papildomą kiekį w_1, o į kairę pusę – atimant papildomą kiekį w_1. Atitinkamai yra inicializuojamos ir tos žemėlapio viršūnės, kurios yra išsidėsčiusios viršutinėje ir apatinėje nuo vidurkio žemėlapio dalyse. Tos viršūnės, kurios nėra išsidėsčiusios tiesiogiai nat ašių, yra inicializuojamos parenkant vektorių w_1 ir w_2 tiesinę kombinaciją.



15 pav. ***Žemėlapio inicializavimas naudojant PCA metodą [38]***

15 pav. didžiausi w_1 ir w_2 koeficientai yra išsidėstę ant žemėlapio rėmelio ir pažymėti *p* ir *q* raidėmis atitinkamai. Tačiau čia iškyla naujas klausimas: kokios turi būti tos *p* ir *q* reikšmės? Kokį duomenų išsibarstymą pagal w_1 ir w_2 ašis mūsų žemėlapis turi apimti? Vienas iš variantų – šias reikšmes susieti su minimalia ir maksimalia reikšmėmis, pasitaikančiomis mokymosi duomenyse [34] (16 pav.). Tačiau tai pilnai neatspindėtų tikrojo duomenų pasiskirstymo (negalėtume atvaizduoti pilno turimų duomenų spektro).



16 pav. ***Tinkamų p ir q reikšmių pasirinkimas***

Tam kad supaprastinti šią situaciją, tarkime kad mūsų duomenys yra normalizuoti, t. y. kiekviena mūsų turimų duomenų dimensija turi vidurkį lygų nuliui ir standartinį nuokrypį lygų vienetui. Tuomet yra tikslinga *p* ir *q* reikšmes įsivaizduoti kaip intervalo, kuris apima visus reikšmingus duomenis, ribines reikšmes. Visa tai mums leistų *p* ir *q* reikšmes apibrėžti naudojant mokymosi duomenų standartinio nuokrypio σ įvertį: $p = q = 2\sigma$. Daugelis autorių [21, 33, 37] tai laiko tinkamiausiu pasirinkimu, nes jų nuomone tai neleidžia neesminiams duomenų elementams išraipyti bendrų rezultatų.

Jeigu mūsų inicializuojamas žemėlapis yra *W* pločio ir *H* aukščio, tai bendra formulė paskaičiuoti žemėlapio m_{ij} viršūnės svorių vektorių yra:

$$m_{ij} = \mu + p_i w_1 + q_j w_2$$

čia : $p_i = 2\sigma * \left(\frac{2i}{W} - 1 \right);$

$q_i = 2\sigma * \left(\frac{2j}{H} - 1 \right).$

Baigiant analizuoti SOM inicializavo principus reiktų pastebėti, kad atsitiktinis duomenų pasirinkimas mokymosi metu nėra niekaip įtakojamas to, kaip mes žemėlapį inicijavome, t.y. duomenų elementų atsitiktinis pasirinkomas nėra ribojamas inicializavimo metu naudotais svorių vektoriais (tokios situacijos yra būdingos kai kuriuose klasteriavimo algoritmuose, tokiuose kaip CLARA. Pastarąjame siekiant sumažinti skaičiavimo sudėtingumą, duomenų klasteriavimui yra naudojama tik maža dalis turimų duomenų [34]).

3. INTELEKTUALAUS KOMPONENTO REALIZACIJA NAUDOJANT SOM TINKLUS

3.1 E-mokymosi domeno modelio aprašymas

Domeno modelis yra laikomas viena pagrindinių, intelektualios e-mokymosi sistemos, dalių. Būtent domeno modelis apibūdina visą mokymosi kurso struktūrą – kokią medžiagą mes turime, kokiu pavidalu ji yra prieinama, kokie yra ryšiai tarp kursą sudarančių temų. Kiekviena intelektuali mokymosi sistema gali turėti savo specifinį domeno modelį. Šiame darbe aš naudosiu prof. A. A. Bielskio siūlomą būdą mokymosi medžiagos struktūrizavimui [3], kuris pagrinde orentuojasi į mokymosi medžiagos pateikimą skirtingais stiliais. Čia yra išskiriami 4 mokymosi stiliai [3]:

1. Pateikiami trys atsakymai, iš kurių vienas yra teisingas, o du klaidingi. Per grįžtamąjį ryšį (*angl.* feedback) pateikiami komentarai kiekvienam atsakymui: neteisingiems atsakymams parodoma, kodėl jie yra neteisingi, nurodant ir teisingą atsakymą, o teisingam atsakymui pateikiamas platesnis paaiškinimas apie tai, kodėl šis atsakymas yra teisingas.
2. Pateikiami ne mažiau kaip 4 atsakymai su trimis teisingo atsakymo 1/3 svorio dalimis, o iš trijų atsakymų sumos yra kaupiamas teisingas atsakymas. Vieno neteisingo atsakymo svoris parenkamas -5. Todėl jo pridėjimas prie dviejų teisingų sumažina įvertį iki <2. Kiekvienas teisingas dalinis atsakymas yra komentuojamas per grįžtamąjį ryšį nurodant, kodėl jis yra teisingas, o neteisingas - kodėl jis yra neteisingas.
3. Pateikiami ne mažiau kaip 7 atsakymai su 5 teisingo atsakymo 2-to svorio dalimis ir ne mažiau kaip 2 neteisingais -4-ių svorio atsakymais. Teisingo atsakymo ne mažiau kaip 3 dalys yra paimamos iš tai potemei mokytis tinkančio sukurtojo intelektualizuoto e.patarėjo arba objekto intelektualus valdymo programos taikymo komandų, kurios yra susiję su šia poteme. Atsakymo likusios teisingos dalys imamos iš teorinės medžiagos. Neteisingi atsakymai yra konstruojami iš e.patarėjo, o taip pat ir iš teorinės medžiagos. Per grįžtamuosius ryšius pateikiama mokymosi medžiaga nurodant, kodėl kiekvienas dalinis teisingas atsakymas yra teisingas, o kodėl kiekvienas neteisingas atsakymas yra neteisingas.
4. Pateikiami ne mažiau kaip 7 atsakymai naudojantis individualiu e.laboratoriniu darbu iš kurio imami ne mažiau kaip 5 atsakymai , ir teorine medžiaga, iš kurios imami ne mažiau kaip 2 atsakymai. Teisingų atsakymų svoris lygus +2 , o netesingų atsakymų svoriai imami -4. Per grįžtamuosius ryšius yra pateikiama mokymosi medžiaga nurodant, kodėl teisingas atsakymas yra teisingas, o kodėl neteisingas atsakymas yra neteisingas.

Kuriant studento ir padagoginį modelius mums yra būtina žinoti kiek temų sudaro visą mokymosi kursą ir kokiais stiliais ta medžiaga yra paruošta. Turėdami tokio pobūdžio informaciją (ją mes gauname iš domeno modelio) galime pereiti prie studento ir pedagoginio modelių kūrimo.

3.2 E. mokymosi savaime susitvarkančio žemėlapio projektavimas

SOM tinklas kuriamas tam, kad būtų galima realizuoti studentų klasifikavimą į atskiras grupes pagal jų bendrus požymius. Turint tokias studentų grupes būtų galima sekti jų mokinimosi eigą ir pagal tai nustatyti optimalų kurso medžiagos išmokimo planą. Tačiau čia iškyla klausimas – kokie turi būti tie požymiai pagal kuriuos mes galėtume klasifikuoti studentus? Mano manymu, šiuos požymius būtų galima parinkti analizuojant besimokinančio "kitimą" mokymosi proceso metu. Tai būtų galima pavaizduoti pavyzdžiu: ką tik naują kursą pradėjęs mokintis studentas turi žymiai mažiau žinių apie tą kursą, negu tas kuris jau yra išmokęs vieną ar daugiau to kurso temų. Analogiškai būtų galima laikyti ir tai, kad studentas jau baigiantis išmokti visą kursą turi daugiau žinių negu tas, kuris dar tik įpusėjo kurso mokymasi. Atsižvelgdamas į visa tai aš nusprendžiau, kad geriausias požymis, pagal kurį būtų galima studentus klasifikuoti, yra jų pasiektų rezultatų, besimokinant konkretų kursą, įvertinimas. Čia rezultatų įvertinimas yra suprantamas kaip studento gauti įvertinimai mokinantis konkrečias kurso temas.

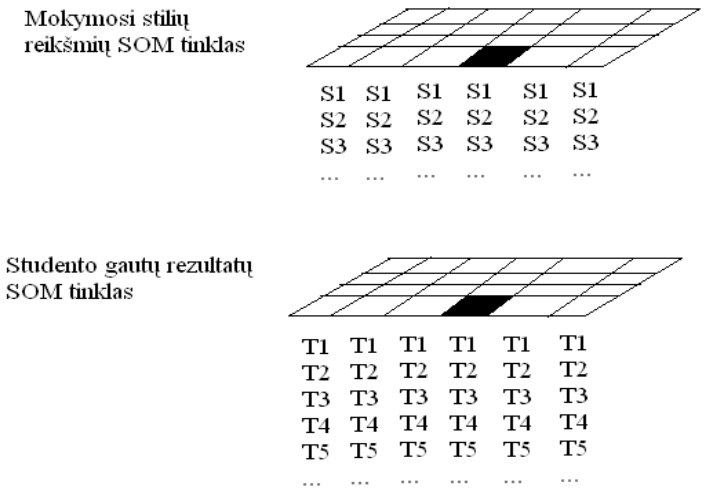
Tarkime, jeigu mokymosi kursą sudaro 10 temų (temų skaičių mums nusako domeno modelis) tai studento (kurso mokymosi pradžioje) pasiekti rezultatai bus:

TEMOS ID	1	2	3	4	5	6	7	8	9	10
Įvertinimas	0	0	0	0	0	0	0	0	0	0

Išmokus pirmą temą ir už testo sprendimą iš šios temos gavus 8 studento mokymosi rezultatai bus:

TEMOS ID	1	2	3	4	5	6	7	8	9	10
Įvertinimas	8	0	0	0	0	0	0	0	0	0

Norint sukurti klasifikatorių pagal aukščiau aptartus studento duomenis mes turime sukurti SOM tinklą, kurio kiekvienas neuronas savyje talpins 10 – ies dimensijų vektorių (tiek kiek yra kurse temų). Reikia pastebėti, kad šis SOM tinklas tik klasifikuos studentų rezultatų duomenis į tam tikras grupes, tačiau nieko nepasakys apie tai, kokiu stiliumi studentui turėtų būti pateikiama medžiaga. Siekiant išspręsti pastarąjį uždavinį, turime sukurti dar vieną SOM tinklą, kuris kauptų duomenis apie gautus rezultatus mokinantis vienu ar kitu stiliumi. Pastarasis žemėlapis (toliau jį vadinsiu stilių žemėlapiu) pagal savo plotį ir ilgį turi būti lygus pirmajam SOM tinklui (17 pav.). Čia gali skirtis tik žemėlapį sudarančių vektorių dimensijos: stilių žemėlapyje neurono vektoriaus dimensija yra lygi mokymosi stilių skaičiui (jį taip pat mes galime sužinoti iš domeno modelio), o ne kursą sudarančių temų skaičiui.



17 pav. ***Mokymosi stilių ir gautų rezultatų SOM tinklų sąryšis***
(čia Sx žymima mokymosi stiliai, o Tx – kurso temos)

17 pav. vaizduojamų SOM tinklų juodai pažymėti neuronai reiškia, kad kiekvienas rezultatų žemėlapio neuronas turi atitikmenį stilių žemėlapyje (neurono lokalizacijos žemėlapyje koordinatės sutampa).

3.3 E. mokymosi SOM tinklo apmokymo algoritmas

SOM tinklo apmokymas yra glaudžiai susijęs su intelektualaus komponento vartotojo modeliu. Šis modelis pasižymi tuo, kad jis turi valdyti ir saugoti visą reikalingą informaciją apie besimokinančio asmens pasiektus rezultatus. Taip pat reiktų atkreipti dėmesį į tai, kad mokymosi procese mes neišvengiame vėlavimo, t.y. sistema pateikusi studentui mokymosi medžiagą tam tikru stiliumi, negali žinoti ar jos sprendimas buvo teisingas tol, kol studentas nepatikrino savo įgytų žinių spręsdamas testą. Dėl šios priežasties SOM tinklo mokymas prasideda tik tuomet kai mes turime visus reikalingus duomenis:

- Studento pasiektų rezultatų vektorius **T** (jis bus naudojamas kaip įėjimo į SOM tinklą vektorius);
- BMU taško žemėlapyje koordinatės;
- Mokymosi stilių vektorius **S**, saugantis informaciją apie tai, kaip studentui sekėsi mokintis konkrečią temą vienu ar kitu stiliumi.

