Tallinna Ülikool
Matemaatika- loodusteaduskond
Informaatika osakond

Õpihaldussüsteemide koostalitlus IVA SQI liidese näitel

Bakalaureusetöö

Autor: Tanel Toova
Juhendaja: Jaagup Kippar

Autor:....."..... "....2006.a.
Juhendaja:....."..... "....2006.a.
Osakonna juhataja:..... "..... "....2006.a.

Tallinn 2006.a.

Sisukord

Sissejuhatus

Seoses internetiühenduste arvu plahvatuslikku kasvuga viimastel aastatel ning hariduspoliitika ümberorienteerumisega individuaalõppele, aga ka järjest suureneva ümber- ning täiendõppe vajadusega, on e-õpe muutunud väheste aktivistide pärusmaast oluliseks ning järjest rohkem populaarsust saavutavaks alternatiiviks senistele kinnistunud õppemeetoditele.

Käesoleva bakalureusetöö teemavalik on tingitud e-õppe populaarsuse kasvust ning sellest tulenevast vajadusest kvaliteetsete ning mitmekesiste e-õppe materjalide kättesaadavuse järele. Ülemaailmse arvutivõrgu, Interneti, struktuur on küll hea lahendus info vabaks ja segamatuks edastamiseks – ehk siis täitmaks eesmärke, milleks see algselt loodi, kuid info killustatus, struktureerimatus ning sageli ka tõespärasuse kaheldavus ei võimalda efektiivselt kvaliteetsete õppematerjalide hankimist. Ometi on vajalikud kvaliteetsed oma ala professionaalide poolt loodud õppematerjalid sageli täiesti olemas. Need paiknevad erinevates nii kommertslikes kui vabavaralistes õpihaldussüsteemides, mis on enamasti loodud kasutamiseks vaid konkreetse asutuse või organisatsiooni töös. Seega pole probleemiks mitte niivõrd e-õppe materjalide, õpiobjektide olemasolu, kui nende leitavus ning kättesaadavus. Siit tulenebki vajadus erinevate õpihaldussüsteemide koostalitluse järele.

Õpihaldussüsteemide koostalitlusega seotud problemaatika on huvitav ning siiani suhteliselt vähekäsitletud uurimisala, mistõttu koosnebki käesolev bakalaureusetöö teoreetilisest osast, mis annab ülevaate teemaga seotud mõistetest, õpihaldussüsteemidest, õpiobjektidest ja andmevahetusprotokollidest ning praktilisest osast, mis kirjeldab kahe õpihaldussüsteemi (IVA ja EducaNext-i) vahelise koostalitluse loomise protsessi.

Töö eesmärgid võib sõnastada järgmiselt:

- selgitada õpihaldussüsteemidega seotud mõisteid,
- anda ülevaade olemasolevatest õpihaldussüsteemidest,
- anda ülevaade probleemidest ning olemasolevatest lahendustest õpihaldussüsteemide koostalitluse tagamisel.
- anda ülevaade koostalitluse tagamiseks vajalikest andmevahetusstandarditest, pöörates erilist tähelepanu SQI standardile,
- kirjeldada ning analüüsida IVA SQI liidese loomise protsessi,
- luua IVA SQI liidese dokumentatsioon .

1. Õpihaldussüsteemide koostalitluse võimaluste tutvustus

1.1. Mõisted

Käsitlemaks õpihaldussüsteemide koostalitluse võimalusi tuleb esmalt defineerida kasutatavad terminid. See esmapilgul lihtne ülesanne on siiski üsna komplitseeritud, kuna vaadeldavaid mõisteid võib käsitleda mitmel tasandil ning mitmest seisukohast lähtuvalt. Seega toon ära enimlevinud ning minu arvates täpseimad definitsioonid, millest kokku peaks moodustuma tervikpilt käesolevate mõistete tähendustest.

1.1.1. Õpihaldussüsteemi mõiste

Õpihaldussüsteem (Learning Management System) on tarkvarapakett, mis võimaldab õppematerjalide haldamist ja õpilasteni toimetamist. Enamik õpihaldussüsteeme on veebipõhised, et võimaldada juurdepääsu õppematerjalidele igal ajal ja kohas.

Suur osa õpihaldussüsteeme võimaldavad õpilaste registreerimist, e-õppe materjalide otsimist, kasutamist ja allalaadimist, testide koostamist ja lahendamist ning juhendajaga tundide läbiviimist. Keerulisemad süsteemid võimaldavad veel virtuaalsete tundide läbiviimist, õpiressursside haldust, õpilaste sertifitseerimist jpm. Suur osa selliseid süsteeme on suunatud iseõppijatele, võimaldades iseseisvat registreerumist ning juurdepääsu õppematerjalidele.

Üldjuhul loetakse õpihaldussüsteemide hulka kuuluvaks ka õppematerjali loomist võimaldavaid süsteeme, mis lisaks õppematerjali loomise vahenditele sisaldavad veel organiseerimise ja õpihaldussüsteemi toimetamise vahendeid.

Enamasti põhinevad õpihaldussüsteemid Java EE ja Microsoft .NET arhitektuuridel, kasutades mõnd enamlevinut andmebaasisüsteemi. Kuigi enamik õpihaldussüsteemidest on tasulised, toimub viimasel ajal selles vallas tugev liikumine avatud lähtekoodiga tasuta tarkvara poole.

Tuntuimaid õpihaldussüsteeme:

- EducaNext
- Moodle
- Olat
- WebCT

[Allikas 3]

1.1.2. Koostalitluse mõiste

Koostalitlusvõime kohta on loodud mitmeid definitsioone, mis on järgnevalt ära toodud.

"Funktsionaalüksuste võime omavahel suhelda, programme täita või andmeid edastada nii, et kasutaja ei pea tundma nende üksuste eriomadusi."

[Allikas 5, lk 116]

"Toodete, süsteemide või äriprotsesside omadus töötada koos ühise eesmärgi saavutamiseks."

[Allikas 6]

"Tarkvara valdkonnas tähendab koostalitlus erinevate programmide suutlikust vahetada andmeid ühiste protseduuride abil, lugeda ja kirjutada samu failiformaate ning kasutada samu protokolle. Koostalitluse puudumine viitab standardiseerituse puudumisele tarkvara tootmisprotsessis."

[Allikas 6]

"Kahe või enama süsteemi või komponendi võime informatsiooni vahetada ning vahetatud informatsiooni kasutada."

[Allikas 15]

„... see tähendab, et eraldiseisvalt arendatud tarkvarakomponendid suudavad vahetada informatsiooni viisil, et neid saab koos kasutada.“

[Allikas 16, lk 35]

"Koostalitlus on võime efektiivselt edastada ja kasutada sama informatsiooni mitmes organisatsioonis ja infotehnoloogia süsteemis."

[Allikas 17, lk 15]

"Infosüsteemide võime samaaegselt koos töötada, mis hõlmab kommunikatsiooniprotokollide, riistvara, tarkvara, rakenduste ja andmete ühilduvust."

[Allikas 18]

"Olukord, mis eksisteerib, kui erinevused infosüsteemide vahel ei ole takistuseks ülesannete lahendamisel mitmete süsteemide koostöö abil."

[Allikas 19]

1.1.3. Õpiobjekti mõiste

Õpiobjekt on e-õppe korduvkasutatavate õppematerjalide ja neid kirjeldavate metaandmete terviklik ühik. Et õpiobjektid oleks kasutatavad erinevates kontekstides ja virtuaalsetes õppekeskkondades, on materjali esitamisviis eraldatud (st. ei sisaldu õpiobjektis endas) muust mittenähtavast materjalist ja metaandmetest.

Tüüpiline õpiobjekt koosneb järgnevatest komponentidest:

- kursust kirjeldavad andmed - kursuse identifikaator, sisu keel, õppeaine, kirjeldus, märksõnad;
- elutsükkel – versioon, staatus;
- õppematerjal - tekstid, veebilehekülјed HTML formaadis, pildid, helid, videod;
- terminite sõnastik – terminid, definitsioonid, akronüümid;
- testid ja hindamisjuhendid – testid, vastused;
- õigused – hind, autoriõigused, kasutamistingimused;
- seos teiste kursustega – eelduskursused;
- haridustase – klass, vanusegrupp, õppimisele kuluv aeg, raskusaste.

Tuntumaid õpiobjektide formaate:

- MERLOT – "Multimedia Educational Learning and Online Teaching" on tasuta ning avatud ressurssidel põhinev peamiselt üliõpilastele suunatud õpiobjektide formaat. MERLOT pakub kõige muu seas ka linke internetis leiduvatele õppematerjalidele koos lühiülevaadete ja harjutustega.
- UK LOM Core – United Kingdom Learning Object Metadata Core on hetkel veel mustandi staadiumis olev Suurbritanniast pärinev õpiobjektide formaat, mis on välja töötatud võrreldes 12 metaandmete skeemi. Metaandmed on jagatud kolme kategooriasse: vajalikud, valikulised ja soovituslikud.
- CanCore – on sarnaselt UK LOM Core-le õpiobjektide formaat, kuid sesaldab üksikasjalikke juhendeid iga LOM standardi mainitud andmeelemendi kirjeldamiseks ja tõlgendamiseks.

[Allikas 7]

1.2. Tuntumad õpihaldussüsteemid

1.2.1. Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment) on vabavaraline e-õppe platvorm, mille arendamist alustati 1999.a. Tegemist on ühe populaarsema õpihaldussüsteemiga, mida on tõlgitud 61 keelde ning millel on üle kahe poole miljoni registeeritud kasutaja. Moodle töötab kõigi operatsioonisüsteemidega, mis toetavad PHP-d ja mõnd levinumat andmebaasimootorit nagu näiteks MySQL, PostgreSQL või mõnd muud läbi ADO ühendusliidese töötavat andmebaasi.

Moodle ülesehitus ja kasutusloogika on tugevasti mõjutatud sotsiaalse konstruktivismi filosoofiast, mille kohaselt toimib konstruktivistlik õppimine kõige efektiivsemalt ühiselt arendatavas õppekeskkonnas. Konstruktivistlik õppimine omakorda tähendab, et õpitakse toetudes oma kogemusele, õpetaja loob vaid kogmuse omandamiseks vajaliku keskkonna. Moodle-s väljendub see jagatud kasutajarollide kasutuselevõtus, mis lubavad igal kasutajal olla nii õpilane kui õpetaja. Õpetajana pole kasutaja ülesanne olla ainult teadmiste pakkujaks, vaid ka omamoodi arvamusliidriks või lihtsalt vestluspartneriks, mõjutades niimoodi õpilast liikuma kursuse eesmärgi täitmise poole.

[Allikas 8]

Moodle arhitektuur on modullaarne, võimaldades lihtsat ning kiiret kohandamist erinevateks vajadusteks ning uue funktsionaalsuse lisamist. Moodle pakett sisaldab järgmisi mooduleid:

- leheküljehaldus – lehekülje disain, aktiivsed moodulid, kasutatavad keeled;
- kasutajahaldus – kasutajate registreerimine, rollide määramine;
- kursusehaldus – uute kursuste loomine, haldamine;
- ülesannete moodul – uute ülesannete loomine, õpilaste poolt täidetud ülesannete haldamine;
- jututoa moodul – reaalajaline tekstipõhine suhtluskanal kasutajatele;
- foorumi moodul – sisaldab kõiki üldlevinud foorumivahendeid;
- küsitlusmoodul – moodul küsitluste läbiviimiseks;
- testide moodul – testide koostamine, neile vastamine, vastatud testide haldamine;
- ressursihaldusmoodul – meediaressursside haldamine ja esitamine;
- töötoamoodul – virtuaalne keskkond grupitöö läbiviimiseks.

[Allikas 9]

1.2.2. OLAT

OLAT (Online Learning And Training) on õpihaldussüsteem, mis toetab igasugust veebipõhist õpet ilma didaktiliste piiranguteta. OLAT on avatud lähtekoodiga ning kõigile tasuta kättesaadav, seda on arendatud alates 1999.a. Zürihi Ülikoolis. See õpihaldussüsteem on täielikult realiseeritud Java vahenditega ning põhineb modulaarsel struktuuril – uue funktsionaalsuse lisamine on tehtud võimalikult lihtsaks. OLAT toetab mitmeid e-õppe standardeid nagu näiteks IMS Content Packaging, IMS QTI ja SCORM.

[Allikas 10]

Mõned OLAT-i tähtsamad omadused:

- põhineb täies ulatuses Javal, vajab ainult Tomcat serverit ja mõnda levinumat andmebaasimootorit (MySQL, PostgreSQL, HSQL);
- ainult HTML-il põhinev kasutajaliides;
- mehhanism täienduste lisamiseks ja kasutaja vajadustele vastavaks muutmiseks;
- mitmekeelne (täielikult UTF-8 toetav);
- turvalisus põhineb õigustel, mis tagab paindlikuma halduse, kui rollidel põhinev turvalisus;
- personaliseeritud portaalilaadne kasutajaliides;
- omab virtuaalset failisüsteemi, mis lubab ligipääsu kõigile failidele üle ühe ühenduspunkti;
- iga kasutaja saab jagada oma materjale üle HTTP või WebDAV protokolli;
- omab kursuste loomise tööriista, mis võimaldab luua täpselt kasutaja vajadustele vastavaid kursusi – ühtegi pedagoogilist kontseptsiooni ei eelistata ega suruta kasutajale peale;
- omab võimalusterikast kasutajate ja gruppide haldus- ning kommunikatsioonivahendit – õpivahendite võimaldamine kasutajatele ja gruppidele, isikliku õppekava loomine ja kursustele registreerumine, automatiseeritud teadetesüsteem RSS voo ja e-maili abil. Sisaldab integreeritud reaalajas töötavat tekstipõhist suhtluskanalit;
- omab testide ja ülesannete haldusüsteemi – erinevad testitüübid, automaatne või käsitisi hindamine, personaliseeritud ülesanded;
- dokumentatsioon nii arendajatele kui kasutajatele.

[Allikas 11]

1.2.3. WebCT

WebCT (Web Course Tools) on tasuline veebipõhine e-õppekeskkond, mis on suunatud peamiselt ülikoolidele ja teisele kõrgematele õppeasutustele. Kuna uute kursuste loomise vahend on äärmiselt paindlik, sobib see süsteem hästi algajatele kui ka juba kogenud e-õppe materjali loojatele. Õppejõud saavad oma WebCT kursustele lisada interaktiivseid tööriistu nagu näiteks jututubasid, foorumeid ja meililiste samuti ka staatilist materjali nagu veebilehekülgi ja PDF dokumente. WebCT-d arendati algselt Briti Kolumbia Ülikoolis Murray W. Goldberg-i poolt. Esimene WebCT versioon valmis 1997.a. Süsteemi edust kannustatuna asutas Murray firma, WebCT Educational Technologies Corporation, mis 1999.a. pakkus oma toodet juba 2-3 miljonile õpilasele 30 riigis. 2002.a. ostis ULT (Universal Learning Technology) WebCT ning praeguseks on sellel väidetavalt üle 10 miljoni kasutaja 80 riigis.Hetkel pakutakse WebCT-st kahte versiooni: WebCT Vista ja WebCT Campus Edition. Esimene neist on kõigi vahenditega täislahendust pakkuv versioon ja teine mõeldud organisatsioonidele, millel on juba olemas andmete hoidmis- ja jagamissüsteem ning kursustele registeerumise süsteem. Tarkvaral pole määratud hinda, hind sõltub kasutusmeetodist ja organisatsiooni suurusest.

WebCT Vista sisaldab:

- virtuaalsete kursuste keskkonda;
- õpiobjektide haldusvahendit;
- kasutajate haldusvahendit;
- PowerLinks Kit – tarkvaraarendusvahendit;
- PowerSight Kit – raportite loomise vahendit.

WebCT Campus sisaldab:

- virtuaalsete kursuste keskkonda;
- tarkvara tugistruktuuri – andmebaasisüsteeme, meediaservereid jne.;
- võimalust integreerida WebCT Vista mooduleid.

Seda tarkvara kasutatakse ka elektrooniliseks publitseerimiseks. Selleks, et kasutada mõningaid WebCT formaadis õpikuid või muid õppimisvahendeid, nõuavad mõned materjalide loojad, et õpilased ostaksid materjali kättesaamiseks salasõna. Tarkvara võimaldab lokaalselt loodud materjali ja kommertsmaterjali ühildamist.

[Allikas 12]

1.3. Õpihaldussüsteemide koostalitluse vajadusest

Internet annab kõigi oma kasutajate käsutusse suure hulga õppematerjali, kuid tänu informatsiooni üleküllusele, paljude olemasolevate õpisüsteemide, meediaserverite ja grupitöötarkvara kinnisele struktuurile, on nende materjalide leidmine sageli keeruline ja ebaefektiivne. Samas on juba olemasolevate õpiobjektide loomiseks ja kirjeldamiseks kulutatud märkimisväärseid ressursse.

Kuna suur hulk õpiobjekte on loodud tasuta kasutamiseks ning levitamiseks, on nende loojad huvitatud oma loomingu võimalikult laiast levikust. (või vähemasti ei ole selle vastu) Õpiobjektide lai levik ja lihtne leitavus tagaks ka uute materjalide loojate suurema huvi ning kaasatuse, mille tulemuseks on järjest kvaliteetsemate ning mitmekesisemate õpiobjektide loomine. Samuti tagab suurem õpiobjektide valik õpihaldussüsteemide kasutajate suurema huvi ning on seega abiks e-õppe levikule ja arengule.

Kokkuvõtlikult võib vabavaraliste õpihaldussüsteemide koostalitluse vajalikkuse põhjused välja tuua järgnevalt:

- õpiobjektide loomine on kulukas;
- õpiobjektide kirjeldamine (metaandemete loomine) on kulukas;
- õpiobjektide loojad on huvitatud oma loomingu võimalikult laiast levikust;
- õpihaldussüsteemide kasutajad on huvitatud võimalikult laiast õpiobjektide valikust.

[Allikas 7]

Kuni viimase ajani on õpihaldusüsteemide loomine ning rakendamine olnud väikeste töögruppide pärusmaa – nende sihtgruppideks on sageli vaid ühe kooli õpilased või ettevõtte personalikoolitusosakond. Seoses internetiühenduste kättesaadavuse kasvuga on vajadus õpitehnoloogia järele muutunud globaalseks. Seega on uuest tehnoloogiast ning loomulikult ka turuosast huvitatud suured tarkvaratootjad, kelle poolt pakutavad kommertstooted lisaksid valikuvõimalusi ning aitaksid kaasa e-õppe veelgi populaarsemaks muutumisele. Globaalsed lahendused aga ei saa eksisteerida ilma standarditeta. Muuhulgas tagavad hästidefineeritud standardi ka koostalitluse, mis aitab saavutada järgmisi hüvesid:

- kasutaja perspektiivist kaob vajadus valida ainult ühe pakkuja teenuseid; kulud on madalamad kuna kaob vajadus kohandatud süsteemide järele; suurem tarbijate hulk muudab tõenäolisemaks, et õpiobjektide tootjad investeerivad rohkem oma toodangu

mitmekesistamiseks – isegi väga spetsialiseeritud valdkondades;

- töövahendite tootja perspektiivist vaadatuna kaob vajadus erinevate toodete jaoks täiesti erinevaid kasutajaliideseid luua. See toob kaasa madalama tootmiskulu ja suurendab potentsiaalset turgu. Töövahendite tootjad saavad omavahel võistelda oma toodete kvaliteedi ja väärtuslikkuse alusel jättes kõrvale vormi;
- õpiobjektide tootja seisukohalt tagavad ühtsed standardid õpiobjektide võimalikult laia leviku ja kasutatavuse;
- õppija perspektiivist tekib rohkem valikuvõimalusi erinevate toodete ja sisu vahel. Samuti muutub võimalikuks sujuv üleminek ühelt õpihaldussüsteemilt teisele;
- disaineri jaoks tekib rohkem taaskasutatavat koodi ning kaob vajadus erinevate süsteemide loomise järele – piisab uue funktsionaalsusega moodulite loomisest.

