INTELLIGENTSED SÜSTEEMID (ITB8827) - KURSUSE KORRALDUS 2020 KEVADEL

Versioon 6.05.2020

Jaak Tepandi, Jekaterina Tšukrejeva, TTÜ Tarkvarateaduse instituut

Selle materjali viimane versioon: https://moodle.taltech.ee/course/view.php?id=4764 kursus "ITB8827 Intelligentsed süsteemid" ja https://tepandi.ee/

Käesolev materjal selgitab kursuse korraldust, annab iseseisvate tööde struktuuri ning selgitab teisi kursuse korralduse tegevusi.

NB! 2020.a. korraldus erineb oluliselt eelmiste aastate korraldusest. Materjal võib sisuliselt muutuda enne semestrit ja semestri esimesel poolel, peale seda võib tulla väiksemaid täpsustusi detailides. Viimane versioon on kättesaadav eksamisessiooni alguseks. Kasutada tuleb materjali viimast versiooni.

1. ÕPPEAINED, HINDAMINE, TÄHTAJAD JA EESMÄRGID

Õppeaine hinne moodustub hindepunktide summa põhjal. Järgnevas tabelis on toodud hindepunktide teemad, hindepunktide saamise ajavahemik ning antud teema maksimaalselt arvestatav punktide arv.

Hindepunktide teema	Ajavahemik (k.a.)	Max punkte	Märkused
1. töö esitamine	6. nädala lõpuni*	10	Vt jaotis 2. Esitatakse Moodle foorumisse.
1. tööle tehtud ja saadud retsensioonid	9. nädala lõpuni*	2	Vt jaotis 3. Retsensioon on individuaalne, esitatakse Moodle foorumisse. Tehtud retsensioon annab selle autorile max 1 p. Saadud retsensioon annab tiimile max 1 p.
2. töö esitamine	11. näd lõpuni*	10	Esitatakse Moodle foorumisse.
2.töö retsens-d	13. näd lõpuni*	2	Vt 1. töö retsensioonid.
3. töö esitamine	15. näd lõpuni*	10	Esitatakse Moodle foorumisse.
3.töö retsens-d	16. näd lõpuni*	2	Vt 1. töö retsensioonid.
Testid ja kaastöö auditooriumis	16. näd lõpuni*	8	Testid toimuvad loengute või harjutuste ajal. Peale 16. nädalat teste ei ole.
Ettekanded	16. näd lõpuni*	8	Vt jaotis 4.
Eksam		50	Vt jaotis 5

* "Kuni X. nädala lõpuni" = "töö jõuab kohale hiljemalt X. õppenädala pühapäeva õhtul". Näide: "kuni 5. nädala lõpuni" = "töö on kohale jõudnud hiljemalt 1.03.2020 kell 23:59.59" (aga ärge jätke saatmist viimasele hetkele, et vältida ootamatusi).

Hindamine ja vastamine:

- Tööde esitamine tähtajale järgneva nädala jooksul annab 50% max punktide arvust, peale seda - 0p.
- Peale tähtaega esitatud retsensioonid ja ettekanded annavad 0p.
- Õppejõud võib anda lisapunkte õppetöö käigus tehtavate testide, kaastöö ja osavõtu eest.
- Hinde saamiseks on kohustuslikud vähemalt kolm tööd ja eksam. Eksamile saamiseks on vaja töödest koguda vähemalt 15 punkti (vt tabel ülal).
- Eksamieelduste ja eksami punktisummad summeeritakse, hinde vastavus summale: 51…60 punkti - hinne 1; 61 … 70 punkti - hinne 2; 71 … 80 punkti - hinne 3; 81 …90 punkti - hinne 4; 91 ja enam punkti - hinne 5.
- Hindamine toimub jooksva semestri reeglite järgi (nt nõuded töödele, iseseisvate tööde esitamine jne; saadud hindepunktid kehtivad jooksva semestri eksamisessiooni lõpuni).
- Eksamit või arvestust saab vastata ainult siis, kui õppeinfosüsteemis on võimalik hinne välja panna (kui aine on vastatud ja osutub, et hinnet välja panna ei saa, muutub hinne kehtetuks).
- Lisainfot vastamise ja hindamise kohta on Moodle's ja aadressil tepandi.ee. Kehtivad ka muud reeglid, muuhulgas vastab 1 EAP 26 tunnile tööle, mille üliõpilane on õppeks kulutanud (https://www.riigiteataja.ee/akt/126092017010).

Õppeaine eesmärk on saada aru sellest, kuidas intelligentsete süsteemide meetodid ja vahendid muudavad äriprotsesse, strateegiat ja poliitikat ning osata rakendada seda arusaamist ettevõtte eesmärkide saavutamiseks.

Kursuse edukalt läbinud üliõpilane:
1. kirjeldab ja selgitab intelligentsete süsteemide, tehisintellekti ja andmeanalüüsi põhiseisukohti;
2. selgitab ja rakendab teadmiste esitamise ja järeldamise, masinõppe, andmeanalüütika ja otsingu meetodeid;
3. oskab analüüsida igapäevaelu probleeme ja otsustada, milliseid ülesandeid on otstarbekas realiseerida intelligentsete süsteemide meetoditega;
4. oskab spetsifitseerida ja realiseerida lihtsaid intelligentsete süsteemide prototüüpe ning hinnata nende tulemusi.

Võimalik on kavandada ka individuaalne kursus, mis sisaldab käesoleva õppeaine põhilisi mõisteid ning käsitleb süvitsi valitud intelligentsete süsteemide eriteemasid. Selline kursus, selle materjalid ning tööd tuleks õppejõuga kirjalikult kooskõlastada semestri alguses.

2. ISESEISVAD TÖÖD

2.1. Ülevaade, autorid, maht, vormistus

Kursuse käigus esitatakse kolm iseseisvat tööd, igaüks sisaldab kolm osa. Iga töö sisaldab järgmisi osi (viimanes töös on lisaks ka kokkuvõte):

- lähteolukorra ja loodava süsteemi ülesande püstitus,
- esimese prototüübi realisatsiooni kirjeldus,
- testimine ja hinnang.

Eeldatakse, et loodavad süsteemid kasutavad tehisintellekti teooriat ja meetodeid (vrd kursuse materjal, sh jaotis 1.3).

Igal tööl on soovitatavalt neli autorit. Kui autoreid on rohkem, tuleb see e-posti teel kirjalikult juhendajaga kooskõlastada. Tööde mahu arvulised näitajad (näiteks nõuete arv) ei sõltu autorite arvust. Kui kahel järjestikusel tööl on sama lähteolukorra või selle komponentide kirjeldus, siis peaksid neil olema samad autorid.