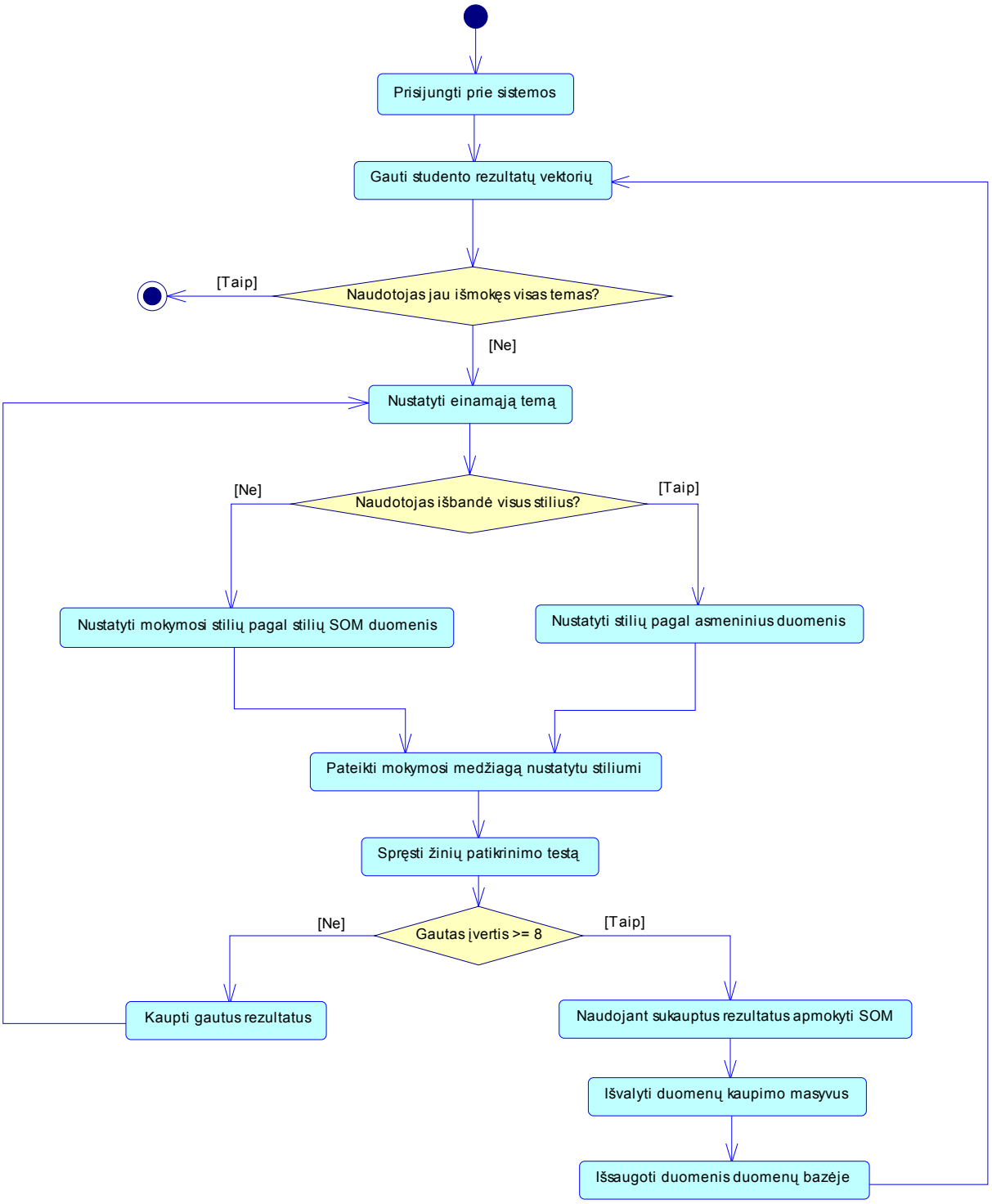
Visą studento mokymosi algoritmą būtų galima užrašyti tokia veiksmų seka:

1. Sistemos naudotojas autentifikuodamasis prisijungia prie sistemos ir tuo metu jam yra pateikiamas jo mokymosi metu pasiektų rezultatų vektorius **T**, nustatoma kokia temą studentas turėtų mokintis.
2. Vektoius **T** yra pateikiamas kaip įėjimo vektorius į e-mokymosi SOM tinklą ir pagal šį vektorių yra ieškoma BMU taško. Taip yra nustatoma kuriai grupei studentas priklauso.
3. Pagal gautas BMU taško koordinates atitinkamai yra paimamas mokymosi stilių vektorius **S**. Pastarasis **S** vektorius saugo mokymosi stilių "gerumo" reikšmes (t.y. tie stiliai kuriais studentai spręsdami gavo geresnius įvertinimus turės didesnes reikšmes negu tie, kuriais gauti įvertinimai buvo žymiai prastesni).
4. Iš **S** vektoriaus nustatoma, kuriuo stiliumi reikia pateikti mokymosi medžiagą. Čia yra paimamas tas stilius kurio sukaupta reikšmė buvo didžiausia.
5. Intelektualus e-mokymosi komponentas informuoja hypermedia komponentą (arba bet kurį kitą, atsakingą už medžiagos pateikimą naudotojui) apie tai kokiu stiliumi ir kokia tema turėtų būti pateikta besimokančiajam.
6. Baigęs studijuoti medžiagą studentas tikrina įgytas žinias spręsdamas testą, pateikiamą atitinkamu stiliumi. Gautas rezultatas yra perduodamas intelektualiajam komponentui ir pastarasis sprendžia ar studentas išmoko temą ar dar ne (šiame darbe aš laikiau, kad studentas išmoko temą jeigu jo gautas rezultatas >= 8). Jeigu studentas temos neišmoko tuomet yra atliekami tokie veiksmai (už šių veiksmų vykdymą yra atsakingas vartotojo modelis):
    a. Laikiname rezultatų masyve yra kaupiami gauti įvertinimo rezultatai **T1** bei mokymosi stilių rezultatai (gautas stilio gerumo "atlygis") **S1**.
    b. Tos pačios temos mokymuisi yra parenkamas sekantis pagal "gerumą" mokymosi stilius. Jeigu jau visi mokymosi stiliai išbandyti, bet tema dar neišmokta, tai temą siūloma mokintis tuo stiliumi, kuriuo studento pasiektas rezultatas mokinantis šią temą buvo didžiausias.
    c. Grįžtama prie 5 žingsnio vykdymo.
7. Studentas temą išmoko, kas reiškia kad mes turime visus mums reikiamus duomenis kad galėtume apmokyti tiek rezultatų tiek ir mokymosi stilių SOM tinklus. Apmokant SOM tinklus atliekami šie veiksmai:
    a. Pagal BMU taško koordinates prie studentų rezultatų vektoriaus **T** yra pritaikomi pats BMU ir jo kaimyniniai neuronai.
    b. Pagal BMU taško koordinates prie mokymosi stiliaus "gerumo" vektoriaus **S1** yra pritaikomi mokumosi stilių žemėlapio BMU taškas ir kaimyniniai jo neuronai.

c. Į studento mokymosi rezultatų vektorių **T**, ties ką tik išmokta tema, yra įrašomas skaičius, lygus vidurkiui visų gautų pažymių už testus, spręstus mokinantis šią temą.

d. Išsaugomi SOM tinklų pakeitimai. Grįžtame į 2 – ą žingsnį.

Šio algoritmo achema yra pavaizduota 18 pav:



18 pav. ***Studento mokymosi ir SOM tinklo apmokymo schema***

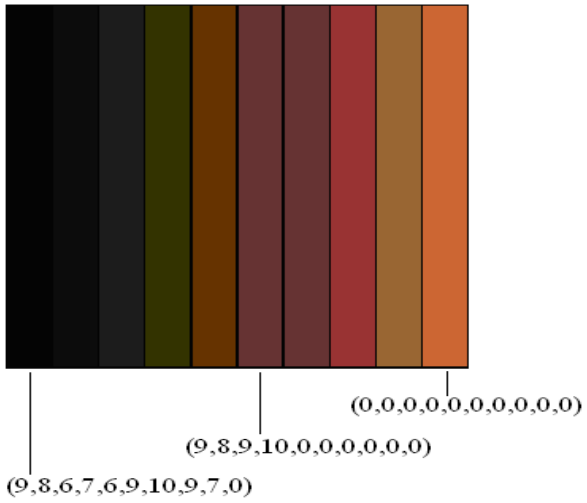
Kaip matome iš algoritmo, SOM apmokymo etape naujais duomenimis yra papildomas studento sukauptų rezultatų vektorius **T**, ko pasekoje, mokinantis sekančią temą, bus gautas visai kitas BMU taškas, o pagal jį ir nauji nurodymai, kokiu stiliumi geriausia būtų mokytis naująją temą.

3.3 SOM tinklų inicijavimas

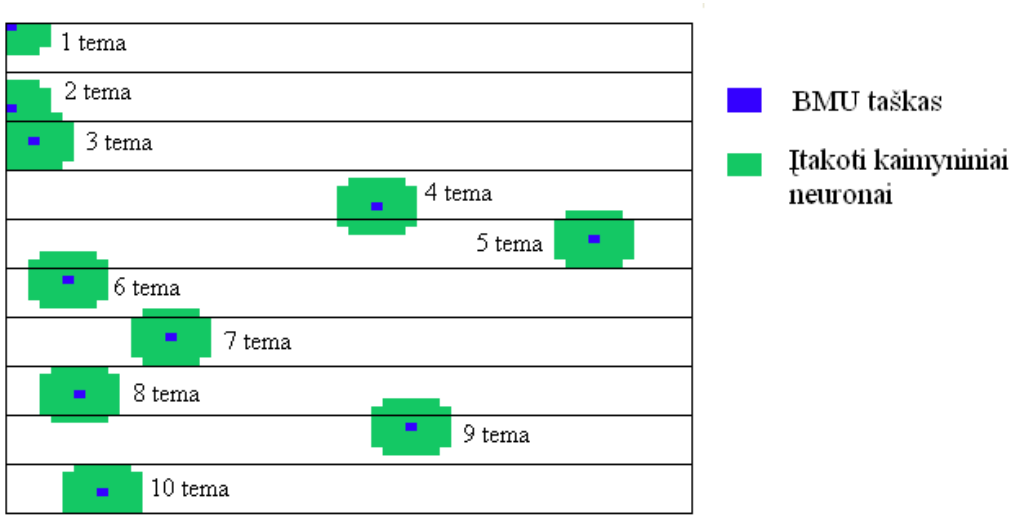
Kuriant SOM tinklus vienas iš esminių etapų yra jų inicijavimas. Nuo to kaip mes užpildome tinklą sudarančių neuronų reikšmes labai priklauso kaip greitai mūsų SOM tinklas apsimokys ir ar iš viso mes gausime pageidaujamą rezultatą. Dažniausiai naudojami SOM inicializavimo metodai buvo aptarti anksčiau šiame darbe ir kaip pats paprasčiausias buvo paminėtas metodas, kuomet neuronų reikšmės yra užpildomos visiškai atsitiktiniais skaičiais.

Siekdamas optimalesnio varijanto aš mokymosi rezultatų SOM tinklą inicializavau atsižvelgdamas į tam tikras prielaidas:

- Kadangi požymis, pagal kurį klasifikuosime studentų duomenis, yra gautas įvertinimas žinių patikros metu, tai nesunkiai galima nustatyti galimų reikšmių ribas: vertinant išmokimą dešimties balų sistema galimų reikšmių intervalas yra [0...10].
- Praktikoje dažniausiai mokymosi medžiaga yra pateikiama dėsningai, t.y. temos tarpusavyje būna daugiau ar mažiau susijusios. Dėl šios priežasties žemėlapis inicijavimo metu buvo suskirstytas į tam tikras zonas pagal temas (19 pav.). Tokiu būdu, po kiekvienos temos išmokimo, naujais duomenimis apmokant SOM tinklą, bus įtakojami tik išmoktosios temos ir gal būt, maža dalis jai artimų temų, neuronų (žr. 20 pav.)



19 pav. ***Mokymosi rezultatų SOM inicijavimas, žemėlapį suskaidant pagal temas į sritis***



20 pav. ***Mokymosi rezultatų SOM tinklo vaizdas po vieno studento viso kurso išmokimo.***

Iš 20 pav. galime matyti, kad ieškant BMU taško pradedant mokintis naują temą, šis taškas (pavaizduotas mėlyna spalva) yra randamas būtent tos temos zonoje ir tik gana nedidelė dalis BMU taško kaimyninių neuronų patenka į kitos temos tariamą zoną (tačiau tai nėra labai svarbu, kadangi vykstant SOM tinklo apmokymui, kaimynų radiusas nuolatos mažėja ir tinklo mokymo pabaigoje jis praktiškai tampa lygus 1, t.y. pačiam BMU taškui).

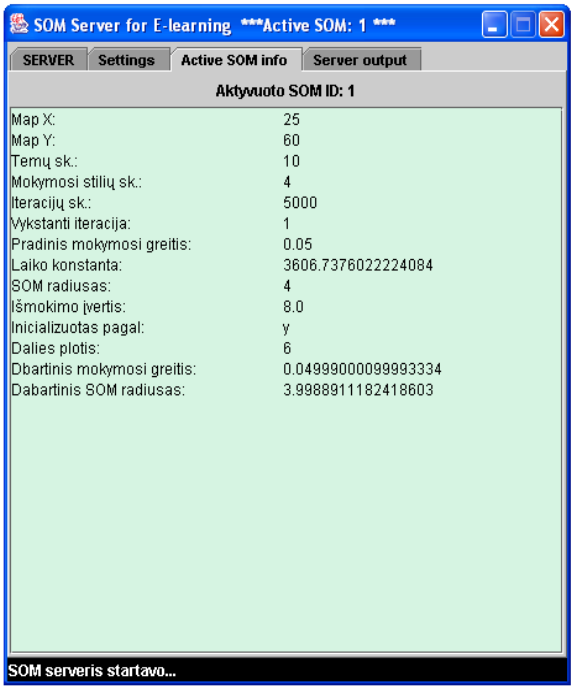
Mokymosi stilių SOM tinklas sistemos apmokymo pradžioje yra inicijuojamas visų neuronų vektorių reikšmes užpildant 0, kas reiškia, kad mes dar neturime duomenų kuriuo stiliumi konkrečią temą mokytis yra geriausia.