[Allikas 13, lk 1]

1.4. Lahendused koostalitluse tagamiseks

Lihtsaim lahendus koostalitluse tagamiseks on uute õpisüsteemide loomisel rahvusvaheliselt tunnustatud standardite järgimine. Seda pealtnäha lihtsat nõuannet on aga praktikas palju keerulisem rakendada, sest nagu selgub, on erinevaid standardeid palju ning neid tekib järjest juurde. Lisaks sellele pole ühest standardite komplekti kõigi õpisüsteemide osade jaoks – nii peavadki õpisüsteemide arendajad tegema valikuid oma nägemuse, õpisüsteemi koostalitluse võime ja selle tehniliste lahenduste keerukuse astme vahel.

Hea lahendus õpisüsteemi koostalitluse tagamiseks on ehitada terve süsteem üles modulaarsel arhitektuuril, mis lubab võimalikult lihtsalt ja väikeste kuludega lisada süsteemile uut funktsionaalsust. Nii suureneb tõenäosus, et isegi juhul, kui igasugune ühilduvus teiste süsteemidega jääb algul loomata, on valminud õpisüsteemil vähemalt tulevikus potentsiaal koostalitlusvõime saavutamiseks.

Lisaks modulaarsusele on oluline veel täpse ja üksikasjaliku dokumentatsiooni olemasolu. Süsteemi põhjalik kirjeldamine ja selle kirjelduse teistele arendajatele kättesaadavaks tegemine suurendab oluliselt tõenäosust, et tulevikus loodavad samalaadsed süsteemid on olemasolevaga koostalitlusvõimelised, mis omakorda tagab kasutajate suurema huvi mõlema süsteemi vastu.

2. Andmevahetuse standardid

Nagu ülal kirjeldatud on parimaks lahenduseks õpihaldusüsteemide koostalitluse tagamisel ühiste standardite kasutuselevõtmine. Alljärgnevad standardid on väljapakutud Sun Microsystems-i poolt. E-õppe standardid võib vastavalt Sun Microsystem-i poolt väljapakutud e-õppe mudelile (vt. Lisa 8) jagada järgmistesse kategooriatesse:

[Allikas 13, lk 2]

2.1. Metaandmed

Sisu (õppematerjalid) on e-õppe keskne osa. Õppematerjalid ja nende kogumid peavad olema varustatud järjepidevate andmetega, et tagada indekseerimine, hoiustamine, otsimine ja kättesaadavus erinevate töövahendite poolt läbi mitmete repositooriumite. Selleks otstarbeks kasutatavaid andmeid nimetataksegi metaandmeteks. Metaandemete standardeid on loodud mitmeid:

- LOM (Learning Object Metadata) – IEEE Õpitehnoloogiate Komitee poolt loodud standard, mis on tunnustatud ja omaksvõetud selliste organisastioonide poolt nagu: The IMS Global Learning Consortium, The Advanced Distributed Learning Initiative, The Alliance of Remote Instructional and Distribution Networks for Europe jpt.;
- The Dublin Core Metadata Initiative on loonud oma metaandmete standardi, mis oli algselt mõeldud küll raamatukogudele, kirjastustele, valitsusasutustele ja teistele sedalaadi organistatsioonidele, kuid millest leidub ka hariduslikuks kasutamiseks sobiv versioon. Hetkel töötab ülalmainitud organisatsioon koos IEEE-ga et luua ühist raamistikku mõlemale standardile;
- IMS Learning Design meeskond töötab haridusalaste modelleerimiskeelte kallal (Educational Modelling Languages), mis on mõeldud kogu kursust hõlmava pedagoogilise metodoloogia kirjeldamiseks. Hetkel tegeldakse kõrgtasemekeelte ja masintõlgendatavate keelte ühildamisega. Projekt on veel algstaadiumis, kuid sedagi võib vaadelda metaandmete standardina.

[Allikas 13, lk 3]

2.2. Sisu pakkimine

Sisu pakkimise spetsifikatsioonid ja standardid lubavad kursuste transporti ühest õpisüsteemist teise. See on äärmiselt oluline, sest õppematerjale võib luua ühe töövahendiga, muuta teisega, hoida suvalises repositooriumis ja esitada hoopis mingi kolmanda vahendiga. Sisu pakkimine hõlmab nii õpiobjekte kui seda, kuidas neist moodustada suuremaid õppematerjale. Samuti võivad need sisaldada reegleid, kuidas sisu õpilaseni toimetada. Olemasolevad spetsifikatsioonid ja standardid sisu pakkimiseks:

- The IMS Content Packaging spetsifikatsioon (Microsoft on sellest loonud kommertsversiooni LRN, mida kasutavad mitmed tootjad);
- The IMS Simple Sequencing specification (hetkel väljatöötamisel);
- Aviation Industry CBT komitee juhised ja soovitused arvutihallatavate intruktsioonide jaoks;
- ADL-i (The Advanced Distributed Learning initative) SCORM (Sharable Content Object Reference Model).

Hinnangud ja küsimused, millel need põhinevad, on eriline õppematerjal, mida toetavad teistsugused spetsifikatsioonid, kuid tulemus on sama – küsimused, testid ja ülesanded võivad olla loodud ühes keskkonnas, kuid on kasutatavad ka teistes. Selliste õppematerjalide pakkimiseks on enimlevinud standard QTI (The IMS Question and Test Interoperability spetcification).

[Allikas 13, lk 4]

2.3. Õppija profiil

Õppija profiili standardid lubavad erinevatel õpihaldussüsteemidel jagada andmeid õppija kohta. Õppija profiil võib sisaldada isiku kohta käivaid andmeid, õppeplaani, õpingute ajalugu, sertifikaate ja kraade, hinnanguid omandatud teadmiste kohta ja hetkel pooleli olevate õpingute staatust. Tähtsaimad standardid õppija profiili jaoks on:

- LIP – (The IMS Learner Information Package),
- PAPI – (The Personal and Private Information).

Lisaks ülaltoodud kahele võib standarditena arvestada veel vCard-i ja SPEEDE/Express-i.

[Allikas 13, lk 4]

2.4. Õppija registreerimine

Õppija registreerimisandmed lubavad administreerimise ja õpimaterjali kohaletoimetamisega tegelevatel moodulitel kindlaks teha, milliseid materjale pakkuda ja milliseid materjale vajatakse. Enimlevinud standard on:

- IMS Enterprise töögrupi poolt loodud spetsifikatsioon õppematerjali pakkumise ja õppijate registreerimise andmete jagamiseks, mis töötab K-12 keskkonnas School Interoperability Framwork-i toel.

[Allikas 13, lk 5]

2.5. Sisu edastamine

Uue sisu lisamisel on vajalik edastada andmed selle kohta õppijale. Kui õppija sisuga töötab, tekib tema töö põhjal uus informatsioon, mille sisestab õpetaja, või mis genereeritakse süsteemi poolt automaatselt õppematerjali looja poolt antud instruktsioonide alusel – kursuse hinne, hinnang ülesande täitmisele vms. Uue sisu lisamise kohta käivate andmete, õpitegevuse andmete ja tulemuste jagamiseks erinevate komponentide vahel on samuti vaja standardiseeritust.

Sel alal arendatavad standardid lubavad komponentidel jagada andmeid väga erinevatel tasemetel: individuaalsete ülesannete üksikutest küsimustest kuni kursuse hinde või täitmisprotsendini. See saavutatakse luues standardiseeritud kommunikatsiooniprotokollid ja andmemudelid, mis lubavad õppematerjalidel edastada andmeid süsteemidele, kust nad ise pärinevad. Hetkel töötatakse kahe lahenduse kallal:

- SCORM 1.1 sisladab JavaScript-i API-t, mis lubab kommunikatsiooni õppematerjali edastusüsteemi ja veebilehitsejale edastatud sisu vahel;
- The Aviation Industry CBT komitee CMI (computer managed Instruction) spetsifikatsioon sisaldab ka komponentide vahelise kommunikatsiooni kohta käivat osa.

[Allikas 13, lk 5]

- Z39.50 ja Zing – Z39.50 on klient-server protokoll andmete otsimiseks ning informatsiooni edastamiseks. Tegemist on küllalt vana standardiga – selle esimene versioon valmis 1988.a. Esialgu oli see mõeldud kasutamiseks raamatukogudes. Õpiüsteemide juures kasutatakse tänapäeval Z39.50 edasiarendust ZING (Z39.50 International: Next Generation).

[Allikas 14]

3. SQI iseloomustus

SQI (Simple Query Interface) on õpiobjektide repositooriumitesse päringute esitamiseks loodud API (Application Program Interface). SQI põhineb mitmete rahvusvaheliste töögruppide ja asutuste koostööl ning on osaliselt finantseeritud CEN/ISSS (European Commitee of Standardisation / Information Society Standardization System) poolt.

[Allikas 1, lk 1]

Kuna SQI on loodud väga erinevate e-õppe süsteemide ja õpiobjektide repositooriumite loojate koostöös, on selle olulisemateks omadusteks paindlikkus erinevate süsteemide ühendamisel ning kasutuselevõtu lihtsus. SQI on loodud pidades silmas järgmisi põhimõtteid:

- SQI on neutraalne vastuseformaatide ja päringukeelte suhtes: repositooriumid, mida SQI ühendab, võivad olla oma olemuselt äärmiselt erinevad – seega ei tee SQI mingeid eeldusi vastuseformaatide ja päringukeelte osas;
- SQI toetab nii sünkroonseid kui asünkroonseid päringuid, et tagada võimalikult lai kasutatavus;
- SQI võimaldab nii olekuga kui olekuta implementatsiooni (vt. lk. 18);
- SQI põhineb lihtsal sessioonihalduse kontseptsioonil, et eraldada autentimise probleemid päringuhalduse omadest.

API enda disain põhineb kahel järgmisel põhimõttel:

- käsu ja päringu eraldatuse printsiip (Command-Query Separation Principle),
- lihtne ja laiendatav käsustik.

[Allikas 2, lk 2]

SQI päringuteenuse API jaoks on vaja kindlaks määrata hulk meetodeid, mida repositoorium oskab täita, et saada päringuid ja vastata päringuele, mida esitavad teised rakendused. Et eristada pärijat süsteemist, mis vastab, kasutatakse mõisteid "allikas" ja "sihtmärk".

Metaandmeid on võimalik hoida erinevate vahendite abil, näiteks failipõhised repositooriumid, relatsioonilised andmebaasid, XML repositooriumid või RDF tööriistad, mis kokku moodustavad mitmekesise keskkonna. Selleks, et muuta õpiobjektide repositooriumeid koostalitlusvõimelisteks, ei piisa ainult ühtse liidese defineerimisest, vaid tuleb määrata ka ühine päringukeel koos ühtse vastuseformaadiga õpiobjektide kirjelduste (ja teiste metaandmete) jaoks.

Sellised koostalitlusvõime aspektid nagu ühine päringuskeem ja vastuseformaat on osa õpisüsteemi semantilisest mudelist.

Päringuteenust kasutatakse eelnevalt kindlaksmääratud ühtses päringukeeles päringu esitamiseks sihtmärgile. Järgmiseks tagastatakse samuti eelnevalt kindlaksmääratud vastuseformaadis päringuvastused allikale. Rakendustasandil võib osutuda vajalikuks luua mähised, et tõlkida päring päringukeelest X päringukeelde Y ning viia päringuvastused ainult ühe süsteemi poolt toetatud formaadist ühisesse formaati ja vastupidi.

Lisa 4 illustreerib andmevahetusprotsessi, kus õpiobjektide repositoorium A (allikas) saadab päringu õpiobjektide repositooriumile B (sihtmärk). Eeldatakse, et mõlema süsteemi osas on eelnevalt kokkulepitud ühine päringukeel. Päringulauses kasutatud mõisted on osa ühisest päringuskeemist. Õpiobjektide repositooriumis B võib osutuda vajalikuks, et liidese osa peab tõlkima päringulause ühisest päringukeelest repositooriumi spetsiifilisse päringukeelde. Samuti on võimalik, et liides vajab ülekandeid ühisest päringuskeemist kohalikku päringuskeemi enne kui alustatakse otsinguprotsessi. Seda ülesannet täidavadki mähised. Kui otsinguprotsess on andnud tulemusi, edastatakse need allikale ühises vastuseformaadis.

[Allikas 4, lk 2]

3.1. Päringukeel ja vastuste formaat

Selleks, et saavutada täielikku päringute esitamise funktionaalsust, tuleb SQI-le lisada järgnevad kokkuleppelised andmed:

- päringus kasutatavad atribuudid ja sõnavarad;
- päringukeel ja selle esitusviis;
- päringut rahuldavate õpiobjektide nimekirja esitusviis;
- õpiobjektide metaandmete esitusviis.

Nagu näha on SQI päringukeelte ja vastuste formaatide osas neutraalne – igasugune kokkulepe kahe või enama repositooriumi vahel on SQI jaoks sobiv. Selliseid kokkuleppeid võib näiteks väljendada RDF (Resource Description Framework) või muude XML (Extensible Markup Language) skeemide abil.

[Allikas 2, lk 1]

3.2. Sünkroonsed ja asünkroonsed päringud

SQI-d on võimalik kasutada kahel erineval viisil:

1) sünkroonsete päringute korral tagastab sihtmärk (repositoorium, millele saadetakse päring) päringu tulemused allikale (repositoorium, õpihaldussüsteem või otsingusüsteem, mis teeb päringu). Seega on tulemuste tagastamine algatatud allika poolt päringu saatmisega ning teiste meetoditega, mis lubavad allikale ligipääsu päringu tulemustele. (vt. Lisa 5)
2) asünkroonsete päringute puhul on tulemuste tagastamine algatatud sihtmärgi poolt. Kui leitakse määratud hulk sobivaid tulemusi, siis saadab sihtmärk need allikale. Sellise kommunikatsiooniviisi toetamiseks peab allikal olema tulemuste kuular. Samuti peab allikas olema võimeline üheselt identifitseerima iga päringu, mis on mingile sihtmärgile saadetud (ka siis, kui sama päring on saadetud mitmele sihtmärgile).Vastasel juhul ei ole allikas võimeline eristama mitmest sihtmärgist saadud otsingutulemusi ning sihtmärgile varem saadetud päringuid. (vt. Lisa 6)

Sünkroonse päringu korral suudab päringuliides teha sessiooni jooksul mitu päringut korraga. Asünkroonse päringu puhul annab allikas päringule identifikaatori, mille abil on võimalik siduda tagastatud päringutulemused konkreetse päringuga. Ka asünkroonse päringu korral võivad sessiooni jooksul aktiivsed olla mitu päringut.

[Allikas 2, lk 1-2]

3.3. Sessioonihaldus

Nagu eelpool öeldud, põhineb SQI lihtsal sessioonihalduse kontseptsioonil. Eeldatakse, et enne igasuguse andmeedastuse toimumist on loodud sessioon. Selline spetsifikatsioon eraldab päringuhalduse ja töötlemise autentimisest.

Sünkroonse päringuliidese korral loob allikas sihtmärgile sessiooni kasutades seejuures sihtmärgilt saadud sessiooni identifikaatorit, et ennast kommunikatsiooni käigus identifitseerida. Autentimine ei pea põhinema salasõnadel, sest on võimalik luua ka anonüümseid sessioone.

[Allikas 2, lk 3]

3.4. Olekuga ja olekuta kommunikatsioon

Olekuga ja olekuta on omadused, mis kirjeldavad, kas repositooriumid on loodud jälgima ja talletama ühte või enamat eelnevat sündmust suvalises interaktsioonide jadas. Olekuga tähendab, et sihtmärgiks olev repositoorium talletab interaktsiooni seisundit - näiteks hoides eelnevalt esitatud päringu tulemusi vahemälus. Olekuta tähendab, et eelnenud interaktsioonidest andmeid alles ei hoita ja iga interaktsiooni koheldakse ainult vasatavalt sellega kaasnevale informatsioonile.

[Allikas 2, lk 3]

3.5. Käsu ja päringu eraldatuse printsiip

Selle printsiibi kohaselt on iga meetod, kas käsk, mis täidab mingi tegevuse, või päring, mis tagastab andmed pärijale, kuid mitte mõlemad korraga. Ehk täpsemalt, meetodid tagastavad väärtuse vaid siis, kui nad on läbipaistvad (töödeldavate andmete sisu ei sõltu kodeerimis- või edastusviisist) ja seega ei põhjusta mingeid kõrvalnähte. Tänu sellele tekib disainistiil, mille tulemuseks on lihtsamad ja palju kergemini mõistetavad liidesed. Näiteks andmete pärimisel baasist ja kuvamisel kasutajaliidesesse ei tehta seda mitte ühe meetodiga, sest selline meetod sisaldaks nii andmete edastamist kui andmete muutmist. Vastavalt käsu ja päringu eraldatuse printsiibile võiks sellise tegevuse täita kolme meetodiga. Esimene meetod pärib baasist andmed. Teine meetod viib need kasutajaliidese jaoks sobivale kujule ning kolmas meetod kuvab andmed kasutajaliidesesse. Selline lahendus muudab ka koodi lihtsamaks, sest igale terviklikule tegevusele vastab oma meetod. Samuti on nüüd baasi ja kasutajaliidese andmete formaat teineteisest sõltumatu. Kui baasi või kasutajaliidese andmete formaat muutub, tuleb muuta ainult meetodit number kaks.

Käsu ja päringu eraldatuse printsiip on pärit objektorienteeritud programmeerimise vallast. Printsiibi looja on Bertrand Meyer.

[Allikas 2, lk 3]

3.6. Lihtne ja laiendatav käsustik

Et muuta liidest kergesti laiendatavaks, kasutatakse lähenemist, mille kohaselt minimaliseeritakse erinevate meetodite parameetrite arv, selle asemel et minimaliseerida meetodite endi arvu. Liidese variatsioonid on kergesti loodavad lisades uusi funktsioone, samal ajal kui pole vaja teha muudatusi juba kasutuselevõetud meetoditesse. Seega on lihtsam tagada ka tagurpidiühilduvust.