Töö lõppu võib lisada autorite rollid ja nende poolt tehtud töö mahu (panuse) protsendid. Kui seda pole tehtud, eeldatakse et kõikide autorite panus on võrdne. Panuse põhjal arvutatakse iga autori punktid.

Vormistus peaks jälgima allpool toodud struktuuri. Tööd peaksid olema vormistatud nagu projekti dokumentatsioon (kõik tulemused on selgitatud ja põhjendatud). Käesolevas korralduse failis toodud selgitavaid tekste pole vaja töösse panna. Jaotise lõpus on iseseisva töö tiitelleht, kus töö pealkiri peaks kajastama töö sisu.

Paljudel juhtudel on lähteolukord eelnevalt olemas, näiteks on autori(te)l oma töö juures vaja luua uut süsteemi. Kui seda ei ole, võib lähteolukorra fantaseerida. See on ka ainuke osa tööst, mille võib välja mõelda – süsteem ja järgnevad osad peavad lähteolukorrast lähtuma ja süsteemi tehisintellekti komponendi prototüübid peavad tegelikult realiseeritud, õppimisandmed peaksid olema realistlikud (muidu ei tule realistlikku õppimist) jne. Realiseerida tulevad käesoleva kursusega seotud süsteemi osad (tehisintellekti meetoditel põhinevate komponentide prototüübid), mitte süsteemid ise.

Realiseeritud osade failid tuleb teha kättesaadavaks analüüsile: kas esitada GitHub vm keskkonnas ja viidata sellele töös, lisada eraldi zip-failis, panna töö lõppu lisasse vms. Kõigi tööde dokumenteerimise puhul peaks lähtuma sellest, et realisatsioon oleks töö kirjelduste põhjal teostatav ning hinnatav.

Varasemate tööde kopeerimine on keelatud, töödele tehakse vajadusel plagiaadikontroll.

Alltoodud kirjeldused annavad töö struktuuri ja ei ole piisavad töö tegemiseks - vajalikud teadmised ja oskused antakse kursuse materjalis, loengutes, praktikumides, lisamaterjalides.

2.2. Esimene töö: teadmustöötlus (nt reeglisüsteem)

Esimeses töös kasutatakse püstitatud ülesande lahendamiseks teadmuse esitamise ja töötlemise meetodeid ja vahendeid. Allpool on näitena toodud reeglisüsteemid, aga võib kasutada ka muid vahendeid.

2.2.1.Lähteolukord ja loodava süsteemi ülesande püstitus

Kirjeldatakse lähteolukorda – tavaliselt organisatsiooni, mille jaoks süsteemi luuakse. Miks hakatakse süsteemi arendama, kes süsteemi loomise on algatanud, kes süsteemi finantseerib, kes tegeleb süsteemi hoolduse ja kaasajastamisega, kes hakkab süsteemi kasutama.

Lähteolukorra kirjeldus on väga oluline, sest ta annab aluse süsteemi kavandamiseks ja hindamiseks.

Paljudel juhtudel on lähteolukord eelnevalt olemas, näiteks on autori(te)l oma töö juures vaja luua uut süsteemi. Kui seda ei ole, võib lähteolukorra fantaseerida. See on ka ainuke osa tööst, mille võib välja mõelda – süsteem ja järgnevad osad peavad lähteolukorrast lähtuma ja süsteemi tehisintellekti komponendi prototüübid peavad tegelikult realiseeritud, õppimisandmed peaksid olema realistlikud (muidu ei tule realistlikku õppimist) jne.

Realiseerida tulevad käesoleva kursusega seotud süsteemi osad (tehisintellekti meetoditel põhinevate komponentide prototüübid), mitte süsteemid ise.

Kirjeldatakse süsteemi - mida ta teeb, milliseid vajalikke probleeme lahendab, mida ettevõttele ja inimestele annab, kes on kasutajad. Süsteemi sobivuse käesolevasse kursusse määrab muuhulgas:

- see, kuivõrd ülesanne mahub intelligentsete süsteemide mõiste alla (vt ka kursuse materjal - kaks mõistet), sealhulgas - kui selgelt on olemas lahendusalgoritm või päring. Kui selget algoritmi /päringut ei ole, siis võiks ülesanne sobida. Kui selge lahendusmeetod on teada, pole selle kursuse meetodeid mõtet kasutada;
- kas see süsteem on kellelegi niivõrd kasulik, et keegi (kes?) hakkab seda reaalselt arendama (reaalsesse kasutusse mineva süsteemi arenduse töömaht võib olla päris suur) ja uuendama.

Mõned juhendid sobiva ülesande valikuks on kursuse põhimaterjalis.

Järgnevad teemad, kui nad on realiseeritud lihtsa reeglisüsteemina, võivad võtta hindepunkte alla: arvuti valik; telefoni valik; külmkapi valik; reisi / raamatu / filmi /ajakirja / puhkusereisi / hobi valik. Põhjused: neid on tihti töödesse valitud (arvutid); nende kasulikkust potentsiaalsele kasutajale on raske põhjendada („tean ise, miliseid filme vaatan“); nende jaoks on raske pakkuda arendajat, kes oleks kaasnevate kulutuste tegemisest huvitatud. Selliseid ülesandeid on siiski võimalik tihti esitada nii, et nad muutuvad sobivaks (näiteks, muutes ülesannet konkreetsemaks või orienteerides selle konkreetsele huvipoolele).

Lisakitsendusi ülesande keerukusele ei ole – keerukus tuleneb töödele esitatavatest nõuetest. Näiteks, kui süsteem on kavandatud realistliku ja kasulikuna, siis ei saa ta olla triviaalne. Teisest küljest, püstitatud ülesanne võib olla kuitahes keerukas, peaasi et seda oleks võimalik spetsifitseerida ning et leiduks alamülesanne, mida tuleb projekteerida ja realiseerida kursuse meetodeid kasutades.

Esitatakse vähemalt 10 sisend- ja 4 väljundmõistet. Töös antakse antud sisend- ja väljundkeele (mõistete) kirjeldus tabelina või muus selgelt eristatavas vormis.