3.4 SOM tinklų pagrindinių parametrų parinkimas

Anksčiau šiame darbe jau buvo aptarta, kad mokymosi rezultatų SOM tinklo neuronų dimensijos dydis yra lygus mokomo kurso temų skaičiui, o mokymosi stilių SOM tinklo – turimų mokymosi stilių skaičiui. Dabar nustatysime kitus parametrus reikalingus kuriant Kohoneno SOM tinklą, skirtą e-mokymosi sistemos intelektualiajam komponentui. Tai būtų šie parametrai (21 pav.):

- *Žemėlapio plotis ir aukštis.* Parenkant šiuos parametrus būtina atsižvelgti į tai, kiek pas mus yra mokymosi duomenų, t.y. kiek mes turime vektorių kuriuos apmokydami SOM paduosime kaip įėjimo vektorius. Čia kuriamoje sistemoje vieną tokį vektorių sudaro duomenys gauti vienam studentui išmokus vieną temą. Šis skaičius iš anksto nėra apibrėžiamas, nes šie duomenys yra gaunami studentui besimokant, tačiau atlikus keletą bandymų su testiniais duomenimis ir atsižvelgiant į tai, kad kursą sudaro 10 temų, nuspręsta šį skaičių apibrėžti ties 5000 riba. Apytiksliai žinodami turimų SOM tinklo apmokymo

duomenų kiekį, žemėlapio plotį ir aukštį parenkame lygų 25 x 60 (matavimo vienatai yra tinklo neuronų skaičius). Tokių išmatavimų SOM tinklą sudaro 1500 neuronų, o tai yra žymiai mažiau už turimų apmokymo duomenų kiekį (5000). Pasak SOM autoriaus, Kohonen'o, ši sąlyga yra būtina kuriant SOM tinklą. Kadangi žemėlapis yra inicijuojamas sritimis pagal temas, tai parenkant žemėlapio plotį ar aukštį, reiktų imti tokį skaičių kuris be liekanos dalinasi iš kurso temų skaičiaus (taip tiksliau apskaičiuojamos temų sričių ribos).



21 pav. ***E. mokymosi SOM tinklo pagrindiniai parametrai.***

- *SOM apmokymo iteracijų skaičius*. Nuo šio parametro priklauso SOM apmokymo greičio ir kaimyninių neuronų radiuso kitimas vykdant SOM tinklo apmokymą. Kadangi pasirinkta 5000 apmokymo duomenų, tai norint visus juos pateikti SOM tinklui bus reikalinga 5000 iteracijų (po vieną duomenų vienetą kiekvienos iteracijos metu).
- *Pradinis mokymosi greitis*. Šis parametras nusako kiek "stipriai" BMU taško ir jo kaimyninių neuronų reikšmes padaryti panašesnes į įėjimo vektorių. Kadangi kuriamasis SOM tinklas yra gana nemažas ir jame yra pakankamai neuronų, kad skirtingi įėjimo vektoriai atrastų skirtingus į save panašius neuronus, tai nėra reikalinga labai "stipriai" keisti SOM tinklo, atsižvelgiant į įėjimo vektorius, todėl pradinis mokymosi greitis parenkamas santykinai nedidelis: 0,05.
- *SOM radiusas*. Tai parametras nusakantis, kokiu mąstu BMU taško kaimyniniams neuronams yra dromas poveikis, SOM tinklo apmokymo pradžioje. Vykstant tinklo apmokymui šis radiusas nuolat mažėja. Mokymo pradžioje kaip šio parametro reikšmę

parenkame 4 (kadangi pagal ankstesnius SOM tinklo aprašymus gaunasi, kad tariama temos sritis ant žemėlapio yra 6 – ių neuronų pločio, tai apmokant tinklą svarbiausia yra įtakoti tik atitinkamos temos ir jai gretimų temų neuronus (20 pav.). Radiusas lygus 4 – iems apima 7 neuronus, ko visiškai pakanka siekaimo tikslo įgyvendinimui).

- *Išmokimo įvertis.* Šiuo parametru yra nurodoma kiek mažiausiai studentas turi gauti spręsdamas tam tikros temos testą, kad būtų laikoma, jog jis išmoko tą temą. Jeigu studento gautas įvertis X už atliktą žinių patikrinimo testą yra didesnis arba lygus šiam parametrui, tai stiliui, kuriuo buvo pateikiama mokymosi medžiaga ir testas, suteikiamas atlygis (*angl.* reward) lygus $\frac{X}{10}$, priešingu atveju mokymosi stiliui suteikiamas atlygis paskaičiuojamas pagal formulę: $\frac{10-X}{10}*(-1)$.

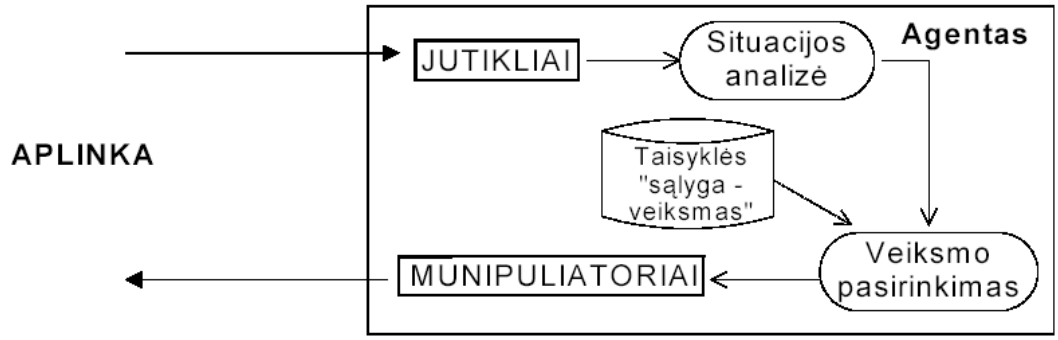
Tai buvo pagrindiniai parametrai, kuriuos reikia nurodyti kuriant naują e. mokymosi SOM tinklą. Kiti kintamieji (tokie kaip laiko konstanta), naudojami SOM apmokymo algoritme yra paskaičiuojami pagal formules, kurios buvo aptartos analitinėje šio darbo dalyje.

3.5 Intelektualaus komponento realizavimo principai ir priemonės

Pagal aukščiau išvardintus parametrus sukurtas tinklas naudoja santykinai nedaug resursų: visų SOM tinklo naudojamų duomenų išsaugojimui binarine forma pakanka apie 250 Kb atminties. Sukūrus tinklą visi šie duomenys yra saugomi atitinkamuose masyvuose (tokiu būdu mes visus duomenis turime kompiuterio operatyvynėje atmintyje, ko pasekoje mes greitai galime atlikti visas reiklaingas operacijas, tokias kaip mokymo stiliaus nustatymas, SOM tinklo apmokymas ir pan.). Tačiau savaime suprantama, kad adaptyvi e. mokymosi sistema privalo būti realizuota internetinėje aplinkoje (*angl.* web based). Tokiu atveju atsiranda problemos siekiant realizuoti SOM tinklus: jeigu mes su kiekviena internetinės naršyklės užklausa skaitysime SOM tinklų duomenis iš duomenų bazės, juos saugosime masyvuose ir po to atliksime informacijai išgauti/pakoreguoti reikiamas operacijas, tai net keli, vienu metu dirbantys vartotojai labai apkraus serverį. O jei dar vartotojui reikia atlikti dešimtis tokių operacijų, kurios inicijuojamos vis naujomis naršyklės užklausomis? Beto atsiranda problema siekiant užtikrinti, kad keli, vienu metu dirbantys vartotojai turėtų pačius naujausius SOM tinklų duomenis. Tokiu būdu būtų tikslinga panaudojant agentines technologijas [27] sukurti agentą, kuris būtų atsakingas už visą kuriamo intelektualaus komponento veikimą.

Agentas apibrėžiamas kaip esybė, kuri veikia savarankiškai, vykdydama sau ir/arba kitoms esybėms reikalingas užduotis [25]. Mano darbo kontekste yra kalbama apie agentą – programų

sistemą. Tai yra intelektualizuotas agentas, kas reiškia kad jis nagrinėjamoje situacijoje pasirenka geriausią galimą veiksmą [25]. Taip pat intelektualizuotu agentu yra vadinamas agentas, kuris yra gyvybingas, adaptyvus, reaktyvus, generatyvus, gebantis analizuoti saugomus duomenis, siekti savo tikslų, mokytis ir asistuoti vartotojui [25, 28]. Mano siekiamų tikslų įgyvendinimui reiktų sukurti agentą pasižymintį reaktyvaus, sugebančio mokytis ir asistuoti vartotojui, agento savybėmis. Reaktyvaus agento principinė schema pateikiama 22 pav.:



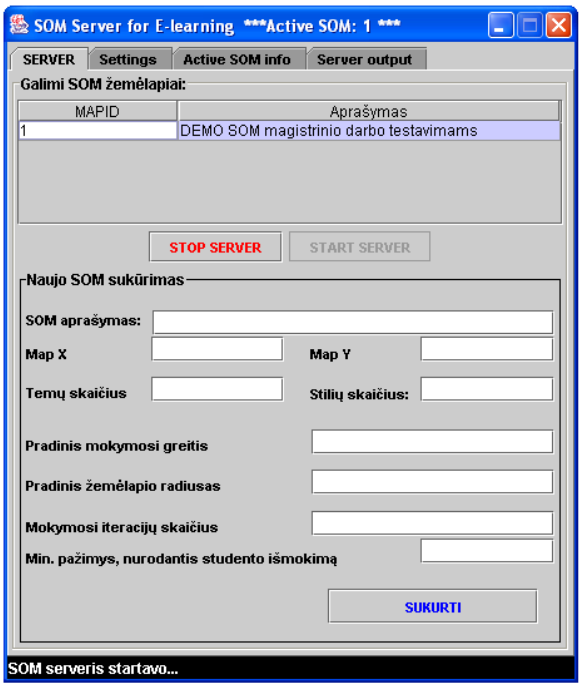
22 pav. ***Reaktyvaus agento principinė schema [25]***

Reaguojantis agentas per savo jutiklius gauna informaciją iš aplinkos ir pagal esamą situaciją priima sprendimus. Bendruoju atveju sprendimui priimti toks agentas naudoja "sąlyga-veiksmas" taisykles, tačiau šiame darbe sprendžiamame uždavinyje agentas sprendimą priima naudojantis Kohoneno SOM tinklais. Pastarasis agentas kaip informaciją iš aplinkos gauna studento sukauptus rezultatų duomenis ir pagal juos turi nustatyti kuriuo stiliumi studentui geriau mokytis tam tikrą temą (tai yra agento grąžinama informacija). Tam kad būtų galima realizuoti e. mokymosi sistemos intelektualų komponentą naudojant SOM tinklus, agentas turi tenkinti šiuos pagrindinius reikalavimus:

- Gebėti pagal studento duomenis priimti su mokymosis stiliumi susijusius sprendimus;
- Apsimokyti pagal studentų mokymosi duomenis;
- Būti pasiekiamas per kompiuterių tinklą;
- Gebėti vienu metu aptarnauti keletą vartotojų;
- Centralizuotai valdyti SOM tinklų duomenis, tam kad užtikrinti jog visi vartotojai dirba su naujausia informacija;
- Užtikrinti duomenų korektiškumą ir saugumą sistemos sutrikimo atveju.

Pagal šiuos reikalavimus galima pastebėti, kad kuriamam reaguojančiam agentui yra priskiriama serverio rolė [30]. Šis agentas nepasižymi mobilumo savybėmis, t.y. jis veikia tik viename kompiuteryje ir nekeliauja keliavimo (judėjimo) kompiuterių tinkle.

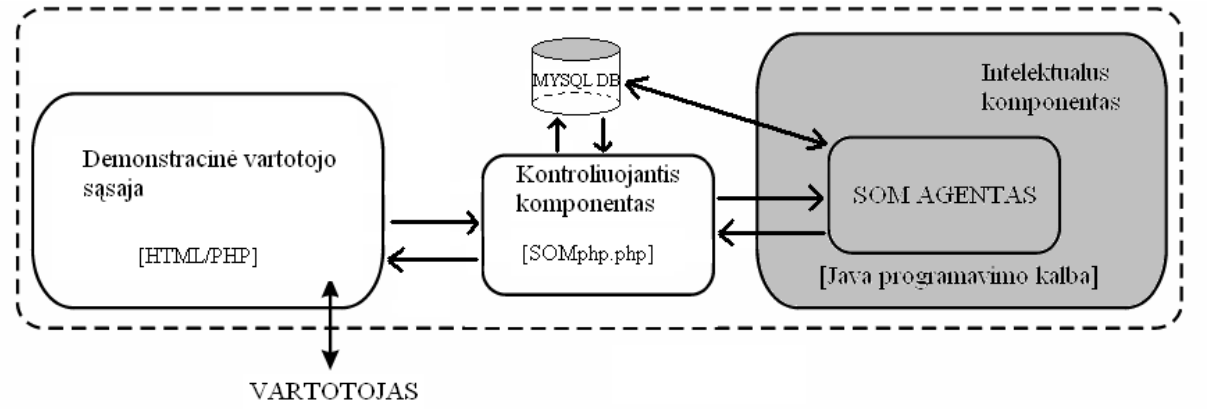
Sukūrus agentą, atitinkantį visus aukščiau keliamus reikalavimus, ir siekiant padaryti jo valdymą (paleidimą/sustabdymą) paprastesniu, buvo sukurta vartotojo sąsaja, kuri suteikė galimybes valdyti agentą, nustatyti reikiamus parametrus, įvesti naujus SOM tinklų duomenis (23 pav.). Visi agento veikimui reikalingi duomenys buvo saugomi duomenų bazėje, kurios struktūra yra aprašyta sekančioje potemėje.