[Allikas 2, lk 4]

3.7. Probleemid SQI liidese teostamisel

Reaalse SQI liidese loomisel puututakse kokku mitmete probleemidega ning peab vastavalt olemasolevatele ressurssidele ja vajadustele tegema mitmeid otsuseid loodava liidese osas. Olulisemad küsimused on:

- Kas teostada SQI liides nii, et see toetab nii sünkroonseid kui mittesünkroonseid päringuid?
- Kuidas loodud SQI liidest rakendada tagamaks koostalitlust juba olemasolevate või tulevikus loodavate SQI liidest omavate süsteemidega?
- Kas on vajalik, et loodav SQI liides toetab kõiki SQI standardis loetletud meetodeid või on käesoleval juhul mõistlik teostada vaid osa meetoditest – ehk siis, milline on SQI rakendusprofiil?

Erinevalt asünkroonsetest päringutest saab sünkroonsete päringute puhul SQI-d rakendada nii olekuga kui olekuta teenusena. Olekuta SQI teenus unustab päringu vastuse kohe, kui see on allikale toimetatud. Juhul, kui päritakse veel tulemusi, esitatakse päring täies mahus uuesti ja toimetatakse allikani kõik lisatulemused.

Olekuga SQI teenus hoiab eelmiste interaktsioonide olekuinformatsiooni alles. Näiteks kui vajatakse lisatulemusi juba esitatud päringule, siis pole päringu uuesti täies mahus läbitöötamine enam vajalik – lisatulemused antakse teenuse poolt hoitavast vahemälust. Olekuga SQI teenus indentifitseerib päringuid nende räsi järgi.

SQI on kõige lihtsamini rakendatav veebiteenusena. Et kindlustada koostalitllust erinevate SQI-d toetavate rakenduste vahel, on loodud WSDL sidum SQI jaoks. Tegemist on Sourceforge projektiga, mis on kättesaadav aadressilt: http://sqi-wsdl.sourceforge.net/

SQI-d on võimalik rakendada ka osaliselt. See tähendab, et kasutatakse ainult mingit alamhulka kõigist SQI spetsifikatsioonis väljapakutud meetoditest. Sellisel juhul annavad mittekasutatud meetodid SQI veakoodi METHOD_NOT_SUPPORTED. Näiteks võib eksisteerida kahest sõlmest koosnev SQI võrk, mis toetab ainult sünkroonseid päringumeetodeid ja ei kasuta päringuparameetrite konfigureerimiseks olemasolevaid meetodeid (päringuparameetrid on üheselt ettemääratud). Lisaks sellele võib SQI rakendusprofiil sisaldada veel informatsiooni süsteemide toetatavate päringukeelte ja vastuseformaatide osas.

[Allikas 2, lk.16]

3.8. SQI API

Kokku sisaldab SQI API kolmeteist meetodit (vt. Lisa 14), mida võib kooskõlas lihtsa sessioonihalduse põhimõttega jagada esmalt kaheks:

1. sessioonihaldusmeetodid;
2. päringumeetodid.

Päringumeetodeid saab omakorda jagada kolmeks:

1. päringu parameetrite konfigureerimise meetodid;
2. sünkroonse päringu meetodid;
3. asünkroonse päringu meetodid.

3.8.1. Sessioonihaldusmeetodid

createSession() - meetod loob sessiooni ja tagastab SessionID (sessiooni identifikaatori). Meetodil on kaks parameetrit:

- UserID (string), mis on kasutajanimi kasutaja identifitseerimiseks;
- password (string), mis on salasõna kasutaja autentimiseks.

Meetod võib tekitada kahte tüüpi vigu:

- WRONG_CREDENTIALS – vigane UserID ja/või password;
- METHOD_FAILURE – sessiooni loomine ebaõnnestus mõnel muul põhjusel.

createAnonymousSession() - meetod loob uue sessiooni, vajamata seejuures kasutajaandmeid. Meetod tagastab sessiooni identifikaatori ja ei oma parameetreid. Meetod võib luua vea:

- METHOD_FAILURE – sessiooni loomine ebaõnnestus mõnel muul põhjusel.

destroySession() - sessiooni loonud süsteem saab kasutada seda meetodit sessiooni hävitmiseks. Meetod ei tagasta midagi ja omab ühte parameetrit:

- sessionID (string), mis identifitseerib hävitatava sessiooni.

Meetod võib tekitada kaks viga:

- NO_SUCH_SESSION – sessionID on vigane;
- METHOD_FAILURE – sessiooni hävitamine ebaõnnestus mõnel muul põhjusel.

Päringu parameetrite konfigureerimise meetodid on:

setQueryLanguage() - meetod määrab päringukeele, mille abil saab allikas kontrollida päringulause süntaksit. Meetod ei tagasta midagi ning omab kahte String andmetüüpi parameetrit:

- targetSessionID, mis näitab, millisele sessioonile päringukeel määratakse;
- queryLanguageID, mis näitab, milline päringukeel määratakse.

Meetod võib tekitada kolme tüüpi vigu:

- NO_SUCH_SESSION – targetSessionID on vigane;
- QUERY_LANGUAGE_NOT_SUPPORTED – sihtmärk ei toeta allika poolt kasutatavat päringukeelt;
- METHOD_FAILURE – operatsioon ebaõnnestus mingil teisel põhjusel.

setResultsFormat() - see meetod võimaldab allikal määrata sihtmärgi poolt tagastatavate vastuste formaadi. ResultsFormat parameeter sisaldab tavaliselt kas URI-t (Universal Resource Identifier) või eelnevalt kokkulepitud väärtusega stringi.Meetod ei tagasta midagi ja omab kahte parameetrit:

- targetSessionID (string), mis näitab, millisele sessioonile päringukeel määratakse;
- resultsFormat (string), määrab tagastatavate tulemuste formaadi.

Meetod võib tekitada kolme tüüpi vigu:

- NO_SUCH_SESSION – targetSessionID on vigane;
- RESULTS_FORMAT_NOT_SUPPORTER – sihtmärk ei toeta sisestatud formaati;
- METHOD_FAILURE – operatsioon ebaõnnestus mingil teisel põhjusel.

setMaxQueryResults() - meetod määrab ära päringu poolt tagastatavate tulemuste maksimaalse arvu. Vaikimisi on see 100. Meetod ei tagasta midagi ja omab kahte parameetrit:

- targetSessionID (string), mis näitab, millisele sessioonile päringukeel määratakse;
- maxQueryResults(täisarv), mis määrab ära tagastatavate tulemuste maksimaalse arvu. Selle parameetri väärtus peab olema 0 või suurem täisarv. 0 tähendab, et allikas ei soovi vastuste arvu piirata.

Meetod võib tekitada kolme tüüpi vigu:

- NO_SUCH_SESSION – targetSessionID on vigane;
- INVALID_MAX_QUERY_RESULTS – sisestatud arv ei vasta nõuetele;
- METHOD_FAILURE – operatsioon ebaõnnestus mingil teisel põhjusel.

setMaxDuration() - meetod võimaldab allikal seada ajapiir asünkroonsele päringule. Aeg

antakse millisekundites. Vaikimisi väärtuseks on 0, mis tähendab, et ajapiiri seadmise õigus antakse üle sihtmärgile.Meetod ei tagasta midagi ja tal on kahte parameetrit:

- targetSessionID (string), mis näitab, millisele sessioonile päringukeel määratakse;
- maxDuration (täisarv), mis määrab ajapiiri asünkroonsele päringule. Selle parameetri väärtus peab olema 0 või suurem täisarv.

Meetod võib tekitada kolme tüüpi vigu:

- NO_SUCH_SESSION – targetSessionID on vigane;
- INVALID_MAX_DURATION – sisestatud arv ei vasta nõuetele;
- METHOD_FAILURE – operatsioon ebaõnnestus mingil teisel põhjusel.

3.8.2. Sünkroonse päringu meetodid

setResultsSetSize() - meetod määrab ühes tulemustekollektsioonis sisalduvate tulemuste maksimaalse arvu. Vaikimisi väärtuseks on 25. Meetod ei tagasta midagi ja omab kahte parameetrit:

- targetSessionID (string), mis näitab, millisele sessioonile päringukeel määratakse;
- resultsSetSize (täisarv), mis määrab tulemustekollektsiooni maksimaalse suuruse. Väätus peab olema 0 või suurem, kusjuures 0 tähendab, et kollektsioonis sisalduvad kõik tulemused.

Meetod võib tekitada nelja tüüpi vigu:

- NO_SUCH_SESSION – targetSessionID on vigane;
- QUERY_MODE_NOT_SUPPORTED– sihtmärk ei toeta asünkroonseid päringuid;
- INVALID_RESULTS_SET_SIZE – sisestatud arv ei vasta nõuetele;
- METHOD_FAILURE – operatsioon ebaõnnestus mingil teisel põhjusel.

synchronousQuery() - meetod esitab sihtmärgile päringu.Ühe sessiooni vältel saab edastada mitu päringut. Meetod tagastab kõik päringule vastavad metaandmete kirjed ja omab kolme parameetrit:

- TargetSessionID (string), mis näitab, millisele sessioonile päringukeel määratakse;
- queryStatement (string), mis sisaldab konkreetset päringut;
- startResult (täisarv) – parameeter, mis määrab, milline tulemustekollektsiooni liige on esimene, selle väärtuseks võib olla täisarv ühest kuni maksimaalse lubatud tagastatavate tulemuste arvuni.

Meetod võib tekitada seitset tüüpi vigu:

- NO_SUCH_SESSION – targetSessionID on vigane;

- INVALID_QUERY_STATEMENT – päringulause ei vasta päringukeele süntaksile;
- QUERY_MODE_NOT_SUPPORTED– sihtmärk ei toeta asünkroonseid päringuid;
- INVALID_START_RESULT – sellise numbriga tulemust ei leidu;
- NO_MORE_RESULTS – kui startResult on null ja rohkem tulemusi ei leidu;
- METHOD_FAILURE – operatsioon ebaõnnestus mingil teisel põhjusel.

getTotalResultsCount() - meetod tagastab päringu tulemuseks saadud vastuste arvu.Meetod omab kahte parameetrit:

- TargetSessionID (string), mis näitab, millisele sessioonile päringukeel määratakse;
- queryStatement (string), mis määrab millise päringulause tulemuste arv tagastatakse.

Meetod võib tekitada nelja tüüpi vigu:

- NO_SUCH_SESSION – targetSessionID on vigane;
- QUERY_MODE_NOT_SUPPORTED– sihtmärk ei toeta asünkroonseid päringuid;
- INVALID_QUERY_STATEMENT – päringulause ei vasta päringukeele süntaksile;
- METHOD_FAILURE – operatsioon ebaõnnestus mingil teisel põhjusel.

3.8.3. Asünkroonse päringu meetodid

asynchronousQuery() - meetod lubab saata sihtmärgile päringu nii, et tulemused tagastatakse asünkroonsel meetodil. Meetod ei tagasta midagi ja omab kolme parameetrit:

- TargetSessionID (string), mis näitab, millisele sessioonile päringukeel määratakse;
- queryStatement (string), mis määrab millise päringulause tulemuste arv tagastatakse;
- queryID (string), mis võimaldab päringutulemusi hiljem päringuga siduda.

Meetod võib tekitada viit tüüpi vigu:

- NO_SUCH_SESSION – targetSessionID on vigane;
- QUERY_MODE_NOT_SUPPORTED– sihtmärk ei toeta asünkroonseid päringuid;
- NO_SOURCE_LOCATION – enne päringu esitamist ei ole allika asukoht määratud (meetodi setSourceLocation abil);
- INVALID_QUERY_STATEMENT – päringulause ei vasta päringukeele süntaksile;
- METHOD_FAILURE – operatsioon ebaõnnestus mingil teisel põhjusel.

setSourceLocation() - meetod määrab allika päringutulemuste kuulari asukoha. Seda meetodit tuleb asünkroonse pärimisviisi puhul kasutada alati enne päringu esitamist. Meetod ei tagasta midagi ja omab kahte parameetrit:

- TargetSessionID (string), mis näitab, millisele sessioonile päringukeel määratakse;
- sourceLocation (string), mis määrab allika päringu tulemuste kuulari asukoha URL kujul:

Meetod võib tekitada kolme tüüpi vigu:

- NO_SUCH_SESSION – targetSessionID on vigane;
- QUERY_MODE_NOT_SUPPORTED– sihtmärk ei toeta asünkroonseid päringuid;
- METHOD_FAILURE – operatsioon ebaõnnestus mingil teisel põhjusel.

queryResultsListener() - sihtmärgi poolt initsialiseeritud meetod, mis edastab päringutulemuste kollektsioonid allikale. Meetod ei tagasta midagi ja omab kahte parameetrit:

- queryID (string), mis sisaldab päringu identifikaatorit, mis on vajalik päringu ja päringutulemuste sidumiseks;
- queryResults(string), mis sisaldab päringu tulemusi

Meetod võib tekitada kolme tüüpi vigu:
INVALID_QUERY_RESULTS – päringutulemuste kollektsioon on allika jaoks tõlgendamatu;
NO_SUCH_QUERY – queryID on vigane;
METHOD_FAILURE - operatsioon ebaõnnestus mingil teisel põhjusel.

[Allikas 2, lk 5-13]

4. SQI liidesed teiste rakenduste juures

Alates SQI spetsifikatsiooni avaldamisest 2004.a. märtsis, on see paljude õpiobjektide repositoorimumite juures kasutusele võetud. Järgnevad on vaid mõned projektidest, mille juures on SQI-d implementeeritud.

[Allikas 4, lk 5]

4.1. Ariadne

Ariadne Foundation loodi eesmärgiga luua ja arendada vahendeid ning metodoloogiaid, mille abil saab luua, hallata ja taaskasutada arvutipõhiseid pedagoogilisi elemente ning audiovisuaalseid vahendeid toetavaid treeningmaterjale.

Ariadne KPS-i (Knowledge Pool System) tuumaks on õpiobjektide repositooriumite hajusvõrk, mis levitab nii õpiobjekte endid kui nende metaandmeid. Selle infrastruktuuri peale on loodud mitmeid õpiobjektide loomiseks ja kasutamiseks vajalikke töövahendeid, mida Ariadne oma liikmetele pakub.

Iga hajusvõrgu liige haldab relatsioonilist metaandmete kogumit, mille suhtes on lubatud teha nii sünkroonseid kui asünkroonseid SQI päringuid.

Sünkroonsed SQI päringud võimaldavad KPS-i kasutamiseks loodud kolmanda osapoole rakenduste lihtsamat kasutuselevõttu – päringu API loomine on märksa lihtsam. Sellisel kasutusjuhul saadab rakendus päringu ainult ühele sünkroonset pärimist võimaldavale sihtmärgile – lisanduvaid päringutulemusi laetakse alla vastavalt vajadusele.

Asünkroonse pärimismeetodi puhul on eesmärgiks parem koostalitlus mitmete vahendusteenustega. Kuna need teenused esitavad suure arvu päringuid mitmetele repositoorimitele, on asünkroonne kommunikatsioon vähem veatundlik, sest päringutulemusi kogutakse ainult ühe kuulari poolt. See omakorda tagab lihtsama halduse ja on mugavam.

Tagamaks Ariade liikmetele ligipääsu teistele (väljaspool KPS-i asuvatele) repositooriumitele on loodud ühendatud otsingumootor. See otsingumootor pakub sünkroonset SQI liidest kõigile kasutajaliidestega. Näiteks kasutab seda repositoorimitele päringute esitamiseks SILO – Ariande otsingu ja indekseerimisvahend. Otsingumootor töötab saates päringu erinevatele SQI liidesega repositooriumitele. Hetkel on võimalik otsida järgnevatest repositooriumitest: Ariadne, EdNA

Online, EducaNext, Merlot, Pond, RDN, SMETE, VOCED. Kuna kõik ülalmainitud repositooriumid toetavad erinevaid päringukeeli, mis pole kergesti teineteiseks tõlgitavad, võeti kasutusele strateegia, mille käigus implementeeriti teistest kõige erinevam päringukeel – st. Kõik pärigud esitatakse selles keeles ja iga repositoorium tõlgib loodud mähiste abil selle endale arusaadavaks.. Sellise lähenemise korral koosneb päring ainult sellistest otsinguterminitest, mida iga repositoorium oskab endale arusaadavaks päringuks tõlkida.

Ariadne vajab edukaks ühilduvuseks teiste repositooriumitega järgmisi LOM XML kodeeringus metaandmete välju: URL, identifikaator, pealkiri ja originaalrepositooriumi identifikaator. URL peab lahenduma õpiobjektiks. Kui juurdepääs õpiobjektile on piiratud, peab URL lahenduma kontakti informatsiooniks. Originaalrepositooriumi identifikaatorit on vaja tulemuse andnud repositooriumi õigeks identifitseerimiseks. Lisaks ülaltoodud metaandmete väljadele on kõik ülejäänud LOM-i metaandmete väljad valikulised.

[Allikas 4, lk 5]

4.2. Celebrate ja iClass

iClass-i projekti alustati eesmärgiga luua intelligentne kognitiiv-õppel põhinev e-õppesüsteem ja keskkond, mis on kohandatav iga individuaalse õppija vajadustele. Iclass-i projekti rahastakse European Community Framework Programme for Research, Technological Development and Demonstration poolt.

iClass adapter on iClass (Intelligent distributed Cognitive-based Learning System for Schools) õpihaldussüsteemide võrgustiku osa, mis võimaldab võrgustikuvälistel, kuid Celebrate föderatsiooni kuuluvatel õpihaldussüsteemidel otsida ning kasutada iClass-i materjale (metaandmeid ja õpiobjekte). iClass hoiab metaandmeid metaandmete repositooriumite võrdõigusvõrgustikus nimega "sisuserver" ("content server"). Õpiobjekte hoitakse õpiobjektide repositooriumite võrdõigusvõrgustikus nimega "sisujaotus süsteem" ("content distribution system"). Õpiobjekti kättesaamine on kaheosaline protsess:

1) metaandmete otsimine ja hindamine: metaandmetes sisalduvate kirjelduste põhjal valitakse välja õpiobjektid, mis vastavad kasutaja vajadustele;
2) õpiobjektide hõivamine: valitud õpiobjektide kättesaamine metaandmetes viidatud asukohast.

Tagamaks võimalikult suure hulga õpihaldussüsteemide ühilduvust iClass-iga on äärmiselt oluline, et liides oleks avatud ja standardiseeritud. SQI lihtsus, sõltumatus päringukeelest ja

vastuseformaadist ning asünkroonse pärimisviisi toetus teevad sellest hea kandidaadi iClass "sisuserveri" liideseks.

iClass-i metaandmed annavad õpiobjekti asukoha asemel hoopis selle identifikaatori, sest iClass-i õpiobjektide adaptiivne ja multimeediale orienteeritud olemus kombineerituna sellega, et sisujaotus süsteem on võrdõigusvõrgustikul põhinev, teeb otsese ligipääsu neile keeruliseks. See ongi põhjuseks, miks metaandmete põhjal valitud õpiobjektide asukoha lahendamiseks on vaja eraldi protsessi. Selle "asukoha lahendamise" protsessi käigus levitatakse õpiobjekti asukoha päringut repositooriumist repositooriumisse kuni vajalik õpiobjekt leitakse – "sisujaotussüsteemile" esitatakse asünkroonne päring, mis sisaldab õpiobjekti identifikaatorit ja andmeid päringu allika kohta. Seejärel liigutatakse leitud õpiobjekt päringu tegijale kõige lähemasse serverisse ja URL, millelt selle õpiobjekti kätte saab, tagastatakse sisujaotus süsteemile. Kuna see protsess võtab teatava aja, vastab sisujaotus süsteem päringule asünkroonselt, et tagada pärija, praegusel juhul iClass adapteri, optimaalne töövõime. Seega on õpiobjekti asukoht teada ja see tagastatakse ainult juhul, kui õpiobjekt on juba saadavalt konkreetses serveris.

