KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Aistė Jukniūtė

Interaktyvių e – mokymo sistemų pritaikymas paslaugoms orientuotai Grid architektūrai

Magistro darbas

Darbo vadovas
doc. dr. Vytautas Rėklaitis

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Aistė Jukniūtė

Interaktyvių e – mokymo sistemų pritaikymas paslaugoms orientuotai Grid architektūrai

Magistro darbas

Recenzentas
dr. Rimantas Kavaliūnas
2007 05 24

Vadovas
doc. dr. Vytautas Rėklaitis
2007 05 22

Atliko
IFM-1/2 gr. stud.
Aistė Jukniūtė
2007-05-20

Kaunas, 2007

Turinys

Interaktyvių e – mokymo sistemų pritaikymas paslaugoms orientuotai Grid architektūrai

Santrauka

Daugėjant informacijos ir paslaugų kiekiui internete tradicinės elektroninio mokymo architektūros vis sunkiau patenkina iškylančius integracijos, išplėtimo, resursų valdymo poreikius. Nauja paslaugom orientuota architektūra, įgyvendinama žiniatinklio paslaugų technologijomis bei Grid aplinka yra tinkamas būdas efektyviam ir kokybiškam e-mokymui užtikrinti.

Šiame darbe nagrinėjama kokias naujas galimybes e-mokymo srityje atveria paslaugom orientuota Grid architektūra. Tam kad būtų galima pasinaudoti naujomis galimybėmis turimą sistemą reikia modifikuoti. Pateikiamos rekomendacijos egzistuojančios interaktyvios e-mokymo taikomosios programos adaptacijai Grid paslaugų architektūrai. Šios rekomendacijos realizuoja esamą sistemą kaip paslaugą ar paslaugų rinkinį. Sudarytos rekomendacijos praktiškai išbandomos pritaikant realią nuotolinę testavimo sistemą Grid paslaugų aplinkai. Eksperimentinėje dalyje yra įvertinta nauda, kurią suteikia sistemos realizavimas Grid paslaugomis išplečiamumo, lankstumo bei pakartotinio panaudojimo požiūriu.

Interactive e-learning system adaptation for service oriented Grid architecture

Summary

The increasing amount of information and services on the internet continues to grow, traditional e-learning architectures are unable to satisfy the need of integration, flexibility and resource management. The new service oriented architecture based on web services technologies and Grid environment is the effective and valuable way to develop e-learning.

In this master thesis new e-learning opportunities based on service oriented Grid architecture are analyzed. In order to use new approaches the existing system has to be modified. Recommendations for the existing interactive e-learning applications adaptation for the service oriented Grid environment are presented. These recommendations implement the current systems as a service or set of services. The proposed recommendations are tested practically adapting the real distant assessment system for Grid service environment. The added value of this new approach is evaluated taking into consideration scalability, flexibility and reusability of Grid services in the experimentation part.

1. Įvadas

Elektroninio mokymo idėja paplito pasaulyje kaip labai patogus ir priimtinas būdas mokyti/s ir dalintis žiniomis. Interneto atsiradimas leido gimti e-mokymo idėjai, tačiau dabar vis sunkiau patenkinti poreikius, kuriuos kelia e-mokymas. Yra sukurta daug e-mokymo sistemų, kurios veikia skirtingose platformose, naudoja įvairias technologijas. Vartotojai naudojasi kažkokia konkrečia sistema, o tokios sistemos paprastai būna uždaros, autonomiškos. Mokymo resursai ir vartotojai paprastai saugomi šių sistemų viduje. Taigi šiuo metu e-mokymo sistemas riboja tokie klausimai kaip - pakartotinis panaudojimas, integracija, resursų bei dalyvių valdymas, saugumas[1]. Pačios sistemos didėja, tampa vis labiau funkcionalesnės. Tradicinės e-mokymo sistemų architektūros nepajėgios išspręsti iškylančius uždavinius. Šių uždavinių sprendimas atvedė prie POA – paslaugom orientuototų architektūrų (angl. SOA – Service Oriented Architecture) Grid aplinkoje.

Šiuolaikinės informacinės valdymo sistemos, paskirstytų skaičiavimų bei duomenų apdorojimo sistemos vis dažniau įgyvendinamos pagal naują architektūrinį principą kaip paslaugoms orientuotos programinės sistemos. Paslaugoms orientuotoje architektūroje ir sisteminės valdymo funkcijos ir probleminės srities taikymai įgyvendinami naudojant žiniatinklio paslaugų (angl. Web Services) technologiją.

Tam tikros paskirties ar probleminės srities POA sistemą sudaro silpnai tarpusavyje susietų žiniatinklio paslaugų visuma, atskiros žiniatinklio paslaugos ar jų grupės gali būt paskirstyti plataus tinklo mazguose. Kiekviena žiniatinklio paslauga kaip autonominis programinis komponentas skirtas atlikti tam tikrą taikomąją probleminės srities funkciją. Vartotojo užklausas aptarnaujančios paslaugos keičiasi pranešimais, kuriais perduodami duomenys ir tokiu būdu įgyvendinama užklausa, kitaip tariant vartotojas gauna paslaugą.

E-mokymo srityje paslaugoms orientuota architektūra gali palengvinti greitą pritaikomų sistemų vystymą, kuris gali būti optimizuotas pagal specifinį tikslą, užduotį ar pedagoginį reikalavimą. POA leidžia lengvai pridėti papildomus komponentus, ar sujungti jau egzistuojančius nauju norimu būdu, realizuojant greitesnį norimo funkcionalumo įdiegimą taip įgalinant bendravimą tarp skirtingų mokymo sistemų ar institucijų.

Daugelis institucijų bei organizacijų naudoja elektroninį mokymą, tačiau naudojama aparatūrinė ir programinė įranga labai skiriasi, yra nevienoda. Tai sukelia problemas norint laisvai dalintis mokymo/si resursais. Galimas sprendimas šiom problemom yra Grid paslaugos virtualiose organizacijose. Tradiciškai Grid aplinka naudojama paskirstytiems dideliems skaičiavimams, tačiau vystantis Grid technologijoms atsiranda naujos galimybės panaudoti jas kitose srityse, tame tarpe ir elektroniniame mokyme. Pastaraisiais metais Grid

tinklai priėmė į paslaugas orientuotą požiūrį, priimdami bendrus standartus. Žiniatinklio paslaugos leidžia integruoti skirtingas sistemas, suteikia lankstumo, o Grid infrastruktūra užtikrina saugų informacijos dalinimąsi bei dinamišką resursų, vartotojų valdymą taip išsprendžiant e-mokyme iškylančias problemas. Žiūrint iš POA perspektyvos, Grid siūlo išskirtinį modelį paskirstytiems resursams ir informacijai – esminę POA modelio savybę. Žiūrint iš Grid perspektyvos, POA siūlo alternatyvius, bet lankstesnius metodus priimant Grid architektūros sprendimus, tuo pačiu leidžiant juos pritaikyti platesniame skirtingų platformų ir aplinkų rate.

Besivystant internetinėms technologijoms informacijos kiekis Internete vis didėja. Visame pasaulyje, tame tarpe ir Lietuvoje, daugėja Internetu prieinamos informacijos bei paslaugų kiekis. Neabejotinai ši tendencija išliks ir ateityje. Elektroninio mokymo srityje atsiranda galimybė kurti dinamiškas mokymo paslaugas pagal poreikius, apjungiant mokymo objektus, integruojant skirtingas sistemas. Šios paslaugos yra natūraliai besivystančios, atviros pokyčiams, lanksčios bei palengvinančios sistemų sąveiką. Naudojant aktyvius mokymo objektus bei iš jų komponuojant interaktyvias mokymo paslaugas, priešingai nei vien pateikiant statinį turinį, galima pasiekti aukštesnę mokymo/si kokybę ir efektyvumą. Taigi, šiame kontekste iškyla egzistuojančių interaktyvių e-mokymo sistemų ar taikomųjų programų pritaikymo Grid paslaugų architektūrai klausimas.

1.2. Darbo tikslas

Pagrindiniai darbo tikslai:

- Analizuoti ir apibendrinti kokias naujas galimybes e-mokymo taikymams atveria POA bei žiniatinklio ir Grid technologijos.
- Sudaryti rekomendacijas interaktyvių e-mokymo taikomųjų programų bei sistemų pritaikymui bei restruktūrizacijai paslaugoms orientuotai architektūrai.
- Praktiškai panaudoti rekomendacijas egzistuojančios interaktyvios e-mokymo sistemos pritaikymui paslaugom orientuotai Grid architektūrai.

1.3. Reikalavimai darbui

- Analizuoti paslaugoms orientuotų programinių sistemų architektūrą.
- Ištirti Grid aplinkas ir pasirinkti tinkamą platformą realizuojančią POA.
- Sudaryti rekomendacijas interaktyvios e-mokymo sistemos pritaikymui Grid POA pasirinktoje platformoje. Apibendrinti rekomendacijas.
- Pasinaudojant sudarytom rekomendacijom pritaikyti, adaptuoti realią interaktyvią e-mokymo sistemą Grid paslaugų architektūrai.

- Išanalizuoti kokią naują kokybę, lankstumą, išplėtimo galimybes adaptuotai sistemai suteikia POA ir Grid, apibendrinti gautus rezultatus.

1.4. Darbo apimtis ir struktūra

Šį darbą sudaro įvadas ir 6 pagrindiniai skyriai, literatūros sąrašas, terminų ir santrumpų žodynas, publikacijų sąrašas ir priedai.

Pirmame skyriuje – supažindinama su pagrindiniu darbo tikslu bei darbe keliamais reikalavimais, apžvelgiama darbo struktūra.

Antrame skyriuje – apžvelgiamos naudojamos technologijos, galimi sprendimai. Argumentuojamas sprendimo pasirinkimas.

Trečiame skyriuje – formuluojamos rekomendacijos egzistuojančios interaktyvios e-mokymo sistemos pritaikymui paslaugom orientuotai Grid architektūrai.

Ketvirtame skyriuje – magistratūros studijų metu sukurtos programinės įrangos – žinių vertinimo paslaugos techninės-projektinės dokumentacijos esminiai aspektai.

Penktame skyriuje – sukurtos programinės įrangos patobulinimo bei išplėtimo galimybių tyrimas

Šeštame skyriuje – realizuotos žinių vertinimo paslaugos eksperimentinis tyrimas. Eksperimentiškai tiriamos sudarytų rekomendacijų pritaikymas bei sistemos išplėtimo galimybės.

Septintame skyriuje – pateikiamos darbo išvados: projektavimo metu priimti sprendimai, atliktų tyrimų esmė ir autoriaus gauti rezultatai

2. POA ir Grid technologijų analizė, galimų sprendimų apžvalga

Toliau šiame skyriuje yra apžvelgiama POA architektūra, Grid technologijos, šių abiejų požiūrių susiliejimas, bendrų standartų apžvalga. Analizuojama kokią naudą šios technologijos gali teikti e-mokymui.

2.1. POA – paslaugoms orientuota architektūra

Šiuo metu į paslaugas orientuota architektūra pristatoma kaip sekantis žingsnis programinės įrangos architektūros evoliucijoje leidžiantis IT organizacijom išspręsti iškylančius sudėtingus uždavinius. Tradicinės architektūros pateikia statišką turinį, paskirstytus, bet nesusietus resursus, o tuo tarpu POA leidžia dalintis informacija ir resursais bei integruoti skirtingus paskirstytus resursus[4]. Iš esmės POA tai paslaugų rinkinys. Šios paslaugos yra autonominės ir bendrauja tarpusavyje. Paslaugų sąveika yra nepriklausoma, laisvai susieta bei apibrėžiama naudojant aprašymų kalbą. Šis architektūros stilius leidžia pakartotinį panaudojimą makro (paslaugų), o ne mikro (objektų) lygyje, taip pat supaprastina egzistuojančių sistemų integraciją bei sąveiką.

Iš esmės POA neturi kažkokio oficialaus principų rinkinio atspindinčio šią architektūrą, tačiau egzistuoja labiausiai įprasti principai, kurie daugiausiai susiję su orientacijos i paslaugas idėja. Šie principai yra tokie [6]:

- **Paslaugos yra pakartotinai panaudojamos.** Nežiūrint į tai ar konkrečiu atveju iškart galima atlikti pakartotinį panaudojimą, paslaugos yra projektuojamos taip, kad potencialiai pakartotinis panaudojimas būtų įmanomas.
- **Formalios taisyklės**. Tam kad paslaugos bendrautų, jos turi dalintis tik formaliom, standartais pagristom taisyklėm, kurios apibūdina kiekvieną paslaugą ir tai kaip jos keičiasi informacija.
- **Paslaugos yra laisvai susietos.** Paslaugos turi būti projektuojamos taip, kad jų sąveika nereikalautų glaudaus sąryšio ar priklausomybės.
- **Paslaugos paslepia vidinę logiką.** Vienintelė paslaugos dalis matoma išoriniam pasauliui yra aiškiai apibrėžta jos sąsaja. Vidinė logika, kurią paslauga realizuoja, yra nematoma ir neaktuali tiems, kurie kreipiasi į paslaugą.
- **Paslaugos yra komponuojamos**. Vienos paslaugos gali apimti kitas. Tai leidžia sukurti atitinkamus abstrakcijos lygius, pakartotinį panaudojimą, bei išskaidyti logiką į

skirtingus lygius.

- **Paslaugos yra autonomiškos**. Paslaugos įgyvendinama logika turi aiškias ribas. Paslauga visiškai gali kontroliuoti tai kas yra tarp šių ribų i nepriklauso nuo kitų paslaugų tam kad įvykdyti savo teikiamas funkcijas.
- **Paslaugos nesaugo būsenos.** Paslaugos sudarytos taip kad būsena nesaugoma kartu su paslauga.
- **Paslaugos yra surandamos.** Turi būti galimą surasti paslaugą pagal jos aprašymą. Aprašymas turi būti suprantamas žmonėm arba paslaugos klientam, kurie gali norėti pasinaudoti paslaugos teikiama logika.

Esminiais iš šių principų gali būti laikoma autonomiškumas, laisvas susiejimas, abstrakcija ir formalios taisyklės. Šie keturi principai sudaro POA pagrindą ir palaiko kitų principų realizaciją.

2.1.1. POA ir tradicinės architektūros

Tradicinės architektūros leido išspręsti daugelį problemų ir vystėsi nuo didžiųjų kompiuterių, kliento – serverio, interneto architektūros modelių iki dabar pristatomo POA modelio. POA atėjimas reiškia monolitinių architektūrų ir jų palaikymo problemų pabaigą. Tačiau POA nereikalauja, kad būtų visiškai atsisakoma tradicinių architektūrų, POA gali pridėti papildomo lankstumo patenkinant iškylančius IT poreikius. Lentelėje pateikiame tradicinių ir į paslaugas orientuotų architektūrų palyginimą.

1 lentelė lentelė

Tradicinės architektūros	**SOA architektūros**
Projektuojamos visam laikui	Projektuojamos pasikeitimam
Tvirtai susietos	Laisvai susietos, prisitaikančios, lanksčios
Nelanksčios	Sudarytos iš paslaugų
Ilgi kūrimo ciklai	Interaktyvūs ir iteratyvūs kūrimo ciklai
Palaiko homogenines technologijas	Palaiko heterogenines technologijas

Į paslaugas orientuota architektūra neturi nieko bendro su aparatūrinės, programinės įrangos naudojimu ar specialiais protokolais – tai požiūris į sistemų ir veiklos procesų projektavimą ir organizavimą suteikiantis lankstumą ir galimybę valdyti pasikeitimus. POA idėja iš esmės nepriklauso nuo realizacijos, šis architektūros stilius gali būti įgyvendinamas bet kokia tinkama technologija.

Žiniatinklio paslaugos (angl. Web Services), pagrįstos standartais (WSDL, SOAP,

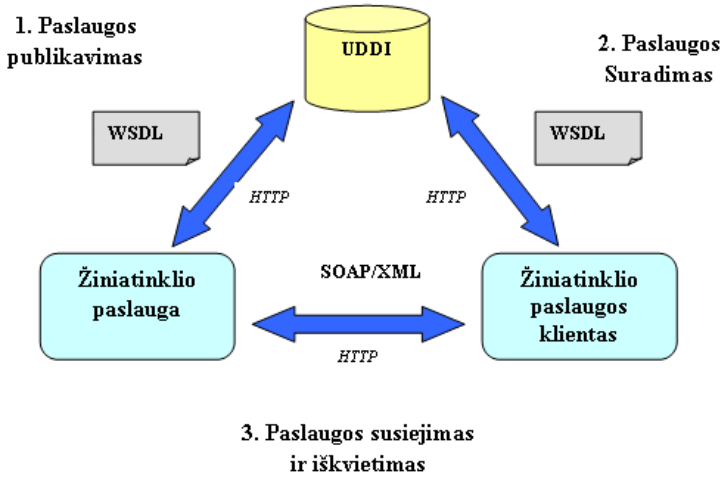
UDDI) tampa labiausiai įprasta POA realizacija [6].

2.2. Žiniatinklio paslaugos

Dažniausiai sutinkama POA realizacija – žiniatinklio paslaugos (angl. Web services). Žiniatinklio paslaugos yra palyginti nauja sparčiai besivystanti paskirstytų skaičiavimų technologija, kuri taikymo prasme pradeda vyrauti lyginant su žinomomis CORBA, Java RMI, EJB technologijomis. Žiniatinklio paslauga suprantama kaip programinis komponentas, kuris aprašomas savo įėjimo taškais EPR (End Point Reference). Per tuos taškus paslauga priima, apdoroja ir išsiunčia pranešimus. Paslaugos paremtos tokiais standartais kaip SOAP, WSDL, UDDI:

- SOAP (Simple Object Access Protocol). SOAP tai XML protokolas, kuris supakuoja XML duomenis į voką persiuntimui žiniatinklio infrastruktūra (pvz. per HTTP).
- WSDL (Web Service DescriptionLanguage). Aprašo žiniatinkio paslaugas XML forma pagal XML schemą, t.y., nusako kaip su ją sąveikauti. WSDL apibūdina žiniatinklio paslaugas kaip rinkinį tinklo galinių taškų turinčių apibrėžtą sąsaja. WSDL dokumente išskiriami trys lygmenys:
  - Duomenų lygmuo – aprašomi duomenys, perduodami pranešimais tarp komunikuojančių paslaugų;
  - Pranešimų formavimo ir apdorojimo lygmuo – tai SOAP formato pranešimai, kurie nukreipiami pagal WSDL aprašo *wsdl:portType* ir *wsdl:binding* apibrėžimus;
  - Pranešimų transportavimo lygmuo, dažniausiai tai HTTP protokolu perduodami duomenys SOAP formato pranešimais.
- UDDI (Universal Descritpion Discovery and Integration). UDDI – tai nepriklausantis nuo platformos, XML pagrįstas žiniatinklio paslaugų registras internete, panašus į baltųjų ar geltonųjų puslapių paslaugas. UDDI palaiko paskelbimo, paieškos ir susiejimo operacijas; paslaugos teikėjas apibrėžia ir publikuoja paslaugos detales registre, tie kurie nori pasinaudoti paslauga užklausia registrą norint surasti paslaugos tiekėją, atliekamas paslaugų susiejimas pasinaudojant UDDI teikiamom techninėm detalėm. UDDI priima SOAP užklausas ir teikia priėjimą prie publikuotų WSDL dokumentų, apibūdinančių reikalingas protokolo sąsajas ir pranešimų formatus norint sąveikauti su paslaugomis užregistruotomis registre.

Pagal WC3 žiniatinklio paslaugos tai programinės įrangos komponentai, sukurti tam, kad leistų kurti kintamo dydžio, laisvai susietas, nepriklausančias nuo platformos programas[]. Žemiau pateiktame paveikslėlyje pateikiamas tipinis žiniatinklio paslaugų iškvietimo scenarijus (1 pav.).



1 pav. *Tipinis žiniatinklio paslaugų iškvietimo scenarijus*

Šios paslaugos turi sąsają, kuri apibrėžiama kompiuteriam suprantamu formatu, tokiu kaip WSDL. UDDI suteikia mechanizmus žiniatinklio paslaugų klientams dinamiškai rasti kitas paslaugas. Kitos sistemos sąveikauja su paslaugomis būdu, kuris aprašytas jų sąsajose naudodamos pranešimus, kurie gali būti perduodami, supakuoti į SOAP voką. Šie pranešimai dažniausiai perteikiami naudojant HTTP protokolą, ir dažniausiai naudoja XML kartu su kitais standartais. XML žiniatinklio paslaugos naudoja XML pranešimus tam, kad gauti ir siųsti duomenis, o tai leidžia programom, parašytom skirtingom kalbom ar veikiančiom skirtingose platformose, bendrauti.

Programos parašytos įvairiose programavimo kalbose, veikiančios įvairiose platformose, gali naudoti žiniatinklio paslaugas, tam kad keistis duomenimis internete, atlikti specifines užduotis naudojant standartinius WWW mechanizmus. Žiniatinklio paslaugos gali būti surastos, iškviestos arba gali būti realizuota kelių paslaugų kompozicija, naudojant gerai apibrėžtą darbo srauto (angl. workflow) modeliavimo karkasą. Žiniatinklio paslaugos žymiai padidino WWW architektūros galimybes, leisdami automatišką programų bendravimą, paslaugų suradimą. Žiniatinklio paslaugos suteikia „on-the-fly“ programinės įrangos komponavimą, naudojant laisvai susietus, daugkartinio panaudojimo komponentus.