Näide 1 - riski hindamise käigus teisendatakse olukorra parameetrid riski hinnanguteks, nt https://www.provenir.com/decision-engine-software/credit-scoring-models/

Näide 2 - reeglisüsteemid võivad "tõlkida" kasutaja probleemi (esitatud kasutaja terminites) mingitesse muudesse mõistetesse (muude "keelde"). Näiteks, probleem *"tahan osta külmkappi, mul on peres kolm inimest, köök on väike ja asub toa kõrval, eelistan osta hulgi koju ja tarbida pika aja jooksul, söön palju lihatooteid, ei taha väga kallist, elan Tallinnas jne"* teisendatakse sooviks *"soovin osta järgmist külmkappi: Kõrgus/laius/sügavus: maksimaalselt …/…/…, Hind maksimaalselt …EEK, Brutomaht: minimaalselt… liitrit , Kasulik maht: minimaalselt … liitrit, Energiakulu: maksimaalselt …kWh/24h, Energiaklass: B, Kompressorite arv: minimaalselt …, Müratase: maksimaalselt … dB, Uste*

arv: … jne)". Viimase kirjelduse põhjal saab juba teha päringu külmkappide andmebaasi, et leida sobiv toode.

Sisendkeele mõisteteks on toodud näites külmkapi ostu soov, inimeste arv perekonnas, köögi ja tubade suurus ning asetus, tarbimisharjumused, ostujõud, elukoht jne.

Väljundkeele mõisteteks on kõrgus/laius/sügavus, hind, brutomaht, kasulik maht, energiakulu, energiaklass, kompressorite arv, müratase, uste arv jne.

Üldjuhul pole mõtet teha reeglisüsteemi, milles viimasena toodud mõisted on sisendiks ning külmikute tüübid on väljundiks – see on andmebaasi ülesanne.

Veel näide – reeglisüsteemi sisenditeks on töötaja vanus, tööstaaž, eelmised kätte saadud puhkused, puhkuseaja soovid, teiste töötajate soovid, firma projektide tähtajad jne, väljund – puhkusepäevade aeg ja arv. Sisenditeks võivad olla ka inimese vanus, analüüside jm andmed; väljundiks diagnoos ja ravimid.

Veel näide: arvutite müügi veebisaitidel pakutakse arvuteid tihti stiilis *„Firma Seeria Versioon i7-8750H / 15.6" FHD AntiGlare / 16GB / 1TB+SSD 128GB / RTX2060 6GB / Win, hind 1399.-" või "… 10Molecule N435 / 3,7 GHz - RAM 4 GB - HDD 500 GB - GMA 3150 - WLAN : 802.11b/g/n - MeeGo - 10.1, Hind : 540.-".* Tavakasutajale jääb see arusaamatuks, pole selge, millest tulenevad hinnavahed ja millist arvutit nad peaksid ikkagi ostma. Mõistlikum oleks pakkuda kasutajale valik kasutusviisile orienteeritud küsimustest, näiteks: *Milleks vajate arvutit? Kas vajate arvutit endale või firmale? Kas soovite tekstitöötlust? tabelarvutust? mängida mänge? vaadata filme?… Kas kuvarit on vaja? Millist? …..Milline hinnavahemik on vastuvõetav?...* ning anda siis vastus stiilis: *…Teie nõudmisi rahuldavad …… (hind 750.-) või …… (veidi aeglasem, hind 560.-)*. Jällegi on tegemist kahe keelega – üks on tavakasutaja keel ning teine on tehniline keel. Süsteemi ülesanne on tõlkida ülesanne kasutaja keelest tehnilisse keelde.

Märkus: Võiks rääkida lihtsalt sisendi(te)st ja väljundi(te)st. Tavaliselt moodustavad need siiski süsteemi (keele, mõistestiku, ontoloogia,...) ja selliselt on ülesannet parem analüüsida.

Võrdlus olemasolevate konkureerivate süsteemidega: kas on süsteeme, mis lahendavad sama ülesannet? Kas neid saaks kasutada? Kui ei, siis mis seal on puudu?

Väga oluline on, et süsteem oleks tõepoolest realistlikuna ja kasulikuna kavandatud. Selleks on vaja põhjendada kasutatavaid meetodeid. Näiteks kui tegemist on arvutite, kursuste, autode või kinnisvara valiku süsteemiga ja mingi muu analoogiline vahend on juba tegelikult realiseeritud, peab süsteemi koostamisele eelnema analüüs: mida saaks olemasolevas lahenduses paremini teha? Kas selleks saab kasutada kursuse meetodeid või vahendeid? Kui jah, siis kui palju see maksab ja kas see tasub ära (raskem küsimus)? Võib olla ka nii, et vastused neile küsimustele tekivad alles peale iseseisva töö valmissaamist - siis tuleks seda märkida kokkuvõttes.

Esitatakse sõltuvuste/reeglite komplekt (vähemalt 10), mis täpsustavad, kuidas sisendmõistetest saadakse väljundmõisted. Näiteks, tarbimisharjumused mõjutavad külmkapi mahtu, kuid arvatavasti mitte eeldatavat mürataset. Võib kasutada kui-siis reegleid, otsustuspuid, närvivõrgu struktuuri analüüsi, freime jne. Lisaks sobivad ka algoritmid, päringud, programmid, kuid kui piisab ainult nendest, siis võiks juba rakendada programmeerimist või andmebaase.

Intelligentse süsteemina realiseerimise otstarbekus. Lähtudes eelnevast analüüsitakse, kas antud kujul formuleeritud ülesannet on mõtet realiseerida intelligentse süsteemina (võrreldes teiste meetoditega on see tavaliselt kulukam) või saab selleks kasutada teisi, soodsamaid meetodeid. Veel näiteid halvasti sobivatest ülesannetest ja lahendustest:

- Võib tunduda, et reeglisüsteemi saaks esitada lihtsa kaalude muutmise abil – vastavalt sisendite väärtustele liidetakse väljundite kaaludele positiivseid või negatiivseid väärtusi. Matemaatiliselt võiks sellisel juhul süsteemi väljundit (väljundväärtuste vektorit) kujutada sisendväärtuste vektori ja kaalude maatriksi korrutise vm tehete abil. Nii koostatud süsteemi on siiski väga raske siluda. Sellise süsteemi tegemiseks on otstarbekam kasutada matemaatilisi funktsioonide lähendamise või õppimise meetodeid. Ka on sellist süsteemi (tehted vektorite ja maatriksitega) lihtsam realiseerida mõnes programmeerimiskeeles - reeglisüsteemid on selliseks realisatsiooniks ebavajalikud ja kohmakad.

- Kui on olemas selge idee, kuidas mingi ülesande lahendamiseks kasutada andmebaasi (näiteks kirjutades andmebaasi päringuid), siis pole mõtet kasutada intelligentseid süsteeme.
- Kui on olemas selge algoritm mingi ülesande lahendamiseks programmi abil, siis pole mõtet kasutada intelligentseid süsteeme.
- Ka otsustuspuud on realiseerimisel lihtsamad ja kui nende abil saab ülesande lahendada, siis pole mõtet kasutada intelligentseid süsteeme.