Kuna sellise protsessi asünkroonselt läbiviimiseks ei leidunud avatud liidest ja et vältida uue liidese loomist, otsustati kasutusele võtta SQI; iClass adapteri poolt toetatavatele keeltele ja vastuseformaatidele lisati uus "päringukeel" (asukoha leidmiseks) ja "vastuseformaat" (asukoha tagastamiseks). Loodud päringukeele nimeks sai "ICLASS-LO-ID". Päring selles keeles koosneb metaandmetest saadud õpiobjekti identifikaaatorist. Vastuseformaadi nimeks sai "URL". Tagastatav vastus selles formaadis koosneb ainult õpiobjektile viitavast URL-ist.

Selline lahendus lubab minimaliseerida õpiobjekti asukoha leidmise protsessi implementatsiooniga seotud kulusid nende repositooriumite jaoks, mis juba kasutavad SQI-d metaandmete otsimiseks. Hetkel on see implementeeritud Celebrate SQI lüüsina.

[Allikas 4, lk 5]

4.3. ELENA

ELENA projekt sai alguse Euroopa Komisjoni poolt finantseeritud uurimis-initsiatiivist, milles osales interdistsiplinaarne uurimismeeskond, kelle eesmärgiks oli luua "Smart Spaces for Learning (TM)", mis kujutavad endast avatud keskkondi, mis toetavad õppijaid optimiseerides nende õppeprotsesse.

[Allikas 23]

Saavutamaks erinevate õpihaldussüsteemide koostalitlust võeti ELENA projekti käigus kasutusele uudne semantilise veebi tehnoloogiatel põhinev infrastruktuur ja tarkvaralahendus. Infrastruktuur ehitati üles järgnevail põhimõtetel:

1) päringute tegemiseks on ühine liides – SQI koos veebiteenustel põhineva kasutajaliidesega;
2) ühine semantiline mudel päringute ja päringutulemuste esitamiseks, mis on realiseeritud XML ja RDF skeemide abil;
3) korduvkasutatavad komponendid integreerimaks QEL ja XQuery päringukeeli kasutavaid süsteeme minimaalsete jõupingutustega.

Infrastruktuuri eesmärk on luua "Smart Space for Learning", mis lubab integreerida erisuguseid semantilises võrgus asuvaid hariduslike materjalide allikaid ja võimaldada neile "arukat" juurdepääsu. Koos õpieesmärkide defineerimise, isikustatud otsingute ja tagasiside andmise vahenditega mängib haridusalane semantiline võrk olulist rolli koostööle orienteeritud personali arengu toetamisel. Õpiobjektide lai valik võimaldab märkimisväärselt avardada valikuid õppematerjali valimisel.

Kõigi selle projekti raames ühendatud süsteemide jaoks loodi üldine sidumisskeem, mis võimaldab edastada päringuid kõigile süsteemidele ning võtta neist vastu ka päringutulemusi. Skeem on kirjeldatud RDF-i abil ning päringukeelena kasutatakse QEL-i. Kõik ELENA projekti raames ühendatud süsteemid on ligipääsetavad HCD Online portaalist (http://www.hcd-online.com/ubp).

Ühtse sidumissüsteemi loomine oli keeruline, kuna erinevas süsteemid kasutavad juba omakorda erinevaid sidumissüsteeme ning metaandmete esitlusviise. Näiteks ULI Campus hoiab metaandmeid RDF failides samas kui EducaNext, CLIX ja paljud kursuste andmebaasid hoiavad metaandmeid relatsioonilistes andmebaasides. Samuti leidub süsteeme, mis hoiavad metaandmeid XML failides.

QEL päringute Xquery päringuteks tõlkimiseks loodi terve uus tõlketehnika, mille abil suudeti integreerida veel ka teisi süsteeme nagu näiteks LASON ja Knowledgebay, mis kasutavad XML andmebaasi eXist. Ühendamaks meediallikaid nagu Amazon oli vaja veelgi erinevamat lähenemist - QEL päringud tõlgiti Amazoni otsinguobjektideks.

[Allikas 4, lk 6]

5. SQI liidese loomine

5.1. EducaNext-i iseloomustus

EducaNext on mitmekeelne akadeemiliste materjalide vahetusportaal, kus kõrgemate õppeasutuste, teadusorganisatsioonide ja teiste professionaalide ühenduste liikmed saavad jagada, leida ning taaskasutada õpperesursse.

EducaNext on loodud Euroopa Komisjoni sponsorluse toel UNIVERSAL töögrupi poolt. UNIVERSAL projekti näol on tegemist on ühe ambitsioonikaima e-õppe projektiga Euroopas ja kogu maailmas.

EducaNext-i peamiseks ülesandeks on aidata kõigil akadeemilistesse ringkondadesse kuuluvail isikuil, kes on huvitatud oma teadmiste täiendamisest ning teistega jagamisest läbi hajuskoostöö, saada lihtne ning kiire juurdepääs akadeemilistele materjalidele.

EducaNext on teadmispõhistele kooskondadele suunatud teenus, mille eesmärgiks on saavutada parem koostöö õppematerjalide tootjate vahel, et tagada loodavate materjalide parem kvaliteet ja mitmekesisem sisu. Kooskonna ülesandeks on järjepidevalt luua, jagada ja uuendada õppematerjale. Arvestades õpiressursside loomisele kuluvat aega ja kiiret tempot, millega need ressursid vananevad, on kvaliteetsete materjalide pakkumine võimalik ainult tänu suurele inimressursile.

EducaNext-i teenus on tasuta ja avatud kõigile akadeemilise kogukonna liikmetele, kes nõustuvad EducaNext-i kasutamistingimuste ja nendega seotud käitumisreegitega. Registreerunud kasutajad pääsevad ligi kõigile EducaNexti poolt pakutavatele teenustele ja materjalidele. Mitteregistreerinud kasutajatele pakutakse piiratud juurdepääsu, mis hõlmab ainult õpiressursse, mis on lubatud avalikuks jagamiseks.

EducaNext-i poolt pakutavad teenused põhinevad Universal Brokerage Platform-il (UBP). UBP on modulaarne tarkvara, mis on loodud arvestades olemasolevaid metaandmete standardeid, kaasaarvatud avatud liideseid, mis põhinevad W3C Web Service standarditel, et tagada koostalitlus juba olemasolevate õpihaldussüsteemidega. Tarkvara funktionaalsust täiendatakse pidevalt. Eesmärgiks on saavutada koostalitlus selliste õpihaldusüsteemidega nagu Ariadne, suurendada pakutavate teenuste mahtu ja kaastata uuenduslikke tehnoloogiaid, mis vastavalt EducaNext-i eesmärgile lubavad luua, vahetada ja integreerida kõrge kvaliteediga õppematerjale.

[Allikas 20, lk 1-3]

5.1.1. EducaNext-i struktuur

Nagu eelnevalt mainitud põhineb EducaNext samuti UNIVERSAL töögrupi poolt väljatöötatud UBP tehnoloogial. UBP-l on klassikaline kolmekihiline arhitektuur (vt. Lisa 1). Esimesel kihil on teostatud kasutajaliides, mis põhineb Java-l. Äriloogika on teostatud rakendusserveril, Java servlettide abil, mida tõlgendavad Tomcat (Java servlettide mootor) ja veebiserver. Platvormi operatsioonisüsteemiks on Linux. Kolmandal kihil asub andmebaas, millega võetakse ühendust läbi JDBC (Java Database Connection).

UBP arhitektuur põhineb XML-il, objektid on modelleeritud kasutades RDF-i. RDF annab üldise mudeli märkimaks omadusi ja ressursside omaduste väärtusi. XML-i kasutatakse õpiressurssude RDF-mudeli põhiste metaandmete kodeerimiseks UNIVERSAL platvormil.

Suure hulga erinevate ning mitmekesistel metaandmete skeemidel põhinevate õppematerjalide tarnesüsteemide integratsioon eeldab väga paindliku metaandemete lahenduse kasutuselevõttu. Kasutades XML-i, on metaandmete kirjeldused lihtsasti mõistetavad, taaskasutatavad ja muudetavad standardsete töövahendite abil. RDF annab raamistiku kirjeldamaks metaandmeid paindlikul moel kasutades kirjelduselemente mitmest erinevast kirjeldusskeemist. RDF-i kasutamine võimaldab lisaks ülaltoodule ka metaandmete hajutatud otsimist ja kasutamist

Õpiressursi kohaletoimetamine sõltub tarnesüsteemi tüübist(vt. Lisa 3). Enamasti kasutatakse õpiobjektide kohaletoimetamiseks veebiservereid. Niimoodi kohaletoimetatud õpiobjektide kasutamiseks on sageli vajalikud mitmed lisad ,nagu näiteks PDF reader ja PowerPoint Viewer, kasutajapoolsele tarkvarale (veebilehitsejale). Hetkel on streaming audio ja video jaoks ainult üks UBP-ga ühilduv tarnesüsteem, milleks on Real Server.

UBP koosneb erinevatest tarkvaramootoritest (vt. Lisa 2), mis pakuvad teenuseid kasutajatele ja välistele süsteemidele – nagu näiteks tarnesüsteemid ja hindamissüsteemid. Süsteemi olulisemas osad on metaandmete mootor, lepingumootor, tarnemootor, kasutajaprofiili mootor ja ka administreerimismootor.

Metaandmete mootor tegeleb õpiressurssidega seotud andmete haldamisega. Mootor suhtleb õpiressursside pakkujatega läbi veebipõhise kasutajaliidese või kasutab mõnd protokolli suhtlemiseks kohalike õpihaldussüsteemidega. Lisaks ülaltoodule tegeleb metaandmete mootor veel ka juurdepääsu lubamisega õpiobjektide repositooriumile. Seega lubab süsteem ühest küljest pakkuda paindlikku otsinguteenust, lubades kasutajal sisestada keerukaid päringuid nagu võimas otsingumootor. Teisest küljest aga jääb kasutajale ka võimalus lehitseda hästistruktureeritud, rahvusvahelisel taksonoomial põhinevat kataloogi.

Lepingumootor tegeleb õpiressurrside pakkumiste ja nõudmiste töötlemisega pannes kokku sobivad nõuded ja pakkumised ning hoolitsedes ka materjalide hõivamise eest. Nende transaktsioonide käigus tegeleb süsteem kasutajate autentimise, tehingute kontrollimise ja võimalik, et tulevikus ka arvete esitamisega.

UBP struktuuris asub vahendus- ja tarnesüsteemi vahel asub õhuke liidesekiht, mis pakub kommunikatsioonialast funktsionaalsust nagu näiteks autentimist, autoriseerimist, tarne korraldamist ja tarne kontrolli.

Kasutajaprofiili mootori ülesandeks on kasutajate adiminstreerimine – näiteks uute kasutajate registreerimine ja vanade registratsioonide tühistamine. Kasutajaprofiili mootor pakub teenuseid teistele mootoritele (eriti õpiressursiide metaandmete mootorile), et kasutajaprofiile korralikult uuendada ja hallata. Kasutajapõhine sisselogimine süsteemi võimaldab pakkuda personaliseeritud juurdepääsu õpiobjektide repositooriumile.

Administreerimismootor tegeleb andmete hoidmisega sisselogimiste kohta, samuti muutuste kohta repositoorimumis. Lisaks sellele annab administreerimisootor hinnangusüsteemile tagasisidet kvaliteedi määramiseks.

Kokkuvõtteks võib öelda, et UBP struktuur on loodud pöörates tähepanu sellistele kõrge kvaliteedi saavutamiseks vajalikele omadustele nagu avatud disain, turvalisus, koostalitlusvõime ja kasutatavus jätmata sealjuures ka tähelepanuta jõudlust ja mastabeeritavust(skaleeritavust). XML:RDF lähenemine valiti, et kindlustada avatust ja paindlikust õpiobjektide, agentide, õiguste, taksonoomiate, lühikirjelduste ja tarnesüsteemide andmemudelite loomisel. Avatus on äärmiselt oluline efektiivse andmevahetuse tagamiseks.

Ka turvalisus on usaldusväärse vahendusteenuse loomisel oluline komponent. Standardsed internetiprotokollid ei paku kuigivõrd kaitset, et tagada privaatsust, konfidentsiaalust, täielikkust ja autentsust. Autentimine on defineeritud kui kindlaks tegemine, et andmed pärinevad just sellest allikast, kust neid väidetakse pärinevat või et võrdõigusvõrgustiku sõlm on just see, kes ta väidab end olevat. Põhilised nõuded käesoleval juhul autentsuse tagamiseks on:

- vahendusteenuse autentsuse tagamine – kasutajatel peab olema võimalus identifitseerida serverit, millega nad suhtlevad;
- kasutajate autentsuse tagamine – UBP peab autentima õpiobjektide kasutajaid ja pakkujaid. Autentimine on alus kasutaja autoriseerimiseks.

[Allikas 21, lk 4-6]

5.2. EducaNext-i ja UBP installeerimine

Käesoleva bakalaureusetöö praktilise poole üheks osaks oli õpiobjektide vahendusportaali ja repositooriumi EducaNext installeerimine ja konfigureerimine, et seda saaks kasutada teise osapoolena Tallinna Ülikooli õpihaldussüsteemi IVA SQI liidese väljatöötamise juures. IVA SQI liidese loomine on osa suuremast rahvusvahelisest iCamp projektist, mille eesmärgiks on tagada koostalitlus enamlevinud avatud lähtekoodiga õpihaldussüsteemide, õpiobjektide repositooriumite ja digitaalsete raamatukogude vahel.

Paigaldamiseks ettenähtud tarkvara lähtekood ja installeerimisjuhend pärinevad UNIVERSAL projekti meeskonnas CVS serverist. Märkimisväärne on, et kuigi EducaNext 2.0 on kuulutatud avatud lähtekoodiga tarkvaraks, pole selle lähtekood siiski vabalt saadaval, vaid selle saamiseks tuli saavutada kokkulepe arendusmeeskonnaga.

Pagaldusplatvormiks oli valitud järgmise riistvarakonfiguratsiooniga SUN Fire server:

- AMD Opteron(tm) Processor 254 2800 Mhz
- 4 GB RAM
- 200 GB HDD [RAID5]

Serveri operatsioonisüsteemiks on vaikimisi seadetega installeeritud Red Hat Enterprise Linux.

Vastavalt installeerimisjuhendile, vajab EducaNext 2.0 korrektseks toimimiseks järgmisi tarkvarakomponente:

- Java 2 Platform, Standard Edition (J2SE) 1.4.2;
- Apace HTTP server 2.0;
- TomCat 4.1;
- TomCat Connector;
- Firebird 1.5.

Nõutud komponentidest oli juba koos operatsioonisüsteemiga installeeritud Apache veebiserver – versioon 2.2. Ülejäänud komponendid said probleemideta installeeritud vastavalt installatsioonijuhendile. Lisaks installeeriti veel ka Apache ANT 1.6.5 – kuigi seda installatsioonijuhendis vajaliku tarkvara nimekirjas polnud, selgus juhendi edasisel lugemisel, et saadud EducaNext 2.0 koodi kompilleerimiseks tuleb kasutada just seda vahendit.

UNIVERSAL projekti CVS serverist allalaetud EducaNext 2.0 kood on järgmise kataloogistruktuuriga (seletused installatsiooijuhendist):

- /bin – skriptid käsureapõhiste administreerimisvahendite käivitamiseks;
- /conf – rakenduse konfiguratsioonifailid;
- /database – EducaNext-i andmebaasi fail(id);
- /html – HTML mallid, mis moodustavad EducaNext-i veebirakenduse;
- /htmlCompiled – eelkompilleeritud HTML mallid;
- /images – EducaNext-i veebiliidese pildid;
- /logs – EducaNext-i logifailid;
- /src – EducaNext-i lähtekoodi;
- /styles – CSS style-sheet failid;
- /WEB-INF – rakenduse Java klassid, JAR teegid ja veebirakenduse kirjedldusfail TomCat-i jaoks;

5.2.1. Ilmnenud probleemid

Vastavalt installatsioonijuhendile anti süsteemsetele kasutajatele apache ja tomcat vajalikud õigused kataloogidele ja failidele. Edasi muudeti konfiguratsioonifaili "build-user.properities" sisu vastavalt juhendile. Lisaks juhendis lahtiseletatutele sisaldas see fail veel mitmeid väärtusi, kuid kuna nende olemuse kohta igasugune informatsioon puudus, jäeti need muutmata.

Esimesel kompilleerimisel Apache ANT-iga ilmnes 2 viga:

1) puudus kataloog WEB-INF/classes – lahenduseks piisas vastava kataloogi loomisest,
2) puudus fail ItemCountResults.tag

Peale kontakteerumist EducaNext-i arendusmeeskonnaga selgus, et CVS serveris oli üleval mittetöötav kood. Kood uuendati ja edasine töö jätkus sellega. Küsimustele, mis puudutasid konfiguratsioonifailis "build-user.properties" olevaid kommenteerimata väärtusi, ei vastatud.

Järgmine probeem ilmnes EducaNext-i veebirakenduse konfiguratsiooni juures. Peale konfiguratsioonifailide muutmist vastavalt installatsioonijuhendile selgus katse-eksitus meetodil, et näites antud formaat andmebaasi faili asukoha kirjeldamiseks on vale.

Näites antud formaat:

jdbc:firebirdsql://127.0.0.1/C:/Your/path/database.gdb.

Õige formaat:

jdbc:firebirdsql://127.0.0.1//Your/path/database.gdb

Olles teinud veel vajalikke muudatusi konfiguratsioonifailides ja edukalt eelkompilleerinud EducaNext-i veebirakenduse mallid käivitati TomCat-i veebiserverist EducaNext-i rakendus. Rakendus ei töötanud andes logifaili rea veateateid. Peale uuesti kontakti võtmist EducaNext-i arendajatega selgus, et vajalik on teha järjekordne koodiuuendus CVS serverist.

Seoses uue koodi kasutuselevõtmisega ilmnesid uued vead – EducaNext-i veebirakenduse mallide eelkonfigureerimise käigus saadi terve hulk veateateid. Peale konsulteerimist EducaNext-i arendajatega selgus, et tegu on aegunud andmebaasistruktuuriga – st. uue koodiga oli kaasa antud vana andmebaas. Ka vana andmebaasi asendamine uuega ei lahendanud kõiki vigu – mallide eelkonfigureerimine ebaõnnestus veateatega lubamatu märgi kohta andmebaasis. Täpsemal uurimisel selgus, et kuna mallide eelkompilleerimisel võetakse tekstid vastavalt keelele andmebaasist ja luuakse niimoodi mallide abil erinevates keeltes html leheküljed, siis oli andmebaasi sattunud hispaania keeles kasutatav sümbol, mille sisestamisega HTML formaadis faili installatiooniskript mingil põhjusel hakkama ei saanud – lahenduseks konfiguratsioonifailis hispaania keele väljajätmine kasutajaliidese keelte hulgast.