Žiniatinklio paslaugų standartai palaikomi nepriklausomų, ne pelno siekiančių standartų organizacijų. WC3 konsorciumas koncentruojasi ties esminėm infrastruktūros specifikacijom[]. OASIS koncentruojasi į aukštesnio lygio funkcionalumą – kūrimą, technologijų suliejimą, e-biznio standartų priėmimą, funkcinį suderinamumą veikimo metu tarp platformų, aplinkų, programų ir programavimo kalbų[].

2.3. Grid tinklas

Grid sąvoka atsirado apie 1990 metus kaip metafora - padaryti kompiuterių resursų panaudojimą tokį pat lengvą, kaip elektros tinklų. Žodžio **Grid** kilmė siejama su elektros tinklų analogija (angl. electric power Grid), kuri realizuoja visuotinį elektros paplitimą ir prieinamumą, nepriklausomai nuo to, kurioje elektrinėje elektra yra gaminama.

Grid – tai paskirstyta sistema, kuri leidžia geografiškai nutolusių, autonomiškų resursų dinamišką dalinimąsi, pasirinkimą ir kaupimą, priklausomai nuo jų prieinamumo, galimybių, veikimo, kaštų ir vartotojo reikalavimų paslaugų kokybei [15].

Grid sistemos idėja yra sena, būtinos šios sistemos savybės:

- Vienodas ir saugus priėjimas prie kompiuterinių resursu, kurie yra geografiškai nutolę ir labai skirtingi.
- Resursai ir naudotojai pastoviai keičiasi.
- Decentralizuotas valdymas.
- Sistema apjungia resursus priklausančius skirtingiems savininkams.

Grid aplinką paprastai kuria programinis sluoksnis įdiegtas kompiuteriniuose, duomenų saugojimo ar kituose resursuose.

Grid tinklų evoliuciją galima suskirstyti į tris skirtingus etapus: pirma karta, antra karta ir trečia karta [19]. Pirmos kartos Grid sistemos gali būti laikomos Grid tinklų pirmtakais. Antros kartos Grid sistemos – tai tarpinė programinė įranga palaikanti didelio mąsto duomenų ir skaičiavimo resursus skirtingose aplinkose. Trečioji Grid karta įgyvendina POA modelį orientuojantis į globalų bendravimą ir virtualias organizacijas. Toliau apžvelgsime kiekvieną kartą detaliau.

Pirma karta: 1980 – 1998

Grid tinklo sąvoka atsirado apie 1980 m., tuo metu kai labai sparčiai vystėsi programinė ir aparatūrinė įranga, algoritmai, skirti lygiagretiesiems skaičiavimams, kompiuteriams. Grid skaičiuojamieji tinklai atsirado kaip paskirstytos skaičiavimo sistemos. Tuo metu buvo sukurta programinė įranga, kuri lygiagretiesiems skaičiavimams leido valdyti bendravimą tarp procesorių ir vykdymo aplinkų tarp skirtingų mašinų. Lygiagrečios virtualios mašinos – PVM (Parallel Virtual Machine), pranešumų perdavimo sąsaja – MPI (Message Passing Intefrace), HPF (High Performance Fortran) ir OpenMPI buvo sukurta bendravimui tarp paskirstytu sistemų, vykdomi projektai skaičiuojamųjų resursų sujungimui. Terminas metaskaičiaviamas (angl. Metacomputing) buvo naudojamas tokių projektų apibūdinimui kai internetu susisiekiantys išteklius – tinklus, procesorius, kaupiklius ir panašiai buvo stengiamasi sujungti į vieną visumą, sprendžiant bendrą uždavinį. Pagrindinis tokiu

metaskaičiavimų projektų tikslas buvo teikti skaičiavimo resursus, daug resursų reikalaujančioms programoms. Svarbiausi du metaskaičiavimų projektai buvo FAFNER ir I-WAY abu prasidėję 1995m. Kiekvienas iš jų, tam tikru/ savu būdu įtakojo keletos pagrindinių Grid technologijų projektų vykdomų šiuo metu evoliuciją.

FAFNER (Factoring via Network-Enabled Recursion) buvo sukurtas RSA130 faktorizavimui naudojant naują metodą NFS Number Field Sieve naudojant skaičiuojamuosius Web serverius. Uždavinys buvo išskaidomas į mažas dalis, paskirstomas ir tokiu būdu net nelabai galingi kompiuteriai, galėjo pasitarnauti sprendžiant šį uždavinį. Šis projektas buvo inicijuotas RSA Data Security Inc. 1991 m., siekiant patikrinti RSA viešo rakto saugumą. Vėliau 1995 Syracuse University ir Co-oparating Systems pradėjo vykdyti FAFNER projektą. Šis projektas buvo sėkmingas ir tapo pirmtakas tokiems projektams kaip Seti@HOME ir Distributed.NET.

I-WAY (Information Wide Area Year) tikslas buvo superkompiuterių sujungimas naudojant jau egzistuojančius tinklus, t.y., nekuriant naujų tinklų. Viena iš I-WAY projekto naujovių buvo skaičiuojamųjų resursų brokerio sukūrimas, panašiai kaip resursų brokerio naudojamo šiuo metu. I-WAY stipriai įtakojo tokius projektus, kaip Globus, kuris šiuo metu beveik tapo pagrindine Grid platforma, taip pat LEGION projektui – alternatyviam požiūriui į paskirstytus skaičiavimus.

Antra karta: 1998 – 2003

Grid tampa ne tik sistema teikianti skaičiavimo resursus, bet infrastruktūra paskirstytoms sistemom, kuri palaiko saugų, patikimą resursų paskirstymą bei dalinimąsi skirtingose, sudarytose iš daugelio organizacijų virtualiose organizacijose. Grid platformos siūlo daugelį priemonių/įrankių paskirstytiems skaičiavimams: panaudojant resursus (saugojimo, komunikavimo, programinės įrangos, ir t.t), CPU, resursų suderinimą, planavimą, darbo paskirstymą tarp skirtingų mašinų.

Duomenų infrastruktūra gali susidėti iš įvairių tipų tinklo resursų – nuo kompiuterių ir saugyklų iki duomenų bazių ir specialių mokslinių instrumentų. Taip pat egzistuoja skaičiavimo resursai, tokie kaip superkompiuteriai ir klasteriai. Tradiciškai labai platus duomenų ir skaičiavimo resursų spektras apibūdina Grid sistemas, tačiau pagrindiniai principai yra tokie:

- Administracinė hierarchija – Kiekviena Grid aplinka kažkokiu būdu susidoroja su potencialiai globalia erdve, nusako kaip administracinė informacija keliauja Gris aplinkoje.
- Bendravimo paslaugos – Grid sistemų bendravimo poreikiai yra įvairūs, kintantys nuo patikimo taškas į tašką bendravimo iki viešo transliavimo. Komunikacijos

infrastruktūra turi palaikyti protokolus, kurie naudojami duomenų perdavimui, komunikacijai ir t.t. Taip pat tinklo paslaugos Grid aplinkoje nusako svarbius paslaugos kokybės - QoS (Quality Of Service) parametrus, tokius kaip patikimumas, pralaidumas, klaidų tolerancija ir pan.

- Informacijos paslaugos - Grid tai dinamiška aplinka, kur prieinamos paslaugos, jų vieta ir tipas pastoviai keičiasi. Pagrindinis tikslas yra leisti bet kokiam sistemos procesui bet kada rasti norimą resursą, nežiūrint į tai kur yra jo vartotojas Grid informacijos (registracijos ir suradimo) paslaugos suteikia mechanizmus informacijos apie aplinkos resursus, struktūrą, paslaugas registravimui ir išgavimui.
- Vardų suteikimo paslaugos (angl. naming) – Grid kaip ir bet kurioje kitoje sistemoje vardai suteikiami įvairiems objektams – kompiuteriams, paslaugoms, duomenims. Suteikiama vienoda vardų sritis visai paskirstytai sistemai. Tipinius vardų paslaugas teikia X.500 vardų schemos arba DNS.
- Paskirstytos failų sistemos – paskirstytos aplikacijos dažnai reikalauja priėjimo prie failų paskirstytų tarp daugelio serverių. Svarbu kad toks failas turėtų unikalią vardų sritį (angl. namespace), palaikytų eilę I/O protokolų, nereikalautų programos pakeitimų, nebent minimalių ir leistų įgyvendinti veikimo optimizavimą.
- Saugumas ir autorizacija – bet kokia paskirstyta sistema apima keturis saugumo aspektus: konfidencialumas, integralumas, autentifikacija ir autorizacija. Saugumas Grid aplinkoje tai sudėtingas uždavinys reikalaujantis įvairius skirtingus resursus autonomiškai administruoti, kad galėtų vykti sąveika, kuri neįtakotų resursų panaudojimo, nesudarytų saugumo skylių tiek individualiose sistemose, tiek bendroje aplinkoje. Saugumo infrastruktūra yra esminė dalis Grid aplinkoje.
- Sistemos būsena ir klaidų tolerancija – svarbu atlikti resursų ir programų stebėjimą tam, kad teikti patikimą ir tvirtą aplinką. Tuo tikslu turi būti įdiegtos atitinkamos programos ir įrankiai, kurie atlieka srtebėjimą.
- Resursų valdymas ir planavimas – procesoriaus laiko, atminties, tinklo ir kitų komponentų valdymas Grid aplinkoje labai svarbus. Bendras tikslas – veiksmingas ir efektyvus programų, kurios naudojasi paskirstytais resursais planavimo ir valdymo sudarymas. Iš vartotojo pusės šis procesas turi būti skaidrus ir apribotas vien programos pateikimu.
- Vartotojo ir administracinė sąsaja – paslaugų ir resursų vartotojo sąsaja turėtų būti intuityvi.

Trečia karta: 2003 - iki dabar

Antra Grid karta suteikė sąveikumą (angl. interoperability), kuris leido įgyvendinti plataus mąsto skaičiavimus. Toliau ieškant vis naujų Grid sprendimų atsirado nauji Grid inžineriniai aspektai. Kuriant naujas Grid sistemas atsirado poreikis pakartotinai panaudoti jau egzistuojačius komponentus ir informacijos resursus ir lanksčiai juos apjungti. Šio uždavinio sprendimui panaudojamas į paslaugas orientuotas modelis ir didėjantis dėmesys metaduomenims. Į paslaugas orientuotas modelis ir metaduomenys tampa svarbiausios trečiosios kartos charakteristikos.

Nors Grid aplinka tradiciškai siejama su plataus mąsto duomenų apdorojimu ir dideliais skaičiavimais, šiuo metu atsirado nauji terminai siejami su Grid sąvoka. I. Foster savo straipsnyje „Anatomy of Grid“ panaudoja tokias sąvokas kaip paskirstytas bendradarbiavimas ir virtuali organizacija [8]. Vėliau šie apibrėžimai buvo praplėsti koncentruojantis į koordinuotą resursų dalinimąsi ir sprendimų priėmimą virtualiose organizacijose sudarytose iš daugelio institucijų

Grid skaičiavimų tinklai išsivystė iki svarbios šakos kompiuteriniame pasaulyje atsiskirdamas nuo paskirstytų skaičiavimų teikdamas vis didesnį dėmesį resursų dalinimuisi, koordinacijai ir valdymui. Resursų pasidalinimas yra laikomas Grid problema ir apima problemas susijusias su resursų pasidalinimų tarp individualių vienetų ar grupių. Šis resursų pasidalinimas varijuoja nuo paprasto failo perdavimo iki sudėtingų reikalaujančių bendradarbiavimo problemos sprendimų ir yra įgyvendinamas naudojant kontroliuojamas, apibrėžtas sąlygas ir politikas. Šiuo požiūriu kritiniai aspektai yra resursų suradimas, autentifikacija ir autorizacija bei priėjimo mechanizmai. Resursų pasidalinimas ir toliau išlieka sudėtinga problema, kai įvairios aplikacijos ir resursai tampa prieinami, pasidalinami, resursai Grid aplinkoje.

2.4. Resursų suradimas

Grid aplinkoje sujungiami įvairus geografiškai nutolę skaičiavimo ir duomenų resursai. Šie resursai yra paskirstomi skirtingoms vartotojų bendruomenėms. Resursai gali priklausyti skirtingoms institucijoms, turėti skirtingas priėmimo teises ir turėti skirtingus reikalavimus priimamoms užklausom. Prieš tai kai bet koks resursas gali būti surastas ir panaudotas darbui, vartotojas turi pasirinkti tokį resursą, kuris atitinka to darbo reikalavimus. Šis procesas, kai parenkami resursai pagal vykdomo darbo reikalavimus vadinamas resursų suradimu (resource matching). Grid aplinkoje, kur resursai kinta dinamiškai, yra norima ir kartais būtina resursų suradimo procesą automatizuoti, kad tiksliai patenkinti specifinio darbo reikalavimus [20].

Su resursų suradimo problema susiję informacijos sistemos – informacijos apie

paslaugas bei resursus suradimui, kaupimui, paviešinimui ir užklausimui, pvz.: Globus MDS, UDDI. MDS yra plačiai naudojama Grid aplinkoje, resursų suradimui, o UDDI naudojama WWW aplinkoje paslaugų suradimui.

Tiek MDS, tiek UDDI palaiko paprastas užklausų kalbas. Tačiau jos nesiūlo nei apibūdinimo paslaugų nei sudėtingų suradimo (atitikimo) galimybių. Tokiose aplinkose resursų tiekėjai paviešina resursų savybes informacijos registracijoje (angl. registry). Tuomet resurso vartotojas kreipiasi į registraciją, kad identifikuoti norimus resursus, prieš atlikdamas tikrą resurso užklausą.

Resurso pasirinkimas, pagrįstas sugrąžinta užklausa, gali būti atliekamas arba vartotojo arba tam tikro algoritmo.

2.5. Virtuali organizacija

Virtuali organizacija (VO) tai dinaminė grupė susidedanti iš individų, grupių ar organizacijų, kurie nustato sąlygas ir taisykles resursų dalinimuisi. Įvairūs organizacijos aspektai gali būti virtualizuoti: vieta, sąsajos ir ribos, procesai, struktūra, produktai/paslaugos.[11]

VO idėja – esminė Grid sąvokai. I. Foster apibūdina VO kaip rinkinį individų ir/ar institucijų apibrėžtų pagal dalinimosi taisykles koordinuojant resursų dalinimąsi ir problemų sprendimą dinaminėse aplinkose sudarytose iš daugelio institucijų ar organizacijų [8]. Dalinimasis suprantamas kaip tiesioginis priėjimas prie įvairių skaičiuojamųjų resursų: procesorių, duomenų, programinės įrangos, aparatūros ir t.t. Tai reikalauja prieinamų resursų virtualizacijos ir pilnos kontrolės juos naudojant ir teikiant. Dalinimosi taisyklės nusako kas yra dalinamasi, kas turi teisę pasinaudoti pateiktais resursais, kada galimas priėjimas ir kiti apribojimai.

Visoms virtualioms organizacijoms svarbios tos pačios charakteristikos bei klausimai ir reikalavimai, kurie gali skirtis dydžiu, apimtimi, trukme, socialiniu aspektu bei struktūra.

Tipinės charakteristikos egzistuojančios virtualioje organizacijoje:

1. Egzistuoja taisyklės ir reikalavimai resursų dalinimuisi.
2. Resursų dalinimasis yra sąlyginis, apribotas laike ir priklausantis nuo sudarytų taisyklių.
3. VO dalyviai dinamiškai kinta.
4. Resursų pasidalinimas priklauso nuo atviro ir aiškiai apibrėžto sąveikų ir priėjimo taisyklių.

Bet kokios VO dalyviai resursais dalinasi priklausomai nuo taisyklių ir sąlygų, kurios nustatytos virtualioje organizacijoje. Tuomet dalyviai resursai dalinasi bendroje VO resursų erdvėje dinamiškai sukurtame loginių ir fizinių resursų rinkinyje.

Vartotojų, resursų ir organizacijų ir skirtingų sričių (angl. domein) priskyrimas į VO esminis techninis klausimas šiuo metu.

Ši užduotis reikalauja nustatyti bei pritaikyti tinkamus resursų suradimo mechanizmus, dalyvių priskyrimo sąlygas ir taisykles, saugumo taisykles - federacija ar delegacija ir priėjimo kontrolę tarp dalyvių.

2.6. Grid paslaugos

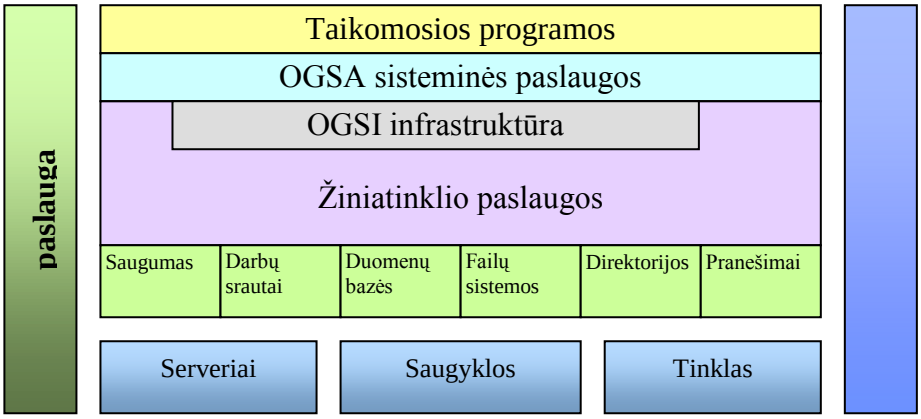
Vystantis Grid technologijoms bei kuriantis vis naujiems Grid tinklams iškilo būtinybė įvesti standartus, kad užtikrinti skirtingų Grid tarpusavio integravimo galimybes. Taip atsirado OGSA (Open Grid Services Architecture) žiniatinklio paslaugų pagrindu sukurta Grid tinklų architektūra, bei infrastruktūra - OGSI (open Grid Service Infrastructure). Galiausiai Grid tinklai visiškai priėmė į paslaugas orientuotą požiūrį 2005m. išleisdami Globus Toolkit versiją realizuojančią į paslaugas orientuotą požiūrį – WSRF (Web Service Resource Framework).

2.7. Grid paslaugų architektūra

Norit pasiekti tikrai paskirstytą resursų dalinimąsi tarp skirtingų ir dinaminių virtualių organizacijų, Grid tinklai šiuo metu koncentruojasi į lengvesnę integraciją, saugumą bei paslaugų kokybės aspektus. OGSA gali būti apibūdinama kaip sprendimas norint įgyvendinti šiuos uždavinius [3]. Egzistuojančių Grid standartų suliejimas su besivystančia POA, o taip pat internetu atvedė prie naujos architektūrinės idėjos. OGSA suteikia vienodą būdą Grid paslaugų apibūdinimui ir apibrėžia vieningą jų veikimo modelį. Ji apibūdina Grid paslaugos elgesio, apibrėžimo mechanizmus ir susijusių protokolų informaciją naudodama žiniatinklio paslaugas kaip tinkamą technologiją. Ši architektūra naudoja geriausias savybes tiek iš Grid, tiek iš žiniatinklio paslaugų technologijų.

2.7.1. OGSA

OGSA tai POA realizuota žiniatinklio paslaugomis Grid aplinkoje. OGSA palaiko virtualios organizacijos resursų ir paslaugų sukūrimą, valdymą ir jų dinaminį komponavimą [7].



2 pav. *OGSA architektūra*

Standartinės sąsajos, kuriuos apibrėžia OGSA nusako yra dinaminis paslaugos sukūrimas, gyvavimo laiko valdymas, nuorodų valdymas, suradimą, notifikacijas ir t.t. Taip pat aibė aukštesnio lygio paslaugų apima: paskirstytų duomenų valdymą, darbo veiksmų seką, darbo apkrovimą, diagnostiką, resursų monitoringą ir t.t.

Pagrindinis OGSA komponentas yra OGSI. Tai Grid programinės įrangos infrastruktūros standartas pagrįstas žiniatinklio paslaugų standartais. Visi resursai, tiek fiziniai, tiek loginiai OGSA modelyje yra laikomi paslaugomis. Šios paslaugos yra projektuojamos naudojant POA ir žiniatinklio paslaugų architektūrą. Praktiškai Grid paslauga – tai žiniatinklio paslauga. Žiniatinklio paslaugos viena iš technologijų, kuri leidžia kurti laisvai susietas paslaugas ir klientus. Todėl tai tampa labiausiai tinkama technologija naujos kartos Grid paslaugų įgyvendinimui. Tokiu būdu Grid paslaugos gali naudotis pranešimų perdavimo (angl. messaging), paslaugų sąsajos apibrėžimo (angl. description) ir suradimo (angl. discovery) modeliais. Pagrindinis tikslas yra pernešamumas pranešimų lygyje, tai pasiekiama naudojant XML formatą pranešimų keitimuisi.