Saamaks aru, kas ülesanne sobib iseseisvateks töödeks, võib kasutada järgmist testi: kas saab kohe teha andmebaasi päringu või programmi? Kui näiteks arvuti valiku puhul on sisenditeks HDD 1 TB või ekraan 24'', siis saab nende põhjal teha andmebaasi päringu ja leida arvutid. Kui sisendid iseloomustavad kasutajat või töö laadi, siis tavaliselt sellist päringut kohe teha ei saa. Rusikareegel: kui sisendkeel on objekti omaduste mõistetes (nt koera suurus) ja väljund on objektid (nt koeratüübid), siis on enamaltjaolt tegemist andmebaasi ülesandega. Kui sisendkeel on subjekti mõistetes (nt koeraomaniku iseloom või elulaad) ja väljundid on objektid (nt koeratüübid), võib olla tegemist intelligentsete süsteemide ülesandega.

Täpsemalt on ülesande sobivuse kriteeriumid kirjeldatud kursuse materjalis.

2.2.2.Realisatsioon, selle kirjeldus töös

Kokkuvõte ja maht: realisatsioonis peaks olema vähemalt 40 alamkomponenti (nt reeglit).

Realisatsioon. Reeglisüsteemi järeldusmootoreid pakuvad mitmed firmad, neid on nii tasulisi kui tasuta, paljude programmeerimiskeelte ja -keskkondade jaoks. Valik võib sõltuda kursuse läbimise eesmärkidest, ülesande laadist, olemasolevast kogemusest, olemasolevatest ressurssidest, olemasolevast töökeskkonnast jne (soovitussüsteemi viis sisendmõistet on juba olemas☺).

Kui põhieesmärk on saada aru järeldamise loogikast reeglisüsteemis ja eelnev programmeerimise kogemus on väike, siis sobivad näiteks OpenRules (http://amazon.openrules.com:8080/OpenRulesAnalyzer/), CLIPS (https://en.wikipedia.org/wiki/CLIPS). Mõningaid neist selgitatakse harjutustundides. Kindlustegurite kasutamist võimaldab EXSYS (http://www.exsys.com/, sel aastal klassides ei ole), mingil määral ka CLIPS. Veidi keerukama ülesande lahendamine süsteemiga OpenRules eeldab teatavat Java kogemust.

Kui lisaks järeldamise loogikast arusaamisele on vaja kohe lahendada mõni praktiline ülesanne või on soov kasutada tuttavat programmeerimiskeskkonda või -keelt, siis võib kaaluda eeltoodud süsteeme või teha otsingu „rule engine *keel/keskkond*“.

Näiteks on Python põhine reeglisüsteem durable-rules kättesaadav aadressil https://pypi.org/project/durable-rules/. Java-põhine järeldusmootor Jess on süsteemi CLIPS laiendus ja arusaamiseks Jess loogikast võib alustada nii temast endast kui ka CLIPSist. Omakorda, Oracle ärireeglite süsteem (Oracle Business Rules, http://www.oracle.com/technetwork/middleware/business-rules/overview/index.html) põhineb Jess süsteemil. Praktiliselt kasutatavad, on veel mitmed Javal põhinevad süsteemid (nt Drools); saab kasutada erinevaid Pythoni ja muude keelte põhiseid järeldusmootoreid.

Järeldusmootori valik võib olla ka tootja- või arenduskeskkonna põhine. Näiteks, kui olla kogenud Microsofti, IBMi või Amazoni arenduskeskkonna kasutaja, siis võib kaaluda nende firmade poolt pakutavaid järeldusmootoreid. Sageli saab valida mitmete arenduskeskkondade vahel (järeldusmootorid ILOG BRMS, Blaze Advisor jpt). Mitmed firmad pakuvad spetsialiseeritud reeglite halduse vahendeid (nt Facebook Ad Rules Engine).

Tihti on sellised süsteemid keerukamad kui mitteprogrammeerijaile mõeldud järeldusmootorid.

CLIPS on tasuta, durable-rules on tasuta. OpenRules on tasuta kasutamiseks liivakasti versioonis, allalaadimiseks küsitakse minimaalset tasu, millest võib proovida vabaneda (http://openrules.com/why_fee.htm). Firmade ärireeglisüsteemid, OpenRules, EXSYS ja paljud teised on praktiliste realisatsioonide puhul tasulised.

Kui realiseeritavad omadused nõuavad teistsuguseid vahendeid (nt loomuliku keele mõistmine, kõnetuvastus jne), siis kasutatakse neid, kooskõlastades seda eelnevalt õppejõuga.

Kui reaalne vajalik maht on suurem ettenähtust, võib realiseerida vajaliku mahu olulisematest komponentidest ning kirjutada selgituseks, millised osad on realiseeritud.

Realisatsiooni kirjeldus töös. Realiseeritud osade failid tuleb teha kättesaadavaks analüüsile: kas esitada GitHub keskkonnas ja viidata sellele töös, lisada eraldi zip-failis või panna töö lõppu lisasse.

Kasutusjuhend peaks kindlasti sisaldama kirjelduse loodud komponentidest ja nende füüsilisest asukohast. Vajadusel tuleks kirjeldada süsteemi installeerimist, käivitamist ning kasutamist (kui see erineb standardsest).

Kõigi tööde dokumenteerimise puhul peaks lähtuma sellest, et realisatsioon oleks töö kirjelduste põhjal teostatav ning hinnatav.

2.2.3.Testide projekteerimine, testimine, hinnang

Testide projekteerimine - vähemalt 10 testi süsteemi hindamiseks. Üks osa tööst oli luua praktiline ülesande püstitus, testimine peaks toimuma sellest lähtudes. Seega valitakse testid lähtuvalt reaalse kasutamise olukordadest (kasutamise stsenaariumist), mitte kavandatavast realisatsioonist. Iga testi puhul kirjeldatakse järgmisi komponente:

- testi identifikaator
- testolukord (sisuline!) - kirjeldada konkreetset kasutajat konkreetses probleemolukorras
- millist omadust testib
- testi sisendid
- kirjeldatud kasutajale tegelikult kasulikud väljundid (eksperdi hinnang)

Funktsionaalsete testide projekteerimisel püütakse valida võimalikult erinevaid kasutuse stsenaariume. Näiteks külmikute puhul: üksik/perega tudeng; suur/väike pere maal; madala/kõrge sissetulekuga pere linnas; jne.