EducaNext 2.0 rakenduse taaskäivitamisel selgus uus probleem. Sümptomiks olukord, Kus rakenduse veebiliides on osaliselt kättesaadav aadressilt http://www.server.xx:8080/EducaNext, kuid mitte selleks tegelikult mõeldud aadressilt http://www.server.xx/EducaNext/ubp

Nimelt kasutab EducaNext 2.0 veebirakendus nii staatilist kui mittestaatilist sisu. Staatilise sisu all on mõeldud muutumatu sisu ja struktuuriga html lehekülgi ja pilte. Mittestaatilise sisu moodustavad Java servlett leheküljed, mille sisu on pidevalt muutuv. EducaNext töötab nii, et staatilist sisu esitab kasutajale Apache veebiserver ja mittestaatilist TomCat. Tehnilise lahendusena on kasutatud Jakarta Apache Tomcat Connector-it, mille eesmärgiks on siduda Apache veebiserver ja TomCat, nii, et kui Apache veebiserver saab päringu esitada JSP lehekülge, siis edastab ta selle päringu TomCat-ile, mis vastavalt olukorrale loob mittestaatilise JSP lehekülje alusel selleks korraks staatilise sisu ja saadab selle tagasi Apache veebiserverile. Tulemuseks on olukord, kus kasutajale paistab nagu saaks ta kogu sisu Apache veebiserveri käest.

Arvestades ülalmainitud sümptomeid oli selge, et probleem on Jakarta Apache TomCat Connectoris. Siiani oli kasutuses JK Connectori versioon 1.2.8, mis tuli eelkompilleerituna kaasa EduaNext-iga. Konfigureeritud oli see samuti vastavalt EducaNext-iga kaasatulnud installeerimisjuhendile. Kuna mainitud versioon JK Connectorist polnud viimane, üritati

kasutusse võtta versiooni 1.2.15, kuid selle kompilleerimine ebaõnnestus tänu sobimatusele Apache veebiserveri versiooniga 2.2. Konsultatsiooni käigus EducaNext-i arendajatega jõuti järeldusele, et Apache veebiserver 2.2 tuleb välja vahetada vanema versiooni vastu. Uueks serverile installeeritud Apache veebiserveri versiooniks sai 2.0.48.

Kuid ka selline lahendus ei saavutanud edu. Lõpuks sai katse-eksitus meetodil ning alternatiivseid installatsioonijuhendeid lugedes selgeks, et EducaNext-iga kaasasolnud JK Connectori installatsiooniõpetus on täiesti ebapädev.

Selle asemel, et kopeerida mod_jk.conf ja workers.properties failid TomCat-i konfiguratsioonikataloogi nagu seda käseb installatsioonijuhend, on lahenduseks mod_jk.conf faili sisu kirjutada otse Apache veebiserveri konfiguratsioonifaili ja sealt viidata workers.properties faili asukohale. Peale selle probleemi lahendamist oli EducaNext lõpuks töökorras ja installeeritud. Lisaks tuli veel installeerida Java Delivery System, mis on vajalik õpiobjektide lisamiseks. JavaDS installatsioon möödus sündmustevaeselt. Ainus probleem tekkis JavaDS-i registreerimisega, mille käigus tekitas segadust, et mõnede URL-ide puhul oli "http://" alguse lisamine vajalik teiste puhul mitte.

5.2.2. Lõpptulemus

Peale JavaDS installeerimist selgus, et EducaNext koos JavaDS-iga vaikimisi SQI-d ei toeta. Ilmnes, et vajalik on installeerida veel üks tarnesüsteem, mis põhimõtteliselt kujutabki endast SQI liidest, lubades edastada õpiobjekte erinevate SQI-d toetavate süsteemide vahel. Saanud EducaNext-i SQI liidese arendaja käest liidese koodi, millega ei kaasnenud seekord mitte mingisugust juhendit, asuti uut tarnesüsteemi installeerima varustatuna vaid vihjega, et see peaks töötama läbi Apache AXIS-e. Peale Apache AXIS-e tööpõhimõttega tutvumist selgus, et arendaja käest saadud JAVA failid on vaja kompilleerida. Siin tekkis uus probleem, nimelt kasutab arendaja oma koodis Java tavapakettides mitteleiduvaid klasse. Selgitanud probleemi arendajale, saadi vastuseks probleemi osaliselt lahendavad JAR failid, mis sisaldasid ühte kahest puuduvast klassist. Küsimustele ühe puuduva klassi kohta arendaja siiani (16.04.2006) vastanud ei ole. Samuti ei õnnestunud selle klassi kohta leida informatsiooni internetist.

Mainida tasub veel, et igasugune veahaldus EducaNextiga töötades on suhteliselt olematu. Kasutajalt eeldatakse, et ta teab täpselt, mis järjekorras peab kõiki tegevusi täitma. Juhul kui kasutaja üritab teha midagi, mida ta veel või enam tegema ei peaks (kuid võimalus selle tegevuse läbiviimiseks on tal täiesti olemas), siis ei ole enamike juhtudel tulemuseks isegi mitte veateade, vaid tühi lehekülg ja rida veateateid logifailis. Halvemal juhul jookseb veebirakendus ka kokku. Näited sellistest situatsioonidest on järgmised:

- Kui EducaNext-i administraator pole veel registreerinud ühtegi tarnesüsteemi, siis kasutaja, kes üritab vaadata enda poole lisatud õpiobjekte, saab tulemuseks tühja lehekülje ja hangunud süsteemi.
- Kui tõlkija õigustega kasutaja üritab tõlkida mõnd kasutajaliidese osa enne, kui on lisatud vähemalt üks tõlkekeel (kusjuures, mitte kuskil ei hoiatata kasutajat, et keel oleks vaja lisada) on tulemuseks üldine veateade ja süsteemi hangumine.

Ülal said kirja pandud ainult probleemid, mis tekkisid otseselt EducaNext-i koodi vigadest või installatsioonijuhendis kirjas olevatest valedest andmetest. Mainimata on jäetud kõik vead, mis tulid mitmetimõistetavustest ja süsteemid pinnapaelasest kirjeldusest.

Kokkuvõtteks võib väita, et kuigi EducaNext-i on võimalik edukalt installeerida ja korrektselt installeerituna süsteem ka töötab, siis pole see kindlasti jõukohane ülesanne algajale administraatorile või kogenud arvutikasutajale. Installatsioonijuhend on puudulik ja kohati suisa vale. Paljud lahendused probleemidele saavutati ainult tänu koostööle EducaNext-i arendusmeeskonnaga.

5.3. Ariadne KPS-i installeerimine

Eelnevalt käesolevale bakalaureusetööle, oli päevakorras ka Ariadne õpiobjektide repositooriumi installeerimine.

Sarnaselt EducaNext-ile on Ariadne KPS (Knowledge Pool System) kolmekihilise arhitektuuriga (vt. Lisa 7):

1) andmekihil paikneb Oracle 8i andmebaas;
2) teenuskihil paikneb AWS (Ariadne Web Services), mis töötab Java veebiserveri abil (Jboss või TomCat);
3) rakenduskihil paiknevad erinevad teenuskihti kasutavad rakendused nagu näiteks SILO (Search & Index Learning Objects), WEBLE ja KPS Client.

Erinevalt EducaNext-ist on Ariadne installatsioonifailid veebist kättesaadavad – tõsi küll ainult organisatsiooniga liitunutele. Märkida tuleb, et liitumine on tasuline.

Korrektseks toimimiseks vajab Ariadne:

- Oracle 8i või uuemat andmebaasi;

- Java veebiserverit – käesoleval juhul valiti selleks TomCat.

Eelinstalleeritud Oracle 8i veebiserverile loodi Ariadne KPS Installer-i abil andmebaas – tänu lihtsale graafilisele kasutajaliidesele toimus andmebaasi ning sellega kaasnevate liideste ja teenuste loomine ning konfigureerimine kiiresti ja veatult.

Vastavalt installatsioonijuhendile installeeriti ka TomCat-i veebiserver. Peale seda asuti paigaldama Ariadne Web Service-t. Kuigi tegutseti installatsioonijuhendit täht-tähelt järgides oli tulemus negatiive. Ariadne Web Service ei hakanud ettenähtud viisil toimima – tööle ei hakkanud isegi graafiline konfigureerimisliides.

Juhendina kasutati Ariadne kodulehel olevat materjali. Nagu ülalmainitud õnnestus läbi viia Ariadne andmebaasi installatsioon, mis toimus täpselt nagu õpetuses kirjeldatud (http://www.ariadne-eu.org/tikiWiki/tiki-page.php?pageName=developingTools_kpsManager). Samuti õnnestus täita Ariadne AWS-i esmase installatioooni juhend (http://www.ariadne-eu.org/tikiWiki/tiki-page.php?pageName=developingTools_aws_installation).

Edasi pidanuks vastavalt manuaalile (http://www.ariadne-eu.org/tikiWiki/tiki-page.php?pageName=developingTools_aws_configuration) toimuma AWS-i konfigureerimine, kuid erinevalt manuaalis kirjasolevast ei olnud kasutajaliidesesse võimalik isegi mitte sisse logida. Manuaalis mainiti taolise olukorra kohta vaid, et ilmselt pole TomCat-i server sisselülitatud või pole AWS õigesti paigaldatud – samas olid mõlemad tingimused kindlasti täidetud.

Kuna Ariadne dokumentatsioonist ega ka vastavasisulistest foorumitest abi ei leitud ja projektiga tegeles ainult käesoleva bakalaureusetöö autor oma vabast ajast, siis jäeti Ariadne installeerimine katki.

5.4. IVA-le loodud SQI liides

Kuna esialgselt kavas olnud IVA ja EducaNext-i ühendamine viibis EducaNext-i SQI liidese kohta käiva oskusteabe puudumise tõttu, otsustati prooviks luua liides IVA ja WordPress blogimootori jaoks.

Esialgne versioon liidesest koosnes viiest Vahur Rebase poolt loodud Python programmeerimiskeelsest failist

1) SQITool.py (vt. Lisa 9) – IVA SQI liides;

2) SQICore.py (vt. Lisa 10) – sisaldab kasutatavaid SQI pärimismeetodeid;
3) SQIInterface.py (vt. Lisa 11) – sisaldab kõigi võimalike SQI meetodite loetelu;
4) SQISessionManager.py (vt. Lisa 12) – sisaldab kasutatavaid SQI sessioonihalduse meetodeid ja klassi SQISession();
5) Exceptions.py (vt. Lisa 13) – SQI API meetodite poolt produtseeritavad vead.

Fail SQITool.py sisaldab klasse nimedega "SQITool()" ja "SQISessionManager()".

Klass SQITool sisaldab järgmisi meetodeid:

konstruktor - klassi külge lisatakse eksemplar klassist SQISessionManager ja käivitatakse klassi SQICore konstruktor;

standard_error_message – tagastab muutujate error_traceback, error_type, error_message ja error_value väärtused;

_processQuery – käivitab meetodi _execQuery ja tagastab selle meetodi käest saadud tulemused;

_execQuery – etteantud muutujate sessID ja queryID väärtuste järgi leitakse meetodite _getSession ja getQuery abil sobiv sessioon ja selle päring. Päringulausest leitakse vastavalt märksõndele otsitavad väärtused ning nende alusel koostatakse RSS formaadis päringuvastus. Käesoleval juhul tagastatakse mittetühi päringutulemus ainult siis, kui märksõna types üheks väärtuseks on blog;

make_query – meetod SOAP/SQI päringu esitamiseks läbi WSDL-i. Läbi WSDL teenuse luuakse teise osapoole (antud juhul WordPress-i) SQI liidesega uus sessioon (SQI liidese meetodi createSession abil). Määratakse päringukeel (setQueryLanguage), milleks sellel juhul on VSQI. Esitatakse sünkroonne päring (synchronousQuery), mis tagastab kohe tulemused. Tulemus viiakse sobivasse formaati ja tagastatakse;

Klass SQITool initsialiseeritakse käsuga Globals.InitializeClass(SQITool).

Klass SQISessionManager sisaldab ainult geneerilise klassi SQISessionManager (asub failis SQISessionManager.py) konstruktorit ja meetodit _doUserCheck.

konstruktor – annab loodud instantsile identifikaatori väärtusega "sessionmanager"

_doUserCheck – kontrollib sessiooni loomisel sihtsüsteemis kasutajanime ja parooli sobivust.

Fail SQISessionManager.py sisaldab klasse sqiSession ja sqiSessionManager. Klass sqiSession sisaldab järgmisi meetodeid:

konstruktor – algväärtustatakse järgmised objekti sqiSession muutujad (enamik neist on SQI API nõuetest tulenevad vaikeväärtused):

- self.id=sessionID – klassi instantsi loomisel antakse instantsile identifikaator;
- self.sessionID=sessionID – sama identifikaator antakse ka klassi muutujale sessionID (seda muutujat kasutatakse reaalselt sessiooni tuvastamiseks);
- self.isAnon=isAnon – loogikamuutuja, mis näitab kas tegu on anonüümse sessiooniga;
- self.username=username – kasutajanimi, on oluline kui tegu pole anonüümse sessiooniga;
- self.started=time.time() – muutuja, mis hoiab instantsi loomise aega, vajalik SQI asünkroonse päringu sessiooni lõpetamiseks vastavalt meetodiga setMaxDuration määratud ajale;
- self.lastAccess=time.time() - muutuja, mis hoiab instantsi viimase kasutamise alguse aega, vajalik SQI asünkroonse päringu sessiooni lõpetamiseks vastavalt meetodiga setMaxDuration määratud ajale;
- self._queryLanguage = 'vsql' – vaikimisi päringukeeleks määratakse vsql;
- self._resultsFormat='RSS' – vaikimisi päringuvastuste formaadiks määratakse RSS;
- self._maxQueryResults=100 – vastavalt SQI API-le määratakse vaikimisi maksimaalseks päringutulemuste arvuks 100;
- self._maxDuration=100 – sellega määratakse vaikimisi maksimaalseks asünkroonse päringusessiooni kestvuseks 100 millisekundit. Vastavalt SQI API-le peaks see vaikeväärtus olema 0;
- self._resultsSetSize=25 – vastavalt SQI API-le määratakse tulemustekollektioonis sisalduvate tulemuste maksimaalseks arvuks 25;
- self._sourceLocation='' - see muutuja peab sisaldama asünkroonse päringu kuulari asukohta URL-ina;
- self._queries={} - massiiv sessiooni käigus esitatud päringute hoidmiseks;
- self._results={} - massiiv sessiooni käigus saadud päringute tulemuste hoidmiseks;

_setQueryLanguage – meetod määrab sessiooni päringukeele;

_setResultsFormat – meetod määrab sessiooni päringu vastuste formaadi;

_setMaxQueryResults – meetod määrab sessiooni maksimaalse päringutulemuste arvu;

_setMaxDuration – meetod määrab asünkroonse päringu sessiooni maksimaalse kestvuse millisekundites;

_setResultsSetSize – meetod määrab asünkroonse päringu tulemustekollektsioonide suuruse;

_setSourceLocation – meetod määrab asünkroonse päringu puhul tulemuste kuulari asukoha URL-ina;

addQuery – lisab sessiooni päringute massiivi päringu kasutades indeksina päringu identifikaatorit. Sünkroonse päringu identifikaator on alati 0;

getQuery – tagastab sessiooni päringumassiivist vastavalt etteantud indeksile päringu;

delQuery – kustutab vastavalt etteantud indeksile sessiooni päringumassiivist päringu;

addResults – lisab sessiooni päringutulemuste massiivi päringutulemused kasutades indeksina päringu identifikaatorit;

getQueries – tagastab sessiooni päringumassiivist kõik päringud;

getResults – tagastab sessiooni päringutulemuste massiivist kõik päringutulemused.

Klass SQISessionManager on geneeriline klass, mis sisaldab järgmisi meetodeid:

createAnonymousSession – SQI API-s nõutud sessioonihaldusmeetod, mis loob anonüümse sessiooni (kutsudes välja meetodi _createNewSession parameetriga isAnon=1) ja tagastab selle;

createSession – SQI API-s nõutud sessioonihaldusmeetod, mis loob uue sessiooni (kutsudes välja meetodi _createNewSession parameetriga isAnon=0) juhul, kui on antud sessiooni loomiseks vajaminevad õiged kasutajanimi ja salasõna, mille sobivust kontrollitakse meetodiga _doUserCheck. Meetod tagastab uue sessiooni või nulli juhul, kui kasutajanimi ja/või salasõna ei sobi;

destroySession – SQI API-s nõutud meetod, mis hävitab sessiooni vastavalt sisestatud sessiooniidentifikaatorile;

_createNewSession – meetod, mida kasutavad nii createAnonymousSession kui ka createSession uue sessiooni loomiseks. Meetod annab väärtuse muutujale sessID ja loob uue instantsi klassist sqiSession andes sellele sessiooniidentifikaatori (sessID) muutuja isAnon väärtuse ja kasutajanime (mis anonüümse sessiooni puhul on tühi). Loodus sessiooni instants

lisatakse SQISessionManager instantsile. Tagastatakse sessiooniidentifikaator;

_getSession – meetod mis tagastab varemloodud sessiooni sessiooniidentifikaatori alusel;

wsdl – meetod, mis tagastab wsdl faili "sqiSessionManagement.wsdl.pt"

Fail SQICore.py sisaldab ühte klassi nimega SQICore, mis sisaldab enamikku SQI API-s kirjeldatud meetodeid:

Konstruktor – lisab massiivile supportedQueryLanguages väärtuse "vsql" ja massiivile supportedResultsFormat väärtuse "RSS".

SetQueryLanguage – SQI API-s nõutud päringu parameetrite seadmise meetod, mis määrab etteantud identifikaatoriga sessioonile (sessioon leitakse identifikaatori alusel, kasutades meetodit _getSession) päringukeele identifikaatori alusel päringukeele (kasutades klassi SQISession meetodit _setQueryLanguage). Kui määratavat päringukeelt ei eksisteeri toetatud päringukeelte hulgas (massiivis supportedQueryLanguages), siis tagastatakse väärtus 0. Kui ei leidu etteantud identifikaatoriga sessiooni, siis luuakse SQI viga NO_SUCH_SESSION (kirjeldatud failis Exceptions.py). Õnnestumise korral tagastab meetod väärtuse 1.

SetResultsFormat – SQI API-s nõutud päringu parameetrite seadmise meetod, mis määrab etteantud identifikaatoriga sessioonile (sessioon leitakse identifikaatori alusel, kasutades meetodit _getSession) vastuste formaadi identfikaatori alusel vastusteformaadi (kasutades klassi SQISession meetodit _setResultsFormat).Kui määratavat vastuseformaati ei eksisteeri toetatavate vastuseformaatide hulgas (massiivis supportedResultsFormat), siis tagastatakse väärtus 0. Kui ei leidu etteantud identifikaatoriga sessiooni, siis luuakse SQI viga NO_SUCH_SESSION (kirjeldatud failis Exceptions.py). Õnnestumise korral tagastab meetod väärtuse 1.