Programinės įrangos architektūra gali būti suskirstyta į tam tikrus abstrakcijos sluoksnius. Vienas sluoksnis priklauso nuo kito priklausomai nuo tarp jų egzistuojančių sąsajų.

OGSA taip pat sluoksninė architektūra, išskaidanti sudėtingas funkcijas ir komponentus, kur aiškiai atskirtas kiekvieno sluoksnio funkcionalumas (žr. 2 pav.). Esminis architektūros sluoksnis yra OGSI ir platformos paslaugos. Platformos paslaugos apima politiką, loginimą, paslaugų lygio valdymą ir kitas tinklo paslaugas. Aukšto lygio programos ir paslaugos naudoja šiuos žemo lygio komponentus.

OGSA žiūrint iš WSRF perspektyvos turi keturis sluoksnius: resursų, žiniatinklio paslaugų, OGSA platformos paslaugų ir aukšto lygio Grid aplikacijų. Trumpai apžvelgsime

kiekvieną sluoksnį [22]:

1. **Fizinių ir loginių resursų sluoksnis**. Resursų idėja yra pagrindinė OGSA architektūroje, o ir bendrai Grid idėjai. Fiziniai resursai apima serverius, atmintį ir tinklą. Virš fizinių resursų yra loginiai. Jei teikia papildomą funkcionalumą virtualizuojant fizinius resursus. Bendra tarpinė programinė įranga
2. **Žiniatinklio paslaugų sluoksnis**. Tai antrasis architektūrinis OGSA sluoksnis. Svarbus OGSA principas yra tai, jog visi Grid resursai – tiek loginiai, tiek fiziniai – modeliuojami kaip paslaugos. OGSI specifikacija apibrėžia Grid paslaugas naudodama standartines žiniatinklio paslaugų technologijas. OGSI panaudoja tokius žiniatinklio paslaugų mechanizmus kaip XML ar WSDL specifikuojant standartines sąsajas, elgesį ir sąveiką visiems Grid resursam. OGSI išplečia žiniatinklio paslaugų apibrėžimą suteikdama galimybe modeliuoti Grid resursus naudojant dinamiškus, išsaugančius būseną ir valdomus žiniatinklio paslaugas
3. **OGSA sisteminės paslaugos**. Žiniatinklio paslaugų sluoksnis, su OGSI išplėtimais suteikia infrastruktūrą sekančiam sluoksniui – sisteminėm Grid paslaugom. Globalus Grid Forumas – GGF(Global Grid Formum) šiuo metu stengiasi apibrėžti šio lygio Grid paslaugas, tokiose srityse kaip programos vykdymas, duomenų paslaugos ir esminės paslaugos. Kai kurie jau yra apibrėžti ir egzistuoja tam tikros jų realizacijos. Kai atsiras vis daugiau šių paslaugų realizacijų, OGSA taps kur kas labiau naudinga į paslaugas orientuota architektūra (POA).
4. **Grid taikomųjų programų lygis**. Taikomosios programos sudaro ketvirtą aukščiausią OGSA lygį. Grid taikomosios programos naudojasi žemesnio lygio Grid paslaugomis.

2.8. Grid paslaugų standartizacija

Grid standartizavimo srityje yra du pagrindiniai standartai – OGSI ir WSRF [13].

2.8.1. OGSI standartas

OGSI yra pagrindinis OGSA architektūros komponentas. Tai Grid programinės įrangos infrastruktūros standartas pagrįstas žiniatinklio paslaugų standartais. Pagrindinis tikslas yra suteikti kuo daugiau sąveikumo tarp OGSA komponentų.

OGSI pristato sąveikos modelį Grid paslaugoms. OGSI suteikia sąsajas suradimui, gyvavimo ciklui, būsenos valdymui sukūrimui ir sunaikinimui, notifikacijom ir nuorodų valdymui pateikiant vieningą būdą modeliuoti ir sąveikauti su Grid paslaugomis.

OGSI pateikiamos sąsajos ir susitarimai:

- **Fabrikas**. Grid paslaugos, realizuojantys fabriko sąsają leidžia dinamiškai kurti naujas Grid paslaugas. Fabrikas gali sukurti laikinus paslaugų vienetus (angl. instances) turinčius ribotą funkcionalumą, pvz.; paslauga, kuri pristato tam tikro darbo vykdymą, arba dažnai naudojamų duomenų rinkinį. Ne visos Grid paslaugos yra kuriamos dinamiškai, kitos gali būti sukuriamos kaip paslauga atstovaujanti fizinius resursus Grid aplinkoje, pvz.; procesorius ar atminties įrenginiai.
- **Gyvavimo ciklas**. Kadangi Grid paslauga gali būti laikina, Grid paslaugų vienetai sukuriami nurodant tam tikrą gyvavimo laiką. Bet kokios paslaugos gyvavimo laikas gali būti pratęstas jei tik yra poreikis arba jis priklauso nuo komponentų, kurie priklauso nuo arba valdo tą paslaugą. Gyvavimo ciklo mechanizmai buvo suprojektuoti tam, kad išvengti begalinio resursų naudojimo apseinant be globalaus šiukšlių surinkimo (angl. garbage collection).
- **Būsenos valdymas**. Grid paslauga gali turėti būseną. OGSI nusako struktūra šios būsenos pateikimui ir mechanizmus jos paieškai, užklausom ir pakeitimui.
- **Servisų grupės**. Tai Grid paslaugų grupės, kurios apjungtos turint tam tikrą tikslą, pvz. Apjungti paslaugas, kurios naudoja resursus tam tikroje klasterio šakoje.
- **Notifikacijos**. Notifikacijos pritaiko tradicinę publikavimo/užsisakymo (angl. publish/subscribe) paradigmą paslaugų būsenos stebėjimui. Kai atsitinka koks nors įvykis, gali būti siunčiama notifikacija. Grid paslaugos palaiko notifikacijų sąsają, leisdamos kitom Grid paslaugom užsiprenumeruoti pasikeitimus.
- **Nuorodos/ identifikatoriai**. Kai fabrikas sukuria naują Grid paslaugą, jis gražina naujai inicializuotos paslaugos tapatybę. Ši tapatybė sudaryta iš dviejų dalių – paslaugos adreso ir unikalaus rakto. Tai užtikrina kad Grid paslaugos unikaliai identifikuojamos .

2.9. Žiniatinklio paslaugų resursų paketas - WSRF

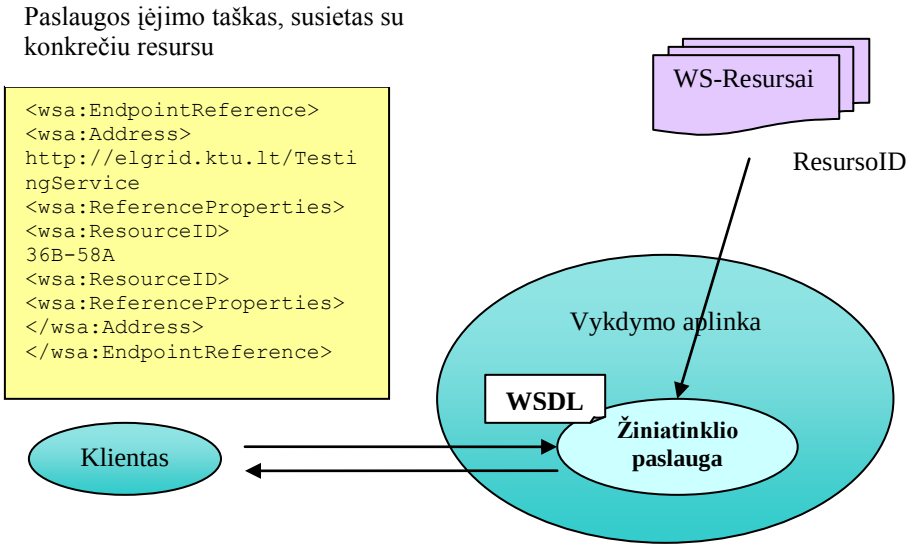
Žiniatinklio paslaugų resursų paketas – WSRF (Web Service Resource Framework) – tai skirtingų specifikacijų rinkinys Grid paslaugom ar kitiems resursam išsaugantiems būseną[5]. WSRF susieja žiniatinklio paslaugų ir Grid standartus. WSRF specifikacija žymiausias indėlis yra Grid ir žiniatinklio paslaugų standartų persidengimas ir jų suderinamumas su POA principais.

Šis suderinamumas – sulyginimas padeda toliau apibrėžti atvirus standartus naudojant funkcionaliai suderinamus veikimo metu pasalugos išplėtimus Grid architektūroje taip padidinant integracijos galimybes.

Kaip jau minėjome, Grid paslauga suprantama kaip žiniatinklio paslauga

atstovaujantis resursą. Šie resursai gali būsi saugantys būseną arba ne (resursas, kurio elgesys nepriklauso nuo būsenos). Paprastai Grid paslaugos atstovauja tam tikrus resursus, kurie turi būseną. O tuo tarpu pati žiniatinklio paslauga nesaugo būsenos (stateless), t.y., neįsimena būsenos tarp užklausų. WSRF siūlo modelį, kurį realizuodama žiniatinkio paslauga, gali išsaugoti būseną. WSRF nustato taisykles būsenos valdymui, tam kad būtų galima patikimai keistis informacija. Kartu su WS-Notification ir kitomis WS-* specifikacijomis Grid resursai tampa prieinami žiniatinklio paslaugų architektūroje.

Ši specifikacija atskiria patčią paslaugą ir jos būseną. Būsenos informacija saugoma visiškai atskiroje esybėje dar kitaip vadinamoje resursu. Kiekvienas resursas turi atskirą identifikatorių, todėl kai tik norime atlikti sąveiką naudodami būseną, tiesiog turime nurodyti tam tikrai žiniatinklio paslaugai naudoti tam tikrą resursą, vykdant operacijas.



3 pav. *WSRF standartą atitinkanti žiniatinklio paslauga*

2.10. Grid platformos

Apie 1980 m. GRID konsepcija suformulavo Ian Foster, Carl Kesselman ir Steve Tuecke, kurie dar vadinami Grid krikštatėviais. Jie suprojektavo Grid infrastruktūros pagrindus, taip vadinama Globus-Toolkit paketą, kuris apima tinklo resursų t.y. CPU ir saugyklų valdymą, darbų paskirstymo sistemą, saugumo sistemą, duomenų migravimą, monitoringą. Laikui bėgant buvo sukurta ir daugiau Grid tinklo valdymo sprendimų (LCG, gLite, ARC, UNICORE, GridBus, Naregi ir kt.), tačiau Globus-Toolkit išliko *standard de facto*.

Kaip e-mokymo aplinkos įgyvendinimo Grid paslaugomis galimybes, apžvelgsime Grid platformas, kurios realizuoja į paslaugas orientuotą modelį. Smulkiau apžvelgsime pagrindine laikoma Grid platformą GT4 bei pateiksime palyginimą su kitom WSRF Grid

realizacijom: pyGridWare, WSRF.Lite ir WSRF.NET.

2.10.1.Globus Toolkit

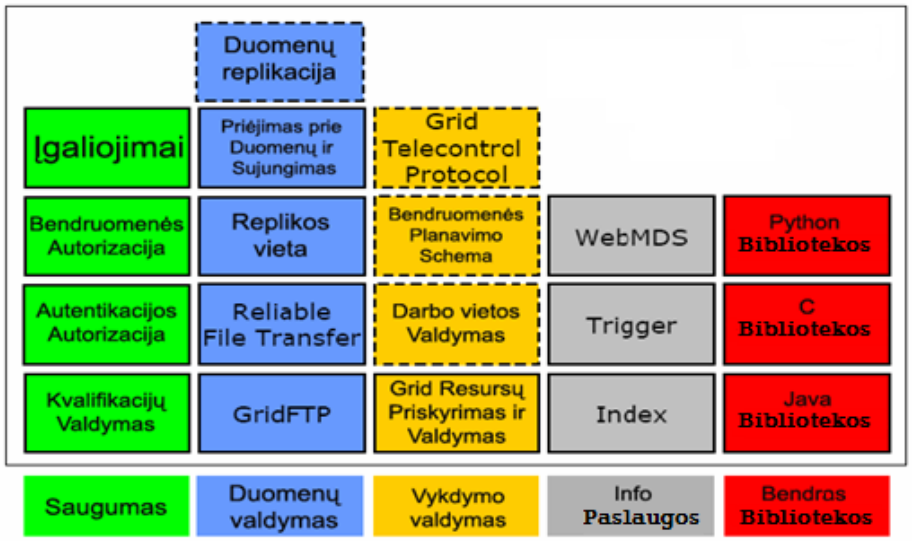
Paskutinė 4 Globus Toolkit (GT4) versija realizuoja WSRF specifikaciją.

GT4 - tai programinė įrangos komponentų ir bibliotekų rinkinys leidžiantis kurti paskirstytas sistemas bei paslaugas Grid aplinkoje [21].

Globus Toolkit teikia įvairius komponentus ir galimybes:

- Realizuotų paslaugų rinkinys infrastruktūros valdymui.
- Įrankiai naujų paslaugų kūrimui Java, C ir Python kalbose.
- Galinga realizuojanti standartus saugumo infrastruktūra
- Programavimo sąsaja - API (skirtingom kalbom) ir komandinės eilutės įrankiai, priėjimui prie skirtingų paslaugų ir galimybių.

Pateikiama GT4 architektūra sudaryta iš atskirtų komponentų (pav. 4). Tai nėra vieninga sistema, atskirus komponentus galima naudoti arba nenaudoti.



4 pav. *GT4 pagrindiniai komponentai*

Pagrindinės komponentų grupės:

- Saugumas
- Duomenų valdymas
- Vykdymo valdymas
- Informacijos paslaugos
- Bendros bibliotekos – programavimo sąsajos ir bibliotekos Java, C ir Python kalbom

GT4 yra kur kas daugiau nei vien tik komponentų, paslaugų rinkinys. Vienodų mechanizmų naudojimas reiškia, kad paslaugos klientas gali sąveikauti su skirtingom paslaugom panašiu būdu, taip įgalinant sudėtingų, didelio sąveikumo sistemų kūrimą ir

pakartotinį panaudojimą. Šis vienodų mechanizmų vienodumas atsiranda keliuose lygiuose:

- WS-I suderinamas SOAP pranešimų keitimąsis tarp žiniatinklio paslaugų ir jų klientų.
- Bendra saugumo ir pranešimų keitimosi infrastruktūra suteikia sąveikumo galimybes tarp skirtingų taikomųjų programų ir paslaugų.
- Galingas ir išplečiamas autorizacijos karkasas, palaikantis daugelį skirtingų autorizacijos mechanizmų.
- Visos GT4 paslaugos realizuoja įprastus mechanizmus paslaugų būsenos nustatymui, priėjimui suradimui ir stebėjimui.

Pateikiame Grid platformų realizuojančių WSRF specifikaciją palyginimą (2 lentelė). Platformos lyginamos skirtingų WS specifikacijų palaikymo atžvilgiu.

2 lentelė WSRF Grid platformų palyginimas

Palaikomos savybės	**GT4-Java**	**GT4-C**	**pyGridWare**	**WSRF::Lite**	**WSRF.NET**
Programavimo kalbos	Java	C	Python	Perl	C++/C#/ VBasic/t.t.
WS-Security	Taip	Ne	Vystoma	Vystoma	Taip
WS-SecurotyX.509 profile	Taip	Vystoma	Taip	Vystoma	Taip
WS-SecureConversation	Taip	Ne	Taip	Ne	Taip
TLS/SSL	Taip	Taip	Taip	Taip	Taip
Autorizacija	Taip	Taip	Taip	Ne	Taip
Išliekantys resursai	Taip	Taip	Taip	Taip	Taip
WS-I Basic Profile	Taip	Taip	Taip	Taip	Taip
WS-I Security Profile	Taip	Taip	Taip	Taip	Taip
Tarnybiniai žurnalai (angl. logging)	Taip log4	Taip log4	Taip log4	Taip log4	Taip log4
WS-ResourceLifetime	Taip	Taip	Taip	Taip	Taip
WS-ResourceProperties	Taip	Taip	Taip	Taip	Taip
WS-ServiceGroup	Taip	Taip	Taip	Taip	Taip
WS-BaseFaults	Taip	Taip	Taip	Taip	Taip
WS – BaseNotification	Taip	Taip	Taip	Ne	Taip
WS-BrokeredNotification	Dalinai	Ne	Dalinai	Ne	Dalinai

WS-Topics	Dalinai	Dalinai	Dalinai	Ne	Dalinai

2.11. Grid ir POA technologijos ir e-mokymas

Dažniausiai Grid tinklo sąvoka yra susijusi su dideliais paskirstytais skaičiavimais, resursų paskirstymu, tačiau tai tik viena sritis, kur Grid aplinka gali būti efektyviai panaudota.

Daugelis mokslo institucijų plačiai naudoja e-mokymą, tačiau skirtingos institucijos naudoja skirtingas aplinkas, skirtingą programinę ir aparatūrinę įrangą. Todėl mokymo medžiagos pakartotinis panaudojimas, skirtingų sistemų bendravimas ir integracija tampa nemaža problema [12]. Sprendžiant šias problemas apjungiant skirtingų institucijų ar organizacijų resursus į virtualias organizacijas, esminiais tampa resursų pasidalinimo, teisių suteikimo ir kiti saugumo klausimai. Grid infrastruktūra gali būti priimtinas sprendimas šiems klausimam įgyvendinti [17].

Grid aplinka yra tinkamas būdas realizuoti efektyvų e-mokymą. Paskutinės kartos Grid technologijos perėmė į paslaugos orientuotą požiūrį. Taigi, iš esmės mes galime paskirstytus skaičiavimo resursus laikyti paslaugomis, kurias teikia įvairios skirtingos organizacijos, tam kad pagerinti mokymosi aplinką.

Naudojant paslaugom orientuotą architektūrą e-mokymo sistema gali būti sudaroma iš atskirų e-mokymo paslaugų su aiškiai apibrėžtomis sąsajomis. Tai sistemai gali suteikti lankstumo bei dinamiškumo. Paslaugas galima panaudoti ar apjungti pageidaujamu būdu, patenkinant tam tikrą reikalavimą ar užduotį [9].

Į paslaugas orientuotų Grid technologijų panaudojimas suteikia tai, ko reikalauja efektyvus elektroninis mokymas – dinamiškas resursų ir vartotojų valdymas, resursų dalinimasis, integracija, suradimas bei saugumo užtikrinimas tarp skirtingų, sudarytų iš daugelio institucijų, virtualių organizacijų (VO).

Grid architektūra yra atvira, o tai pagrindinė prielaida naujų mokymo/si paslaugų kūrimui, jų rato plėtimui ir kokybės gerinimui [18].

2.11.1 Pasirinktas sprendimas

Savaime suprantama, kad visos e-mokymo sistemos negali būti iš naujo perkuriamos ir Grid paslaugų architektūrai. Todėl darbe iškeltas uždavinys pritaikyti jau egzistuojančią interaktyvią e-mokymo sistemą Grid POA aplinkai. Šio uždavinio sprendimui pasirinkta Globus Toolkit Grid platformą, kaip standartinę platformą realizuojančią WSRF specifikaciją. Palyginus kelias žinomiausias WSRF Grid platformų realizacijas tampa aišku, jog Globus Toolkit4 pilnai teikia paslaugų saugumo infrastruktūrą bei palaiko daugiausiai WS specifikacijų. Toliau darbe sprendžiamas uždavinys ir eksperimentiniai tyrimai bus atliekami

naudojant žiniatinklio technologijas ir Globus programinė įrangą, kurianti Grid aplinką.

3. Rekomendacijos interaktyvios e-mokymo sistemos realizacijai Grid paslaugomis

Iki šiol darbe apžvelgtos išanalizuota paskirstyta paslaugų architektūra Grid aplinkoje kaip sprendimas iškylančiom e-mokymo problemom. Šiame skyriuje koncentruojamasi ties autoriaus siūlomais žingsniais, rekomendacijomis egzistuojančios interaktyvios e-mokymo sistemos restruktūrizacijai bei pritaikymui WSRF Grid aplinkai.

Šiuo atveju kalbama apie interaktyvias e-mokymo sistemas ar taikomąsias programas, o ne dideles mokymosi aplinkas kaip WebCT ar Blackboard. Didelės mokymo/si aplinkos paprastai vieningos, išbaigtos sistemos apjungiančios tiek vartotojus, tiek mokymosi proceso ir medžiagos valdymą bei jo pateikimą. Tačiau dažniausiai jose pateikiama statinė mokymosi medžiaga ar kursai. Tokias dideles sistemas pritaikyti paprastai nėra prasmės ir poreikio. Tuo tarpu interaktyvios taikomosios programos gali būti realizuotos kaip paslaugos ir panaudotos apjungiant jas su kitomis paslaugomis bendroje paslaugų architektūroje virtualioje organizacijoje.

Grid aplinka ir į paslaugas orientuota architektūra yra tinkamas būdas elektroninio mokymo paslaugom realizuoti. Jau egzistuojančių mokymo sistemų pritaikymas Grid aplinkai – uždavinys, apimantis sistemos modifikavimą (perprojektavimą bei realizavimą) kaip mokymo paslaugą bendroje paslaugų architektūroje.