23 pav. ***Vartotojo sąsaja skirta agento valdymui, konfigūravimui ir papildomų duomenų įvedimui***

Iš anksčiau darbe pateikto (2 pav.) adaptyvios e. mokymosi sistemos bendro modelio matome, kad vartotojas praktiškai neturio jokio tiesioginio ryšio su intelektualiu komponentu. Čia vartotojas bendrauja su sistema per vartotojo sąsają, kuri savo ruožtu pasitelkdama kitų sistemos komponentų gagalbą, palaiko ryšį su intelektualiuoju komponentu. Kuriant intelektualų komponentą šiame darbe buvo stengiamasi neprisirišti prie vartotojo sąsajos, t.y. intelektualų komponentą padaryti nepriklausomu nuo kitų adaptyvios e. mokymosi sistemos dalių. Visa tai mums suteikia galimybę sukurti savo pageidayjamą vartotojo sąsają ir mokymosi medžiagos išdėstymo struktūrą, nekeičiant intelektualaus komponento.

Šiame darbe nebuvo tikslo sukurti e. mokymosi aplinkos vartotojo sąsają (beto, kyla klausimas, ar verta tai daryti kai egzistuoja tokios pakankamai galingos sistemos kaip Moodle ir pan.), tačiau siekiant patikrinti kokios yra glimybės realizuoti šį komponentą internetinėje terpėje naudojant agentines technologijas, buvo sukurta demonstracinė vartotojo sąsaja realizuojanti studento mokymosi procesui įgyvendinti reikalingas funkcijas. Sukurtos vartotojo sąsajos ekrano vaizdai (*angl.* screenshot) yra pateikiami 1 – ame priede. Principinė tokios sistemos schema yra pateikiama 24 pav.



24 pav. ***Realizuotos adaptyvios e. mokymosi sistemos, principinė schema***

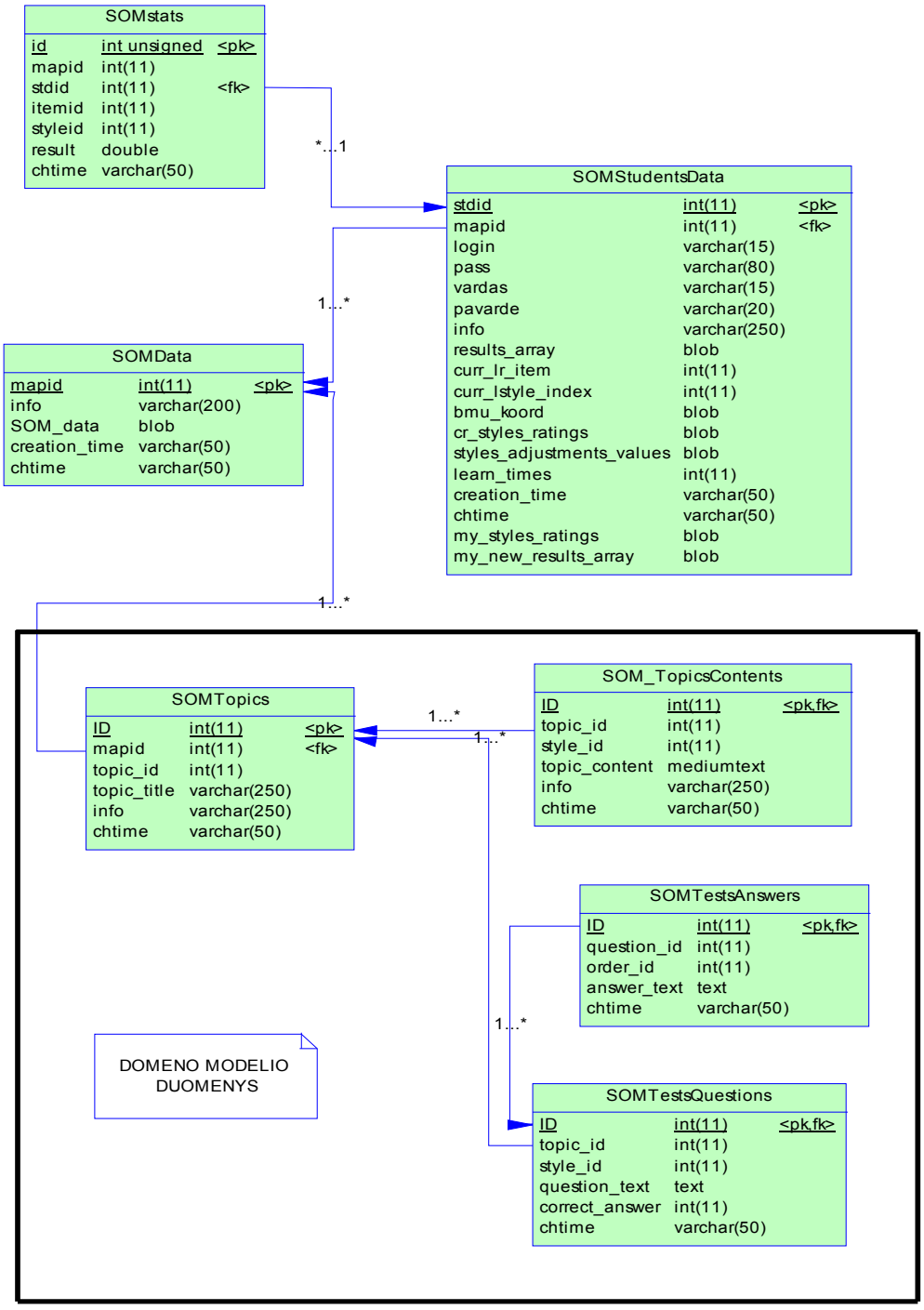
24 pav. pavaizduotos sistemos sukūrimui buvo naudojamos šios darbo priemonės:

- Jbuilder 8 – tai Java kalba paremta programavimo aplinka [16, 44], kurios pagalba buvo sukurtas intelektualaus komponento agentas ir jo valdymo sąsaja.
- MySQL DBVS – laisvai platinama pagal General Public License, duomenų bazių valdymo sistema.
- Mysql Connector/J – tai JDBC tvarkyklė, leidžianti Java programavimo kalbos priemonėmis prisijungti ir dirbti su MySQL DBVS (buvo reikalinga siekiant, kad agentas galėtų saugoti reikiamus duomenis MySQL duomenų bazėje).
- PHP – atviro kodo skriptų kalba [24], kurios pagalba buvo kuriamas tiek kontroliuojantis komponetas (objektas SOMphp, atsakingas už bendravimą su agentu ir mokymosi proceso valdymą), tiek ir vartotojo sąsaja internetinėje terpėje.

3.6 Duomenų bazės, skirtos SOM duomenims saugoti, projektavimas

Siekiant užtikrinti pakankamą duomenų saugumą, duomenų saugojimui buvo pasirinkta MySQL BBVS. Tai yra žymiai patikimiau, nei saugoti duomenis "plokščiame faile". Beto, su šia DBVS puikiai veikia ir PHP, kuri buvo naudojama domonstracinės vartotojo sąsajos kūrimui. Pasirinkus duomenų saugojimo būdą, liko išspręsti dar vieną problemą: kokiu būdu duomenų bazėje saugoti didžiulius skaitinių duomenų masyvus, sudarančius SOM tinklus? Jeigu SOM tinklų dydžiai būtų iš anksto apibrėžiami ir ateityje nekistų, tai gal ir būtų galima saugoti duomenis duomenų bazės lentelėje į kiekvieną lentelės lauką įrašant atitinkamą masyvo elementą (t.y. DBVS mums suteiktų tokias technines galimybes), tačiau toks duomenų saugojimo būdas būtų labai neracionalus ir nepraktiškas. Ši problema buvo išspręsta pasinaudojus Java programavimo kalbos

teikiamomis galimybėmis, tokiomis kaip objektų serializavimas (*angl.* object serialization) [16]. Objekto serializavimas mums leidžia sukurtą objektą paversti binarine duomenų seka, kurią galime saugoti pageidaujamoje formoje, o po to prireikus iš tų duomenų atstatyti tokį patį objektą (*angl.* object deserialization) (t.y. mes galime objektą (kartu su visais jo atributais) išsaugoti, o atsiradus poreikiui – jį sukurti iš anksčiau išsaugotų duomenų). Binarinius duomenis MySQL duomenų bazėje geriausia yra saugoti BLOB tipo lauke.



25 pav. ***Sukurtos adaptyvios e. mokymosi sistemos duomenų bazės schema***

25 pav. pateiktoje duomenų bazės schemoje yra pavaizduotos tiek agento veikimui reikalingų duomenų saugojimui skirtos lentelės, tiek ir domeno modelio duomenų lentelės. Toliau aprašysime tik agento veikimui reikalingų lentelių struktūrą, o detalus domeno modelio lentelių laukų aprašymas yra pateikiamas 2 – ame priede.

Agento veikimui reikalingi duomenys yra saugomi šiose lentelėse:

- SOMData (saugomi SOM tinklų duomenys);
- SOMStudentsData (saugomi vartotojo modelio duomenys);
- SOMStats (saugomi viso mokymosi proceso statistiniai duomenys).

2 lentelė. ***SOMData lentelės laukų aprašymas***

Lauko pavadinimas	**Aprašymas**
MAPID	Unikalus SOM tinklo identifikatorius.
INFO	SOM tinklo aprašymas (paprastai tai yra kurso, kurio mokymui yra kuriamas SOM tinklas, pavadinimas).
SOM_DATA	SOM tinklų duomenys saugomi binarine forma.
CREATION_TIME	SOM tinklo sukūrimo data ir laikas.
CHTIME	Įrašo paskutiniojo keitimo data ir laikas.

3 lentelė. ***SOMStudentsData lentelės laukų aprašymas***

Lauko pavadinimas	**Aprašymas**
STDID	Unikalus naudotojo (studento) numeris
MAPID	SOM tinklo, su kuriuo yra susijęs naudotojo, numeris
LOGIN	Naudotojo prisijungimo vardas.
PASS	Naudotojo prisijungimo slaptažodis.
VARDAS	Naudotojo vardas.
PAVARDE	Naudotojo pavardė.
INFO	Papildoma informacija apie naudotoją.
RESULTS_ARRAY	Naudotojo sukauptų mokymosi rezultatų masyvas.
CURR_LR_ITEM	Temos indeksas, kurią studentas turėtų mokintis šiuo metu.
CURR_LSTYLE_INDEX	Mokymosi stiliaus indeksas kuriuo studentui dabar turėtų būti pateikiama mokymosi medžiaga.
BMU_KOORD	BMU taško, gauto pagal RESULTS_ARRAY vektorių, koordinatės.
CR_STYLES_RATINGS	Pagal BMU_KOORD pasiimtas mokymosi stilių vektorius. Jis saugo bendrai sukauptą informaciją, kokiu stiliumi ir kaip sekėsi mokintis kitiems naudotojams.

STYLES_ADJUSTMENTS_VALUES	Vektorius, kuriame yra kaupiami (naudotojo konkrečios temos mokymosi metu) stilių "uždirbti" koeficientai.
LEARN_TIMES	Kiek kartų studentas jau mokosi vieną ir tą pačią temą.
MY_STYLES_RATINGS	Naudotojo mokymosi metu pagal skirtingus stilius kaupiama informacija, kuriuo stiliumi sekėsi geriau mokytis.
MY_NEW_RESULTS_ARRAY	Masyvas, kuriame kaupiami už išspręstus testus, mokinantis konkrečią temą, gauti įvertinimai.
CREATION_TIME	Įrašo sukūrimo data ir laikas.
CHTIME	Įrašo paskutiniojo keitimo data ir laikas.