Testi väljundite prognoosil tuleks jälgida, et prognoositud väljundid tõesti annaksid adekvaatse tulemuse antud kasutaja ja stsenaariumi jaoks. Näiteks:

- kui tegemist on üksi elava tudengiga, kes soovib ajutist külmkappi, poleks testi oodatava vastusena uus 300-liitrine külmkapp ilmselt õige
- kui soovitakse üllatuskingitust vanaemale, pole testi vastusena raftingu pakkumine tavaliselt mõttekas ega reaalne
- kui stsenaarium on "tahaks aukliku- ja porisevõitu teedega maakodu jaoks autoga aeg-ajalt kipsplaate, muud remondivärki, istikuid jne vedada", siis uus kerge sportauto testi väljundina ei ole adekvaatne
- jne

Mis oleks mõttekad väljundid eelnevate näidete puhul? Selliste stsenaariumide väljapakkumine eeldab, et süsteemi autoril on valdkonnast mingi teadmine. See on ka loomulik, mitteõppiv süsteem ei ole üldjuhul targem kui tema autor.

Testimine. Loodud realisatsiooni testitakse projekteeritud testide põhjal. Selleks antakse loodud süsteemile ette testide sisendid, registreeritakse süsteemi poolt antavad väljundid ning hinnatakse, kas need väljundid on adekvaatsed. Vajadusel lisatakse veel teste.

Kui mõnda mittefunktsionaalset testi pole tehnilistel põhjustel võimalik läbi viia (näiteks, koormustestid), siis püütakse prognoosida või muudmoodi hinnata süsteemi vastavust testis antud nõudmistele. Kirjeldatakse testitulemused: kas oodatavad ja tegelikud tulemused klappisid? Kui ei, siis kui tõsised olid kõrvalekalded?

Esimese realisatsiooni, meetodi ja vahendi hinnangud. Antakse hinnang esimesele realisatsioonile ja testimisele, vastates järgmistele küsimustele:

- Kas ja kuivõrd süsteem rahuldab püstitatud nõudmisi?

- Milline oli testimise põhjalikkus?
- Kas ülesande püstitust, nõudmisi jne tuleks muuta?
- Kas süsteemi saab vastu võtta ja miks?
- Millised probleemid tuleks lahendada järgmises realisatsioonis?

Antakse hinnang meetodile ja vahendile, vastates järgmistele küsimustele:

- Kas kasutatud meetod sobib ülesande lõplikuks lahendamiseks?
- Kas intellektitehnika meetodite kasutamine oli õigustatud (kas saaks piirduda lihtsamate meetoditega)?
- Kas oleks vaja keerukamaid meetodeid?
- Kas vahend sobis, mis olid plussid - miinused, kas mingi teine vahend oleks võinud olla parem?

2.3. Teine töö: masinõppel põhinev süsteem

Teises töös kasutatakse püstitatud ülesande lahendamiseks masinõpet.

2.3.1.Lähteolukord ja loodava süsteemi ülesande püstitus

Lähteolukord ja loodava süsteemi ülesande püstitus kirjeldatakse nagu 1. töös. Kõigis iseseisvates töödes tuleb kirjeldada sisend- ja väljundkeelt (sisend- ja väljundmõisteid) ning nendevahelisi sõltuvusi. Märkus: kui kasutatav masinõppe meetod või tarkvara ei toeta mitut õpitavat väljundit, siis võib valida ühe kõige olulisema väljundi ja kasutada masinõpet selle väljundi jaoks.

Kui mingid osad, näiteks organisatsiooni ja süsteemi kirjeldus, kattuvad 1. töö vastavate osadega, siis võib nendele viidata. Kui kahel järjestikusel tööl on sama lähteolukorra või selle komponentide kirjeldus, siis peaksid neil olema samad autorid.

Ülesande lahendamiseks masinõppe meetodite abil kavandatakse ja esitatakse vähemalt 70 komplekti õppimisandmeid. Sisendite ja väljundite väärtused tulevad kodeerida nii, et neid saaks valitud õppiva süsteemi treenimiseks kasutada. Kodeerimisel on tihti vaja eksperimenteerida, et leida treenimiseks sobiv esitus. Näiteid kodeerimisest närvivõrgu jaoks:

- kui sisend on arv (nt korteri pindala), on see otse kasutatav sisendis, vajadusel normaliseerides
- kui sisend antakse järjestatud kvalitatiivsel skaalal (nt väike, keskmine, suur), saab selle teisendada arvväärtuseks
- kui väljundi väärtus kuulub järjestamata hulka (nt gerbera, roos, tulp), siis on üks võimalus teha iga sellise väärtuse jaoks eraldi väljund, mille suurus iseloomustab antud väärtuse kindlustegurit (nt skaalal 0...1)

Õppimisandmed lisatakse töö dokumentatsiooni või lisasse.

2.3.2.Realisatsioon, selle kirjeldus töös

Realisatsioon. Ülesanne realiseeritakse närvivõrkude vm õppiva süsteem abil, kasutades eelmises osas esitatud õppimisandmeid.

Masinõppe tarkvara on nii tasulist kui tasuta, paljude programmeerimiskeelte ja -keskkondade jaoks. Valik võib sõltuda kursuse läbimise eesmärkidest, ülesande laadist, olemasolevast kogemusest, olemasolevatest ressurssidest, olemasolevast töökeskkonnast jne.

Kui põhieesmärk on omandada esmane kogemus masinõppe ülesannete lahendamisel ja eelnev programmeerimise kogemus on väike, siis sobivad näiteks Weka (http://www.cs.waikato.ac.nz/ml/weka/), See5 (http://www.rulequest.com/see5-info.html) ja mitmed teised. Neid selgitatakse harjutustundides.

Väga palju on Python põhist tarkvara, alustuseks võivad sobida näiteks masinõppe (https://github.com/aimacode/aima-python/blob/master/learning.ipynb), närvivõrkude

(https://github.com/aimacode/aima-python/blob/master/neural_nets.ipynb), kinnitusega õppe (https://github.com/aimacode/aima-python/blob/master/rl.ipynb) ja Markovi otsustusprotsesside (https://github.com/aimacode/aima-python/blob/master/mdp.ipynb) õppematerjalid raamatu "S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach" põhjal.