SetMaxQueryResults – SQI API-s nõutud päringu parameetrite seadmise meetod, mis määrab etteantud identifikaatoriga sessioonile (sessioon leitakse identifikaatori alusel, kasutades meetodit _getSession) maksimaalse päringutulemuste arvu (kasutades klassi SQISession meetodit _SetMaxQueryResults). Kui etteantud identifikaatoriga sessiooni ei leita, siis luuakse SQI viga NO_SUCH_SESSION. Õnnestumise korral tagstav meetod väärtuse 1.

SetMaxDuration – SQI API-s nõutud päringu parameetrite seadmise meetod, mis määrab etteantud identifikaatoriga asünkroonsele sessioonile (sessioon leitakse identifikaatori alusel, kasutades meetodit _getSession) maksimaalse päringu kestvuse millisekundites (kasutades klassi SQISession meetodit _SetMaxDuration). Kui etteantud identifikaatoriga sessiooni ei leita, siis

luuakse SQI viga NO_SUCH_SESSION. Õnnestumise korral tagstav meetod väärtuse 1.

setResultsSetSize – SQI API-s nõutud sünkroonse päringu meetod, mis määrab etteantud identifikaatoriga sessioonile (sessioon leitakse identifikaatori alusel, kasutades meetodit _getSession) ühes tulemustekollektsioonis sisalduvate tulemuste maksimaalse arvu (kasutades klassi SQISession meetodit _setResultsSetSize). Kui etteantud identifikaatoriga sessiooni ei leita, siis luuakse SQI viga NO_SUCH_SESSION. Õnnestumise korral tagstav meetod väärtuse 1.

SynchronousQuery – SQI API-s nõutud sünkroonse päringu meetod, mis esitab etteantud identifikaatoriga sessioonile (sessioon leitakse identifikaatori alusel, kasutades meetodit _getSession) päringulause, mis lisatakse sessiooni päringute massiivi kasutades klassi SQISession meetodit addQuery, käivitab päringu (kasutades klassi SQISession meetodit _proccessQuery) ja tagastab päringutulemused.

getTotalResultsCount – SQI API-s nõutud sünkroonse päringu meetod, mis tagab päringu tulemuseks saadud vastuste arvu. Käesoleval juhul tagastab see meetod alati väärtuse 1. Põhimõtteliselt on meetod teostamata.

asynchronousQuery – SQI API-s nõutud asünkroonse päringu meetod, mis esitab etteantud identifikaatoriga sessioonile (sessioon leitakse identifikaatori alusel kasutades meetodit _getSession) päringulause, mis lisatakse sessiooni päringute massiivi kasutades klassi SQISession meetodit addQuery. Meetod tagastab alati väärtuse 1. Põhimõtteliselt on meetod teostamata.

SetSourceLocation – SQI API-s nõutud asünkroonse päringu meetod, mis määrab etteantud identifikaatoriga sessioonile (sessioon leitakse identifikaatori alusel, kasutades meetodit _getSession) päringutulemuste kuulari asukoha, kasutades selleks klassi SQISession meetodit _setSourceLocation. Kui etteantud identifikaatoriga sessiooni ei leita, siis luuakse SQI viga NO_SUCH_SESSION. Meetodi edukal täitmisel tagastatakse 1.

_getSession – meetod, mis sessiooniidentifikaatori põhjal tagastab sessiooni. Kasutab klassi SQISessionManager samanimelist meetodit.

wsdl - meetod, mis tagastab wsdl faili "sqiTarget.wsdl.pt"

5.4.1. IVA päring WordPressi

```
def make_query(self, REQUEST):
    """ make SOAP query to ... somewhere ... """
    hostname = 'http://localhost/wordpress'
    username = 'admin'
    password = 'admin'
    from SOAPpy import WSDL
    auth = WSDL.Proxy(hostname+'/wp-sqi/sqisession.php?wsdl')
    service = WSDL.Proxy(hostname+'/wp-sqi/sqihandler.php?wsdl')
    sessionID = auth.createSession('admin', 'admin')
    if sessionID == '0':
        return "couldn't log in"
    # logic here
    service.setQueryLanguage(sessionID, 'VSQI')
    query = """<?xml version="1.0" encoding="utf-8" ?><query></query>"""
    res = service.synchronousQuery(sessionID, query)
    auth.destroySession(sessionID)
```

Koodinäide 1: Osa meetodist make_query()

Koodinäide 1 demonstreerib päringu tegemist IVA-st WordPress-i. Esmalt defineeritakse muutujad "hostname", "username" ja "password". Esimest neist on vaja WordPress-i URL-i hoidmiseks ning teised SQI sessioonihaldusmeetodi createSession() parameetrid.

Järgmiseks imporditakse WordPress-i WSDL failist WordPress-i SQI liidese funktsionaalsus: muutuja "auth" kaudu pääseb nüüd ligi SQI sessioonihaldusmeetoditele ja muutuja "service" kaudu SQI sünkroonnse pärringu meetoditele. Muutuja "sessionID" väärtustatakse meetodi createSession() poolt tagastatud uue loodud sessiooni identifikaatoriga. Kui tagastatud identifikaatori väärtus on null, siis WordPressi sisselogimine ebaõnnestus (edasi peaks järgnema veahaldus, kuid see on veel tegemata).

Edasi seatakse meetodi setQueryLanguage() abil loodud sessiooni päringukeeleks VSQI. Nüüd defineeritakse päring – näites on tegemist tühja päringuga. Meetodi synchronousQuery() abil esitakse defineeritud päring, meetodi tulemusega väärtustatakse muutuja "res". Sessioon lõpetatakse meetodi destroySession() abil. Edasi toimub juba päringutulemuste töötlemine vastavalt vajadusele

5.4.2. WordPressi päring IVA-sse

```
from SOAPpy import WSDL

auth = WSDL.Proxy('http://localhost:8080/IVA/sqitool/sessionmanager/wsdl')
server = WSDL.Proxy('http://localhost:8080/IVA/sqitool/wsdl')

sessID = auth.createAnonymousSession()
query = """
<?xml version="1.0" encoding="utf-8"?>
<query>
   <user>vahur</user>
   <category>blog</category>
</query>
"""
results = server.synchronousQuery(sessID, query, 0)
auth.destroySession(sessID)
```

Koodinäide 2: lihtne päring IVA-sse

Koodinäide 2 demonstreerib väga lihtsat WordPress-i päringut IVA-sse. Esmalt väärtustatakse muutujad "auth" ja "session" IVA SQI liideses olevate meetodite wsdl() (tegemist on kahe erineva meetodiga, üks neist paiknev failis "SQICore.py", kust selle impordib klass SQITool() ja teine failis SQISessionManager.py) abil nii, et nende kaudu on kättesaadav IVA SQI liidese funktsionaalus. Sarnaselt eelmisele näitele on SQI sessioonihaldus kättesaadav läbi muutuja "sess" ja SQI sünkroonse päringu haldus läbi muutuja "server".

Meetodi createAnonymousSession() abil luuakse uus sessioon, mille identifikaatoriga väärtustatakse muutuja "sessID". Muutuja "query" väärtustatakse päringuga, milles täpsustakse, et soovitavad andmed kuuluvad kasutajale "vahur" ja on tüüpi "blog" (teist tüüpi andmeid IVA SQI liides ei tagastagi). Meetodi synchronousQuery() abil esitakse päring IVA SQI liidesele. Sama meetod tagastab ka tulemused, millega väärtustatakse muutuja "results". Meetodi destroySession() abil sessioon suletakse. Edasi töödeldakse päringutulemusi vastavalt vajadusele.

Kokkuvõtteks võib väita, et IVA SQI liides on käesoleval hetkel teostatud osaliselt – põhjuseks EducaNext-i installatsiooniprobleemidest tekkinud ajakulu. Praeguse algse versiooni juures on keskendutud liidese osalise töövõime saavutamisele koos WordPress blogimootoriga. Teostamata on veel veahaldus (kasutatakse vaid ühte võimalikku SQI viga). Samuti on asünkroonsete päringutega seotud meetodid teostatud minimaalselt ning käesoleval hetkel asünkroonseid päringuid teostada ei saa. Täiesti puudu on asünkroonsete päringute tulemuste

kuular ja päringu kestvuse kontroll.

Samas võimaldab loodud mudel siiski kõiki SQI API-s kirjeldatud meetodeid sedavõrd täiendada, et tulevikus tagada IVA SQI liidese täielik funktsionaalsus. Vaatamata ilmennud probleemidele projekt IVA SQI liidese loomiseks kestab ning eesmärk, milleks on täisfunktsionaalsusega SQI liides, loodetakse saavutada lähima paari kuu jooksul.

Kokkuvõte

Käesoleva bakalaureusetöö teoreetilise osa peamiseks eesmärgiks oli anda ülevaade olemasolevatest õpihaldusüsteemidest ning selgitada nendega seonduvaid mõisteid, pöörates erilist tähelepanu koostalitlusele ja selle vajadusest tulenevatele probleemidele.

Samuti oli teoreetilise osa eesmärgiks anda detailne ülevaade SQI liidesest jätmata seejuures tähelepanuta ka teisi andmevahetusstandardeid ja kaasusi, mille puhul SQI-d on edukalt implementeeritud.

Autor leiab, et bakalaureusetöö referatiivne osa teenib lisaks enesearendusele ja praktilise osa ettevalmistamisele ka laiemat eesmärki, tehes kättesaadavamaks sissejuhatavat emakeelset materjali, kuna teemakohast eestikeelset materjali praktiliselt ei eksisteeri.

Bakalaureusetöö praktiliseks eesmärgiks sai püstitatud IVA õpihaldussüsteemi ja EducaNext õpiobjektide repositooriumi vahelise koostalitluse tagamise SQI liidese abil. Tegemist oli laiema projektiga ning autori osa projektis piirdus testserveri ning tarkvara installeerimise, konfigureerimise ning haldamisega. Samuti sai seatud eesmärgiks valmiva SQI liidese dokumenteerimine.

Töö praktilist osa hindab autor enda jaoks kõige huvitamaks ning vajalikke kogemusi pakkuvaks ehki mitte küll põhjustel, mida loodeti töö alguses. EducaNext portaali installeerimisel tekkinud probleemide jada on heaks õppetunniks, et projeketitöö käigus ilmneb sageli ettenägematuid asjaolusid ja probleeme, mille lahendamisele kulub ettenähtust märksa rohkem aega. Kuigi portaal sai lõpuks installeeritud, osutus selle funktsionaalsus valitud eesmärgi saavutamiseks ebapiisavaks – lisafunktionaalsuse loomine veel kestab. Samas on selle protsessi käigus omandatud praktilised teadmised äärmiselt väärtuslikud ning autor ei pea projektile kulutatud aega raisatuks, vaatamata sellele, et projekti eesmärki ei ole veel saavutatud.

Töö tulemusena valmis IVA jaoks osalist funktsionaalsust omav SQI liides, mille tööpõhimõtted autor ka projekti käigus dokumenteeris ning liidese edasisel arendamisel ka edaspidi dokumenteerida kavatseb.

Kokkuvõttes peab autor käesolevat bakalureusetööd õnnestunuks, kuna täidetud said kõik teoreetilise osaga seotud eesmärgid ning osaliselt ka praktilised eesmärgid. Töö praktilise osa juures ilmenud probleemid omavad lisaväärtust hoiatusena projektide liiga optimistliku planeerimise eest.

Summary

This bachelor's thesis titled "Interoperability of Learning Management Systems. The Case of IVA" consists of theoretical and practical part. The goals of this thesis are:

- to define terms connected with Learning Management Systems (LMS);
- to give overview of existing LMS-s;
- to describe problems and solutions of making LMS-s interoperable;
- to describe different communication standards, specially SQI;
- to describe the implementation of SQI on IVA;
- to document IVA SQI.

The aim of theoretical part of this bachelor's thesis is to describe and explore different Learning Management Systems (LMS), while giving special attention to their interoperability – systems described are Moodle, OLAT and WebCT. Theoretical part also contains definitions of key terms, such as "interoperability", "Learning Management System" and "learning object".

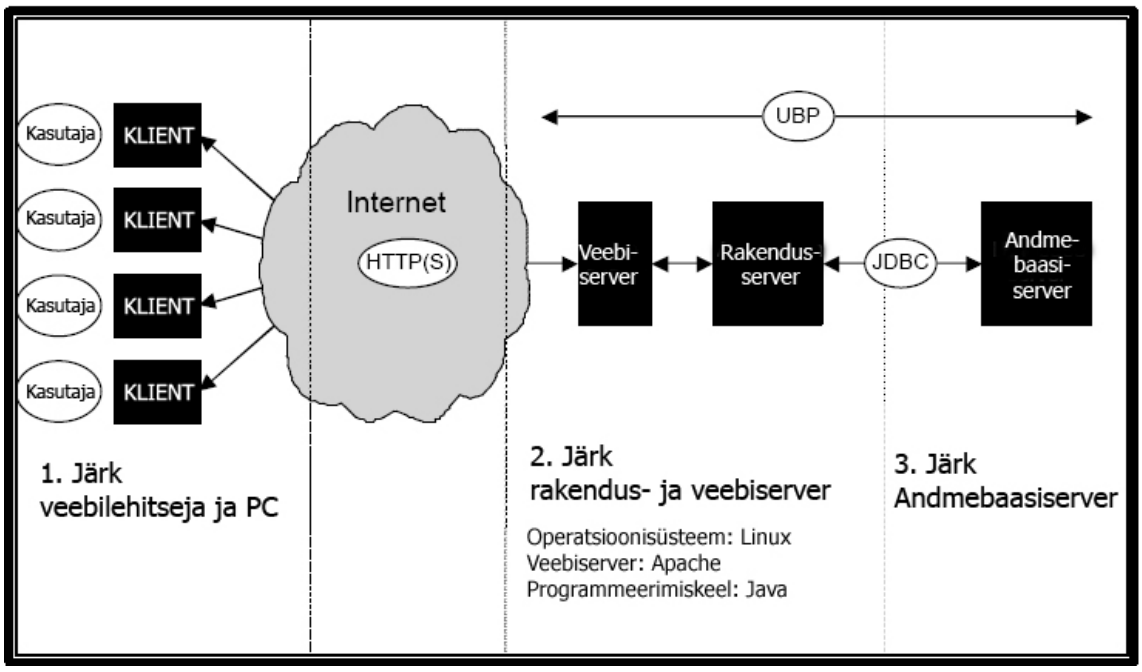
After describing existing LMS-s the thesis takes more theoretical approach giving brief overview of the needs for interoperability and solutions for achieving that. Then the different communication standards such as standards for metadata, content packaging, learners profile and content delivery, are described.

Another important part of theory is thorough description of SQI – it's principles and Application Programming Interface (API). After description of SQI some cases of it's implementation on widely known LMS-s (Ariadne, Celebrate and iClass, ELENA) are described.

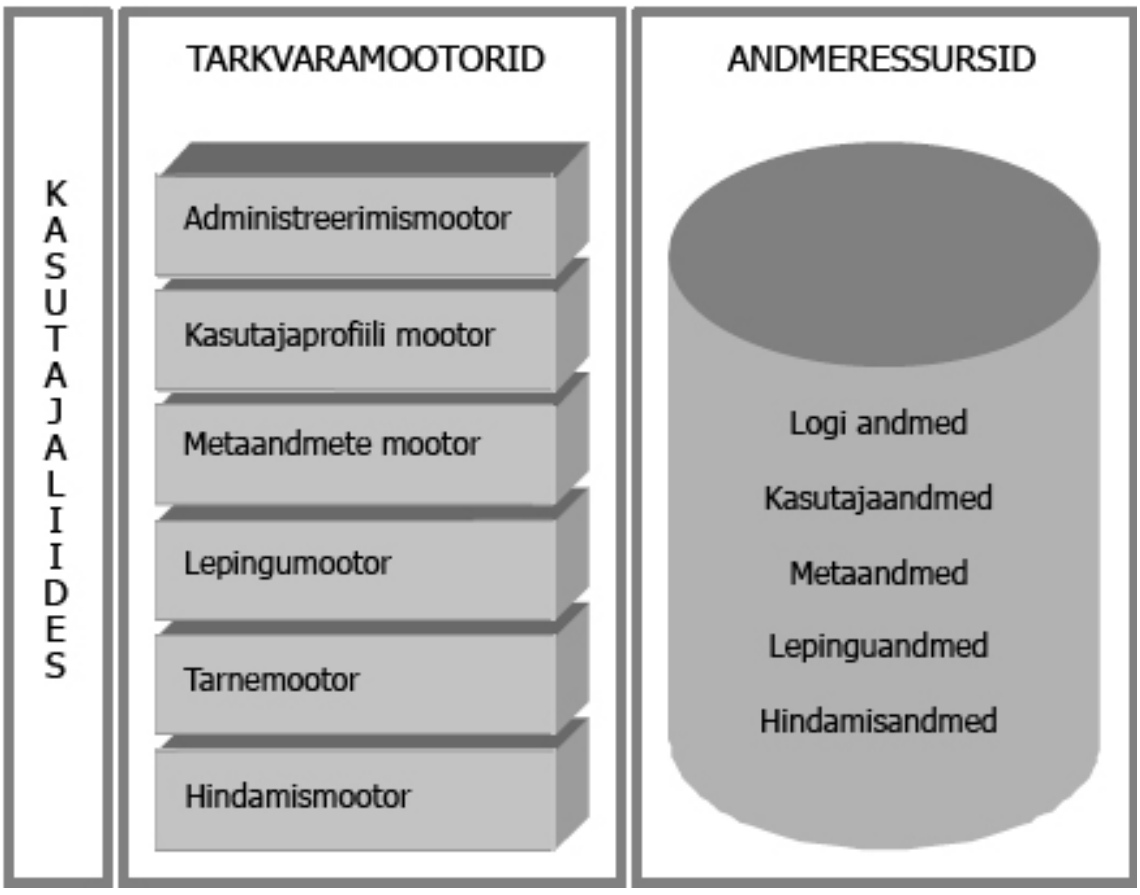
The practical part of this thesis is implementation of SQI on IVA (Tallinn University LMS). At first it was planned as a interface between IVA and EducaNext but because of the large amount of problems with EducaNext installation (although finally successful) the aim of the project was temporarly changed to create a sample SQI interface which succeeded. Since authors part in this project was to be a system administrator and document the created SQI interface, the practical part of this thesis contains overview of EducaNext, it's installation process and problems encountered during it. Also the documentation of IVA SQI and example of it's work is included.