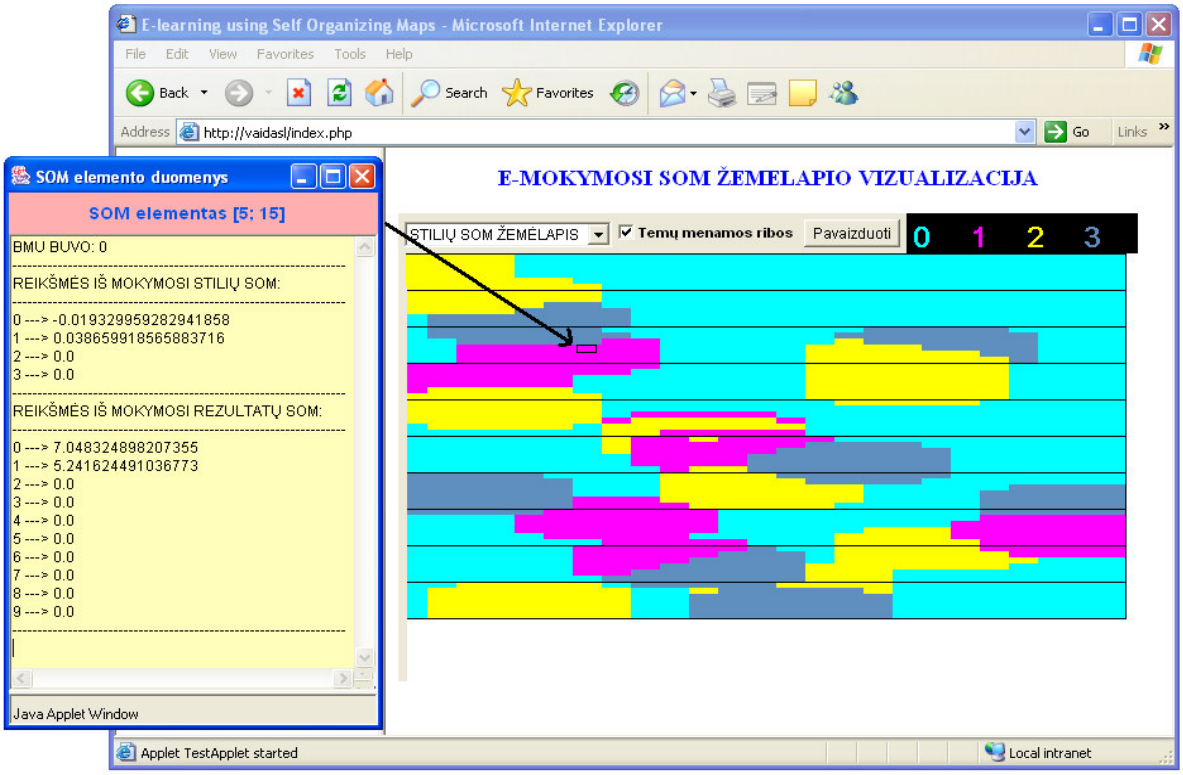
4 lentelė. ***SOMStats lentelės laukų aprašymas***

Lauko pavadinimas	**Aprašymas**
ID	Unikalus, raktinis įrašo identifikatorius.
MAPID	SOM tinklo identifikatorius, pagal kurį buvo vykdomas naudotojo mokymas.
STDID	Naudotojo unikalus identifikatorius.
ITEMID	Kurso temos eiliškumo numeris kuria mokinosi/mokinasi naudotojas
STYLEID	Mokymosi stiliaus identifikacinis numeris, kuriuo buvo pateikta medžiaga.
RESULT	Sprendžiant žinių patikrinimo testą gautas rezultatas.
CHTIME	Įrašo paskutiniojo keitimo data ir laikas.

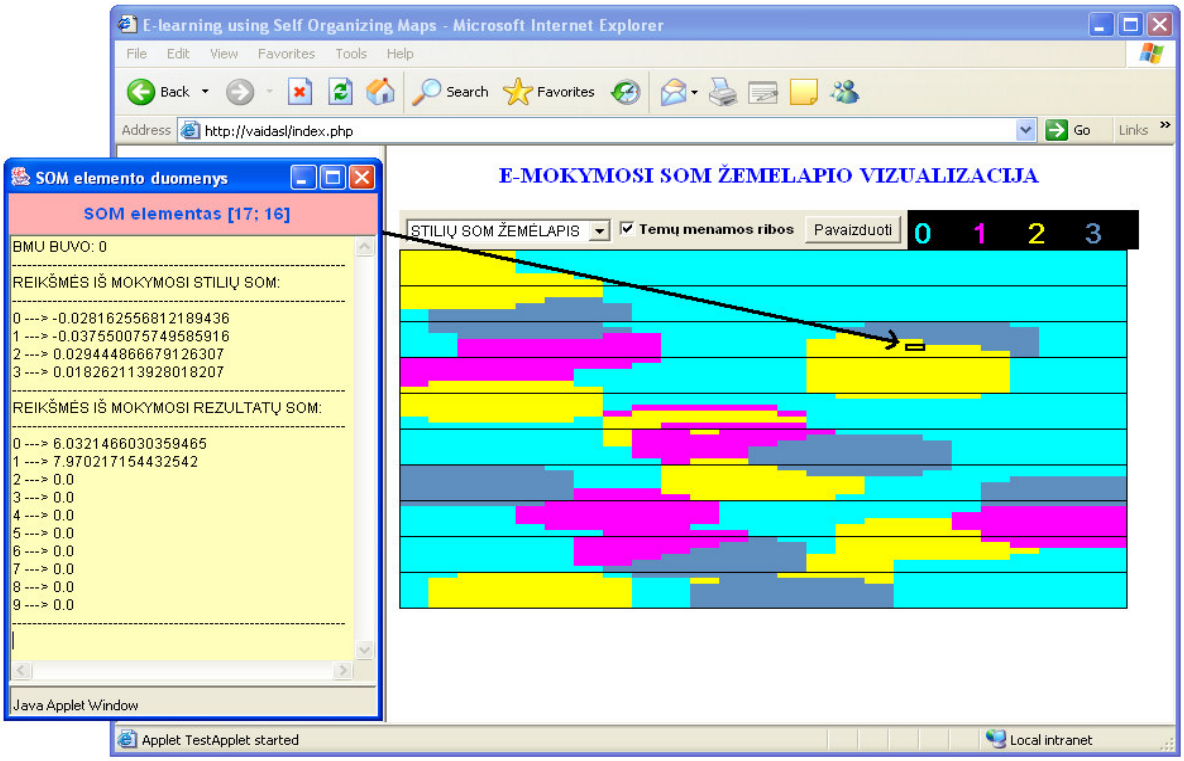
3.7 Gautų rezultatų apibendrinimas

Gautų rezultatų apibendrinimui yra naudojami 26 ir 27 pav., kuriuose yra pavaizduotas dalinai apmokytas e. mokymosi SOM tinklas, bei jo du neuronus charakterizuojantys duomenys. Čia visas SOM žemėlapis yra nuspalvintas keturiomis spalvomis, kurių kiekviena nurodo, kuris mokymosis stilius atitinkamame žemėlapio plote yra geriausias (spalvos ir mokymosis stiliaus indekso priklausomybė yra pavaizduota viršutiniame dešinąjame žemėlapio kampe). Kaip pavyzdys, specialiai buvo paimti du žemėlapio neuronai priklausantys tai pačiai menamai, 3 – ios temos, zonai, tačiau nurodantys skirtingus geriausius mokymosi stilius: pirmasis rodo, kad geriausias mokymosi stilius yra 1 (pavaizduota bordine spalva), o antrasis – tvirtina, kad 2 (geltona spalva). Tuo galima įsitikinti pažvelgus į elemento duomenų langą (sritis "Reikšmės iš mokymosi stilių SOM"): geriausiu stiliumi yra laikomas tas, kurio sukaupta reikšmė yra didžiausia. Norėdami išsiaiškinti kodėl tos pačios temos neuronai rodo skirtingą rezultatą, turime pažiūrėti į elemeto duomenų lango sritį "Reikšmės iš mokymosi rezultatų SOM": čia matome, kad naudotojui, kuris mokindamasis pirmą temą gavo įvertį artimą 7(7,04832), o antrą temą – artimą 5 (5,24162) tai jam

geriau 3 – ią temą mokintis 1 – uoju stiliumi. Tuo tarpu naudotojas gavęs atitinkamai 6 (6,03214) ir 8 (7,90217) turėtų mokintis 3 – ią temą 2 – uoju stiliumi.



26 pav. ***SOM tinklo elemento (5;15)saugomi duomenys***



27 pav. ***SOM tinklo elemento (17;16)saugomi duomenys***

Sprendimas apie tai, kokiu stiliumi reiktų pateikti medžiagą studentui, ankstesnio mokymosi procese gavusiam vienokius ar kitokius įvertinimus, yra priimamas atsižvelgiant į tai, kaip praeityje studentai su analogiškais rezultatais mokinosi naująją (šiuo atveju 3 – iąją) temą.

Šioje adaptyvioje e. mokymosi aplinkoje, sukurtasis SOM tinklas apsimoko tuo greičiau, kuo yra mažiau skirtingų naudotojų vertinant juos pagal jų gaunamus rezultatus. T.y. kuo daugiau vertotojų turės panašius įvertinimus (jų BMU taškai ant žemėlapio pateks į tą tą pačią, ar labai artimą jai, sritį), tuo daugiau informacijos apie mokymosi stilių "gerumą" bus surinkta tai naudotojų grupei. Svaime suprantama, kad informacija, apie mokymosi stilių "gerumą" iš keleto naudotojų, yra žymiai objektyvesnė it tikslesnė negu informacija, gauta tik iš vieno naudotojo.

Čia vienu iš pagringdinių SOM tinklo teikiamų privalumų yra laikoma tai, kad jis sistemai suteikia dinamiškumo – t.y. sistema nuolatos reaguoja į aplinkos pokyčius, ko pasekoje pasikeitus situacijai (tarkim vienu laiko momentu, dauguma studentų konkrečią temą geriausiai išmokdavo vienu stiliumi, o štai kitu laiko momentu, dėl kokių tai priežaščių, kitiems studentams priimtinesnis 3 – iasis mokymosi stilius) adekvačiai keičiasi ir sistema.

Galima pastebėti, kad SOM tinklo apmokymo metu, visas žemėlapis bus suskaidytas į tam tikas zonas (klases) kurių kiekviena turės savus sukauptus duomenis apie mokymosi stilių gerumą, o ši turima informacija bus tuo tikslesnė, kuo daugiau naudotojų bus praėję mokymosi procesą. Iš 26 ir 27 pav. matome kad vieną ir tą pačią temą, sistemos naudotojai pasižymintys skirtingomis savybėmis, gali mokytis skirtingais stiliais. Tokio sprendimo (apie mokymosi stilius) priimti negalime sistemoje naudojančioje Q-learning algoritmą. Pastarojoje sistemoje informacija apie stilių "gerumą" yra kaupiama bendrai visai temai ir nėra atsižvelgiama į besimokinančiojo savybes.

IŠVADOS IR SIŪLYMAI

1. Darbe buvo atlikta adaptyvios e. mokymosi aplinkos analizė, padėjusi susidaryti bendrą vaizdą, nusakantį kokią vietą adaptyvioje e. mokymosi aplinkoje užima intelektualusis komponentas. Atliktoji analizė leido teigti, kad intelektualusis komponentas yra adaptyvios e. mokymosi aplinkos pagrindas.

2. Atlikus detalią intelektualiojo komponento analizę nustatyta, kad norint sukurti mokymosi aplinką, prisitaikančią prie konkretaus naudotojo pagal jo paties poreikius, daugiausia dėmesio reikia skirti naudotojų grupių modeliavimui, kurio dėka galima gautus mokymosi duomenis skirstyti į grupes pagal šių duomenų panašumą, juos analizuoti ir daryti išvadas apie konkrečiai grupei būdingus bruožus. Sukaupta naudotojų grupių informacija ir atskiro naudotojo panašumas (pagal tam tikrus požymius) į tam tikrą grupę leidžia nustatyti kaip turi būti pateikiama mokymosi medžiaga, kad naudotojas kursą išmoktų greičiausiai.

3. Naudotojų ir jų grupių modeliavimas yra glaudžiai susijęs su duomenų klasifikavimo uždaviniu. Šio uždavinio sprendimui yra nemažai metodų, tačiau atlikus keleto iš jų (pvz. AGNES, K-vidurkių, SOM metodų) tinkamumo e. mokymosi aplinkos intelektualaus komponento kūrimui, analizę nustatyta, kad geriausiomis savybėmis pasižymi Kohoneno savaime susitvarkančio žemėlapio neuroninis tinklas. Šis metodas nereikalauja išankstinio grupių apibrėžimo, sugeba prisitaikyti (persitvarkyti) prie naujų duomenų, puikiai tinka procesuose su vėlavimu (šiai grupei priklauso ir e. mokymasis).

4. Vienas iš svarbiausių aspektų SOM tinklų panaudojime – jų inicijavimas. Atlikta SOM tinklų veikimo principų analizė leido nustatyti pagrindinius parametrus (mokymo iteracijų skaičius, turimų duomenų kiekis, pradinis mokymosi greitis ir kaimynų radiusas), įtakojančius SOM tinklų kokybę. Siekaint greitesnio tinklų apsimokymo ir geresnių rezultatų, buvo sukurtas SOM tinklo inicijavimo metodas, kurio metu visas žemėlapis yra padalinamas į menamas zonas pagal kurso temas.

5. Naudojant SOM tinklus, nuolatos yra apdorojami didžiuliais kiekiai duomenų. Siekiant išvengti nuolatinio duomenų skaitymo/rašymo iš duomenų bazės ir užtikrinti jų korektiškumą buvo pasinaudota agentinėmis technologijomis - sukurtas agentas, realizuojantis intelektualųjį komponetą. Agento, pasižyminčio veikimo kompiuterių tinkle savybėmis, sukūrimas mums suteikė galimybes realizuoti vieną iš pagrindinių reikalavimų, keliamų e. mokymosi aplinkai - veikimą internetinėje terpėje.

6. Sukūrus testinę vartotojo sąsają, veikiančią internetinėje terpėje, buvo patikrinta ir įsitikinta tokiais agento gebėjimais, kaip: vienu metu aptarnauti keletą naudotojų, priimti

sprendimus atsižvelgiant į konkretaus naudotojo poreikius, bei užtikrinti sukauptų mokymosi duomenų korektiškumą ir saugų valdymą.

7. Apibendrinant išspręstus uždavinius galime teigti, kad darbo pagrindinis tikslas - sukurti adaptyvios e. moklymosi aplinkos intelektualų komponentą, priimantį sprendimus atsižvelgiant į asmenines sistemos naudotojo savybes, buvo pasiektas.

8. Tolimesniai šio darbo tęstinumo pasiūlymai būtų šie:

- galimybė ieškoti naujų požymių (darbe buvo pasirinkta gauti įvertinimai už žinių patikrinimo testus), kurie duotų kuo geresnius naudotojų klasifikavimo rezultatus;
- išanalizuoti, kokios būtų galimybės tokio intelektualaus komponento susiejimui su puikiomis savybėmis pasižyminčia “Moodle” mokymosi aplinka.