Keerukamate ülesannete lahendamiseks võib kaaluda muuhulgas järgmist tarkvara:

- MXNet: An open source deep learning framework, https://mxnet.apache.org/
- Keras: The Python Deep Learning library, https://keras.io/
- TensorFlow: open source platform for machine learning, https://www.tensorflow.org/

Kui on vaja lahendada praktilist ülesannet või soov kasutada tuttavat programmeerimiskeskkonda või -keelt, siis võib kaaluda eeltoodud süsteeme või teha otsing vastavalt ülesande laadile, näiteks:

- Masinõppe tarkvara: https://en.wikipedia.org/wiki/Machine_learning#Software
- Närvivõrgud: https://en.wikipedia.org/wiki/Neural_network_software
- Süvaõppe tarkvara - https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software
- Python jaoks on suur valik masinõppe tarkvara, näiteks TensorFlow ja Keras: https://machinelearningmastery.com/setup-python-environment-machine-learning-deep-learning-anaconda/, https://github.com/keras-team/keras

Realisatsiooni kirjeldus töös. Kasutusjuhend peaks kindlasti sisaldama kirjelduse loodud komponentidest ja nende füüsilisest asukohast. Vajadusel tuleks kirjeldada süsteemi installeerimist, käivitamist ning kasutamist. Dokumenteerimise puhul peaks lähtuma sellest, et realisatsioon oleks töö kirjelduste põhjal teostatav ning hinnatav.

Projekti dokumentatsiooni lisatakse loodud treenitud süsteemi esitus (nt närvivõrgu puhul kaalude fail).

2.3.3.Testide projekteerimine, testimine, hinnang

Testimine ja hindamine on analoogiline esimese tööga. Enamasti on masinõppe süsteemide puhul võimalik esitada eraldi õppimis- ja testimisandmed ning saada kätte testimise tulemused ja nende analüüs.

Kui testandmed erinesid esimese realisatsiooni testandmetest, tuleb need lisada töösse.

Esitatakse soovitused edasiseks, vastates järgmistele küsimustele:

- Kas esimeses ja teises töös realiseeritud süsteemidest võib mõnda vastu võtta? Kui jah, siis millist?
- Millised on vastuvõtmise riskid?
- Kui ükski loodud süsteemidest pole veel vajalikul tasemel, siis kas püstitatud ülesandeid tasub edaspidi siiski realiseerida?
- Millised vahendid oleksid selleks otstarbekad?

2.4. Kolmas töö: intelligentsed agendid, agendisüsteemid ja simulatsioonid

Kolmandas töös kasutatakse püstitatud ülesande lahendamiseks intelligentseid agente, simulatsioone jm vahendeid autonoomsete süsteemide loomiseks.

2.4.1.Lähteolukord ja loodava süsteemi ülesande püstitus

Lähteolukord kirjeldatakse nagu 1. töös. Kui see on sama nagu 1. töö lähteolukord, võib sellele viidata.

Üldjuhul on esimese ja kolmanda töö süsteemi kirjeldused erinevad. Samas tuleb kõigis iseseisvates töödes kirjeldada sisend- ja väljundkeelt (sisend- ja väljundmõisteid) ning nendevahelisi sõltuvusi. Võimalusel tuleks püstitada realistlikud probleemid, mida loodavad tarkvara agendid ja/või nende simulatsioon püüab lahendada. Näiteid: suurürituse logistika, veebiliikluse analüüs, tellimuste täitmine, maksumuudatuse mõju prognoos, liiklusummikute vähendamine jt.

Simulatsiooni töö puhul on vaja kasutada vähemalt kolme sisend- ning ühte väljundmõistet (atribuuti, muutujat). Samuti tuleks kirjeldada vähemalt viis sisendite ja väljundite vahelist seost. Kui mingid osad simulatsiooni töö ülesande püstitusest kattuvad 1. töö vastavate osadega, siis võib nendele viidata.

Kui tööde on sama lähteolukorra või ülesande püstituse komponentide kirjeldus, siis peaksid neil olema samad autorid.

Näited mudelitest ja nende arendamisest:

NetLogo (vt allpool) pakub palju erinevad näiteid

https://en.wikipedia.org/wiki/Agent-based_model#In_business,_technology_and_network_theory

http://modelingcommons.org/browse/one_model/5086#model_tabs_browse_info

https://en.wikipedia.org/wiki/Model-based_design

http://ccl.northwestern.edu/papers/MEE/

http://www.intro-to-abm.com/index.html

2.4.2.Realisatsioon, selle kirjeldus töös

Saab kasutada näiteks NetLogo (kohandades / kasutades olemasolevaid mudeleid või luues uusi, (https://ccl.northwestern.edu/netlogo/index.shtml, vt ka õppetunde nt https://ccl.northwestern.edu/netlogo/docs/tutorial3.html) või erinevates programmeerimiskeeltes kirjeldatud tarkvara (näide: https://github.com/aimacode/aima-python/blob/master/vacuum_world.ipynb, https://realpython.com/simpy-simulating-with-python/ - vt ka kommentaare!). Vahendite võrdlus: https://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software.

Kavandatud mudel tuleb realiseerida, salvestada töö lisasse ning lisada tööle (töö lisas, eraldi fail, github vms).

2.4.3.Testide projekteerimine, testimine, hinnang

Töös tuleb esitada realisatsiooni, meetodi ja vahendi hinnangud samuti nagu esimese töö puhul.

Loodud mudeli ja realisatsiooni hindamiseks tuleb:

- veenduda, et realisatsioon töötab;
- iseloomustada tulemusi, analüüsida kogemuse või kirjanduse põhjal, kas ja mil määral on saadud väljundid realistlikud;
- analüüsida, kas ja kui suurel määral on süsteemi kirjelduses püstitatud eesmärgid saavutatud.

Soovitusi simulatsiooni hindamiseks: https://en.wikipedia.org/wiki/Agent-based_model#Verification_and_validation.

2.4.4.Meetodite ja vahendite üldhinnang

Analüüsitakse kasutatud meetodeid / vahendeid ning nende otstarbekust:

- Millisel määral realiseeriti intelligentsete süsteemide omadusi?
- Kas sellised realisatsioonid olid otstarbekad või oleks saanud lihtsamalt hakkama (näiteks, kasutades mõnes programmeerimiskeeles esitatud algoritme)?
- Kas oleks vaja hoopis keerukamaid meetodeid?
- Kas vahendid sobisid, mis olid plussid - miinused, kas mingid teised vahendid oleksid võinud olla paremad?

- Kasutatud vahendite võrdlus: milline kasutatud vahenditest oli antud olukorras parem ning miks?
- Autorite poolne tagasivaade – millised olid tehtud tööde head küljed ja vead? Mida tuleks järgmises projektis paremini teha, kuidas?