Šis uždavinys iškilo norint nuotolinę interaktyvią grafinio testavimo sistemą TestTool, plačiai naudojamą Kauno Technologijos Universitete (KTU), realizuoti kaip Mokslo ir Studijų virtualios organizacijos (Mo&St VO) testavimo paslaugą.

Šiuo metu KTU Kompiuterių tinklų katedroje vykdomas Europos struktūrinių fondų paramos projektas VIRTUALIOSIOS MOKSLO IR STUDIJŲ ORGANIZACIJOS KŪRIMAS, VYSTYMAS IR DIEGIMAS GRID TECHNOLOGIJŲ PAGRINDU. Iki šiol Lietuvoje nėra e-paslaugų sistemos žmogiškųjų išteklių kokybės gerinimui mokslinių tyrimų, mokslo ir technologijų plėtros srityse. Todėl šis projektas yra skirtas mokslo ir studijų paslaugų sistemos sudarymui Grid technologijų pagrindu. TestTool sistema buvo restruktūrizuota ir pritaikyta Grid paslaugų architektūrai. Remiantis atliktu pritaikymu sudarytos rekomendacijos interaktyvių e-mokymo sistemų pritaikymui Grid POA aplinkai.

3.1. Sistemos pritaikymo žingsniai

Sistemos pritaikymas daugeliu atvejų negali įvykti neatlikus jos modifikacijos. Norint sistemą adaptuoti Grid aplinkai, integruojant ją į paslaugų architektūrą, turi būti realizuotas tam tikras tarpinis sluoksnis apgaubiantis turimą sistemą. WSRF Grid modelyje tas

apgaubiantis sluoksnis gali būti suprantamas kaip Grid paslauga. Kadangi Grid paslauga kuriama egzistuojančios sistemos pagrindu yra galimybė šį procesą dalinai automatizuoti, taip palengvinant paslaugos kūrimą ir pasinaudojant egzistuojančiu sistemos išeities kodu generuoti tam tikras paslaugos dalis.

Išanalizavus Grid paslaugų architektūrą bei WSRF realizacijas, galime pateikti rekomendacijas ir metodiką egzistuojančios interaktyvios e-mokymo sistemos arba taikomosios programos pritaikymui Grid aplinkai. Tam kad sistema taptų prieinama virtualioje organizacijoje kaip Grid paslauga, reikia kad ji atitiktų tam tikrus reikalavimus. Visų pirmą sistema, realizuojama kaip Grid paslauga ar paslaugų rinkinys turi atitikti OGSI specifikaciją, o taip pat priimti WSRF modelį – paslaugų, kurios saugo būseną kūrimui. Šio proceso metu išskiriamos tokios alternatyvos uždavinio sprendimui:

- Sistemos paleidimas Grid aplinkoje.
- Apgaubiančio sluoksnio realizacija – sistemos pritaikymas panaudojant esamus jos komponentus.
- Pilnas sistemos perprojektavimas ir realizacija

Pirmasis atvejis taikomas kai tiesiog stengiamasi sistemą paleisti Grid aplinkoje be jokių modifikacijų.

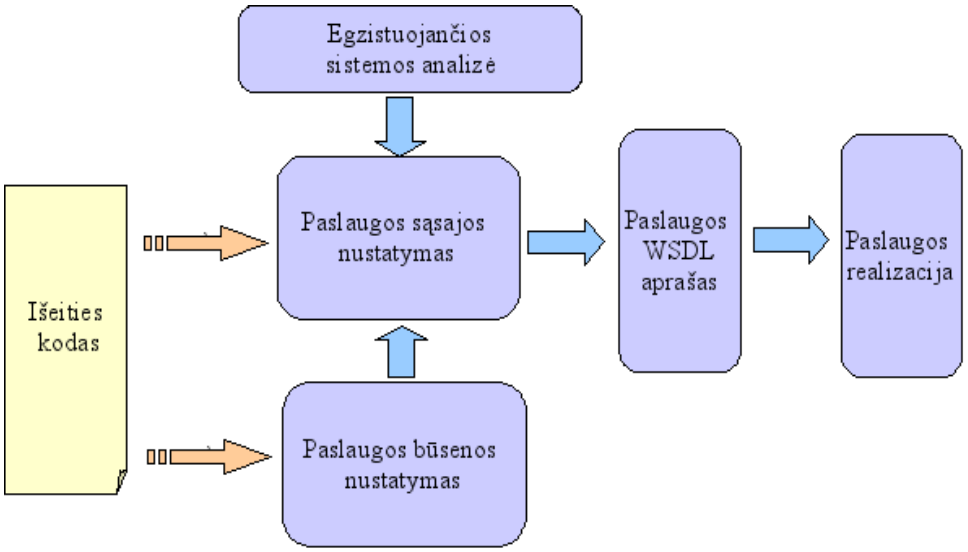
Antrasis būdas taikomas kai esamos sistemos išeities kodas yra prieinamas. Tokiu atveju sistema adaptuojama Grid aplinkai atliekant tam tikras modifikacijas sistemos struktūroj, funkcionalume tačiau neatliekami dideli pakeitimai priklausantys nuo Grid tarpinės programinės įrangos. Ta pati sistema gali veikti tiek Grid tiek ne-Grid aplinkoje.

Trečiasis būdas taikomas, kai sistema kuriama iš naujo. Šis būdas gali būti taikomas kai sistema yra pakankamai sudėtinga ir jos negalima taip lengvai pritaikyti Grid aplinkai. Tokios sistemos nuo pradžios kuriamos Grid aplinkai, išnaudojant visas jos teikiamas galimybes.

Šiame skyriuje panagrinėsime antrąjį atvejį, kai sistema adaptuojama Grid aplinkai. Šiuo atveju nagrinėjama Grid platforma realizuojanti WSRF specifikaciją. Sistemos adaptavimas – tai sistemos kaip Grid paslaugos realizavimas. Sistema nėra perrašoma iš naujo, bet pasinaudojama egzistuojančiu kodu ir veikimo logika, realizuojant Grid paslaugą, kuri integruoja sistemą į bendrą virtualios organizacijos paslaugų architektūrą. Nėra atsižvelgiama į paslaugos tokius aspektus kaip didelių skaičiavimų išlygiagretinimas ar didelių duomenų kiekių apdorojimas. Tam tikri sistemos adaptavimo proceso žingsniai priklauso nuo sistemos realizacijos technologijos. Šiuo atveju pateiksime metodą kaip egzistuojančią sistemą adaptuoti Grid aplinkai. Pritaikymo procesą galime suskaidyti į tokius pagrindinius žingsnius:

- Egzistuojančios sistemos analizė
- Paslaugos sąsajos nustatymas
- Paslaugos resurso savybių nustatymas
- WSDL aprašo sudarymas
- Paslaugos realizavimas
    - Įgyvendinimo klasių realizavimas
    - Konfigūracijos nustatymas
    - Paslaugos įdiegimas
- Paslaugos vartotojo sąsajos programų realizavimas arba pritaikymas

Grafinis proceso pavaizdavimas (žr. pav. 5):



5 pav. *Egzistuojančios sistemos pritaikymo Grid paslaugų architektūrai procesas*

Kadangi Globus platforma yra tapus „de facto“ Grid standartu pasaulyje, čia pateiksime rekomendacijas skirtas konkrečiai Grid aplinkai – GT4 bei paslaugos realizacijai Java kalba. Skyriaus pabaigoje pateiksime apibendrinimą, nepriklausantį nuo Grid platformos ir rekomendacijas automatizavimo įrankio kūrimui.

Norint šį pritaikymo procesą automatizuoti, gana nesunku butų sukurti įrankį, kuris iš egzistuojančios sistemos kodo generuoja WSDL dokumentą, paslaugos kodą, bei pasirinktų kintamųjų suformuoja paslaugos būseną. Tokio įrankio kūrimo galimybės aptariamos skyrelio pabaigoje, tačiau pats įrankis nekuriamas nes pritaikant tik vieną sistemą nėra prasmės tokį įrankį kurti.

3.1.1. Sistemos analizė

Ruošiantis realizuoti sistemą paslaugomis, prieš atliekant bet kokias modifikacijas

būtina atlikti sistemos analizę [3]. Reikia išsiaiškinti kokia dabartinės sistemos architektūra, kokių pakeitimų reikės norit sistemą pritaikyti paslaugom orientuotai Grid architektūrai. Labai svarbu nuspręsti kaip egzistuojanti sistema bus paskirstyta paslaugų architektūroje. Ar jos funkcionalumą realizuos viena paslauga ar keletas, ar keisis naudojamos duomenų struktūros bei duomenų saugojimas. Kadangi sistemos pritaikymas paprastai atliekamas siekiant didesnio lankstumo, dinamiškumo, padidėjusių išplėtimo galimybių, tai sistemą paprastai geriau skaidyti į atskiras paslaugas, kurios atlieka tam tikrą apibrėžtą funkcionalumą ir nepriklauso nuo kitų paslaugų. Bendru atveju, bet kokiai sistemai geriausiai atsisakyti vidinio vartotojų saugojimo iškeliant juos į virtualios organizacijos lygmenį. Reikia spęsti ką palikti sistemoje kaip jos vidinius komponentus, o ką iškelti į išorę kaip kitas paslaugas.

3.1.2. Paslaugos sąsajos nustatymas

Paslaugos sąsaja – tai operacijų, kurias teikia paslauga rinkinys. Paslaugos sąsaja apibrėžiama WSDL dokumente. WSDL dokumentą galima tiesiogiai rašyti ranka arba generuoti iš kokia nors programavimo kalba aprašytos paslaugos sąsajos. Daug patogiau WSDL dokumentą generuoti, nes tokiu atveju reikia mažiau rašyti ir mažesnė tikimybė padaryti sintaksės klaidų. Čia pateikiame paslaugos sąsajos klasės sudarymą kaip pirmą žingsnį po kurio seka WSDL generavimas.

Norint esamą programą realizuoti kaip paslaugą, reikia nustatyti funkcionalumą, kurį teiks naujai sukurta paslauga. Norint tai atlikti reikia išanalizuoti funkcionalumą, kurį teikia esama sistema, t.y., kokios funkcijos prieinamos sistemos klientams. Būtina atskirti funkcionalumą, kurį realizuos pati paslauga, o kuris bus perkeltas į paslaugos klientus. Tai visai nesunku padaryti, kai esama sistema realizuota kliento-serverio principu. Tuomet paslauga realizuoja sistemos serverio dalį.

Iš esmės paslaugos sąsają galima nustatyti dviem būdais:

- rankiniu, tiesiog parašant sąsajos klasę
- sudarant sąsajos klasę pagal egzistuojančios sistemos kodą

Antruoju atveju, pasinaudojant esamos sistemos Java kodu, galima iš egzistuojančio kodo pasirinkti metodus, kurie realizuos paslaugos operacijas. Egzistuojančios sistemos išeities kode pasirenkamos tos klasės, kurios atsakingos už sąsaja su klientu, vartotojo sąsaja, t.y., tos, kurios reprezentuoja sistemos teikiamas funkcijas. Išskyrus šias klases, galima pasirinkti norimus metodus ir atlikti sąsajos sudarymą. Savaime suprantama, kad paslaugos operacijos yra tik nedidelis programos metodų poaibis. Šis žingsnis – metodų bei kintamųjų išskyrimas iš egzistuojančio kodo bei sąsajos sudarymas gali būti automatizuotas.

Pasirinkus operacijas, kurias realizuos paslauga atliekamas operacijos išskaidymas. Operacijos/metodai išskaidomi, nes to reikalauja pasirinkta realizacijos platforma Globus.

Globus nepilnai palaiko sudėtingus duomenų tipus paslaugos realizacijoje. Kadangi Globus remiasi Apache Axis realizacija, todėl nuo to priklauso kokius duomenų tipus galima persiųsti tinklu. Standartiniai atitikmenys (perkėlimai) tarp WSDL ir Java yra tokie[]:

3 lentelė lentelė WSDL ir Java duomenų tipų susiejimas

WSDL	**Java**
xsd:base64Binary	byte[]
xsd:boolean	boolean
xsd:byte	byte
xsd:dateTime	java.util.Calendar
xsd:decimal	java.math.BigDecimal
xsd:double	double
xsd:float	float
xsd:hexBinary	byte[]
xsd:int	int
xsd:integer	java.math.BigInteger
xsd:long	long
xsd:QName	javax.xml.namespace.QName
xsd:short	short
xsd:string	java.lang.String

Jei WSDL dokumente nurodyta, jog elementas gali įgauti NULL reikšmę, t.y., galima grąžinti NULL reikšmę, tuomet primityvūs duomenų tipai pakeičiami apgaubiančiomis klasėmis, tokiomis kaip Byte, Double, Float ir t.t.

Taip pat Axis gali persiųsti keletą kolekcijų - Collection klasių su apribotu sąveikumu. Tarkim tokios klasės kaip Hashtable turi serializatorius, tačiau nėra standartizuoto sąveikumo su kitomis SOAP realizacijomis, taip pat SOAP specifikacija neapima sudėtingų objektų. Norint siųsti sudėtingus, agreguotus objektus, geriausia naudoti masyvus. Nors .NET realizacija nepalaiko masyvų, daugelis kitų SOAP realizacijų gali juos suprasti.

Naudojant Axis neįmanoma siųsti įprastų Java objektų ir tikėtis, kad juos supras priėmėjas. Kadangi neįmanoma užtikrinti, kad tiek siuntėjo, tiek gavėjo pusėje naudojama Java. Todėl galima siųsti tik tuos objektus, kuriems yra užregistruotas Axis serializatorius.

Taigi GT4 aplinkoje galima siųsti Java klases, kurios atitinka JavaBeans modelį. Tereikia nurodyti, kokios klasės atitinka šį modelį. Tačiau pasitaiko atvejų, kai šio modelio neužtenka, kai norima Java klases, neatitinkančias JavaBeans modelio perteikti XML arba kai norima pasirinktą XML schemą tam tikru būdu perteikti Java kalba. Tokiu atveju reikia pasirašyti savo serializatorių/deserializatorių tiems objektams.

Kadangi egzistuoja tokie apribojimai siunčiamiems duomenų tipams, atliekamas

metodų išskaidymas pagal tam tikras taisykles.

Jei iš egzistuojančio kodo pasirinktuose metoduose naudojami sudėtingi duomenų tipai reikia atlikti pakeitimus. Jei sudėtingo tipo klasė atitinka Java Beans modelį, tokiu atveju galimas vienas iš dviejų pakeitimų:

- norimiems sudėtingo tipo klasės kintamiesiems sugeneruoti get/set metodus paslaugos klasėje, jei reikia įvedant identifikatorių, arba;
- naudoti JavaBeans klasę sudėtingiems objektų tipams, tačiau papildyti paslaugą metodu setBean(), kuris nustato Bean klasės reikšmes paslaugoje.

Jei klasė neatitinka JavaBeans modelio reikia:

- rašyti savo serializacijos mechanizmą tam duomenų tipui;
- arba naudoti tik tam tikrus sudėtingo tipo klasės laukus ir juos įtraukti į paslaugą naudojantis set/get metodais.

Kai kuriais atvejais paslauga gali realizuoti ne visas esamos sistemos f-jas, o tik tam tikrą jų dalį. Tai priklauso nuo to, kaip įsivaizduojamos paslaugos teikiamos operacijos. Gali būti, kad perprojektuojant sistemą Grid aplinkai gali tekti atsisakyti kokių nors operacijų (dažnai pasitaikantis atvejis, vartotojų iškėlimas už sistemos ribų), jos gali tapti nereikalingos, arba atsirasti poreikis kažkokiom specifinėm operacijom. Todėl pagal egzistuojančia sistemą sudarytą paslaugos sąsają reikia pakoreguoti, jei reikia pridėti naujų funkcijų.

3.1.3. Paslaugos resurso savybių nustatymas

GT4 platforma įgyvendina WSRF modelį, tai reiškia, kad paslauga gali saugoti būsena. Būsena saugoma resurso savybių pavidalu. Paslauga gali turėti vieną ar daugiau resurso savybių. Taigi, šiame žingsnyje reikia nuspręsti kokios resurso savybės atstovaus paslaugos būseną. Resurso savybės yra apibrėžiamos XML schema ir realizuojamos XML dokumentu vadinamu resursų savybių dokumentu. Resursų savybių dokumentas – standartizuotas būdas išgauti ir modifikuoti paslaugos būseną per paslaugos sąsają. Resurso savybės apibrėžiamos WSDL dokumente ir susiejamos su paslauga per portType atributą. Paslaugos klientai gali modifikuoti paslaugos būseną naudodami standartinius paslaugų pranešimus. Standartiniai pranešimai įtraukiami į WSDL dokumentą kaip paslaugos operacijos. Galiausiai nustatoma, kokias operacijas bus galima atlikti su resurso savybėm (sukurti, išgauti, modifikuoti, trinti ir pan.)

Resurso savybės saugo paslaugos būseną. E-mokymo atveju labai patogu mokymo paslaugos metaduomenis saugoti paslaugos būsenoje kaip resurso savybes. Visų pirmą reikia nustatyti ar egzistuojanti sistema saugo kokią nors būseną, kurią galėtu reprezentuoti kuriama paslauga. Būsena gali nusakyti kokia nors statistinę informaciją, tai paslaugai svarbią

informacija arba informaciją pagal kurią galima rasti norimą paslaugą GT4 paslaugų indekse. Kitaip tariant paslaugos būsenoje galėtų būti saugomi duomenys, kurie kažką pasako apie pasaugą arba tokia informacija kaip paslaugos metaduomenys. Būsena bendru atveju būsena yra sudaryta iš resursų savybių rinkinio. Resurso savybę gali atstovauti bet koks kintamasis, tačiau galioja WSDL ir Java duomenų tipų apribojimai (žr. 3 lentelė).

Resurso savybių nustatymo procesas gali būti skaidomas į tokius žingsnius:

- Resurso savybių nustatymas iš egzistuojančios sistemos kodo
- Resurso savybių papildymas (jei reikia)
- Resurso savybių modifikatorių nustatymas

Taigi, turint egzistuojančia sistemą reikia nustatyti, kokie kintamieji atstovauja jos būseną. Kaip ir paslaugos sąsajos nustatymo atveju, šie kintamieji gali būti išgauti iš esamos sistemos išeities kodo. Galimas atvejis kad egzistuojančių kintamųjų neužtenka norint apibrėžti paslaugos būsena, tokiu atvejų reikia išanalizuoti poreikius paslaugos būsenai ir įvesti naujas resurso savybes.

Resurso modifikatoriai nusako kokius veiksmus galima atlikti su resurso savybėmis. Pasirinkimai yra šie:

- Išgauti resurso savybę - leidžia išgauti bet kokios resurso savybės reikšmę.
- Išgauti daugelį resurso savybių – leidžia išgauti keleto resursų savybių reikšmių vienu metu.
- Nustatyti resurso savybes – leidžia atlikti vieną ar keletą resurso savybių modifikacijų.
    - Atnaujinti – resurso savybės reikšmės atnaujinimas, pakeitimas.
    - Pridėti – naujos resurso savybės su nurodyta reikšme pridėjimas.
    - Ištrinti – pasirinktos resurso savybės ištrynimas.
- Naudoti resursų savybių užklausas – leidžia atlikti sudėtingas resursų savybių dokumento užklausas.

Priklausomai nuo paslaugos, o taip pat ir egzistuojančios sistemos logikos resursų savybių modifikatoriai apriboja veiksmus, kuriuos galima atlikti su resurso būsena. Tarkime jei būsenoje saugoma informacija, kuri nekinta, tokiu atveju reiktų leisti tik išgauti resurso savybes ir neleisti jų modifikuoti, kitais atvejais.

Nustačius paslaugos sąsają ir resurso savybes bei jų modifikatorius galima atlikti paslaugos WSDL aprašo generavimą.

3.1.4. WSDL generavimas

Kai jau turime paslaugos sąsajos klasę, galime atlikti paslaugos aprašymo, WSDL dokumento sudarymą. Iš esmės galima WSDL dokumentą rašyti ranka tiesiogiai, nesudarius paslaugos sąsajos. Tačiau tai yra nelabai patogu, jei paslauga teikia daugelį operacijų. Jei paslaugos sąsaja jau egzistuoja, galima pasinaudoti Apache Axis įrankiu java2wsdl, kuris pagal pateikta sąsają generuoja WSDL dokumentą. Sugeneruotas WSDL dokumentas turi atitinkamus tipus, pranešimus, portType sąsają. Jei sąsajoje naudojamos kitos klasės kaip duomenų tipai, Java2WSDL įrankis sugeneruoja atitinkamus XML tipus, kurie atstovauja tas klases ir paveldėtus tipus. Tačiau reikia turėti galvoje, jog šis įrankis palaiko tik Bean klases, kolekcijas, masyvus ir Holder klases.

Sugeneravus tokį WSDL dokumentą reikia jį šiek tiek modifikuoti, kad jis būtų tinkamas Globus platformai:

- pridėti resurso savybių aprašą
- pridėti atitinkamus resurso savybių modifikatorius.