TERMINŲ IR SANTRUMPŲ ŽODYNĖLIS

AI (Artificial Intelligence) – Dirbtinis intelektas

AQE (Average Quantization Error) – vidutinė kvantavimo klaida

ALE (Adaptive Learning Environments) – adaptyvi mokymosi aplinka (AMA)

AIES (Adaptive and Intelligent Educational Systems) – Adaptyvinės ir Intelektinės Mokymo Sistemos (AIIS)

AHS (Adaptive Hypermedia Systems) – Adaptyvi navigacijos pagalba (ANP)

BLOB (Binary Large OBject) – didelis binarinis objektas.

BMU (Best Matching Unit) – geriausios atitikties elementas

CMS (Course Management System) – Kursų Valdymo Sistema (KVS)

DBMS (DataBase Management System) – duomenų bazės valdymo sistema (DBVS)

ITSs (Intelligent Tutoring Systems) – Intelektualios Mokymo Sistemos (IMS)

LMS (Learning Management Systems) – Mokymo Valdymo Sistema (MVS)

LOM (Learning Object Metadata) – Mokymosi objekto meta duomenys (MOM)

PCA (Principal Component Analysis) ž pagrindinio komponento analizė

RL (Reinforcement Learning) – priverstinis mokymas

SOM (Self Organizing Map) – savaime susitvarkantis žemėlapis

VLE (Virtual Learning Environments) – Virtuali Mokymo Aplinka (VMA)

WBE (Web-based education) – Web paremtas mokymasis

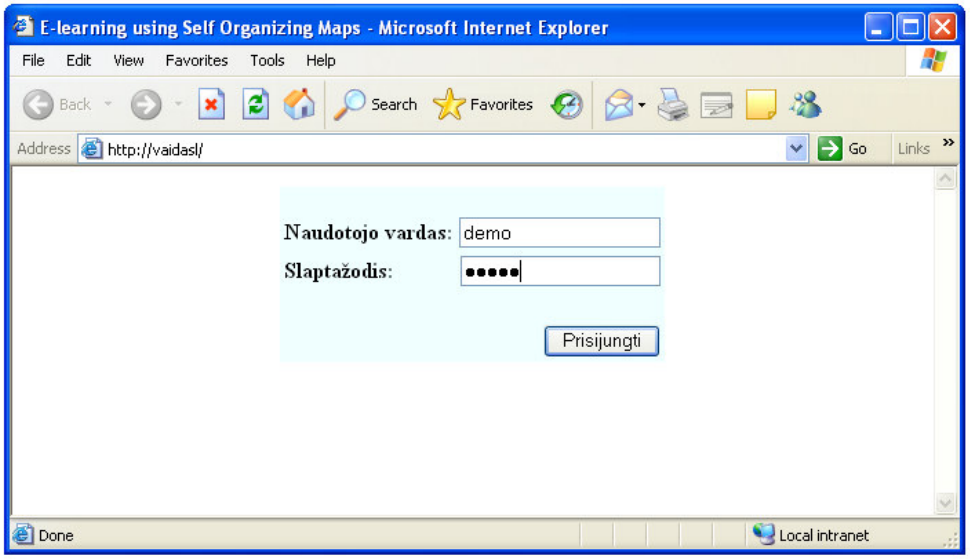
LITERATŪRA

1. Antanaitis A. *Intelektualių e. mokymosi žinių elektroninė verslovė.* 2004. 6 p.
2. Barlow T. W. *Self-organizing maps and molecular similarity.* 1995.
3. Bielskis A. A. *Žinių klasifikatorių taikymas e.mokymosi spartinimui.* Klaipėdos Universitetas, Informatikos katedra. 2004.
4. Brusilovsky P., Schwarz E. *Concept-based navigation in educational hypermedia and its implementation on WWW.* 1997. Proceedings of ED-MEDIA/ED-TELECOM'97, Calgary, Canada.
5. *Cluster analysis.* [interaktyvus]. (žiūrėta 2004.12.10) http://www.mines.edu/~jaelee/dm/Cluster.pdf
6. Conlan O., Dagger D., & Wade V. *Towards a Standards-based Approach to e-Learning Personalization using Reusable Learning Objects.* [interaktyvus] World Conference on E-Learning in Corp., Govt., Health, & Higher Ed. 2002, 210-217 p. (žiūrėta 2004.10.16) https://www.cs.tcd.ie/Owen.Conlan/publications/eLearn2002_v1.24_Conlan.pdf
7. Cristea P. D, Tuduce R. *Test authoring for inteligent e-learning environments*. [interaktyvus]. "Politechnica" University of Bucharest. (žiūrėta 2005.03.14) http://wwwis.win.tue.nl/~acristea/WBE/416-805_WBE-PCristea_RTuduce_6pg.pdf
8. Davis D. N. *Reactive and Motivational Agents: towards a collective minder.* [interaktyvus]. Staffordshire University. 1996. (žiūrėta 2005.12.10) http://www2.dcs.hull.ac.uk/NEAT/dnd/papers/atal96.pdf
9. Dong G., Li J. *Efficient mining of emerging patterns: Discovering trends and differences.* 1999. In proceedings of the fifth international conference on knowledge discovery and data mining. San Diego, USA.
10. Eklund J., Zeiliger R. *Navigating the Web: possibilities and practicalities for adaptive navigational support.* 1996. Proceedings of AUSWEB'96, Southern Cross University Press.
11. Germano T. *Self organizing maps.* 1999, [interaktyvus]. (žiūrėta 2005.03.06) http://davis.wpi.edu/~matt/courses/soms/
12. Gonzalez F., Dasgupta D. *Neuro-Immune and Self-Organizing Map Approaches to anomaly detection: a comparison.* [interaktyvus]. The University of Memphis. 2002. (žiūrėta 2005.02.25) http://www.cs.memphis.edu/~gonzalef/papers/icaris2002.pdf
13. Greenwald A., Hall K. *Correlated-Q Learning* [interaktyvus] Department of Computer Science, Brown University. 2003, (žiūrėta 2004.10.12) http://www.hpl.hp.com/conferences/icml2003/papers/231.pdf
14. Haritopoulos M., Allinson N. M., Yin H. *Multiplicative noise removal using self-organizing maps*. [interaktyvus]. Department of Electrical Engineering and Electronics, Manchester. 2001. (žiūrėta 2005.04.07) http://ica2001.ucsd.edu/index_files/pdfs/022-haritopoulos.pdf
15. Honkela T. *Self-organizing maps in natural language processing*. [interaktyvus]. Helsinki University of Technology. 1997. (žiūrėta 2005.03.06) http://www.cis.hut.fi/~tho/thesis/index.html
16. Horstmann C. S. Cornell G. *Core Java 2 Volume II – Advanced features.* 2002.
17. *How Q-learning works.* [interaktyvus]. (žiūrėta 2004.06.05) http://computing.dcu.ie/~humphrys/Notes/RL/how.q.html
18. Kaski S. *Data exploration using self-organizing maps.* [interaktyvus]. Acta Polytechnica Scandinavica. 1997. (žiūrėta 2005.03.15) http://www.cis.hut.fi/~sami/thesis/index.html
19. Kavcic A. *Adaptation Techniques in Adaptive Hypermedia Systems. I*999, [interaktyvus]. (žiūrėta 2004.10.16) http://lgm.fri.uni-lj.si/~alenka/papers/Mipro1999.pdf
20. Kavcic A. *Adaptive hypermedia learning systems.* 1998. Master's thesis, Faculty of Computer and information science, Ljubljana.
21. Kohonen T. *The Self-organizing map.* Invited paper [interaktyvus]. 1990. (žiūrėta 2005.01.24) http://www.eicstes.org/EICSTES_PDF/PAPERS/The%20Self-Organizing%20Map%20(Kohonen).pdf

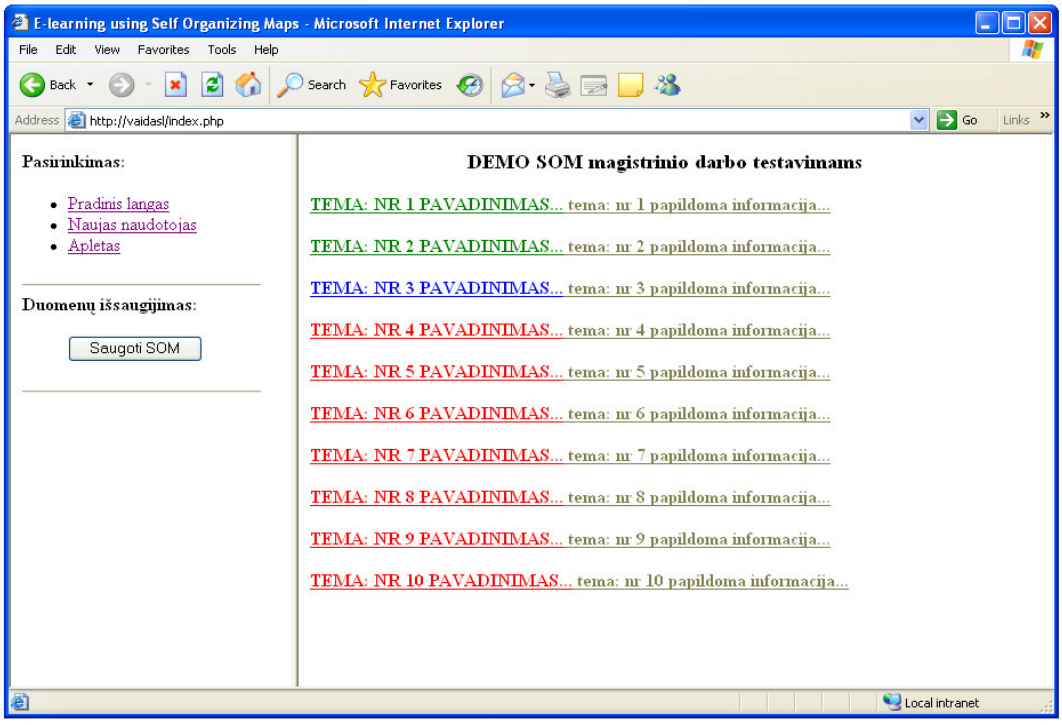
22. Kohonen T., Lagus K. and others *Self-organizing maps of docyment collections: A new approach to interactive exploration.*[interaktyvus]. Helsinki University of Technology. 1996. (žiūrėta 2005.01.10) http://www.ics.uci.edu/~pratt/courses/papers/TextMining/Lagus-KDD-96.pdf
23. Kohonen T. *Self-organizing maps in intelligent systems.* [interaktyvus]. Helsinki University of Technology. (žiūrėta 2004.12.17) http://www.helsinki.fi/~niskanen/tk/koho.html
24. Котеров Д. *Сомоучитель PHP4.* 2001. БХВ-ПЕТЕРБУРГ.
25. Lupeikienė A. *Agentinių technologijų panaudojimo reikalavimai kuriant komponentines verslo, informacines ir programų sistemas.* [interaktyvus]. Matematikos ir Informatikos institutas. 2003. (žiūrėta 2004.12.10) http://www.ktu.lt/lt/mokslas/konf03/konf_02/IT2003/Sekcija06.pdf
26. MAISE Centre. *Making Sense of Learning Specifications & Standards: A Decision Maker's Guide to their Adoption.* [interaktyvus]. eLearning Consortium, Working Group to make Sense of our Standards and Specifications (S3). 2002. (žiūrėta 2004.11.07) http://www.masie.com/standards/S3_Guide.pdf
27. Niederberger C., Gross M. *Hierarchical and Heterogenous reactive agents for real-time applications*. [interaktyvus]. Departement of Computer Science, Zűrich, Switzerland. 2003. (žiūrėta 2005.01.12) http://graphics.ethz.ch/Downloads/Publications/Papers/2003/nie03/p_Nie03.pdf
28. Nolfi S. *Power and the limits of reactive agents.* [interaktyvus]. Institute of Psychology, National research Council. (žiūrėta 2005.04.10) http://gral.ip.rm.cnr.it/nolfi/papers/nolfi.reactive.pdf
29. *Nonhierarchical clustering method: K-Meansclustering method* [interaktyvus]. (žiūrėta 2004.10.22) http://www.snr.missouri.edu/multivariate/lec6note.pdf
30. *Overview of agents and agent environments*. [interaktyvus]. (žiūrėta 2005.04.10) http://www.cse.sc.edu/research/cit/Courses/EECE818/CIS4Agents.ppt
31. Perelomov I., Azcarraga A. P. and others *Using structured Self-organizing maps in news integration websites.* [interaktyvus] National University of Singapore. 2002. (žiūrėta 2005.03.02) http://www2002.org/CDROM/poster/105.pdf
32. Podgornik R., Zafred M., Pajtler A. *A Study of k-means Method where starting conditions are changed: a simulation study.* 2004, [interaktyvus]. (žiūrėta 2005.03.10) http://mrvar.fdv.uni-lj.si/pub/mz/mz1.1/podgor.pdf
33. Rauber A. *Self-Oraganizing maps*. 1998, [interaktyvus]. (žiūrėta 2005.01.23) http://www.ifs.tuwien.ac.at/ifs/research/pub_html/mer_dexa98/node3.html
34. Schatzmann J. *Using Self-Organizing Maps to visualize clusters and trends in multidimensional datasets.*2003, [interaktyvus]. (žiūrėta 2005.02.02) http://mi.eng.cam.ac.uk/~js532/papers/schatzmann03soms.pdf
35. Schenker A., Last M., Bunke H., Kandel A. *A comparison of two novel algorithms for clustering web documents.* 2003, [interaktyvus]. (žiūrėta 2005.02.23) http://www.csc.liv.ac.uk/~wda2003/Papers/Section_V/Paper_16.pdf
36. Serdi-Bouchelaghem H., Sellami M. *Design of an inteligent tutoring system on the WWW to support interactive learning.* 2001, [interaktyvus]. (žiūrėta 2005.01.23) http://www.ineer.org/Events/ICEE2001/Proceedings/papers/522.pdf
37. Simula O. *The self-organizing map as a tool in knowledge engineering, pattern recognition in soft computing paradigm.* 2001. World Scientific Publishing, River Edge, NJ.
38. Smith L. *A tutorial on principal components analysis*. 2002, [interaktyvus]. (žiūrėta 2005.01.23) http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
39. Struyf A., Hubert M., Rousseeuw P. J. *Clustering in an Object-Oriented Environment.* [interaktyvus] Department of Mathematics and Computer Science, U.I.A. 2004. (žiūrėta 2004.10.12) http://www.jstatsoft.org/v01/i04/paper/clus.pdf