2.5. Näiteid probleemidest iseseisvates töödes

Ideid selleks, et hinnata oma töid, on toodud retsenseerimise jaotises 3. Sealsed soovitused teise üliõpilase töö analüüsiks sobivad ka oma töö hindamiseks. Allpool on ka näiteid tüüpilistest probleemidest.

- Organisatsioon / lähteolukord on puudulikult määratletud (nt pole aru saada, kes süsteemi vajab või kes hakkab finantseerima selle haldamist).
- Pole arusaadav, kes ja kuidas hakkab süsteemi kasutama, kus süsteem paikneb.
- Võrdlus teiste süsteemidega on puudu. Tavaliselt mingid vahendid on juba olemas, nad ei pea olema "intelligentsed", aga võib-olla neist piisab? Kui ei, siis miks?
- Ülesande lahendamiseks pole tegelikult vaja intelligentsete süsteemide tehnoloogiat, kuid see ei tule välja projekti lõpus toodud analüüsist. Ülesande laad võib olla alguses ebaselge ning selguda lõpuks – sellisel juhul peab see olema analüüsis märgitud.
- Realiseeritud on kaaludega süsteem, kus sisendite väärtused määravad mingitele vaheparameetritele lisatavad või neist lahutatavad väärtused (vt ülal p 2.2.1). Tulemus on tegelikult määratud kaalude maatriksiga. Sellist süsteemi saaks luua, kasutades näiteks regressioonianalüüsi ja realiseerida lihtsa algoritmi (maatriksi ja vektori korrutised) abil.
- Reeglisüsteemi (või muud kursuse vahendit) kasutatakse selleks, et realiseerida süsteemi loogika juhuslikke komponente, loogikat terviklikult uurimata (kusjuures selline terviklik uurimine oleks võimalik).
- Testid testivad süsteemi loogikat, mitte ülesande loogikat.
- Mingi realisatsiooni testid on tegemata, puudub realisatsioon, selle kirjeldus, hindamine vms.
- Arusaamatu lauseehitus, kirjavead, puuduv sisukord, puuduvad lehekülgede numbrid.

2.6. Korduma kippuvaid küsimusi

Lisaks alltoodud küsimustele vt ka tepandi.ee/kkk.

2.6.1.Millised ülesanded sobivad iseseisvaks tööks?

Sobivad näiteks otsustus- või masinõppe ülesanded, mille lahendamiseks puudub selge algoritm, näiteid on toodud ülal iseseisvate tööde kirjelduses.

Kui ülesannet saab lahendada andmebaasi päringuga (või kui selleks on olemas mõni muu selge algoritm, mida saab programmiliselt realiseerida), siis tasubki nii teha, seega pole mõtet selle kursuse meetodeid kasutada.

Samuti pole mõtet selle kursuse meetoditega lahendada ülesandeid, mille realiseerimiseks piisab, kui iga sisend annab igale väljundile teatud pluss- või miinuspunktid. Sellist väärtuste lisamist tabeli põhjal on lihtne programmiliselt realiseerida.

Üsna sageli osutub ka (näiteks reeglisüsteemi puhul), et süsteemi väljundi põhjal saab moodustada andmebaasi päringu, mis teeb lõpliku valiku.

Liiga keerukad ülesanded pole semestri jooksul lahendamiseks realistlikud, samas on tihti võimalik ülesannet laiendada, kitsendada või teisendada, nii et ta muutuks sobivaks iseseisva töö teemaks.

2.6.2.Milliseid vahendeid kasutada iseseisvates töödes?

Iseseisvates töödes võib kasutada mitmesugust tüüpi vahendeid (vt ülal tööde kirjelduste juures). Närvivõrkude, reeglisüsteemide ja otsustuspuude võrdlus on toodud kursuse materjalis.

2.6.3. Kas võib kasutada teistsugust tööde struktuuri või vormistust?

Ülaltoodud tööde struktuur ja vormistus võimaldab tööst aru saada ja seda hinnata, lähtudes algsest ülesande püstitusest (miks seda ei võimalda näiteks ainult programmi tekst?).

Kui on olemas konkreetne projekt (nt töö juurest), mille puhul on otstarbekas kasutada selle kursuse meetodeid, siis võib seda projekti realiseerida, dokumenteerides vastavalt töökoha nõuetele. Samas peab kogu dokumentatsioon siiski võimaldama projektist ja kogu arenduse loogikast aru saada, lähtudes algsest ülesande püstitusest (see võib siis olla hankekutse, analüüsi dokumendi vms kujul). Selline projekt tuleks õppejõuga eelnevalt kirjalikult kooskõlastada.

3. RETSENSIOON

Iga üliõpilane võib retsenseerida sama kursuse raames samal semestril tehtud töid. Retsenseerijale annab see kogemuse, kuidas vaadata kõrvalt teist analoogilist tööd ning näha selle väärtusi ja probleeme. Töö autorile annab retsensioon head tagasisidet ning võimaluse oma tööd parandada (kui retsensioon on esitatud mõistlikul ajal). Analüüsitakse jooksva semestri töid, ka siis, kui aine oli deklareeritud varem.

Retsensiooni tegemine

- Retsenseerida saab vaid see üliõpilane, kelle oma vastav töö on foorumisse laetud.
- Retsenseeritakse neid töid, millel on kõige vähem retsensioone - eesmärk on vältida olukordi, kus ühel tööl on mitu retsensiooni (suur arv retsensioone ei anna töö autorile erilist lisaväärtust) ja teisel pole ühtegi (autoril jääb saamata tagasiside ja retsenseerimise hindepunktid).
- Retsensiooni võib teha vaid lõpetatud töödele.

Retsensiooni esitamise tähtajad ning hindepunktide arvestus on toodud jaotises 1.