3.1.5. Paslaugos realizavimas

Pateiktos rekomendacijos supaprastina egzistuojančios sistemos adaptavimą Grid aplinkai. Pasinaudojant pateiktomis rekomendacijomis, atsiranda galimybė sugeneruoti dalį Grid paslaugos kodo.

Paslaugos įgyvendinimo klasės gali įgyvendinti vieną iš dviejų modelių WSRF realizacijos atveju. Taigi tam, kad sugeneruoti paslaugos įgyvendinimo klasių karkasą, reikia pasirinkti modelį, kurį realizuos kuriama paslauga. Tai priklauso nuo egzistuojančios sistemos logikos bei poreikių. Trumpai apžvelgsime kiekvieną iš minėtų būdų ir pateiksime rekomendacijas kada reikėtų naudoti vieną ar kitą būdą.

Kaip jau minėta pagal WSRF specifikaciją (žr. 2.7 skyrelį), paslauga ir jos būsena yra atskirta. Paslauga gali turėti būsena arba ne. Jei paslauga turi būsena, tai ta būsena saugoma atskiroje esybėje – resurse. Taigi iš šio požiūrio išplaukia du paslaugos, turinčios būseną arba kitaip tariant resursą, įgyvendinimo – programavimo būdai.

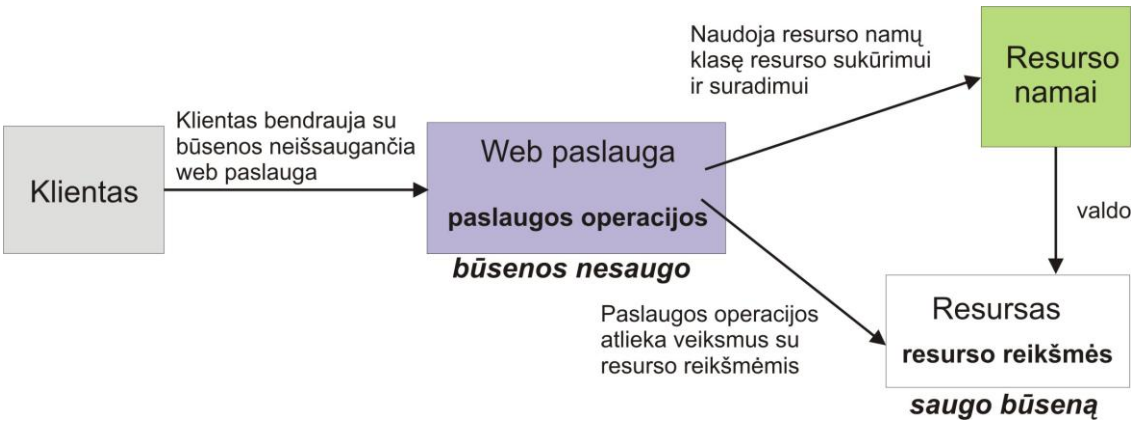
3.1.5.1. Paprasta, vieno resurso paslauga

Šiuo atveju resursas gali būti tik vienas. Paslauga ir resursas programavimo požiūrių gali būti įgyvendinami toje pačioje klasėje, tačiau tai nerekomenduotina, geriau šias klases atskirti. Toks modelis tinka tuomet, kai paslauga turi būsena, kuri nepriklauso nuo skirtingu vartotoju, procesų ir t.t. Paslaugos būsena gali kisti, tačiau ji yra tik viena.

6 paveikslėlyje pateikiama šio būdo realizacija GT4 platformoje.

Paslauga sudaryta iš tokių komponentų:

- Paslauga – realizuojanti sąsaja, su kuria bendrauja klientas.
- Resursas – komponentas atsakingas už paslaugos būsenos saugojimą.
- Resurso namai – skirta valdyti resursus.



6 pav. *Vieno resurso paslauga*

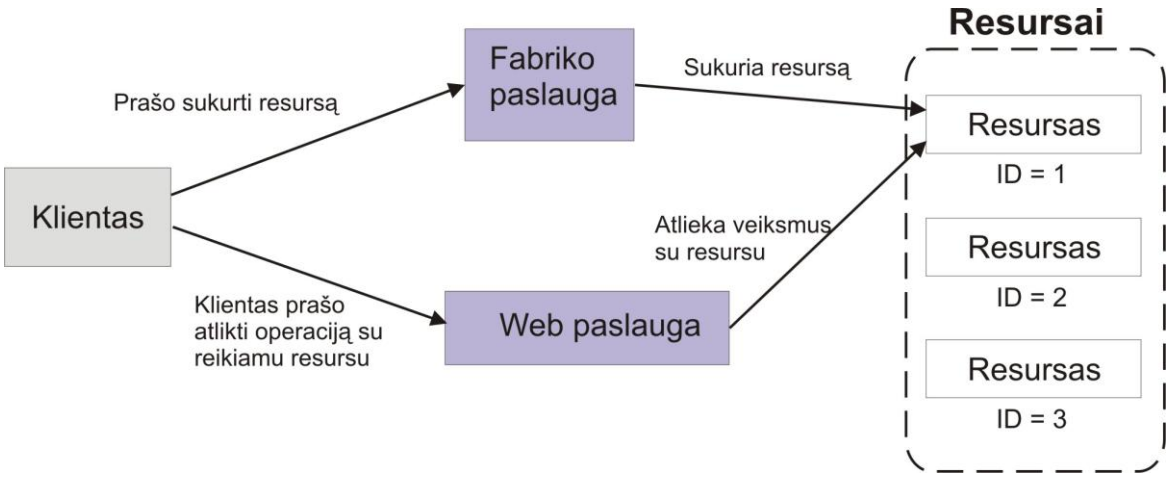
3.1.5.2. Daugelio resursų paslauga – fabriko kūrimo modelis

Kuriant daugelio resursų paslaugą naudojamas projektavimo būdas vadinamas fabrikas/atskiras atvejis (angl. factory/instance) projektavimu.

Naudojant šį programavimo modelį neleidžiama objektus kurti tiesiogiai, bet reikia naudoti *fabriką.* Turint daugelį resursų ir naudojant šį modelį, turime dvi paslaugas: viena – atsakinga už resursų kūrimą (fabriko paslauga) ir kita – faktiškai atsakingas už priėjimą prie informacijos saugomos resurse (“instance” paslauga).

Fabriko modelis leidžia dinamiškai kurti resursus, tuomet kai atsiranda poreikis. Tai naudinga tuomet, kai norima sukurti tam tikrą dinamiškumą sąveikaujant su paslauga. Paslaugos teikiamos operacijos išlieka tos pačios, tačiau jos yra atliekamos su skirtingais resursais. Kiekvienas atskiras resursas gali turėti savo gyvavimo laiką, savo saugumo apribojimus ir pan. Vienas iš pavyzdžių kada tai gali būti patogu - jei norima suteikti tą patį funkcionalumą skirtingiems vartotojams naudojantiems skirtingus duomenis.

Kai klientas norės sukurti naują resursą, jis kreipsis į fabriko paslaugą, kuri bus atsakingas už naujo resurso sukūrimą ir inicializaciją. Kiekvienam sukurtam resursui priskiriamas unikalus raktas. Fabriko paslauga, sukūrus resursą, gražiną WS-Resurso (paslauga + sukurtas resursas) adresą (endpoint reference). Šis adresas leidžia mums unikaliai identifikuoti vieną WS-Resursą – paslaugą sujungtą su atitinkamu resursu.



7 pav. *Daugelio resursų paslauga*

Fabriko paslauga sukurs naują resursą ir grąžins sukurtos paslaugos+ resurso adresą, įėjimo tašką (EPR – endpoint reference).

3.2. Paslaugos klientai

Kaip jau minėta pritaikant sistemą Grid paslaugų architektūrai, sistemos dalis atliekanti visą logiką realizuojama kaip paslauga. Aiškiausiai ši dalis atsiskiria turint kliento serverio architektūrą.

Kai paslauga jau realizuota, kitas žingsnis yra paslaugos kliento sukūrimas. Jei sistema buvo realizuota kliento serverio principu, paprastai kliento programinėje įrangoje reikalingi tik minimalūs pakeitimai, t.y., kreipinius į servisą reikia pakeisti kreipiniais į paslaugą. Reikia pastebėti, jog ne visada paslaugos klientas yra vartotojo sąsajos programa, paslaugos klientu ar gavėju gali būti ir kita paslauga.

Iš esmės visiškai nesvarbu kokiomis technologijomis realizuota paslaugos vartotojo sąsaja. Kadangi žiniatinklio paslaugos užtikrina sveikumą tarp skirtingų platformų ar programavimo aplinkų, paslaugos klientam visiškai nereikia žinoti kokioje aplinkoje veikia paslauga. Taigi paslaugos vartotojo sąsajos gali būti realizuojamos įvairiomis technologijomis ir nepriklauso nuo to, kaip technologiškai realizuota paslauga. Paslaugos vartotojo sąsaja gali būti elementari pateikianti tik tekstinę informaciją arba pakankamai sudėtinga, turinti gausybę valdymo galimybių ir pateikianti rezultatus grafiškai. Galimi atvejai, kai viena paslauga gali turėti kelias sąsajas, ir atvirkščiai - viena sąsaja gali aptarnauti kelias paslaugas.

3.3. Automatizavimo įrankio kūrimo galimybės

Iš esmės pateiktos rekomendacijos nepriklauso nuo esamos sistemos realizacijos

technologijos ar WSRF Grid platformos. Tačiau stengiantis šį adaptavimo procesą automatizuoti, kuriant tam tikrą įrankį, kuris leistų atlikti visus pateiktus proceso žingsnius atsiranda priklausomybė nuo naudojamos technologijos. Tokiu atveju reikia, kad įrankis palaikytų kelias programavimo kalbas, tam kad būtų galima iš išeities kodo išgauti norimus metodus ar kintamuosius, arba kad būtų nesunku pridėti naują programavimo kalbą. Paslaugos sąsaja gali būti sugeneruota kokioje nors kalboje ir transformuota į WSDL aprašą. Realizacijos platformos pasirinkimas gali turėti specifinių realizacijos detalių, todėl įrankis gali būti kuriamas vienai kažkokiai platformai, tarkim žinomiausiai GT4.

4. Magistro darbo projektinės dalies dokumentacija

Magistro darbo projektinės dalies tikslas buvo restruktūrizuoti bei pritaikyti egzistuojančią nuotolinio grafinio testavimo sistemą TestTool paslaugom orientuotai Grid aplinkai. Remiantis 3skyriuje pateiktomis sistemos perprojektavimo Grid paslaugoms rekomendacijomis buvo sukurta nauja programinė įranga bei sukurta visa projekto dokumentacija, nuo reikalavimų specifikacijos iki vartotojo vadovo.
Pateikiame sukurtos techninės – projektinės dokumentacijos svarbiausius aspektus.

4.1. Sistemos apibūdinimas

4.1.1. Programų sistemos funkcijos

Kuriamoje sistemoje egzistuoja trys vartotojų rolės – autorius, dėstytojas ir studentas. Pagal vartotojų tipus pateiktos pagrindinės apibendrintos sistemos funkcijos.

Sistemoje yra trys pagrindiniai vartotojų tipai:

- **Autorius**

  Sistemos vartotojas atsakingas už turimų mokymo objektų/ duomenų failų įkėlimą į duomenų saugyklą.

- **Dėstytojas**

  Dėstytojas atsakingas už testo sukūrimą, klausimų parinkimą iš duomenų saugyklos ir sukėlimą į testą. Jis nustato sukurto testo apribojimus, gali bet kada redaguoti ar ištrinti sukurtą testą taip pat gali peržiūrėti ir redaguoti testo sprendimo įvertinimus.

- **Studentas**

  Sistemos vartotojas sprendžiantis dėstytojo paruoštą testą. Išsprendus testą studentas yra įvertinimas.

Sistemos funkcijos

Sistemos teikiamos paslaugos yra sugrupuotos į logines grupes. Kiekviena grupė apima jai būdingas susijusias funkcijas. Žemiau pateikiami trumpi funkcijų grupių aprašai.

Darbas su duomenų saugykla

Autorius naudodamasis duomenų saugyklos funkcijomis gali:

- Peržiūrėti duomenų saugykloje saugomus objektus
- Įkelti duomenų failą
- Ištrinti duomenų failą

Testo sukūrimas

Dėstytojas naudodamasis testavimo paslauga galės:

- Peržiūrėto sukurtų testų sąrašą
- Sukurti naują testą
- Redaguoti testo duomenis
- Ištrinti testą

Testo sprendimas

Studentas naudodamasis testavimo paslauga gali:

- Pasirinkti testo režimą (praktika/testas)
- Gauti klausimą sprendimui
- Gauti išspręsto klausimo įvertinimą
- Gauti pilną išspręsto testo įvertinimą

Rezultatų peržiūra

Dėstytojas naudodamasis testavimo paslauga gali:

- Peržiūrėti įvertintų testų sąrašą
- Peržiūrėti spręsto testo klausimų įvertinimus
- Redaguoti testo įvertinimą

4.1.2. Vartotojo problemos

Kuriama sistema leis išspręsti resursų valdymo, paskirstymo bei skirtingų sistemų problemas iškylančias e-mokymo sistemose.

Šiuo metu elektroniniame mokyme dėstytojams tampa svarbūs tokie uždaviniai kaip pakartotinis mokymo medžiagos panaudojimas bei mokymo paslaugos komponavimas iš atskirų dalių. Naudojant metaduomenimis aprašytus mokymosi objektus bei paslaugas bus galima efektyviai ir greitai pritaikyti mokymosi paslaugą prie poreikių, kurie gali iškilti atskiriem asmenim ar grupėm.

4.2. Reikalavimai sistemai

Pilni funkciniai sistemos reikalavimai pateikti sitemos reikalavimų specifikacijoje.

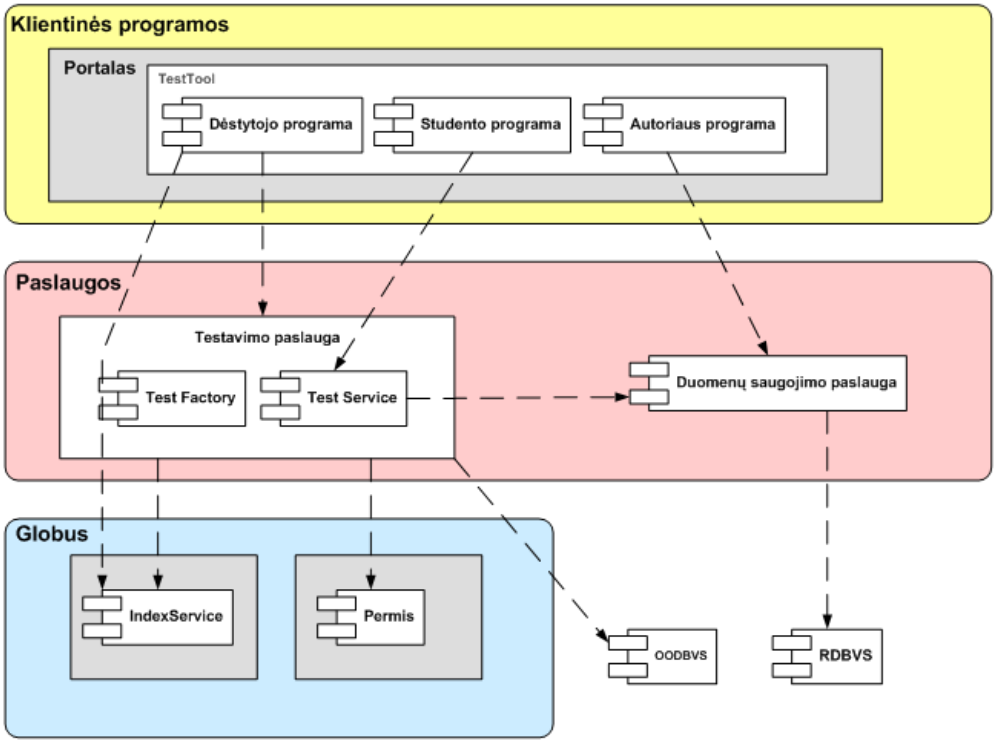
4.2.1. Apribojimai sprendimui

- Sistema turi veikti Grid aplinkoje.
- Sistema realizuojama POA architektūroje
- Kaip Grid tarpinė programinė įranga naudojamas Globus Toolkit 4

4.2.2. Bendradarbiaujančios Mo&St VO posistemės

Kadangi kuriama sistema yra dalis Mo&St virtualios organizacijos, tai sukurtas mokymosi – testavimo servisas bendraus, keisis duomenim su šiomis VO posistemėmis:

- Duomenų saugojimo posistemė – virtualios organizacijos mokymo objektų saugykla.
- Index service - virtualios organizacijos paslaugų – servisų registras
- PERMIS/ Autorizacijos servisas - virtualios organizacijos paslaugų autorizavimo modulis



8 pav. *Žinių vertinimo paslauga Mo&St virtualioje organizacijoje*

4.3. Architektūrinis vaizdas

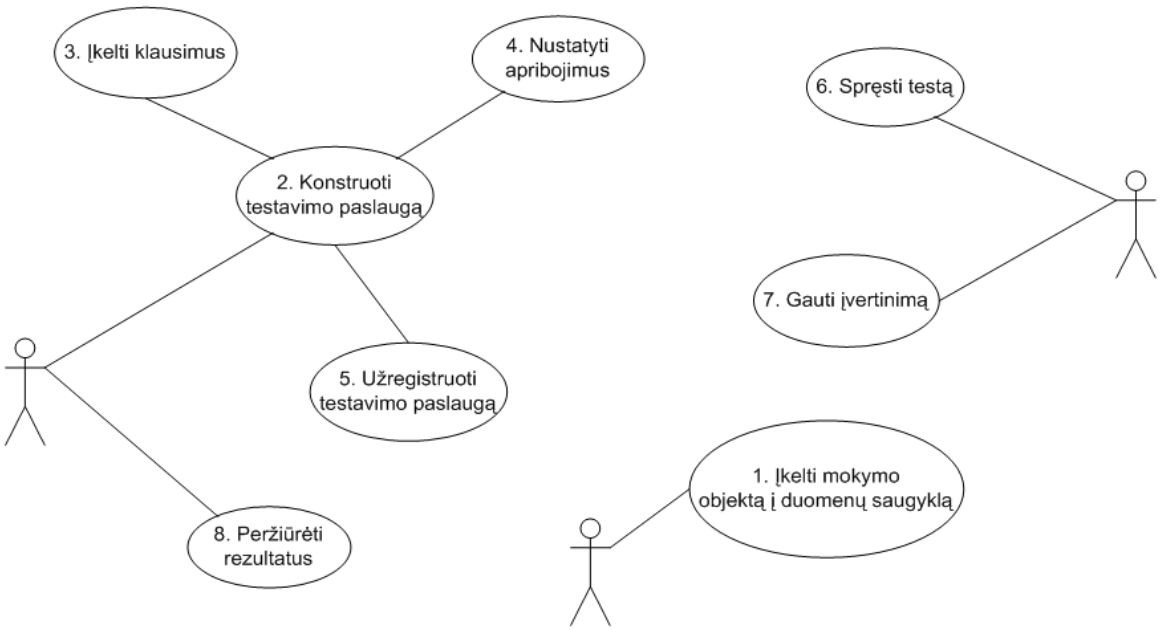
Sistemos architektūra pateikiama keliais vaizdais: panaudojimo atvejų (PA), statinis, dinaminis ir išdėstymo. Šie vaizdai yra pateikiami kaip Rational Rose modeliai naudojant

unifikuota modeliavimo kalba (UML). Sistemos architektūra pateikta remiantis RUP (Rational Unified Process) rekomendacijomis. Sistemos specifikacija pateikta šiais vaizdais kuriems įgyvendinti reikia UML diagramų:

1. Panaudojimo atvejų vaizdas (panaudojimo atvejų diagrama)
2. Sistemos statinis vaizdas (paketai ir klasių diagramos)
3. Sistemos dinaminis vaizdas (būsenų, veiklos, sekų, bendradarbiavimo diagramos)
4. Išdėstymo vaizdas (išdėstymo diagrama)

Toliau pateikiami tik esminiai sistemos architektūriniai vaizdai – panaudojimo atvejai, sistemos suskaidymas į paketus bei išdėstymo vaizdas. Pilnoje projekto dokumentacijoje pateikiami išsamūs ir pilni architektūrinių vaizdų aprašymai.