40. Trausan-Matu S. *Knowledge-based, automatic generation of educational web pages.* [interaktyvus]. "Politechnica" University of Bucharest. 1997. (žiūrėta 2004.09.28) http://www-it.fmi.uni-sofia.bg/larflast/papers/rilw97.pdf
41. Verleysen M. *Self-organizing maps.* 2003, [interaktyvus]. (žiūrėta 2005.02.10) http://www.dice.ucl.ac.be/~verleyse/lectures/elec2870/slides/self-organizing%20maps%20BW%202spp.pdf
42. Vidal R., Sastry S. and others *Generalized Principal Component Analysis (GPCA)*. [interaktyvus]. University of Illinois. 2003. (žiūrėta 2005.01.23) http://robotics.eecs.berkeley.edu/~rvidal/cvpr03-gpca-final.pdf
43. Villmann T., Merenyi E. *Extensions and modifications of the Kohonen-SOM and applications in remote sensing image analysis.* [interaktyvus]. (žiūrėta 2005.01.24) http://www-ece.rice.edu/~erzsebet/papers/SOM-book-chapter.pdf
44. Хабибуллин И. *Сомоучитель Java.* 2001. БХВ-ПЕТЕРБУРГ.
45. Wang D., Ressom H. and others *Double Self-Oraganizing maps to cluster gene expression data.* [interaktyvus]. University of Maine, USA. 2002. (žiūrėta 2005.04.02) http://www.dice.ucl.ac.be/Proceedings/esann/esannpdf/es2002-403.pdf
46. Wenger E. *Artificial inteligence and tutoring systems:computational and cognitive approaches to the communication of knowledge.* 1987. Morgan Kaufmann Publishers, Inc. Los Altos.
47. *Žinių klasifikatorių taikymas e.mokymosi spartinimui.* Bielskis A. A. Vadyba. Mokslo tiriamieji darbai, Nr.2(5), Klaipėda, 2004, 5-11p.

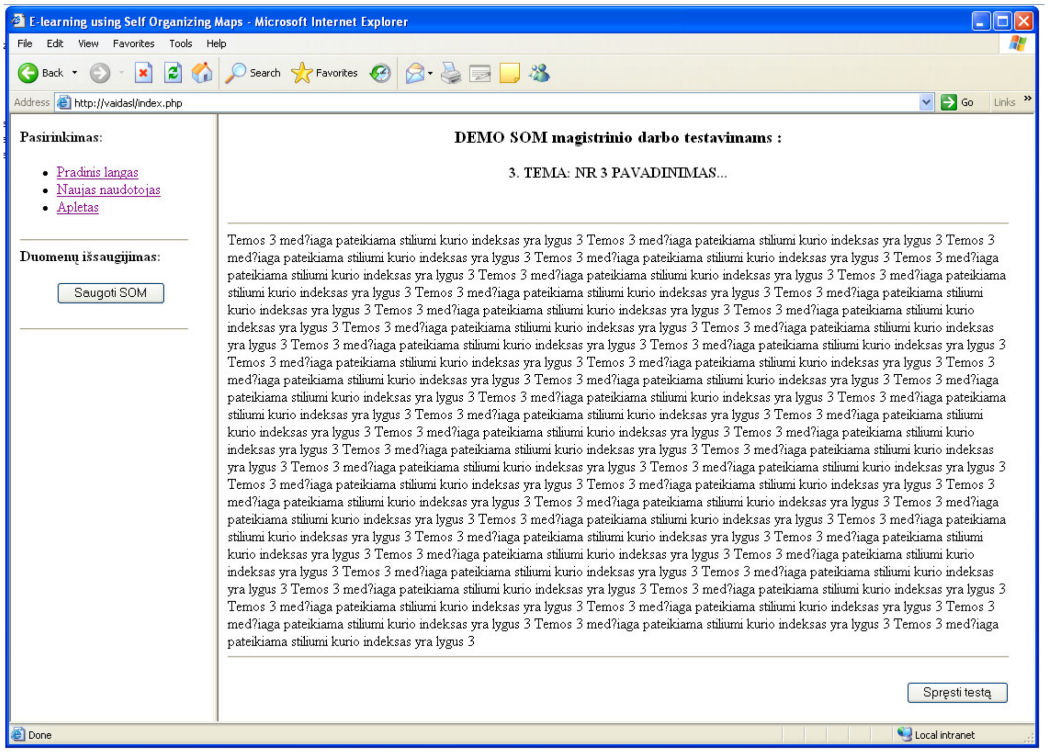
Priedas 1
Demonstracinė vartotojo sąsaja



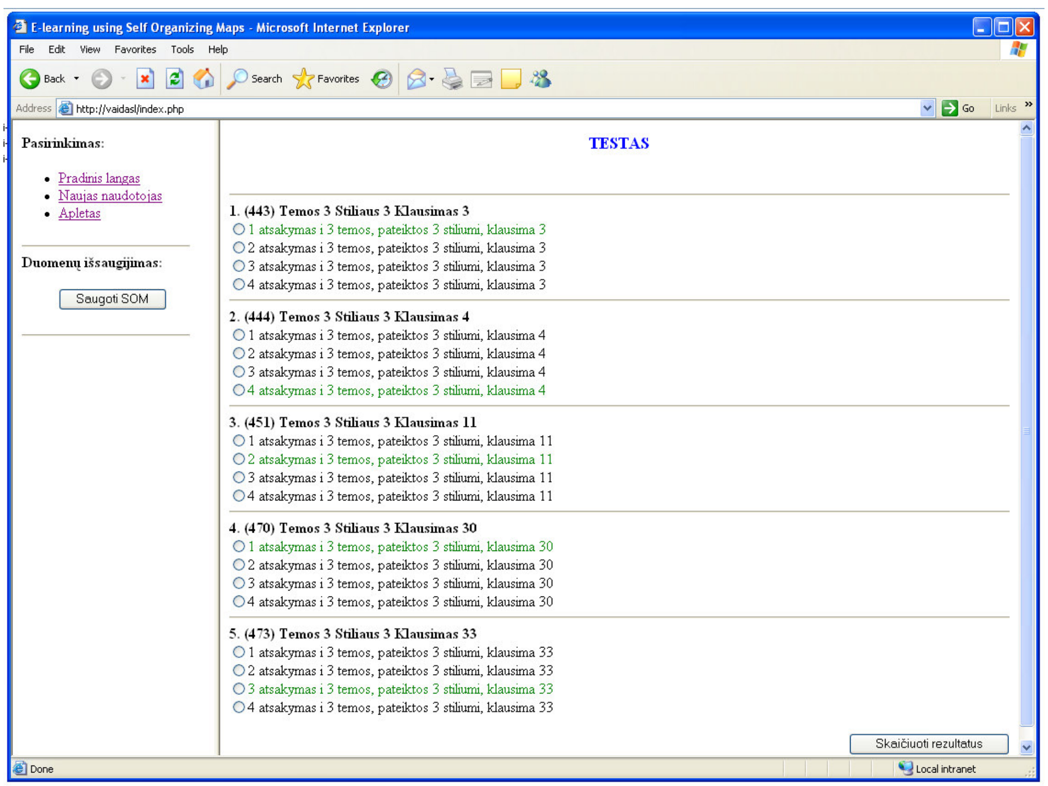
1 *pav.* ***Prisijungimo prie sistemos langas***



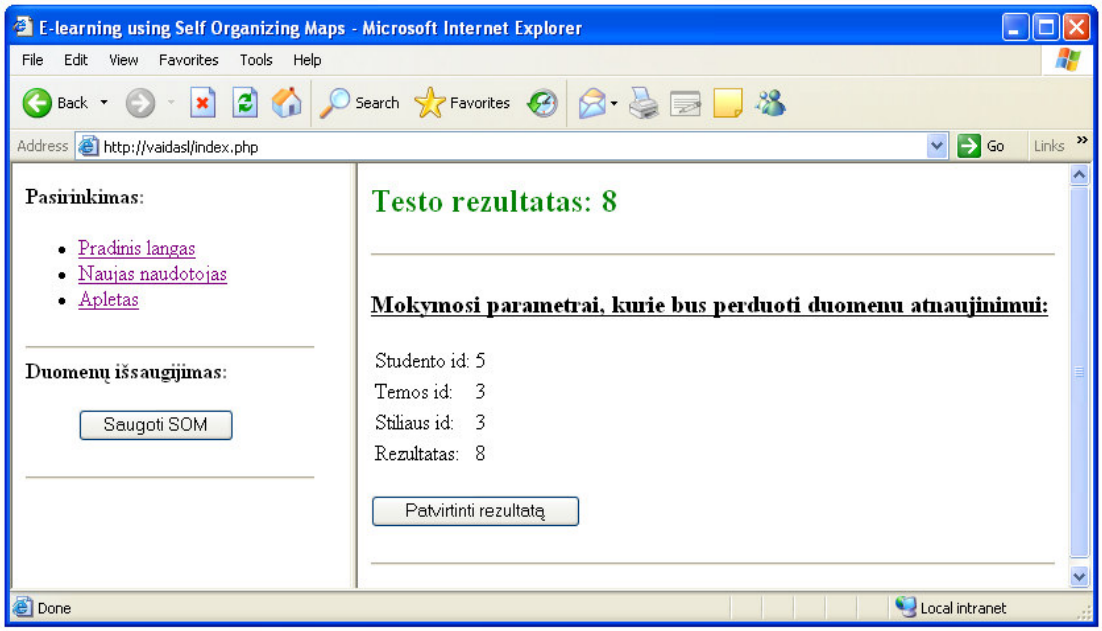
2 *pav.* ***Kurso temų pasirinkimo langas (čia žalia spalva žmimos jau išmoktos temos, mėlyna – tema kurią mokinasi dabar, raudona – dar neišmoktos temos)***



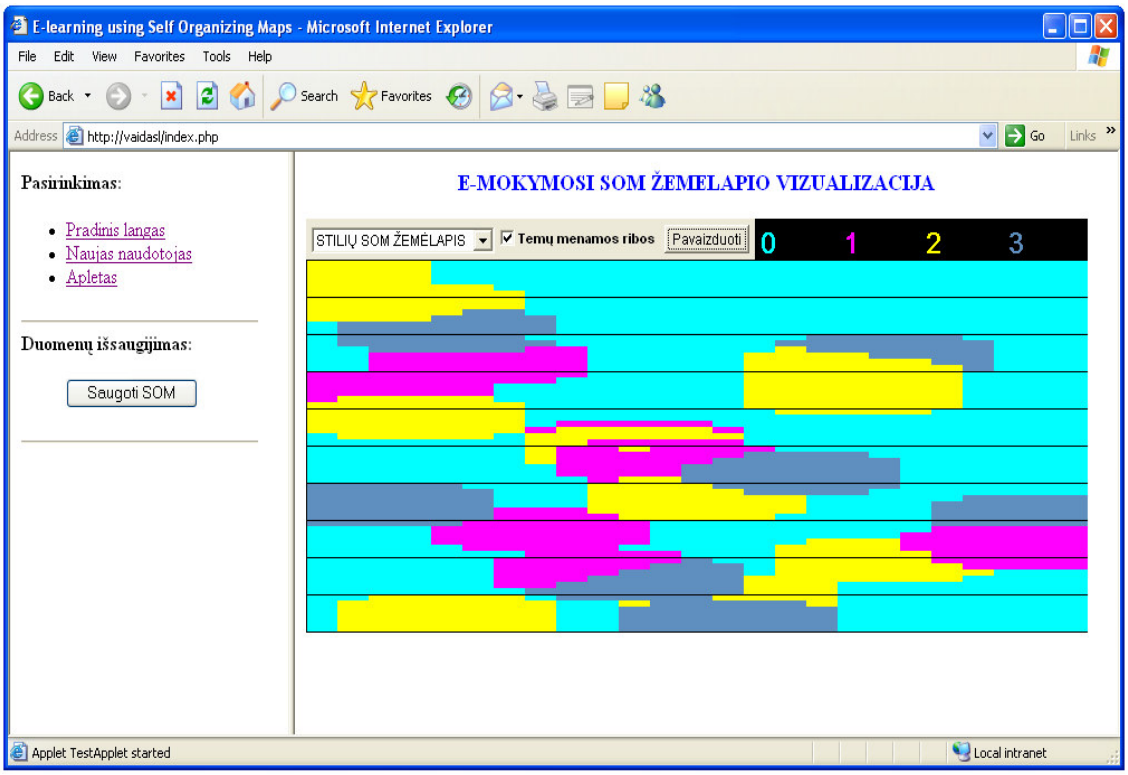
3 pav. ***Pasirinktos temos medžiagos pateikimas***