Analüüsi struktuur on soovitavalt järgmine:

1. Retsenseerija andmed (nimi, rühm, kontakt)
2. Vaadeldava töö andmed
   - autorite nimed, rühm
   - töö liik (1., 2. või 3. töö), nimetus, lühike sisu, vajadusel versioon (vahe-või lõppversioon) või staatus, üleandmise kpv
   - analüüsi sisendid (näiteks: iseseisvad tööd, ainetöö, programm, tekst, dokumentatsioon, testitulemused,...)
3. Töö analüüs
   - Töö analüüs toimub jooksva semestri nõuete järgi
   - Põhimõtted vt allpool
4. Soovitus punktisumma kohta - max 10p
   - lähteolukorra ja loodava süsteemi ülesande püstitus – max 3p (kui teises või kolmandas töös on sama ülesanne kui esimeses, võib nendes töödes anda rohkem punkte realisatsioonile)
   - realisatsioon – max 3p
   - testimine ja hinnangud – max 3p
   - esitus ja vormistus - max 1p

Tööde analüüsil arvestatakse muuhulgas järgmist:

- Ülesande laad ja maht vastavad kursuse nõuetele
- Kõik vajalikud osad on olemas
- Töö mahu arvulised nõuded on täidetud
- On loodud realistlik lähteolukord ja ülesande püstitus, millest järelduvad töö ülejäänud osad. Võrdluses on toodud olulisemad analoogilised süsteemid
- Analüüs on realistlik ning sellest selgub, et intelligentsete süsteemide omaduste, funktsioonide või meetodite kasutamine on põhjendatud - ilma selleta ei saaks lähteolukorrast tulenevaid süsteemi ülesandeid lahendada (vähemalt püstituse ajal pole lahendus selgelt näha). Nende meetodite maksumus ja keerukus on vastuvõetav (või kui ei ole, siis pole see algfaasis silmnähtav)
- Realisatsioon on töö kirjelduste põhjal teostatav ning hinnatav
- Realisatsioonid ja testimised arvestavad eelnenud tegevusi
- Töös kasutatakse kursuse meetodeid/vahendeid
- Kokkuvõte on realistlik, kajastab tulemusi õieti. Näiteks, on täiesti normaalne, kui peale prototüüpide realiseerimist on ülesanne selgem ning seda saab nüüd realiseerida muude vahenditega. See ei tohiks siiski olla silmanähtav töö alustamisel, siis poleks see vastuvõetav teema
- Vormistamine: tiitelleht, sisu arvestavad lähteolukorda. Dokumentatsiooni kõik osad on olemas, sealhulgas vajadusel lähtekoodid (nt. reeglisüsteemi reeglite tekstid või otsustuspuu struktuur). Vormistamine on korralik.
- Näiteid probleemidest iseseisvates töödes on jaotises 2.5.

4. ETTEKANNE

Ettekannete idee on süvendatult tutvuda mõne huvitava kursusega haakuva teemaga ning see koos läbi arutada. Ettekanded tehakse aine eriteemadele, kokkuleppel õppejõuga ka muudele teemadele. Ettekande sisu ja aeg lepitakse eelnevalt kokku, saates õppejõule ettekande teema ja esialgsed materjalid/slaidid vähemalt nädal enne ettekannet. Ettekanded tehakse selles järjekorras, kuidas ettekande materjalid said aktsepteeritud. Teema võib ise välja pakkuda - kursuse konspekti küsimustes on mitmeid teemasid, mille põhjal saaks teha ettekande.

Ettekanne on üldjuhul individuaalne ja see esitatakse semestri jooksul loengute ja harjutuste käigus. Pikem teema võib olla esitajate vahel alapunktide kaupa ära jaotatud, hindamine on ka sel juhul individuaalne. Ettekande pikkus on 10...20 minutit, millele järgneb arutelu ca 10 minutit. Ühel tavalisel loengul / ühes harjutuses saab esineda maksimaalselt 2…3 inimest.

Saadud hindepunktide arv sõltub ettekande kvaliteedist (sisu, esitus, selgus, materjalid), kuulajate arvust ja esituse ajast. Hindepunktide arvu sõltuvus kuulajate arvust: lisaks õppejõule 5 või enam kuulajat – 100%; 2... 4 kuulajat – 60%; 0...1 kuulajat – 20%. Ettekanne, mille materjalid aktsepteeriti ajavahemikus 1. kuni 10. nädalani annab 100% saadud hindepunktidest, ajavahemikus 11. kuni 16. nädalani aktsepteeritud ettekanne - maksimaalselt 75%, peale 16. nädalat - 0 p.

Soovitusi ettekandeks:

- Tutvustage ennast ja ettekande üldist struktuuri
- Jaotage ettekanne osadeks, tutvustage neid
- Andke selge ülesande püstitus
- Lisage slaid "Peale seda ettekannet/õppetundi/arutelu te teate…" ning ettekande sisu kohta käiv(ad) küsimus(ed)
- Andke edasi loogika, kuidas lahendused tulenevad ülesande püstitusest
- Maksimaalselt hinnatav ettekanne sisaldab vähemalt ühte konkreetse vastusega testi (valikvastustega küsimus, arvutusülesanne vms)

- Rääkige kuulajatele ja kuuldavalt
- Kasutage illustreerivat materjali (kui on slaidid, siis nende tekst peaks olema ruumi tagumisest reast loetav)
- Märkige ära, kust ettekande väited tulenevad (autor, ajakiri, raamat, veebileht) ja analüüsige, kas on esitatud ka teistsuguseid väiteid sama küsimuse kohta
- Jälgige kella, jätke aega küsimusteks

5. EKSAM

Kui eksam on suuline, siis esitatakse viis lühikest küsimust või lihtsat ülesannet, igaüks annab maksimaalselt 10 hindepunkti. Seega kogusumma on <= 50 hindepunkti. Vastamine on vestluse vormis, ilma eelneva ettevalmistuse ja materjalideta.

Kirjalik eksam toimub küsimuste või ülesannete vormis. Seoses eriolukorraga võivad eksami toimumise meetodid muutuda.

Eksamiküsimused on toodud kursuse materjali iga jaotise lõpus eraldi alajaotises (küsimusi ja ülesandeid). Olulisemad kordamisküsimused ja ülesanded on eraldi esitatud kursuse materjali viimases jaotises. Eksamiküsimused võivad olla ka iseseisvate tööde kohta.

Kogu materjal on kiiresti muutuv ja lubab erinevaid tõlgendusi. Üldine põhimõte on, et oma seisukohad on ka eksamil teretulnud. Seejuures tuleb neid põhjendada ja osata võrrelda kursuse tõlgendusega, mida niisiis tuleb ka teada.

6. ISESEISVA TÖÖ TIITELLEHE NÄIDE

Esimene autor on põhiline kontakt kirjavahetuses.

Soovitav on mitte panna üliõpilaskoodi tiitellehele.

TALLINNA TEHNIKAÜLIKOOL
TARKVARATEADUSE INSTITUUT

Suuliste käskluste tuvastamise alamsüsteem lennusimulaatorile

1 töö õppeaines “Intelligentsed süsteemid" (ITB8827)

Koostajad: *[esimene autor on põhiline kontakt kirjavahetuseks]*
Tõnu Tamm, õpperühm: ....., e-post.....

Piret Pärn, õpperühm: ....., e-post.....

……., õpperühm: ....., e-post.....

……., õpperühm: ....., e-post.....

Esitatud: .....

Juhendaja: .....

TALLINN 2020