4.3.1. Panaudojimo atvejų vaizdas

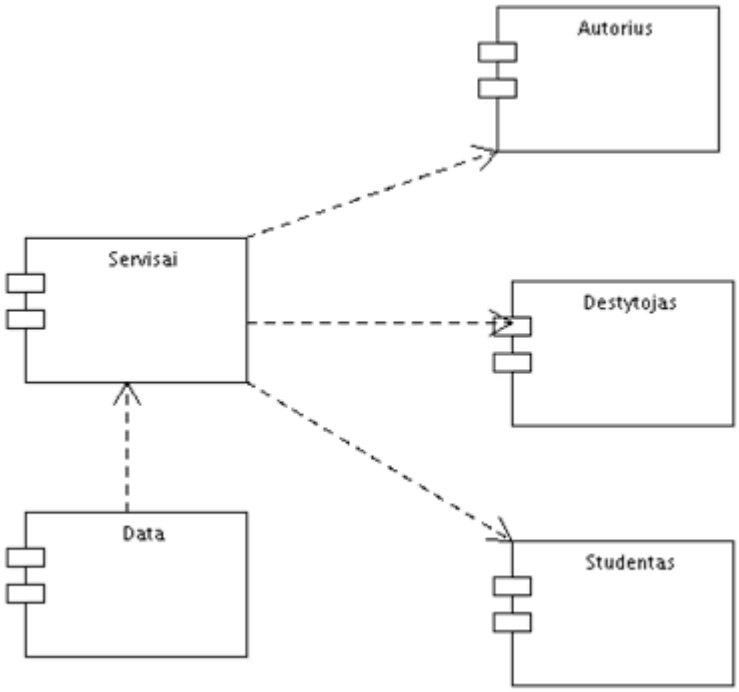


9 pav. *Sistemos ribos*

4.3.2. Loginis vaizdas

Šis skyrius aprašo sistemos loginę sistemos struktūrą. Pateikia sistemos išskaidymą į paketus ir juos sudarančias klases.

Sistema suskaidyta į penkis pagrindinius paketus:

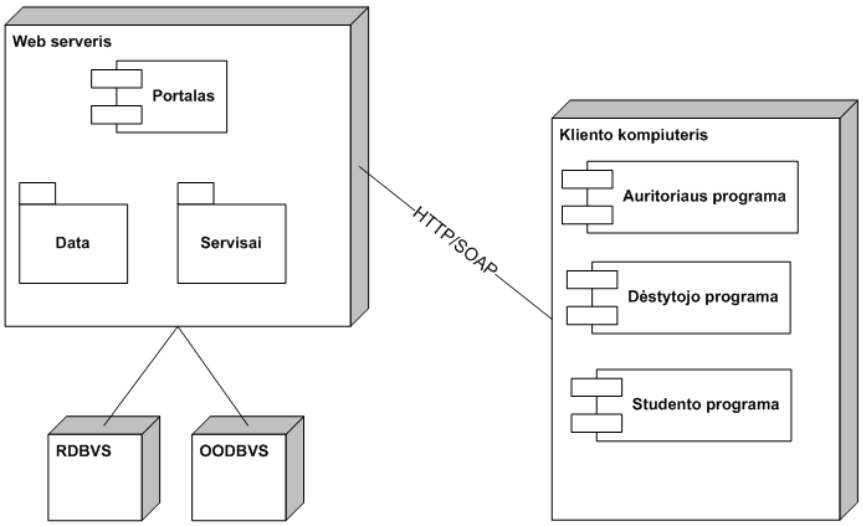


10 pav. *Sistemos suskaidymas į paketus*

Kiekvieno paketo trumpas paaiškinimas:

- *Autorius* – šis paketas atsakingas už autoriaus darbą su sistema. Jame yra klasės realizuojančios vartotojo sąsają, kurių pagalba vartotojas gali atlikti norimus veiksmus (įkelti arba ištrinti mokymo objektą į duomenų saugyklą).
- *Dėstytojas* – šis paketas atsakingas už dėstytojo darbą su sistema. Jame yra klasės realizuojančios vartotojo sąsają, kurių pagalba dėstytojas gali administruoti savo sukurtus testus – sukurti naujus testus, nustatyti jų apribojimus, įkelti mokymosi objektus, šalinti pasirinktus testus bei redaguoti jų duomenis.
- *Studentas* – šis paketas atsakingas už studento darbą su sistema. Jame yra klasės realizuojančios vartotojo sąsają, kurių pagalba testas grafiškai atvaizduojamas ir pateikiamas vartotojui.
- *Servisai* – Pakete pateikiamos klasės realizuojančios visą sistemos funkcionalumą. Jas naudoja kitų paketų klasės, kad atlikti duomenų vaizdavimą. Komponentas atsakingas už duomenų saugyklos funkcijas, testo serviso sukūrimą, apribojimų nustatymą, įvertinimą bei prisijungimą prie testo serviso.
- *Data* – Pakete pateikiamos klasės sudaro sąsają su objektine duomenų baze. Skirtas duomenų saugojimui ir tvarkymui duomenų bazėje.

4.3.3. Išdėstymo vaizdas



Serverio kompiuteris

Sistema įdiegta į KTU *elgrid.ktu.lt* serverio kompiuterį su įdiegta Linux operacine sistema. Web servisai realizuoti Globus aplinkoje. Serveryje turi būti įdiegta Java 1.5+ ir Globus Toolkit 4.1 programinė įranga.

	CPU	Disko vieta	RAM
Java	166 MHz	130 MB	34 MB
Globus toolkit	170 MHz	600 MB	256 MB

Minimalūs serverio reikalavimai:

CPU: 180 MHz

RAM kiekis: 512 MB

Disko dydis 1,5 GB

Kliento kompiuteris

Klientas naudosis sistemos funkcijomis per interneto naršyklę. Kompiuteris gali būti su Linux, Windows operacinėmis sistemomis. Vartotojo kompiuteryje turi būti įdiegta Java vykdymo aplinka Java jre 1.5 ar aukštesnė versija.

Minimalūs reikalavimai:

CPU: 100 MHz

RAM kiekis: 256 MB

Sistema realizuota Grid aplinkoje naudojant XML žiniatinklio paslaugas. Į paslaugas orientuotas požiūris įgalins sąveikumą tarp skirtingų platformų. Bus galima praplėsti sistemą

naudojimui su skirtingais klientų tipais. Didesniam saugumo užtikrinimui nesijungiama tiesiai prie duomenų bazės, o naudojamas programinis sluoksnis sąsajai su duomenų baze. Pasirinkta objektinė duomenų bazių valdymo sistemos OzoneDB , kuri leidžia tiesiogiai saugoti objektus, taip suteikdama didesnį efektyvumą bei integraciją su Java aplinkoje realizuojamomis paslaugomis.

5. Žinių vertinimo paslaugos tyrimas

5.1. Paslaugos išplėtimo galimybių tyrimas

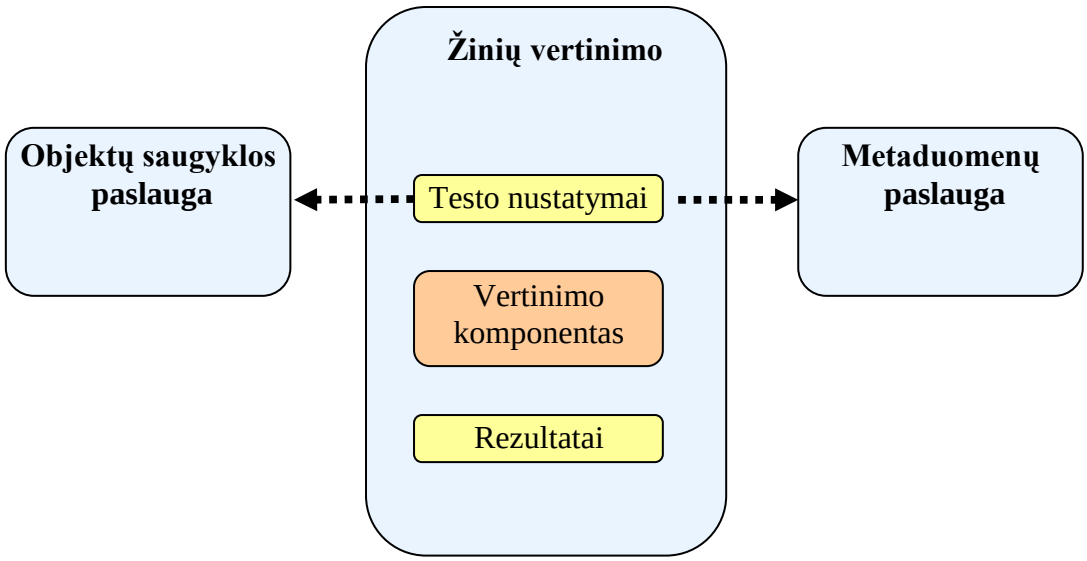
Darbe pateiktos rekomendacijos egzistuojančios interaktyvios e-mokymo sistemos pritaikymui paslaugom orientuotai architektūrai Grid aplinkoje. Pritaikymo rezultatas yra Grid paslauga ar paslaugų rinkinys įgyvendinantis pageidaujamą funkcionalumą. Šis sistemų realizavimas paslaugomis užtikrina lengvesnę sistemų integraciją, išplečiamumą, dinamiškumą bei sąveikumą.

Pagal pateiktas rekomendacijas, atlikome TestTool sistemos pritaikymą paslaugom orientuotai Grid architektūrai. Naujai sukurta sistemos versija buvo kuriama nekeičiant jos funkcionalumo. Atlikus TestTool sistemos analizę ir pasinaudojant esamu jos kodu pagal rekomendacijose pateiktus pritaikymo žingsnius buvo sukurta žinių vertinimo paslauga Mo&St virtualioje organizacijoje. Sukurta paslauga vertinant žinias leidžia naudoti tik TestTool tipo objektus. Pabandysime paanalizuoti esamą paslaugos realizaciją kokybės ir išplėtimo požiūriu.

TestTool sistemos analizės metu, galvojant apie kuriamos paslaugos sąsajos nustatymą, nuspręsta TT sistemos funkcionalumą realizuoti kaip vieną žinių vertinimo paslaugą. Atlikti du esminiai architektūriniai pakeitimai. Nuspręsta atskirti sistemos vartotojus ir naudojamus mokymo objektus nuo pačios paslaugos. T.y., TesttTool sistema buvo autonomiška, savo viduje sauganti vartotojus bei objektus, naudojamus testų kūrimui. Realizuojant žinių vertinimo paslaugą, buvo ruošiamasi ją integruoti į virtualios organizacijos architektūrą, todėl saugoti vidinius sistemos vartotojus nebeliko prasmės. Mokymo objektai reikalingi žinių vertinimo paslaugai taip pat nebesaugomi jos viduje, o jų paieška ir iškvietimas atliekamas naudojantis VO metaduomenų ir duomenų saugojimo paslaugomis.

Šie architektūriniai pokyčiai realizuojant žinių vertinimo paslaugą leido įgauti naują kokybę ir aukštesnį abstrakcijos lygmenį lyginant su buvusia uždara, monolitine sistema. Integruojant paslaugą į MirS virtualios organizacijos architektūrą atsiskleidžia naujos galimybės išplėsti žinių vertinimo paslaugą, laikant ją ne vien TestTool testavimo galimybes realizuojančia paslauga, o bet kokios testavimo sistemos branduoliu su didelėm išplėtimo ir lankstumo galimybėm.

Aukštame abstrakcijos lygyje žinių vertinimo paslauga realizuota TestTool sistemos pagrindu gali būti pavaizduota taip:



11 pav. *Žinių vertinimo paslaugos struktūrair bendraujančios paslaugos*

Žinių vertinimo paslauga yra atsakinga už testo struktūros sudarymą, jos pateikimą vartotojui, vartotojo atliktų veiksmų įvertinimą bei rezultatų saugojimą. Ši paslauga bendrauja su metaduomenų duomenų saugyklos paslaugomis, surasdama ir įkeldama vartotojo pasirinktus objektus į kuriamą testą. T.y. ji naudojasi kitomis VO paslaugomis atlikdama teikiamas paslaugas.

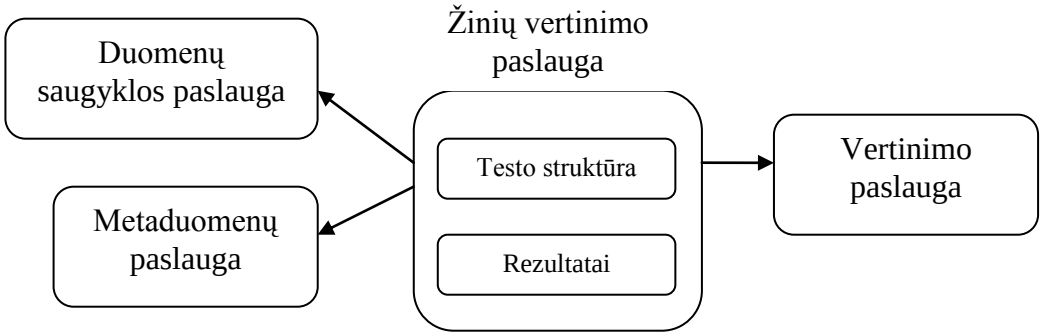
Kaip matome žinių vertinimo paslaugomi išskirti tris pagrindines dalis:

- Testo nustatymai – tai testo struktūros sukūrimas, objektų sukėlimas į testą ir testavimo režimo nustatymas. Ši dalis užtikrina žinių vertinimo paslaugos su sukeltais objektais sukūrimą. Tokia paslauga yra savarankiška, autonomiška, su nustatytu ribotu gyvavimo laiku. Ji yra užregistruojama į paslaugų indeksą, gali būti viešai surasta ir prieinama (jei leidžia VO saugumo politika).
- Vertinimo komponentas – tai sukurtos žinių vertinimo paslaugos įvertinimo etapas jos pateikimo fazėje. T.y., kadangi šiuo atveju mes turime žinių vertinimo – testavimo paslaugas, tai natūralu, jog kai vartotojas atlieka testą ar tam tikrą užduotį, jis turi būti įvertintas. TestTool pagrindu realizuotoje paslaugoje, ši dalis atsakinga už pateikto grafinio testo sprendimo įvertinimą skaitine forma.
- Rezultatai – tai sukurtos testo paslaugos įvertinimų išsaugojimas vėlesnei peržiūrai.

Žinių vertinimo paslauga Grid aplinkoje buvo sukurta naudojantis fabriko modeliu, todėl tai leidžia kurti paslaugų atskirus atvejus (angl. “instances”) kai atsiranda poreikis su skirtingais objektais, apriboti jų gyvavimo laiką bei užregistruoti į paslaugų indeksą kaip atskiras savarankiškas paslaugas, kurias bet kada galima rasti ir iškviesti. Tokia galimybė atveria naujas perspektyvas paslaugos panaudojimui. Iš esmės, galima šia paslauga naudotis

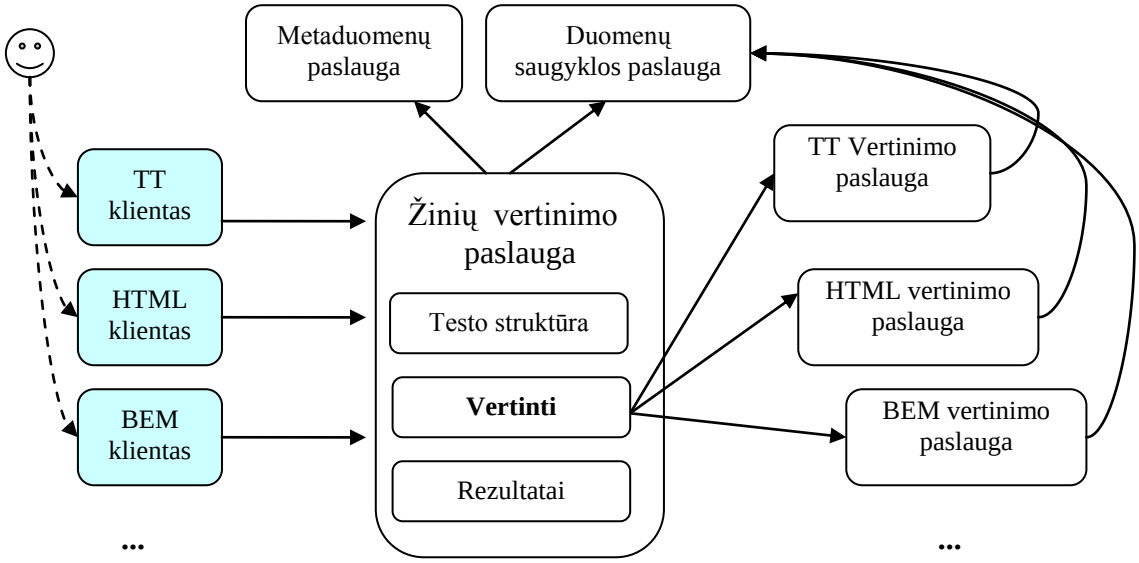
bet kokiam mokomųjų objektų tipui sukuriant paslaugos atskirus atvejus būtent tam objektų tipui.

Tokiu atveju, būtinas sistemos pakeitimas – vertinimo komponento iškėlimas už paslaugos ribų. Priėmus tokį požiūrį žinių vertinimo paslauga gali labai lanksčiai adaptuotis bet kokiam objektų tipui. Kadangi vertinimo komponentas yra specifinis konkrečiam objektų tipui (pvz. TestTool klausimam) tai, sukuriant atskirą vertinimo komponentą kiekvienam žinių vertinime naudojamam objektų tipui galima žinių vertinimo paslaugą išplėsti keletui objektų tipų. Kitaip tariant galima naudotis bazine žinių vertinimo paslauga, testų struktūros sudarymui, jos pateikimui vartotojui, rezultatų saugojimui, o vertinant vartotojo atliktu veiksmus kreiptis į atitinkamą vertinimo paslaugą, mokančią įvertinti to tipo objektus. Savaime aišku, kad tokiu atveju turi egzistuoti ir vartotojo sąsajos programos, mokančios interpretuoti ir pateikti vartotojui tokio tipo objektus.



12 pav. *Vertinimo komponento atkyrimas realizuojant jį atskira paslauga*

Vertinimo paslaugos sąsaja bet kokia objektų tipui turės tik vieną metodą, kuris paima objekto duomenų saugyklos identifikatorių ir vartotojo modifikuotą objektą, bei grąžina skaitinį įvertinimą. Vertinimo paslaugų iškvietimas ir panaudojimas parodytas sekančioje schemoje. Joje pateiktas pavyzdys kai žinių vertinimo paslaugoje galima panaudoti tris skirtingus objektų tipus: TestTool klausimus, HTML failus, BEM (baigtinių elementų) uždavinių failus.



13 pav. *Žinių vertinimo paslaugos išplėtimas skirtingiems objektų tipams*

Vartotojas norėdamas pasinaudoti žinių vertinimo paslauga testo pateikimo metu pasinaudoja vartotojo sąsajos programa, kuri kreipiasi į žinių vertinimo paslaugą, gauna testą sudarančius objektus ir atvaizduoja juos vartotojo sąsajos programose. Vartotojas atlikęs veiksmus yra įvertinamas – vartotojo sąsajos programa kreipiasi į tą pačią žinių vertinimo paslaugą, o ši savo ruožtu į atitinkamą vertinimo paslaugą. Kaip matome tokia schema leidžia labai nesunkiai pridėti vertinimo paslaugas naujam objektų tipui.

Norint pasinaudoti tokia galimybe praktiškai sukurtai sistemai jokių kitų pakeitimų nereikia atlikti, tačiau būtinos tokios sistemos prielaidos:

- Paslaugos vartotojas, dažniausiai dėstytojas, pats yra atsakingas už objektų tipus, kokie yra sukeliami į testą, t.y., jis turi užtikrinti ir žinoti, jog paslauga su tokiais įkeltais objektais bus įmanoma pasinaudoti.
- Egzistuoja interpretatorius – paslaugos vartotojo sąsaja kiekvienam naudojamam (arba pageidaujam) objektų tipui.

Kitaip tariant, tai reiškia, jog mes galime naudoti žinių vertinimo paslaugą testo administravimui ir rezultatų saugojimui. Testų vertinimas turėtų būti iškeliamas į išorė ir realizuojamas kitose vertinimo paslaugose priklausomai nuo objektų tipo. Vartotojo sąsajos programos – interpretatoriai taip pat būtų skirtingi ir priklausytų nuo objektų tipo.

Kadangi patys testą sudarantys objektai nėra saugomi paslaugos viduje, tai leidžia mums atskirti paslaugą nuo objektų tipų, taip padidinant paslaugos abstrakcijos lygį. Priešingai uždarai, monolitinei sistemai, kur objektų tipai yra griežtai apibrėžti ir yra interpretuojami konkrečios vartotojo sąsają teikiančios programos, šiuo atveju turime lanksčią paslaugą, leidžiančią panaudoti skirtingus objektus skirtingiems testams. Aišku tam, kad būtų

galima pasinaudoti paslauga, reikia turėti tą vartotojo sąsajos programą, kuri gebą atvaizduoti to tipo mokomuosius objektus. Tačiau šis klausimas čia nenagrinėjamas.

Daugeliu atveju pritaikant sistemą Grid paslaugų architektūrai vartotojo sąsajos programose užtenka atlikti minimalius pakeitimus, nes visa sistemos logiką teikia paslauga, o kliento programa tik kreipiasi į ją siųsdama ar gaudama duomenis.

Pagrindinis privalumas yra tai, kad bet kada galima pridėti pageidaujamą objektų tipą, nemodifikuojant pačios žinių vertinimo paslaugos. Panašiu būdu galima išplėsti ir kitas žinių vertinimo paslaugos funkcijas, pvz. grafinę klausimų peržiūrą.