Lisad

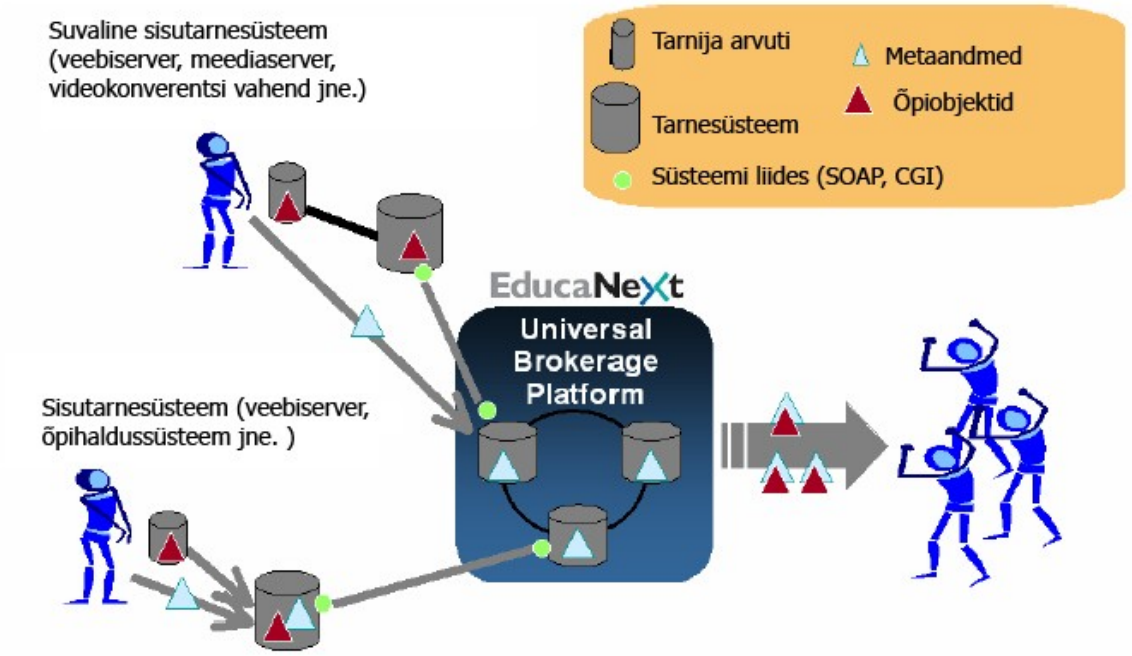
1) UBP kolmekihiline struktuur [Allikas 21, lk 4]



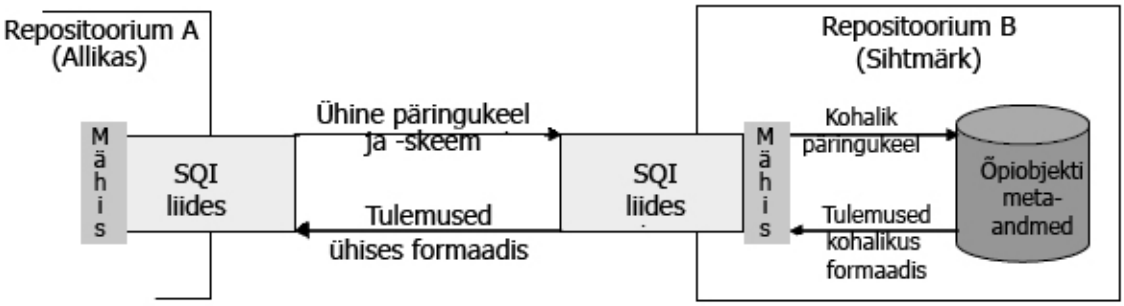
2) UBP tarkvaramootorid ja andmebaasid [Allikas 21, lk 5]



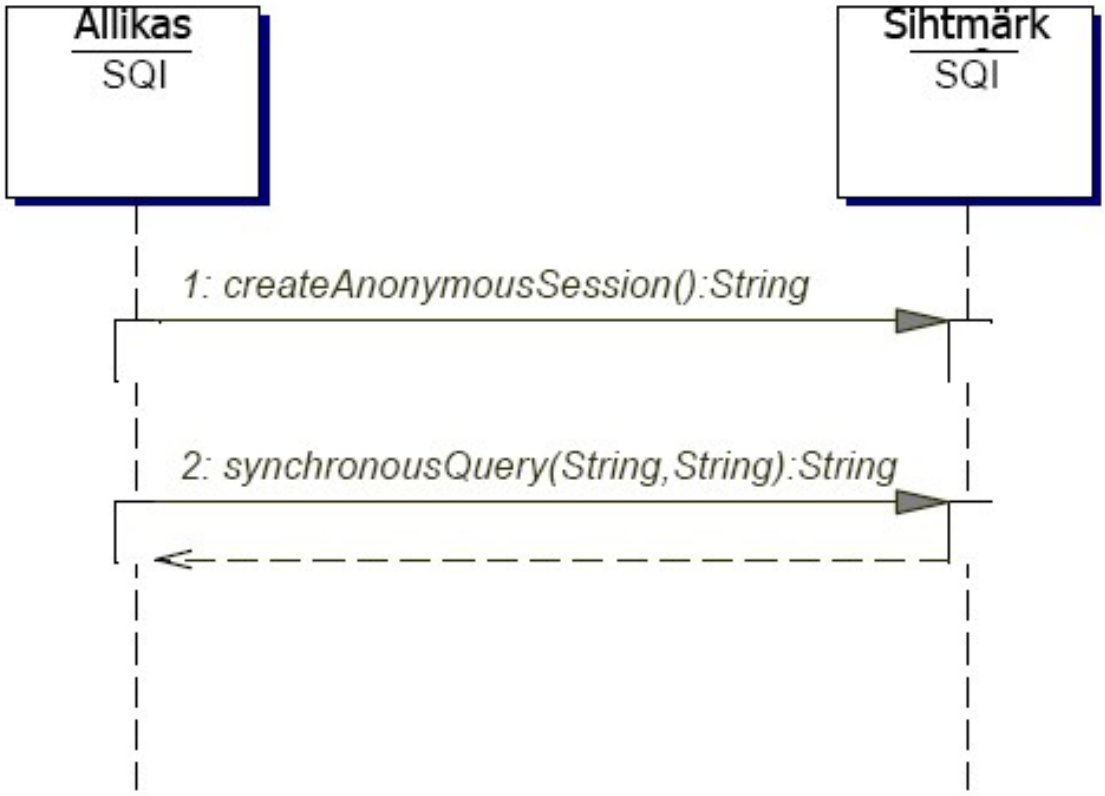
3) EducaNext portaali arhitektuur [Allikas 20, lk. 4]



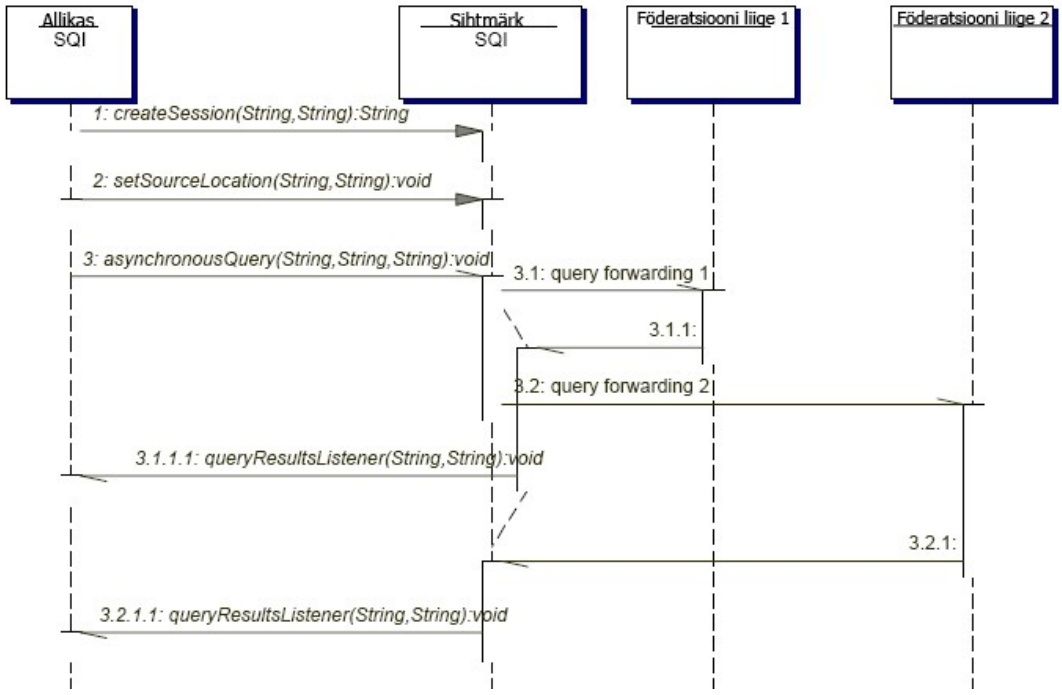
4) Kommunikatsioon kahe õpiobjektide repositooriumi vahel [allikas 4, lk 3]



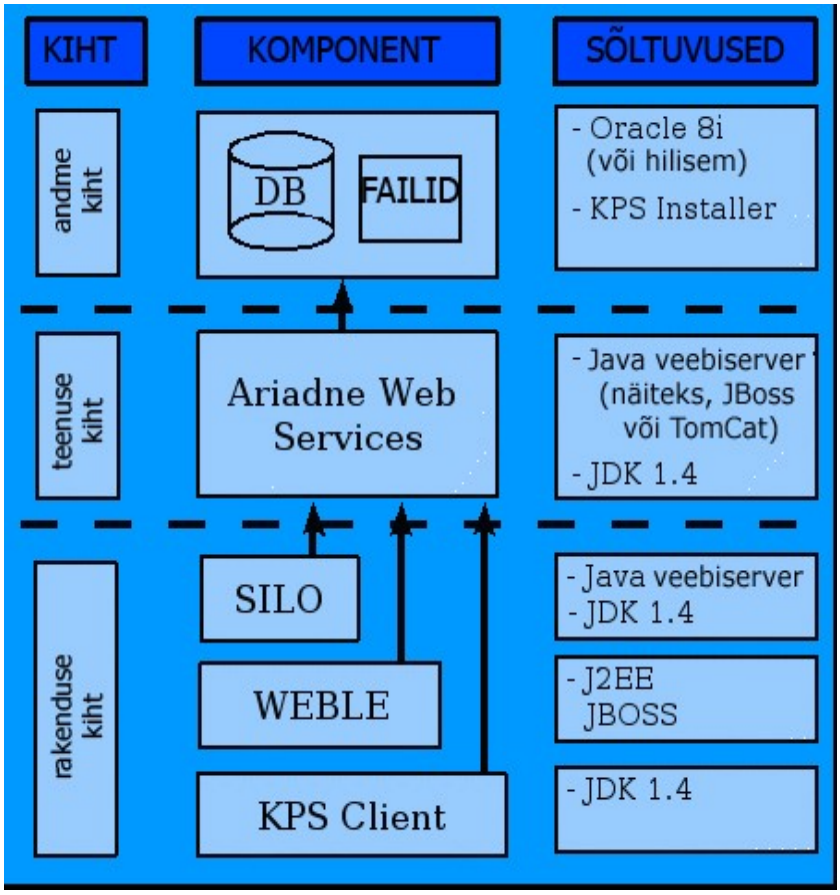
5) SQI sünkroonne pärimisviis [Allikas 2, lk 2]



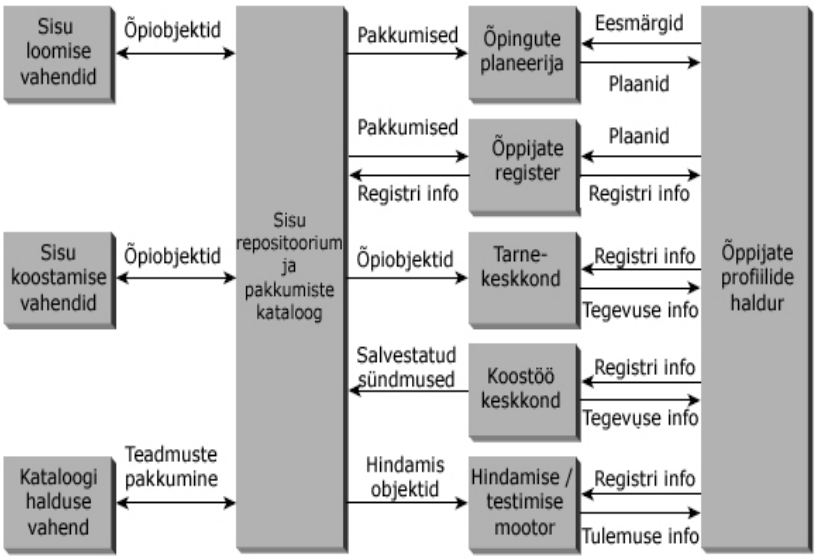
6) SQI asünkroonne pärimisviis [Allikas 2, lk 2]



7) Ariadne struktuur [Allikas 22]



8) E-õppe mudel [Allikas 13, lk 5]



9) Faili "SQITool.py" sisu

```
import Globals
from Globals import Persistent
from OFS import Folder
from AccessControl import ClassSecurityInfo
from Cruft import Cruft
from common import translate, perm_edit, perm_view, mkTime
from Products.SQICore.SQICore import SQICore
from Products.SQICore.SQISessionManager import SQISessionManager
from Products.SQICore.interfaces.SQIInterface import ISQIInterface
from Products.SQICore.interfaces.SQIInterface import ISQISessionManager
from zope.interface import implements

from ImportExport import get_text

class SQITool(Folder.Folder,
        SQICore,
        Persistent,
        Cruft
        ):
    """ SQI Tool """
    implements(ISQIInterface)

    meta_type = 'SQITool'
    security = ClassSecurityInfo()
    security.declareObjectPublic()

    def __init__(self):
        """ initialize SQITool """
        self.id = 'sqitool'
        ssm = SQISessionManager()
        self._setObject(ssm.id, ssm)
        SQICore.__init__(self)

    def standard_error_message(self, error_value='', error_traceback='', error_tb='',
error_type='', error_message='', error_log_url=''):
        """ asd """
        return error_traceback, error_type, error_message, error_value

    def _processQuery(self, sessID, queryID):
        """ execute query, package results as client requested """
        # do query
        results = self._execQuery(sessID, queryID)
        # package results
        #formatted = self._formatResults(sessID, results)
        return results

    def _execQuery(self, sessID, queryID):
        """ execute query """
        sess = self._getSession(sessID)
        query = sess.getQuery(queryID)
        query = query.strip()

        from xml.dom.minidom import parseString
        q = parseString(query)

        user = get_text(q.getElementsByTagName('user')[0])
        types = get_text(q.getElementsByTagName('category')[0])
        course = get_text(q.getElementsByTagName('course')[0])
        objectid = get_text(q.getElementsByTagName('objectid')[0])
        ret_what = get_text(q.getElementsByTagName('return')[0]).split(',')

        cat = None
        cat_query = {}
        if types == 'blog':
            cat = self.blogs_zcatalog
            cat_query['getAuthor'] = user
        if objectid:
```

```
            # return specified object
            pass
        if not cat:
            return []
        results = cat.search(cat_query)

        formatted = """<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0"
    xmlns:content="http://purl.org/rss/1.0/modules/content/"
    xmlns:wfw="http://wellformedweb.org/CommentAPI/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    >
<channel>
\t<title></title>
\t<link></link>
\t<description></description>
\t<generator>IVA</generator>
\t<language></language>\n"""
        u = self.acl_users.getUser(sess.username).__of__(self)
        from AccessControl.SecurityManagement import newSecurityManager,
getSecurityManager
        import time
        newSecurityManager(None, u)
        #print getSecurityManager().getUser()
        for x in results:
            obj = x.getObject()
            cat = ''
            abs_url = obj.absolute_url()
            if 'webtop' in abs_url:
                tmp_u = abs_url.split('/')
                i = tmp_u.index('webtop')
                ccid = tmp_u[i+1][1:]
                cat = self.courses.get_child(ccid).get_name()
            timetime = obj.getTime()
            if obj.hasBeenChanged():
                timetime = obj.getChangeTime()
            # 2006-03-22 10:39:14
            #time_formatted = time.strftime("%a %b %Y-%m-%d %H:%M:%S
+0000",time.localtime(timetime))
            #time_formatted = time.strftime("%a, %d %b %Y %H:%M:%S
+0000",time.localtime(timetime))
            time_formatted = time.strftime("%Y-%m-%d
%H:%M:%S",time.localtime(timetime))
            intro = unicode(x.getIntro, 'utf-8')
            body = unicode(x.getContent, 'utf-8')
            title = unicode(x.getTitle, 'utf-8')
            formatted += '\t<item>\n'
            formatted += '\t\t<title>%s</title>\n' % title
            formatted += '\t\t<link>%s</link>\n' % abs_url
            formatted += '\t\t<comments>%s</comments>\n' % abs_url
            formatted += '\t\t<pubDate>%s</pubDate>\n' % time_formatted
            formatted += '\t\t<dc:creator>%s</dc:creator>\n' % sess.username
            if cat:
                formatted += '\t\t<category>%s</category>\n' % cat
            formatted += '\t\t<guid isPermalink="yes">%s</guid>\n' % abs_url
            formatted += '\t\t<description><![CDATA[%s]]></description>\n' % intro
            formatted +=
'\t\t<content:encoded><![CDATA[<p>%s</p><p>%s</p>]]></content:encoded>\n' % ( intro,
body)
            #formatted += '\t\t<wfw:commentRSS>%s</wfw:commentRSS>\n' % cat
            formatted += '\t</item>\n'
        formatted += '</channel>\n</rss>'
        return formatted

    def _formatResults(self, sessID, results):
        """ package results as requested by client """
        return

    #
    # methods for making SOAP/SQI query
    #
```

```
    def make_query(self, REQUEST):
        """ make SOAP query to ... somewhere ... """
        hostname = 'http://localhost/wordpress'
        username = 'admin'
        password = 'admin'
        from SOAPpy import WSDL
        auth = WSDL.Proxy(hostname+'/wp-sqi/sqisession.php?wsdl')
        service = WSDL.Proxy(hostname+'/wp-sqi/sqihandler.php?wsdl')
        sessionID = auth.createSession('admin', 'admin')
        if sessionID == '0':
            return "couldn't log in"
        # logic here
        service.setQueryLanguage(sessionID, 'VSQI')
        query = """<?xml version="1.0" encoding="utf-8" ?><query></query>"""
        res = service.synchronousQuery(sessionID, query)
        auth.destroySession(sessionID)
        REQUEST.RESPONSE.setHeader('Content-Type', 'text/xml')
        from xml.dom.minidom import parseString
        q = parseString(res)
        q.normalize()
        posts = []
        for x in q.getElementsByTagName('seq'):
            for y in x.childNodes:
                if y.nodeType != 1:
                    continue
                name = y.nodeName
                value = ''
                for c in y.childNodes:
                    if c.nodeType == 4:
                        value = c.nodeValue
                print name, "\t\t", value
        return res

Globals.InitializeClass(SQITool)

class SQISessionManager(Folder.Folder,
        SQISessionManager,
        Persistent,
        Cruft
        ):
    """ SQI session manager """
    implements(ISQISessionManager)

    meta_type = 'SQISessionManager'
    security = ClassSecurityInfo()
    security.declareObjectPublic()
    manage_options = Folder.Folder.manage_options

    def __init__(self):
        """ initialize sqi session manager """
        self.id = 'sessionmanager'

    def _doUserCheck(self, username, password):
        """ check username and password """
        if self.acl_users.authenticate(username, password, None):
            return 1
        if username == 'riho' and password == 'mattias':
            return 1
        return 0
```