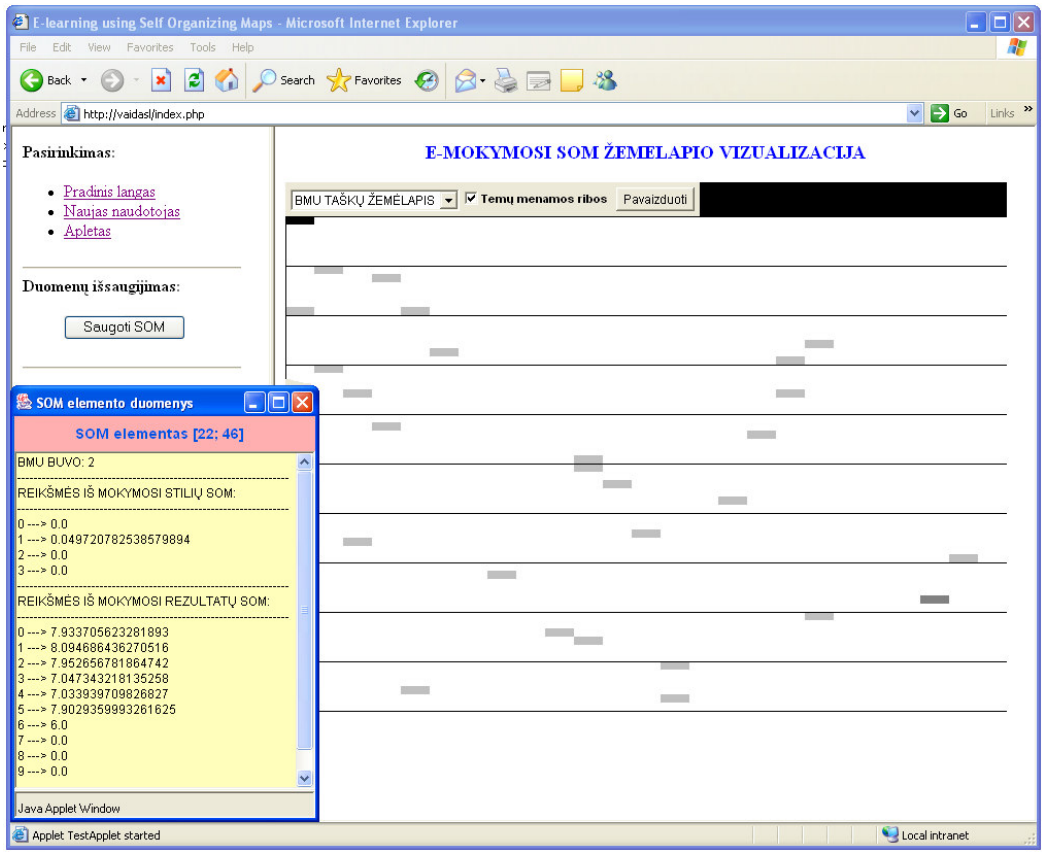
4 pav. ***Žinių patikrinimo testo, iš pasirinktos temos, sprendimo langas (žalia spalva pažymėtas teisingas klausimo atsakymas)***



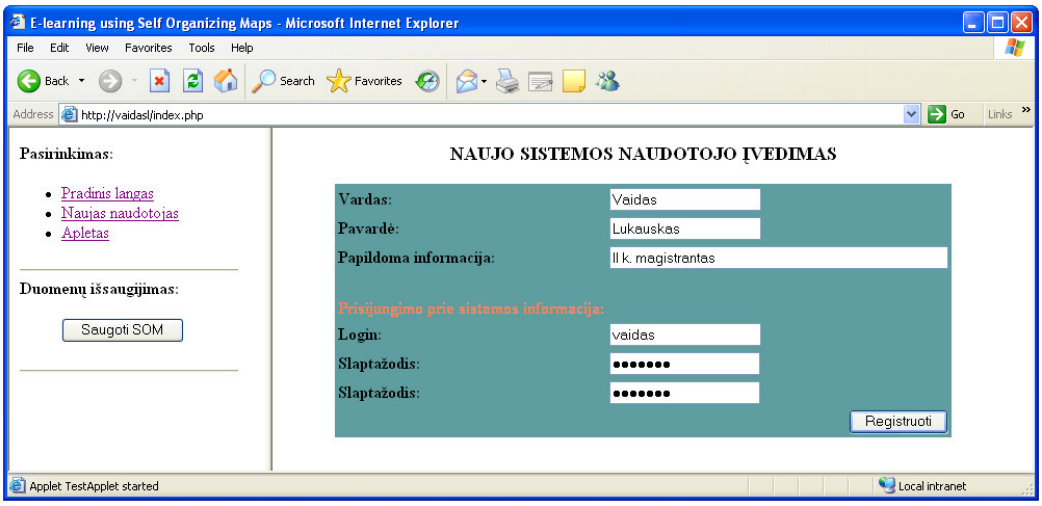
5 pav. ***Gauto įvertinimo ir kitų vartotojo duomenų patvirtinimo langas***



6 pav. ***Adaptyvios e. mokymosi sistemos SOM tnklo vizualizavimas***



7 pav. ***Adaptyvios e. mokymosi sistemos SOM tnklo BMU taškų išsydėstymas (kuo daugiau kartų neuronas buvo BMU tašku, tuo jį vaizduojantis stačiakampis yra tamsesnis)***



8 pav. ***Naujo sistemos naudotojo įvedimo forma***

Priedas 2
Domeno modelio duomenų bazės lentelių aprašymas

1 lentelė. ***SOMTopics lentelės laukų aprašymas (saugoma informacija apie kursą sudarančias temas)***

Lauko pavadinimas	**Aprašymas**
ID	Unikalus, raktinis įrašo identifikatorius.
MAPID	SOM tinklo identifikatorius, su kuriuo yra susieta ši tema.
TOPIC_ID	Temos eilės numeris.
TOPIC_TITLE	Temos antraštė.
INFO	Papildoma (laisvai įvedama) informacija, apibūdinanti temą.
CHTIME	Įrašo paskutiniojo keitimo data ir laikas.

2 lentelė. ***SOMTopicsContents lentelės laukų aprašymas (saugoma kurso temų medžiaga)***

Lauko pavadinimas	**Aprašymas**
ID	Unikalus, raktinis įrašo identifikatorius.
TOPIC_ID	Temos numeris. (išorinis raktas į SOMTopics lentelės ID lauką)
STYLE_ID	Stiliaus, kuriuo yra įvesta temos medžiaga, indeksas
TOPIC_CONTENT	Temos medžiaga (turintys).
INFO	Papildoma (laisvai įvedama) informacija.
CHTIME	Įrašo paskutiniojo keitimo data ir laikas.

3 lentelė. ***SOMTestsAnswers lentelės laukų aprašymas (saugoma informacija apie visus testų klausimų atsakymus)***

Lauko pavadinimas	**Aprašymas**
ID	Unikalus, raktinis įrašo identifikatorius.
QUESTION_ID	Klausimo, kuriam priklauso šis atsakymas, id numeris.
ORDER_ID	Atsakymo eilės numeris (pagal jį nustatomas teisingasis atsakymas)
ANSWER_TEXT	Pats atsakymo tekstas.
CHTIME	Įrašo paskutiniojo keitimo data ir laikas.

4 lentelė. ***SOMTestsQuestions lentelės laukų aprašymas (saugoma informacija apie visus testų klausimus)***

Lauko pavadinimas	**Aprašymas**
ID	Unikalus, raktinis įrašo identifikatorius.
TOPIC_ID	Temos numeris. (išorinis raktas į SOMTopics lentelės ID lauką)
STYLE_ID	Stiliaus, kuriuo yra įvestas testo klausimas, indeksas.
QUESTION_TEXT	Pats klausimo tekstas.
CORRECT_ANSWER	Tesisingo atsakymo indeksas.
CHTIME	Įrašo paskutiniojo keitimo data ir laikas.

Priedas 3

Bendravimui su SOM agentu sukurtos komandos

Žemiau aprašytos komandos yra skirtos bendravimui (apsikeitimui duomenimis) tarp internetinio tinklapio ir intelektualau komponento - SOM agento. Apsikeitimas duomenimis (o tuo pačiu ir komandų perdavimas) vyksta per tinklinius socket'us. Kaip pavysdys yra pateikiamas PHP kodo fragmentas:

```
//prisijungiame prie serverio
$this->data_socket = fsockopen($this->host, $this->port, $this->server_timeout);
//siunčiame prisiloginimo užklausą serveriui
fputs($this->data_socket, "LOGIN;".$loginas.";".md5($passw)."\n");
//priimame mums serverio atsiųstus duomenis
$this->stdid = trim(fgets($this->data_socket));
//siunciame atsijungimo komanda serveriui
fputs($this->data_socket, "EXIT\n");
//uždarome socket'ą
fgets($this->data_socket);
```

1. **LOGIN*;par1;par2*** [autentifikavimosi prie sistemos komanda]

   par1(string) – vartotojo vardas (login);

   par2(string) – vartotojo slaptažodis;

   Grąžinama reikšmė: unikalus sistemos naudotojo numeris (int).

2. **STRFIELD*;par1;par2*** [gauti string tipo informaciją susijusią su prisijungusiu naudotoju]

   par1(int) – unikalus studento numeris (studento id);

   par2(string) – lauko pavadinimas, is lentelės SOMStudentsData, kurio reikšmę mes norim sužinoti;

   Grąžinama reikšmė: nurodyto lauko reikšmė (string).

3. **INTFIELD*;par1;par2*** [gauti int tipo informaciją susijusią su prisijungusiu naudotoju]

   par1(int) – unikalus studento numeris (studento id);

   par2(string) – lauko pavadinimas, is lentelės SOMStudentsData, kurio reikšmę mes norim sužinoti;

   Grąžinama reikšmė: nurodyto lauko reikšmė (int).

4. **NEWUSER*;par1;par2;par3;par4;par5*** [sistemoje užregistruoti naują naudotoją]

   par1(string) – naudotojo prisijungimo vardas (loginas);

par2(string) – naudotojo prisijungimo prie sistemos slaptažodis;

par3(string) – naudotojo vardas;

par4(string) – naudotojo pavardė;

par5(string) – naudotojo papildoma informacija;

Grąžinama reikšmė: naujai užregistruoto vartotojo unikalus numeris (int).

5. **SOMDATA*;par1*** [gauti informaciją susijusią su aktyvuotu SOM tinklu]

   par1(string):

   - MAPID - *Grąžinama reikšmė:*aktyvaus SOM tinklo unikalus numeris (int).
   - INFO - *Grąžinama reikšmė:*aktyvaus SOM tinklo lauko "info" reikšmė. Ji yra tapatinama su mokomo kurso pavadinimu (string).
   - INITPARTSIZE - *Grąžinama reikšmė:*dydis, nurodantis į kokias dalis pagal temas buvo suskirstytas SOM žemėlapis jį inicializuojant (int).
   - XORY - *Grąžinama reikšmė:* 'x', 'y' arba 'none' priklausomai nuo to, pagal kurią koordinatę žemėlapis buvo skaidomas į menamas temų zonas. Tai reikia žinoti piešiant temų ribas, vizualizuojant žemėlapį.
   - TOPICSNO - *Grąžinama reikšmė:*kiek iš viso yra mokomosi kurso temų (int).
   - WININGNODES - *Grąžinama reikšmė:* dvimatis masyvas su duomenimis, kiek konkretus SOM tinklo neuronas buvo BMU taškas (int[][]).
   - ALLMAP - *Grąžinama reikšmė:* trimatis masyvas su mokymosi rezultatų SOM tinklą sudarančiais duomenimis (float[][][]).

6. **STUDENTDATA*;par1;par2*** [gauti int tipo informaciją susijusią su prisijungusiu naudotoju]

   par1(string):

   - RESULTSARRAY - *Grąžinama reikšmė:* string tipo kintamasis (pvz.: 1;1;9;0;0;0;0;0;0;0) su informacija apie išmoktas/mokinamas/neišmoktas temas. Čia 1 – tema jau išmokta; 9 – temą mokinasi dabar; 0 – tema dar neišmokta.
   - LRSTYLE - *Grąžinama reikšmė:*a stiliaus indeksas, kuriuo siųloma studentui mokintis temą (int).
   - LRITEM - *Grąžinama reikšmė:* temos indeksas, kurią dabar mokinasi studentas (string). Jeigu visos temos jau išmoktos – -1.

- OLDSTYLE;param1;param2 [nustatome, kuriuo mokymosi stiliumi studentui su id = param1 sekėsi geriausiai mokytis temą = param2]
    - param1(int) – studento unikalus identifikatorius;
    - param2(int) – jau išmoktos temos indeksas.

  Grąžinama reikšmė: stilius kuriuo nurodytą temą sekėsi mokytis geriausiai (int).

par2(string) – studento unikalus identifikatorius;

7. **SAVERESULT*;par1;par2;par3;par4*** [išsaugoti studento duomenis į duomenų bazę]

   par1(int) – unikalus studento numeris (studento id);

   par2(int) – temos, kurią mokinasi studentas, indeksas;

   par3(int) –stiliaus, kuriuo studentas mokinasi temą, indeksas;

   par4(double) – gautas įvertinimas už testą.

   Grąžinama reikšmė: 0 – jeigu duomenų išsaugoti nepavyko; 1 – jeigu duomenys išsaugoti sėkmingai.