5.2. Rezultatų įvertinimas

Atlikto darbo rezultatų įvertinimas gali būti kiekybinis arba kokybinis. Šiuo atveju darbą įvertinti galima tik tuo požiūriu, kokią naują kokybę ir efektyvumą jis suteikia e-mokymo sistemose. Šiuo atveju nebandėme tirti ir kiekybiškai įvertinti kokią technologinę naudą teikia Grid aplinka naujai sukurtai sistemos versijai. Tai yra nebuvo bandoma pasinaudoti Grid infrastruktūros teikiamu skaičiuojamųjų resursų paskirstymu ar duomenų perdavimu, replikacijomis ir pan. Pritaikant sistemą Grid POA aplinkai buvo siekiama išspręsti sistemų integracijos, pakartotinio panaudojimo problemą, suteikti lankstumo, dinamiškumo realizuojant sistemą kaip bendraujančių savarankiškų paslaugų aibę. Taigi tyrimo metu neatsižvelgiama į sistemos veikimo greitį ar patikimumą, todėl nebuvo matuojami ir įvertinami tokie parametrai kaip sistemos užklausų priėmimo greitis ar tinklo pralaidumas ir pan.

Atliekant šiuo darbo įvertinimą kiekybinis įvertinimas tampa neįmanomas dėl aukščiau išvardintų priežasčių. Darbo rezultatai įvertinami naudojant kokybinius kriterijus. Lyginame kokias naujas savybes įgavo realizuota žinių vertinimo paslauga, lyginant su buvusia sistema TestTool (4 lentelė). Matome, jog žinių vertinimo paslauga yra lankstesnė, lengviau išplečiama nei buvusi monolitinė TT sistema. Kadangi žinių vertinimo paslauga realizuojama paslaugom orientuotoje architektūroje išplėtimas tampa nesunkus. POA suteikia galimybę kurti dideles sistemas iš atskirų komponentų, kurių atviros sąsajos leidžia juos jungti pageidaujamu būdu, įgyvendinant specifinę užduotį.

4 lentelė TT ir žinių vertinimo paslaugos palyginimas

Kriterijus	TestTool	Žinių vertinimo paslauga
Išplečiamumas	Negalima išplėsti sistemos jos nemodifikuojant	Galima išplėsti sistemą, jos nemodifikuojant, prijungiant kitas paslaugas

Mokymo objektų paskirstymas	Naudojami vidiniai objektai, esantys TestTool duomenų bazėje	Naudojami virtualios organizacijos duomenų saugyklose paskirstyti objektai
Skirtingi objektų tipai	Palaiko tik XML, nustatyto formato TT klausimus	Galim naudoti bet kokio tipo objektus, jei egzistuoja to tipo objektų vertinimo paslauga
Mokymo objektų pakartotinis panaudojimas	Gana sunkus, objektai susieti vidiniais ryšiais, bei naudojami tik TT sistemoje	Objektai gali būti naudojami daug kartų, saugomi išorėje
Klientų tipai	Tik Java aplinkos klientai	Bet kokia technologija, kuri leidžia komunikuoti su žiniatinklio paslaugomis, realizuoti klientai

.

5.3. Kitų sistemų pritaikymas paslaugoms orientuotai Grid architektūrai

Šiame darbe išanalizavome ir pateikėme rekomendacijas e-mokymo sistemos pritaikymui Grid POA aplinkai. Ištyrėme kokią naudą ir galimybes paslaugom orientuota architektūra gali pateikti elektroninio mokymo sistemom. Tačiau tai nereiškia, kad pateikti pritaikymo žingsniai negali būti taikomi kitos srities sistemom. Į paslaugas orientuota architektūra šiuo metu yra naujas požiūris programų sistemų projektavime padedantis išspręsti daugelį projektavimo, skirtingų sistemų integracijos, palaikymo bei lengvesnio sistemų pritaikymo išplėtimui ir pakeitimam problemų.

6. E-mokymo sistemos Grid paslaugų architektūroje eksperimentinis tyrimas

E-mokymo sistemos pritaikymo paslaugom orientuotai Grid aplinkai uždaviniui tirti pasirinkta TestTool sistema. Pabandysime praktiškai panaudoti darbe pateiktas rekomendacijas ir modifikuojant egzistuojantį TestTool išeities kodą realizuoti Grid paslaugą.

Realizuosime sistemos pritaikymą atlikdami kiekvieną pritaikymo rekomendacijų žingsnį. Ištirsime žinių vertinimo paslaugos išplėtimą panaudojant skirtingo tipo mokymo objektus.

6.1. TestTool sistemos pritaikymo Grid paslaugų aplinkai tyrimas

6.1.1. TestTool sistemos analizė

Pritaikant TestTool sistemą Grid paslaugų architektūrai, sistema buvo išanalizuota išskiriant dalis, kurios bus realizuotos kaip paslaugos. Sistemos analizės metu nustatyta kokius architektūrinius pakeitimus reiks atlikti bei kokie pakeitimai sistemoje yra būtini, adaptuojant ją paslaugom orientuotai Grid architektūrai.

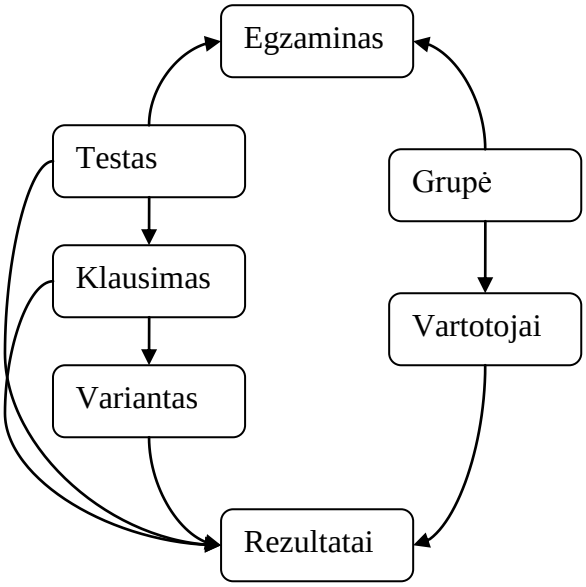
Egzistuojanti TestTool sistema yra realizuota kliento – serverio principu. Pagrindinės sistemos dalys:

- TestTool serveris
- Klientinės programos
    - Dėstytojo programa
    - Studento programa
    - Autoriaus programa

TestTool serveris atsakingas už visos sistemos veikimo logiką. Jis atlieka vartotojų ir testų administravimą, jų duomenų saugojimą, testų įvertinimą bei rezultatų saugojimą.

Vartotojo sąsajos programos (klientai) jungiasi prie TT serverio, kuris atlieka vartotojo pageidaujamus veiksmus. Dėstytojo programa naudojasi dėstytojas tvarkydamas sistemos vartotojus, kurdamas testus bei peržiūrėdamas rezultatus. Studentai, spręsdami testus naudojasi Sudento programa. Autoriaus programa yra savarankiška, autonominė programa, kurios pagalba kuriami grafiniai klausimų objektai, iš kurių sudaromas testas.

Taigi TT yra autonomiška, uždara programa, kuri savo viduje saugo vartotojus ir su jais susietus duomenų objektus. Ši duomenų struktūra abstrakčiai gali būti pavaizuota taip:



14 pav. *Autonomiškos TestTool sistemos duomenų struktūra*

Įvertinus egzistuojančią TT duomenų struktūrą, nuspręstą atlikti tam tikras jos modifikacijas siekiant didesnio lankstumo, pakartotinio panaudojimo bei lengvesnės integracijos bei sąveikos su kitom paslaugom. Siekiant realizuoti sistemą kaip Grid paslaugą buvo atlikti tokie esminiai architektūriniai pakeitimai:

- Atsisakyta saugoti vidinius sistemos vartotojus. Kadangi sistema bus realizuojama kaip VO žinių vertinimo paslauga, tai ji bus prieinama plačiam vartotojų ratui virtualioje organizacijoje, priklausomai nuo joje palaikomos teisių suteikimo ir autorizacijos politikos. Saugoti vidinius vartotojus tampa beprasmiška ir visiškai nenaudinga. Paslauga tampa viešai prieinama, todėl ji turi nepriklausyti nuo to, kokie vartotojai ja naudosis, t.y., šis vartotojų teisių bei priėjimo galimybių klausimas iškeliamas į aukštesnį virtualios organizacijos saugumo politikos lygį.
- Testą sudarančių klausimų objektų saugojimas iškeliamas už sistemos ribų. Kitaip tariant testo struktūrą apibrėžiama ir saugoma paslaugoje, tačiau atsisakoma saugoti pačius klausimų objektus prie paslaugos, vietoj to saugomos tik nuorodos į tuos objektus. Šiuo atveju, tai reiškia, kad testo klausimų objektai gali būti paskirstyti VO priklausančiuose skirtinguose serveriuose ar duomenų saugyklose.
- Apjungti klausimą sudarantys variantai. T.y. klausimas tampa kaip vienas nedalomas vienetas, sudarytas yra keleto variantų. Šis klausimas yra mokymo objektas, turintis savo metaduomenis, pagal kuriuos jį galima surasti VO duomenų saugykloje.

Įvertinus šiuos priimtus sprendimus buvo atliekamas TestTool sistemos perprojektavimas ir realizacija Grid paslaugų architektūrai pagal darbe pateiktas pritaikymo rekomendacijas. Sukurta paslauga integruota į Mo&St virtualią organizaciją.

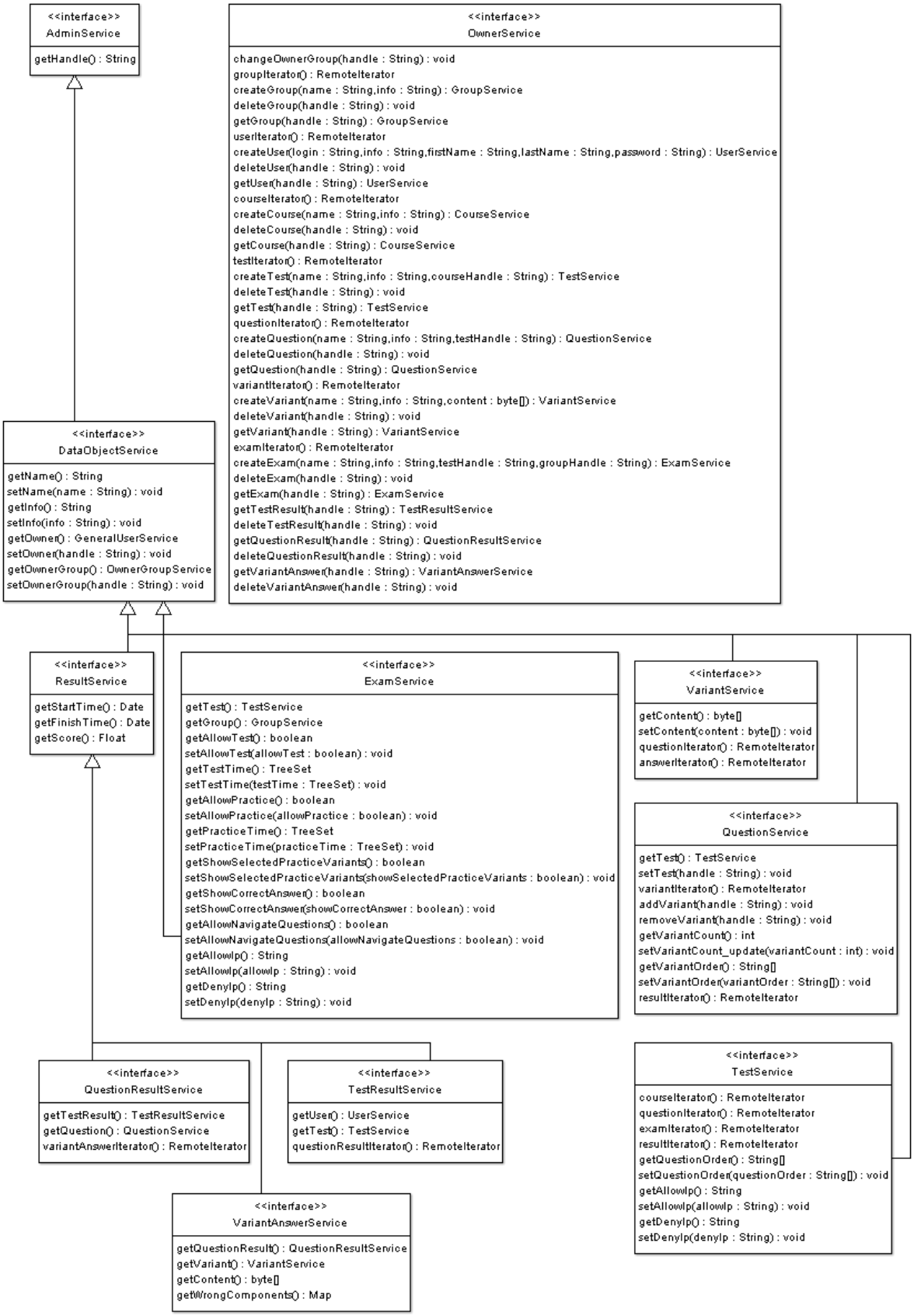
6.1.2. Žinių vertinimo paslaugos sąsajos bei resurso savybių nustatymas

Išanalizavus esamą sistemą ir nusprendus kokius architektūrinius pakeitimus reikės atlikti buvo projektuojama kuriamos paslaugos sąsaja. Šiuo atveju paslauga įgyvendina TT serverio teikiamą funkcionalumą. Pasinaudota TT serverio išeities kodu nustatant paslaugos sąsają.

TT serveris sudarytas iš tokių paketų:

- *data* – objektinę duomenų bazę realizuojančios klasės
- *info* – pagalbinės duomenų struktūros darbui su duomenų objektais
- *util* – pagalbinės klasės, atliekančios tokius veiksmus kaip darbas su XML dokumentais, datos formatais, duomenų laukais ir pan.
- *services* – TT serverio teikiami servisai, į kuriuos tiesiogiai kreipiasi klientai.
- *eval* – vertinimo komponentas, atsakingas už pateikto klausimo įvertinimą.

Pagal sudarytas rekomendacijas pasirenkamas *services* paketas kaip esminis komponentas paslaugos sąsajos sudarymui. Kadangi *services* paketas teikia servisus, prie kurių tiesiogiai jungiasi klientai, tai iš jo galima išskirti operacijas, kurias turės teikti kuriama paslauga. Paslaugos sąsaja bus suformuota pasinaudojant reikalingais metodais iš šio paketo klasių ir atmetant nereikalingus metodus. Žemiau pateikiama *services* paketo klasių diagrama (žr. 15 pav.).



15 pav. *TestTool serverio klasių diagrama*

Iš klasių išskirti ir pasirinkti tokie metodai:

public boolean getAllowPractice() throws RemoteException;

public void setAllowPractice(boolean allowPractice) throws RemoteException;

public boolean getAllowTest() throws RemoteException;

public void setAllowTest(boolean allowTest) throws RemoteException;

public QuestionService createQuestion(String name, String info, String testHandle) throws RemoteException;

public void deleteQuestion(String handle) throws RemoteException;

public QuestionService getQuestion(String handle) throws RemoteException;

public TestResultService getTestResult(String handle) throws RemoteException;

public void deleteTestResult(String handle) throws RemoteException;

public QuestionResultService getQuestionResult(String handle) throws RemoteException;

public void deleteQuestionResult(String handle) throws RemoteException;

public Float evaluateQuestion(String handle, boolean evaluateVariants) throws RemoteException;

public Float evaluateTest(boolean evaluateQuestions, boolean evaluateVariants) throws RemoteException;

public Question[] startExam(String courseHandle, String examHandle, boolean testMode) throws RemoteException;

public byte[] getVariant(String handle) throws RemoteException;

Pagal analizės dalyje pateiktas taisykles, atliekamas sąsajos modifikavimas:

- QuestionService keičiama į:

public void createQuestion (String name, String info, String testHandle) throws RemoteException

public void setQuestionName(String name)

public String getQuestionName ()

public void setQuestionInfo(String name)

public String getQuestionInfo()

- metodai kur naudojama QuestionService keičiami į

public void createQuestion(String name, String info, String testHandle) throws RemoteException

byte[] getQuestionContent(String handle) throws RemoteException

TestResultService keičiama į

public Date getTestStartTime()

public Date getTestFinishTime()

public float getTestScore()

QuestionResultService keičiama į

public Date getQuestionStartTime(String handle)

public Date getQuestionFinishTime(String handle)

public float getScore(String handle)

- Sudaryta paslaugos sąsaja įvertinama, pridedami paslaugai specifiniai metodai, kurie nėra gauti iš egzistuojančios sistemos kodo:

public void getTerminationTime();

public void serTerminationTime();

- Paslaugos resurso savybės parinktos atsižvelgiant į saugomus testo duomenis. Kadangi resurso savybės saugo būseną ir metaduomenis apie paslaugą, pasirinkti tokie

kintamieji resurso savybių įgyvendinimui:

- TestName – atskiro paslaugos atvejo pavadinimas
- TestInfo– atskiro paslaugos atvejo trumpas aprašymas
- TestOwner– atskiro paslaugos atvejo savininkas, VO dalyvio, sukūrusio šį atvejį identifikatorius
- TestMode– atskiro paslaugos atvejo testavimo režimas

Papildomai pridėtos tokios resursų savybės kaip paslaugos metaduomenys

- PaslaugosTipas – fabriko ar atskiro atvejo paslauga
- MokymoSritis – mokymosi dalykas
- InteraktyvumoLygis – aukštas ar žemas interaktyvumo lygis

Resurso savybėm nustatyti tokie modifikatoriai:

- Išgauti resurso savybę
- Išgauti daugelį resurso savybių
- Nustatyti resurso savybes

6.1.3. WSDL sudarymas

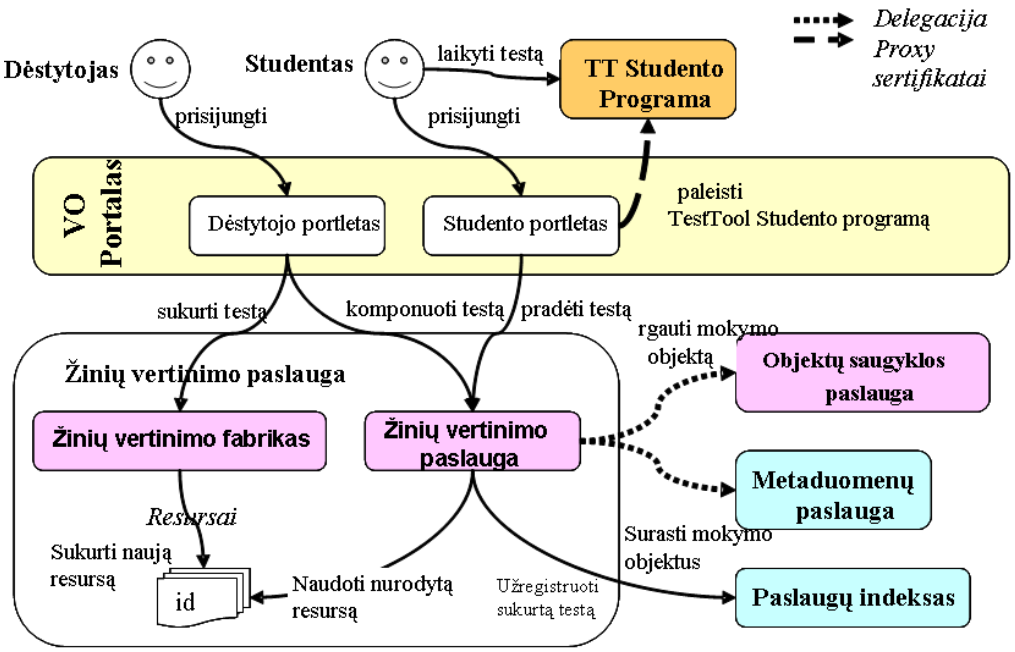
Pagal sudarytą paslaugos sąsają galima rašyti WSDL dokumentą nurodant paslaugos teikiamas operacijas ir duomenų tipus. Šiuo atveju WSDL rankomis nerašėme o pasinaudojome Apache Axis įrankiu JAVA2WSDL.

Sugeneruotas WSDL dokumentas buvo šiek tiek modifikuotas, pridedant specifinius GT4 platformos elementus.

6.1.4. Paslaugos realizacija

Turint paslaugos WSDL dokumentą atliekama jos įgyvendinimo klasių realizacija. Pasirenkamas paslaugos tipas ir atliekami programavimo darbai.

Žinių vertinimo paslauga realizuojama kaip daugelio resursų paslauga pagal fabriko projektavimo modelį. Realizuota paslauga integruojama į Mo&St virtualią organizaciją (16 pav.). Žinių vertinimo paslauga naudoja bendrą VO saugumo infrastruktūra, sisteminias paslaugas.