10) Faili "SQICore.py" sisu

```
from zope.interface import implements
from OFS.SimpleItem import SimpleItem
from interfaces.SQIInterface import ISQIInterface
from Exceptions import NoSuchSessionException as NO_SUCH_SESSION

class SQICore:
    implements(ISQIInterface)

    def __init__(self):
        self.supportedQueryLanguages = ['vsql',]
        self.supportedResultsFormat = ['RSS',]

    def setQueryLanguage(self, targetSessionID, queryLanguageID):
        """ set query language """
        if queryLanguageID in self.supportedQueryLanguages:
            sess = self._getSession(targetSessionID)
            if sess:
                sess._setQueryLanguage(queryLanguageID)
            else:
                raise NO_SUCH_SESSION
        else:
            # FIXME: raise not supported
            return 0
        return 1

    def setResultsFormat(self, targetSessionID, resultsFormat):
        """ set results format """
        if resultsFormat in self.supportedResultsFormat:
            sess = self._getSession(targetSessionID)
            if sess:
                sess._setResultsFormat(resultsFormat)
            else:
                raise NO_SUCH_SESSION
        else:
            # FIXME: raise not supported
            return 0
        return 1

    def setMaxQueryResults(self, targetSessionID, maxQueryResults):
        """ set maximum query results """
        # TODO: int checking
        sess = self._getSession(targetSessionID)
        if sess:
            sess._setMaxQueryResults(maxQueryResults)
        else:
            raise NO_SUCH_SESSION
        return 1

    def setMaxDuration(self, targetSessionID, maxDuration):
        """ set maximum duration """
        # TODO: int checking?
        sess = self._getSession(targetSessionID)
        if sess:
            sess._setMaxDuration(maxDuration)
        else:
            raise NO_SUCH_SESSION
        return 1

    # SYNCHRONOUS QUERY INTERFACE
    def setResultsSetSize(self, targetSessionID, resultsSetSize):
        """ maximum number of results, which will be returned by a single results set
"""
        sess = self._getSession(targetSessionID)
        if sess:
            sess._setResultsSetSize(resultsSetSize)
        else:
            raise NO_SUCH_SESSION
        return 1
```

```
    def synchronousQuery(self, targetSessionID, queryStatement, startResult):
        """ query """
        # queryID here is 0.
        sess = self._getSession(targetSessionID)
        if not sess:
            raise NO_SUCH_SESSION
        sess.addQuery(queryStatement, 0)
        results = self._processQuery(targetSessionID, 0)
        return results

    def getTotalResultsCount(self, targetSessionID, queryStatement):
        """ number of total results """
        return 1

    # ASYNCHRONOUS QUERY INTERFACE
    def asynchronousQuery(self, targetSessionID, queryStatement, queryID):
        """ async query """
        # TODO: should we do some checking here?
        #       allowed characters in queryID
        #       cannot allow 0 here!
        self._getSession(targetSessionID).addQuery(queryStatement, queryID)
        return 1

    def setSourceLocation(self, targetSessionID, sourceLocation):
        """ set the URL where to return query results """
        sess = self._getSession(targetSessionID)
        if sess:
            sess._setSourceLocation(sourceLocation)
        else:
            raise NO_SUCH_SESSION
        return 1
    # helper methods

    def _getSession(self, sessionID):
        """ get session """
        sc = self.sessionmanager._getSession(sessionID)
        return sc

    def wsdl(self):
        """ return wsdl file """
        from Products.PageTemplates.PageTemplateFile import PageTemplateFile
        wsdl = PageTemplateFile('sqiTarget.wsdl.pt', globals())
        return wsdl.__of__(self)()
```

11) Faili "SQIInterface.py" sisu

```
from zope.interface import Interface

class ISQISessionManager(Interface):
    """ Session manager """

    def createAnonymousSession():
        """ create anonymous session """

    def createSession(username, password):
        """ create session - log in """

    def destroySession(sessionID):
        """ destroy session """


class ISQIInterface(Interface):
    """ General interface for SQI """
```

```
    #
    #  QUERY PARAMETER CONFIGURATION
    #
    def setQueryLanguage(targetSessionID, queryLanguageID):
        """
        This method allows the source to control the syntax used in the query
statement by identifying
        the query language. Values for the parameter queryLanguageID are case-
insensitive.

        targetSessionID:string
        queryLanguageID:string
        faults: NO_SUCH_SESSION, QUERY_LANGUAGE_NOT_SUPPORTED, METHOD_FAILURE
        return void
        """

    def setResultsFormat(targetSessionID, resultsFormat):
        """
        This method allows the source to control the format of the results returned by
the target. The
        format according to which the results shall be formatted is specified in the
resultsFormat
        parameter. The parameter is provided via a URI (e.g., the LOM XML Schema
definitions files
        are available at
http://standards.ieee.org/reading/ieee/downloads/LOM/lomv1.0/) or via predefined
        values that are case-insensitive.

        targetSessionID:string
        resultsFormat:string
        faults: NO_SUCH_SESSION, RESULTS_FORMAT_NOT_SUPPORTED, METHOD_FAILURE
        """

    def setMaxQueryResults(targetSessionID, maxQueryResults):
        """
        This method defines the maximum number of results, which a query will produce.
The
        maximum number of query results is set to 100 by default, but can be
controlled via this
        method. maxQueryResults must be 0 (zero) or greater. If the maximum number of
query
        results is set to 0 (zero), the source does not want to limit the number of
maximum query
        results produced.

        targetSessionID:string
        maxQueryResults:integer
        faults: NO_SUCH_SESSION, INVALID_MAX_QUERY_RESULTS, METHOD_FAILURE
        """

    def setMaxDuration(targetSessionID, maxDuration):
        """
        This method enables the source to set a time-out for the query in case of an
asynchronously
        operated query interface. The values of maxDuration must be 0 (zero) or
greater. A source
        delegates the time out management of the query to the target by setting
maxDuration to 0
        (zero). The parameter maxDuration is interpreted in milliseconds. The default
value is zero
        (i.e., time out management is delegated to the target).

        targetSessionID:string
        maxDuration:integer
        faults: NO_SUCH_SESSION, INVALID_MAX_DURATION, METHOD_FAILURE
        """

    #
    # SYNCHRONOUS QUERY INTERFACE
    #
    def setResultsSetSize(targetSessionID, resultsSetSize):
```

```
        """
        This method defines the maximum number of results, which will be returned by a
single
        results set. The size of the results set is set to 25 records by default, but
can be controlled via
        this method. resultsSetSize must be 0 (zero) or greater. A source asks for all
results when the
        maximum number of results is set to 0 (zero).

        targetSessionID:string
        resultsSetSize:integer
        faults: NO_SUCH_SESSION, INVALID_RESULTS_SET_SIZE, QUERY_MODE_NOT_SUPPORTED,
METHOD_FAILURE
        """

    def synchronousQuery(targetSessionID, queryStatement, startResult):
        """
        This method places a query at the target. The query statement is provided via
the
        queryStatement parameter. Within a session identified via targetSessionID
multiple queries
        can be submitted simultaneously. The method returns a set of metadata records
matching the
        query. The startResult parameter identifies the start record of the results
set. The index of the
        result set size starts with 1. The number of results returned is controlled by
        setResultsSetSize and its default value. A valid number for startResult can
range from
        1 to the total number of results. The total number of results produced is
limited by
        setMaxQueryResults and its default value.

        targetSessionID:string
        queryStatement:string
        startResult:integer
        faults: NO_SUCH_SESSION, INVALID_QUERY_STATEMENT, QUERY_MODE_NOT_SUPPORTED,
                METHOD_FAILURE, NO_MORE_RESULTS
        """

    def getTotalResultsCount(targetSessionID, queryStatement):
        """
        This method returns the total number of available results of a query. The
targetSessionID
        identifies the session. The query is provided via the queryStatement parameter
(see Section
        3.1 on Stateful versus Stateless Implementation).

        targetSessionID:string
        queryStatement:string
        faults: NO_SUCH_SESSION, QUERY_MODE_NOT_SUPPORTED, INVALID_QUERY_STATEMENT,
METHOD_FAILURE
        """

    #
    # ASYNCHRONOUS QUERY INTERFACE
    #
    def asynchronousQuery(targetSessionID, queryStatement, queryID):
        """
        This method allows the source to submit a query to the target, while the
results are returned in
        an asynchronous method. The query statement is provided via the queryStatement
parameter.
        A query ID issued by the source is required in order to link the query results
with the query,
        when they are later returned using the results listener. By using unique query
IDs it is possible
        to submit an arbitrary number of queries per active session. The location of
the sourceâ€™s
        results listener is needed and must be provided using setSourceLocation
method.
        Due to the asynchronous nature of this method, query results could still
```

```
arrive from previous
        queries. The query is processed and results are forwarded within the timeframe
specified in
        the setMaxDuration method.

        targetSessionID:string
        queryStatement:string
        queryID:string
        faults: NO_SUCH_SESSION, QUERY_MODE_NOT_SUPPORTED, NO_SOURCE_LOCATION,
                INVALID_QUERY_STATEMENT, METHOD_FAILURE
        """

    def setSourceLocation(targetSessionID, sourceLocation):
        """
        This method is required to be called before a query is submitted in
asynchronous mode. The
        parameter sourceLocation specifies the location of the sourceâ€™s results
listener in order for the
        target to be able to send the results. The sourceLocation must be an URL.

        targetSessionID:string
        sourceLocation:string
        faults: NO_SUCH_SESSION, QUERY_MODE_NOT_SUPPORTED, METHOD_FAILURE
        """

    def queryResultsListener(queryID, queryResults):
        """
        this must be implemented on client not here!

        This target-initiated method forwards the results sets to the source. The
queryID parameter is
        used for linking the query results to previously submitted query, when they
are later return
        using the results listener.
        The queryResults holds a results set consisting of a list of metadata records,
which is
        formatted according to the schema specified in the query.

        queryID: string
        queryResults: string
        faults: INVALID_QUERY_RESULTS, NO_SUCH_QUERY, METHOD_FAILURE
        """
```

12) Faili "SQISessionManager.py" sisu

```
from zope.interface import implements

from OFS.SimpleItem import SimpleItem
from interfaces.SQIInterface import ISQISessionManager
from random import choice
import time

class sqiSession(SimpleItem):
    meta_type = 'sqi_session'

    def __init__(self, sessionID, isAnon=0, username=''):
        """ create session object """
        self.id = sessionID
        self.sessionID = sessionID
        self.isAnon = isAnon
        self.username = username
        # TODO: implement session timeout based on these values
        self.started = time.time()
        self.lastAccess = time.time()
        self._queryLanguage = 'vsql'
        self._resultsFormat = 'RSS'
        self._maxQueryResults = 100
        self._maxDuration = 100
        self._resultsSetSize = 25
```

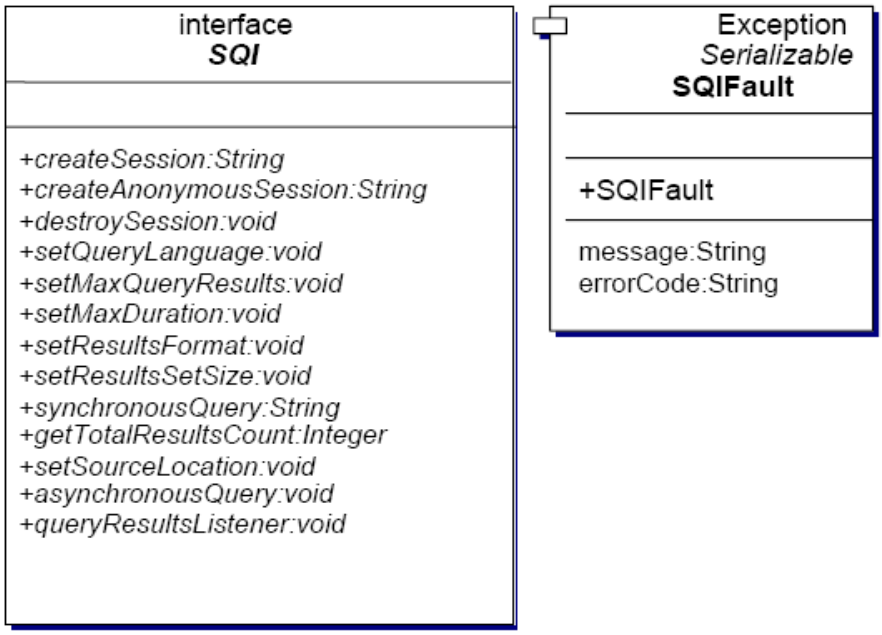
```
        self._sourceLocation = ''

        self._queries = {}
        self._results = {}

    def _setQueryLanguage(self, ql):
        self._queryLanguage = ql

    def _setResultsFormat(self, rf):
        self._resultsFormat = rf

    def _setMaxQueryResults(self, max):
        self._maxQueryResults = max

    def _setMaxDuration(self, dur):
        self._maxDuration = dur

    def _setResultsSetSize(self, size):
        self._resultsSetSize = size

    def _setSourceLocation(self, location):
        self._sourceLocation = location

    def addQuery(self, query, queryID):
        # queryID is 0 when it's syncQuery
        self._queries[queryID] = query

    def getQuery(self, queryID):
        """ return query """
        return self._queries[queryID]

    def delQuery(self, queryID):
        """ delete query """
        del self._queries[queryID]

    def addResults(self, results, queryID):
        # queryID is 0 when it's syncQuery
        self._results[queryID] = results

    def getQueries(self):
        """ return queries """
        return self._queries

    def getResults(self):
        """ return results """
        return self._results

class SQISessionManager:
    implements(ISQISessionManager)

    def __init__(self):
        """ init """
        pass

    def createAnonymousSession(self):
        """ create anonymous session """
        return self._createNewSession(isAnon=1)

    def createSession(self, username, password):
        """ create session. I don't care what the username and password is.
        """
        result = 1
        # we could add zope's internal user authentication here
        if hasattr(self, '_doUserCheck'):
            result = self._doUserCheck(username, password)
        if result:
            return self._createNewSession(isAnon=0, username=username)
        # FIXME: raise an SQIFault!
        return 0

    def destroySession(self, sessionID):
```

```
        """ destroy session eg. logout of something """
        self._delObject(sessionID)

    def _createNewSession(self, isAnon=1, username=''):
        """ create session and return session id """
        from string import letters, digits
        chars = letters+digits
        sessID = ''.join([choice(chars) for i in range(15)])
        s = sqiSession(sessID, isAnon, username)
        self._setObject(s.id, s)
        return sessID

    def _getSession(self, sessID):
        """ get session """
        for x in self.objectValues('sqi_session'):
            if x.sessionID == sessID:
                return x
        return None

    def wsdl(self):
        """ return wsdl file """
        from Products.PageTemplates.PageTemplateFile import PageTemplateFile
        wsdl = PageTemplateFile('sqiSessionManagement.wsdl.pt', globals())
        return wsdl.__of__(self)()
```

13) Faili "exceptions.py" sisu

```
class NoSuchSessionException(Exception): pass
class QUERY_LANGUAGE_NOT_SUPPORTED(Exception): pass
class METHOD_FAILURE(Exception): pass
class RESULTS_FORMAT_NOT_SUPPORTED(Exception): pass
class METHOD_FAILURE(Exception): pass
class INVALID_MAX_QUERY_RESULTS(Exception): pass
class INVALID_MAX_DURATION(Exception): pass
class INVALID_RESULTS_SET_SIZE(Exception): pass
class QUERY_MODE_NOT_SUPPORTED(Exception): pass
class INVALID_QUERY_STATEMENT(Exception): pass
class NO_MORE_RESULTS(Exception): pass
class NO_SOURCE_LOCATION(Exception): pass
class NO_SUCH_QUERY(Exception): pass
```

14) SQI UML-i klassidiagramm [Allikas 2, lk 14]



Allikad

1. http://www.cetis.ac.uk/content2/20040227011926/printArticle ; Wilbert Kraan; "The cockroach of repository interoperability: Simple Query Interface" (11.04.2006)
2. http://nm.wu-wien.ac.at/e-learning/interoperability/SQI_V1.0beta_2005_04_13.pdf ; Bernd Simon, David Massart, Frans Van Assche, Stefaan Ternier, Erik Duval ; "Simple Query Interface Specification version 1.0 Beta" (11.04.2006)
3. http://en.wikipedia.org/wiki/Learning_management_system (11.04.2006)
4. http://nm.wu-wien.ac.at/e-learning/interoperability/www2005-workshop-sqi-2005-04-14.pdf ; Bernd Simon, David Massart, Frans van Assche, Stefaan Ternier, Erik Duval, Stefan Brantner, Daniel Olmedilla, Zoltán Miklós; "A Simple Query Interface for Interoperable Learning Repositories" (11.04.2006)
5. Arvi Tavast, Vello Hanson; "Arvutikasutaja inglise-eesti sõnastik"; Ilo kirjastus, 2004.a. ISBN 9985-57-475-3
6. http://en.wikipedia.org/wiki/Interoperability (11.04.2006)
7. http://en.wikipedia.org/wiki/Learning_Object (11.04.2006)
8. http://en.wikipedia.org/wiki/Moodle (11.04.2006)
9. http://docs.moodle.org/en/Features (11.04.2006)
10. http://en.wikipedia.org/wiki/OLAT (11.04.2006)
11. http://www.olat.org/public/index/features.html (11.04.2006)
12. http://en.wikipedia.org/wiki/WebCT (11.04.2006)
13. http://www.sun.com/products-n-solutions/edu/whitepapers/pdf/eLearning_Interoperability_Standards_wp.pdf ; Sun Microsystems, Inc "e-learning interoperability standards" (11.04.2006)
14. http://en.wikipedia.org/wiki/Z39.50 (11.04.2006)
15. IEEE; "The Authoritative Dictionary of IEEE Standards Terms 7th Edition ";IEEE Standards Press , 2000.a. ISBN 0-7381-2601-2
16. http://www.comsis.fon.bg.ac.yu/ComSIS/Volume01/InvitedPapers/ErikDuval.pdf ; Erik Duval ; "Learning Technology Standardization:Making Sense of it All" (11.04.2006)
17. http://www.agimo.gov.au/__data/assets/file/12298/interoperability_framework.pdf ; "Interoperability Techincal Framework for the Australian Govenrment" (11.04.2006)
18. www.ichnet.org/glossary.htm (11.04.2006)
19. http://www.gils.net/gilsappb.html (11.04.2006)
20. http://www.estandard.no/docs/meetings/EducaNext-WhitePaper-2003-09-18.pdf ; "EducaNext: A Service for Knowledge Sharing" (11.04.2006)
21. http://www.hi.is/~ebba/publications/summit_5b_universal.pdf ; Sigrun Gunnarsdóttir, Saemundur E. Thorsteinsson, Ebba Thora Hvannberg ; "UNIVERSAL: e-Learning brokerage service" (11.04.2006)
22. http://www.ariadne-eu.org/tikiWiki/tiki-page.php?pageName=developingTools_reqDep (12.04.2006)
23. http://www.elena-project.org/en/index.asp?p=1-1 (15.04.2006)