16 pav. *Žinių vertinimo paslauga Mo&St VO*

Fabriko modelis pritaikytas realizuojant lanksčias žinių vertinimo paslaugas. Testų fabriko paslauga yra atsakinga už dinamišką atskirą žinių vertinimo paslaugos atvejo sukūrimą. Šis atskiras atvejis yra nepriklausomas, naudoja atskirus resursus, turi nustatytą gyvavimo laiką ir yra sudarytas iš virtualioje organizacijoje paskirstytų mokomųjų objektų. Tokios atskirų atvejų paslaugos yra komponuojamos į kitas paslaugas virtualioje organizacijoje [2]. T.y., kuriamos agreguotos paslaugos arba iš paslaugų komponuojamas mokymosi kelias. Kai atskiro atvejo paslauga tampa nereikalinga, ji yra sunaikinama.

Žinių vertinimo paslaugos atveju egzistuoja du vartotojo sąsajos tipai TT objektų atvaizdavimui – tai dėstytojo portletas bei Java programa studentui. Mo&St VO vartotojo sąsajos sluoksnis realizuojamas portalu ir portletais. Jei portleto neužtenka funkcionalumui realizuoti galima naudoti ir kitus sprendimus, pvz. Java programa kliento kompiuteryje. Kaip matome paslauga gali turėti įvairius klientų tipus.

6.2. Žinių vertinimo paslaugos išplėtimo tyrimas

Šio darbo metu pritaikėme egzistuojančią sistemą TestTool paslaugom orientuotai Grid aplinkai. Pritaikymo rezultatas – sukurta žinių vertinimo paslauga virtualioje organizacijoje. Sukurtos programinės įrangos tyrimo metu (žr. skyrelį 5) pateiktas kokybės įvertinimas bei sistemos išplėtimo galimybės. Grid technologijos ir paslaugom orientuota architektūra žymiai palengvina skirtingų sistemų sąveiką bei integraciją, suteikia galimybes lengvai išplėsti paslaugos teikiamą funkcionalumą, komponuojant paslaugas kaip atskirus komponentus ir taip pasiekiant galutinį pageidaujamą tikslą ar užduotį.

Čia pabandysime praktiškai ištirti kaip palengvėja sistemos galimybių išplėtimas realizuojant sistemą Grid paslaugomis. Kitaip tariant sukurtą žinių vertinimo paslaugą bandysime išplėsti, kad būtų galima naudoti ne tik TestTool tipo objektus, o bet kokio tipo mokomuosius objektus skirtus testavimui.

Realizuotoje žinių testavimo paslaugoje atliekami tik minimalūs pakeitimai, t.y. vertinimo metode atsiranda papildomas parametras – vertinimo paslaugos adresas. Pagal šį parametrą žinių vertinimo paslauga kreipiasi nurodytu adresu į vertinimo paslaugą, kuri įvertina vartotojo veiksmus su objektu ir grąžina įvertinimą. Kitaip tariant vertinimas iškeliamas iš žinių vertinimo paslaugos ir kiekvienam skirtingam objektų tipui realizuojamas kaip atskira to tipo objektų vertinimo paslauga.

Pabandysime pasinaudoti žinių vertinimo paslauga HTML sintaksės žinių tikrinimui, sudarydami testą iš HTML tipo objektų. Tokio testavimo tikslas yra HTML sintaksės žinių tikrinimas. Vartotojas gaus HTML tekstą ir turės paredaguoti jį taip, kad jis būtu teisingas. Vartotojui pateiktas testas atvaizduos tuos objektus.

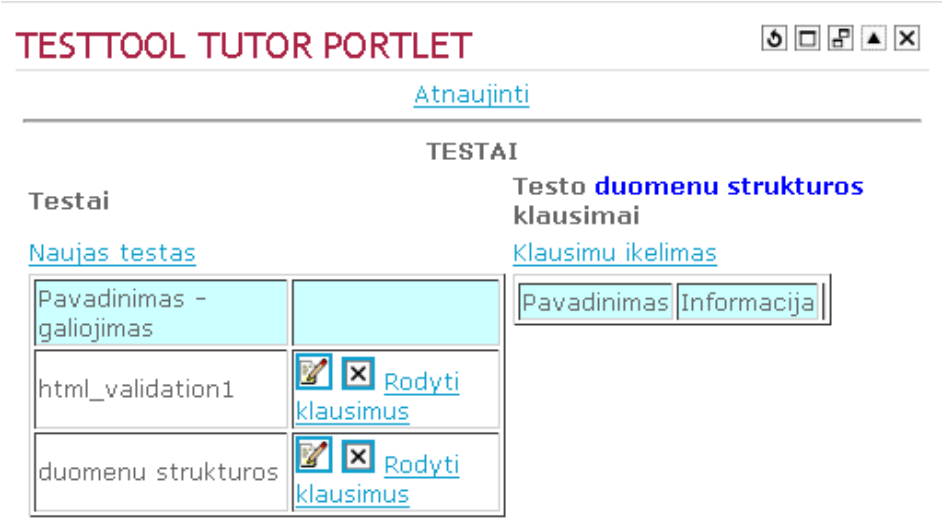
Šiuo atveju HTML vertinimo paslauga bus labai elementari, neteikianti didelių mokymo/si galimybių, nes mums aktualus ne mokymo/si rezultatas ar proceso kokybė, bet sistemos išplėtimo lengvumas. Atviros paslaugų sąsajos apibrėžtos standartais leidžia panaudoti paslaugas kaip atskirus blokus, kuriuos jungiant skirtingu būdu gaunamas skirtingas funkcionalumas.

Pasinaudojant sukurta žinių vertinimo paslauga sukuriame testą, į kurį talpiname jau nebe TestTool, o HTML objektus.

Apsinaudodami žinių vertinimo paslaugos vartotojo sąsaja sukuriame du testus (17 pav.):

html_validavimas1 – į kurį sukeliame html tipo objektus

duomenų_struktūros – į kurį sukeliame TestTool tipo objektus



17 pav. *Žinių vertinimo paslaugos testų administravimo portletas*

Sukurdami testus, nustatome jų gyvavimo laiką, parenkame testavimo režimą bei įvedama papildomą informaciją. Sukūrus testus sukeliame atitinkamus objektus iš duomenų saugyklos į kiekvieną testą.

Norint įsitikinti, kad tikrai sukurti du atskiri žinių vertinimo paslaugos atvejai realizuojantys testus skirtingiems objektų tipams, atliksime Mo&St VO paslaugų indekso užklausą, kuri gražina visas užregistruotas žinių vertinimo paslaugas ir jų resursus.

Kadangi Mo&St virtualioje organizacijoje kol kas nerealizuotas grafinis paslaugų indekso klientas – portretas, pasinaudosime GT4 teikiamu įrankiu wsrf-query, kuris leidžia atlikti paslaugų užklausas. Paslaugų indekso užklausos atsakymas yra XML dokumentas, kuriame surašytos rastos paslaugos, jų adresai, resursai bei metaduomenys.

Čia pateikiami grąžinto užklausos XML dokumento fragmentai, iliustruojantys testų skirtumus.

...
<ns8:Address xmlns:ns8="http://schemas.xmlsoap.org/ws/2004/03/addressing">**http://193.219.159.198:8080/wsrf/services/tt/TestingService**</ns8:Address>
<ns1:TestResourceKey xmlns:ns1="http://elgrid.ktu.lt/services/tt/TestingService">**3EDD7A50-05F9-11DC-84FC-A182314B84BA**</ns1:TestResourceKey>
...
<ns11:AggregatorData>
<ns1:TestName xmlns:ns1="http://elgrid.ktu.lt/services/tt/TestingService">**duomenu_strukturos**</ns1:TestName>
<ns2:TestOwner xmlns:ns2="http://elgrid.ktu.lt/services/tt/TestingService">**owner**</ns2:TestOwner>
<ns3:TestMode xmlns:ns3="http://elgrid.ktu.lt/services/tt/TestingService">**B**</ns3:TestMode>
<ns4:TerminationTime xmlns:ns4="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd">**2007-05-30T14:27:03.774Z**</ns4:TerminationTime>
<ns5:ServiceType xmlns:ns3="http://elgrid.ktu.lt/services/tt/TestingService">**instance**</ns3: ServiceType >
<ns6:LearningSubject xmlns:ns3="http://elgrid.ktu.lt/services/tt/TestingService">**Duomenų struktūrų antras testas**</ns3: LearningSubject >
<ns7:InteractivityLevel xmlns:ns3="http://elgrid.ktu.lt/services/tt/TestingService">**aukštas**</ns3: InteractivityLevel >
<ns8:ObjectType xmlns:ns3="http://elgrid.ktu.lt/services/tt/TestingService">**TestTool**</ns3: ObjectType >

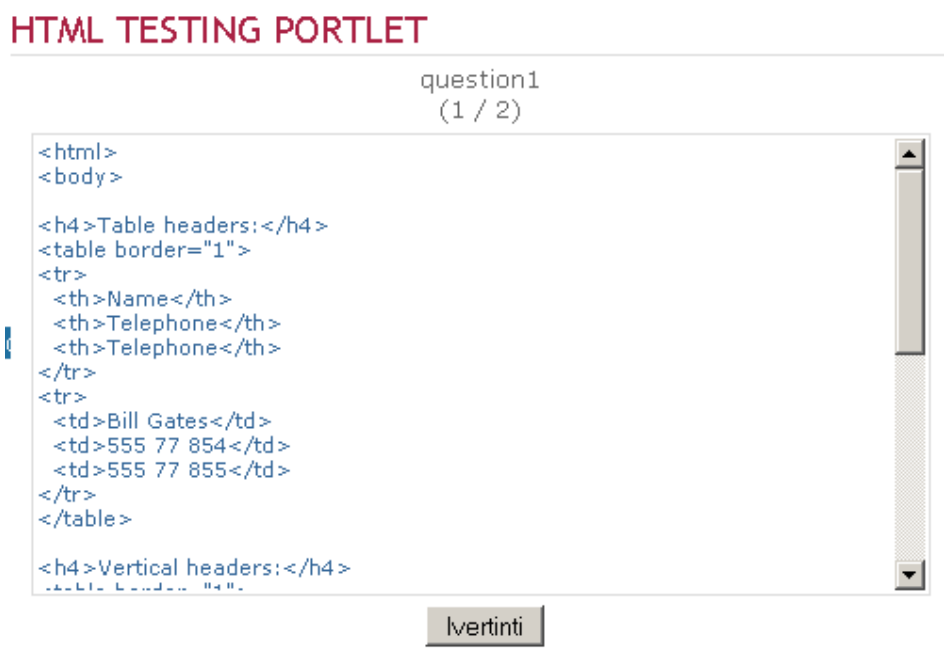
```
  </ns11:AggregatorData>

...
  <ns8:Address
xmlns:ns8="http://schemas.xmlsoap.org/ws/2004/03/addressing">http://193.219.159.198:8080/wsrf/services/tt/TestingSer
vice</ns8:Address>
  <ns1:TestResourceKey xmlns:ns1="http://elgrid.ktu.lt/services/tt/TestingService">BB28EF90-05F9-11DC-84FC-
EFAFF42E1B57</ns1:TestResourceKey>
  </ns9:ReferenceProperties>
...
  <ns11:AggregatorData>
   <ns1:TestName xmlns:ns1="http://elgrid.ktu.lt/services/tt/TestingService">html_validavimas1</ns1:TestName>
   <ns2:TestOwner xmlns:ns2="http://elgrid.ktu.lt/services/tt/TestingService">owner</ns2:TestOwner>
   <ns3:TestMode xmlns:ns3="http://elgrid.ktu.lt/services/tt/TestingService">B</ns3:TestMode>
   <ns4:TerminationTime xmlns:ns4="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-
01.xsd">2007-05-29T14:20:35.775Z</ns4:TerminationTime>
   <ns5:ServiceType xmlns:ns3="http://elgrid.ktu.lt/services/tt/TestingService"> instance </ns3: ServiceType >
   <ns6:LearningSubject xmlns:ns3="http://elgrid.ktu.lt/services/tt/TestingService"> HTML validavimas </ns3:
LearningSubject >
    <ns7:InteractivityLevel xmlns:ns3="http://elgrid.ktu.lt/services/tt/TestingService">žemas</ns3: InteractivityLevel >
    <ns8:ObjectType xmlns:ns3="http://elgrid.ktu.lt/services/tt/TestingService">HTML</ns3: ObjectType >


  </ns11:AggregatorData>
```

Kaip matome paslaugų indekse užregistruojama paslaugos adresas, vardas, savininkas, testavimo režimas, galiojimo laikas, paslaugos tipas, mokomasis dalykas, interaktyvumo lygis bei naudojamų objektų tipas.

Kai yra sukurti du atskiri žinių testavimo apsaugos atvejai galima naudoti juos abu su skirtingais klientais. HTML testo atvejui paslaugų indekse pagal metaduomenis surandamas testas, kurio objektų tipas yra HTML ir jis atvaizduojamas vartotojui (18 pav.):

18 pav. *HTML portleto testų pasirinkimas*

Vartotojas pasirenką režimą – testavimas ar praktika ir tuomet jam atvaizduojamas testo klausimas (19 pav.).



19 pav. *HTML testo klausimo pateikimas*

7. Išvados

1. Šiuo metu elektroniniame mokyme tampa svarbūs tokie uždaviniai kaip pakartotinis mokymo medžiagos panaudojimas, skirtingų sistemų integracija, lankstumas bei mokymo paslaugos komponavimas iš atskirų dalių. Paslaugom orientuota architektūra bei Grid aplinka yra tinkami sprendimai efektyviam ir kokybiškam e-mokymo įgyvendinimui.
2. Nauja paslaugom orientuota architektūra ir Grid technologijos žymiai palengvina skirtingų sistemų sąveiką bei integraciją, suteikia galimybes lengvai išplėsti paslaugos teikiamą funkcionalumą, komponuojant paslaugas kaip atskirus komponentus ir taip pasiekiant galutinį pageidaujamą tikslą ar užduotį.
3. Išanalizavus POA ir Grid technologijas pateiktos rekomendacijos egzistuojančios interaktyvios e-mokymo sistemos pritaikymui Grid POA aplinkai.
4. Darbo metu atliktas eksperimentinis egzistuojančios sistemos TestTool pritaikymas Grid POA aplinkai realizuojant ją kaip žinių vertinimo paslaugą. Sukurta žinių vertinimo paslauga integruota į Mokslo ir Studijų virtualią organizaciją.
5. Naudojant metaduomenimis aprašytus mokymosi objektus bei paslaugas galima efektyviai ir lanksčiai pritaikyti sukurtą žinių vertinimo paslaugą prie poreikių, naudojant skirtingus mokymosi objektų tipus.
6. Pateiktas e-mokymo sistemos pritaikymo Grid POA aplinkai rekomendacijas galima panaudoti ir kitų ne e-mokymo sistemų pritaikymui.
7. Tolimesnis šiame darbe pateiktų rekomendacijų tobulinimas turėtų orientuotis į automatizavimo įrankio sistemų pritaikymui Grid POA aplinkai kūrimą bei semantinių technologijų panaudojimą kuriant mokymosi paslaugas.

Literatūra

[1] Z. Abbas, M. Umer, M. Odeh, R. McClatchey, A. Ali, F. Ahmad. A Semantic Grid-based E-Learning Framework (SELF). 5th IEEE International Symposium on Clust Computing and the Grid Cardiff, Wales, 2005, prieiga internetu [žiūrėta 2007 01 10] http://arxiv.org/ftp/cs/papers/0502/0502051.pdf

[2] M.Agarwal, M.Parashar. Enabling Autonomic Compositions in Grid Environments. 2003, Proceedings of the 4th International Workshop on Grid Computing, Phoenix, AZ.

[3] S. Ambler. Deriving Web services from UML models. IBM, 2005 prieiga internete [žiūrėta 2007 03 25] http://www.ibm.com/developerworks/library/ws-uml1/

[4] M. C. Brown. Build grid applications based on SOA. IBM, 2005 prieiga internete [žiūrėta 2005 11 09] http://www-128.ibm.com/developerworks/grid/library/gr-soa/

[5] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe. The WS-Resource Framework, 2004, prieiga internetu [žiūrėta 2007 01 10] http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf.

[6] T. Erl. Service-Oriented Architecture (SOA): Concepts, Technology, and Design. 2006, Prentice Hall PTR, ISBN-10: 0131858580

[7] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The Physiology of the Grid—An Open Grid Services Architecture for Distributed Systems Integration, 2004, The Globus Alliance, prieiga internetu [žiūrėta 2007 02 10]

http://www.globus.org/research/papers/ogsa.pdf.

[8] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid—Enabling Scalable Virtual Organizations, International Journal of High Performance Computer Applications, 15(3), 200-222 psl., 2001.

[9] I. Foster, S. Tuecke. Describing the elephant: the different faces of IT as service. Queue, Volume 3, Issue 6, 2005, prieiga internetu [žiūrėta 2007 02 10]

http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=319

[10] D. Gannon, J. Alameda, O. Chipara, M. Christie, V. Dukle, L. Fang, M. Farrellee, G. Fox, S. Hampton, G. Kandaswamy, D. Kodeboyina, Ch. Moad, M. Pierce, B. Plale, Al Rossi, Y. Simmhan, A. Sarangi, A.Slominski, S.Shirasuna, T. Thomas. Building Grid Portal Applications From a Web Service Component Architecture. Proceedings of the IEEE, 93(3), 551-563 psl., 2005.

[11] D. Gould. Virtual Organization. 2000, priega internetu [žiūrėta 2007 03 01] http://www.seanet.com/~daveg/Virtual%20Organizing.pdf.

[12] Ch. Yang, H. Ho. An e-Learning Platform Based on Grid Architecture. Journal of Information Science and Engineering, Vol. 21 No. 5, 911-928 psl., 2005

[13] J. Joseph, M. Ernest, and C. Fellenstein. Evolution of grid computing architecture and grid adoption models, IBM Systems Journal Volume 43, Number 4, 2004.

[14] A.Jukniūtė, G. Paulikas. Sands VO: E-learning system architecture for Grid environment". Information technology and control. Vol. 36, No. 1A. 2007, 133-138 psl., 2007

[15] E. Kourpas. Grid Computing: Past, Present and Future, IBM, June 2006.

[16] G. von Laszewski, J. Gawor, S. Krishnan, K. Jackson. Grid Computing: Making the Global Infrastructure a Reality, Commodity Grid Kits - Middleware for Building Grid Computing Environments, Communications Networking and Distributed Systems. Wiley, 639- 656 psl., 2003

[17] V. Pancratius, G. Vossen. Towards e-learning grids: using grid computing in e-learning. IEEE Workshop on Knowledge Grid and Grid Intelligence, 4-15 psl., 2003, prieiga internete [žiūrėta 2006 11 20], www.aifb.uni-karlsruhe.de/BIK/vpa/pankratius_vossen_e-learninggrids.pdf

[18] V. Reklaitis, K. Baniulis, T. Okamoto. Shaping e-Learning Applications for a Service Oriented Grid, Proceedings of 2nd International LEGE-WG Workshop on e-Learning and Grid Technologies : a Fundamental Challenge for Europe, Paris, 2003, prieiga internetu [žiūrėta 2007 02 05], http://www.lege.ktu.lt/pdf/200303_paris.pdf

[19] D. Roure, D., Baker, M., Jennings, N. R. and Shadbolt. The evolution of the Grid. 2003, prieiga internetu [žiūrėta 2007 03 12] http://www.semanticgrid.org/documents/evolution/evolution.pdf

[20] B. Sapkota. Web Service Discovery in Distributed and Heterogeneous Environment. Proceedings of the WWW Service Composition with Semantic Web Services (wscomps05) Workshop, 2005, prieiga internete [žiūrėta 2006 10 16] [http://dip.semanticweb.org/documents/Brahmananda-Web-Service-Discovery-in-Distributed-and-Heterogeneous-Environments.pdf

[21] B. Sotomayor, L. Childers. Globus® Toolkit 4, : Programming Java Servines. Morgan Kaufmann, 2005

[22] J. Unger, M. Haynos. A visual tour of Open Grid Services Architecture, IBM Developwe Works Architecture library, 2005, prieiga internetu [žiūrėta 2007 02 10] http://www-128.ibm.com/developerworks/grid/library/gr-visual/

Terminų ir santrumpų žodynas

CORBA (Common Object Request Broker Architecture)	Standartų sistema koordinuotam kelių programų darbui Internete.
EJB (Enterprise Java beans)	Standartiniai, veikiantys serverio pusėje, daugkartinio panaudojimo Java moduliai.
JAVA	Bendros paskirties, aukšto lygio, objektiškai orientuota, nuo platformos nepriklausoma programavimo kalba sukurta Sun Microsystems firmoje.
GRID	Paskirstyta sistema, kuri leidžia geografiškai nutolusių, autonomiškų resursų dinamišką dalinimąsi, pasirinkimą ir kaupimą, priklausomai nuo jų prieinamumo, galimybių, veikimo, kaštų ir vartotojo reikalavimų paslaugų – servisų kokybei.
HTML (*Hypertext Markup Language)*	Hiperteksto žymėjimo kalba. Tai kompiuterinė žymėjimo kalba, naudojama pateikti turinį internete
HTTP (HyperText Transfer Protocol)	Pagrindinis protokolas pasiekti informaciją pasauliniame tinkle (WWW). Pradinė protokolo paskirtis - pateikti standartinį būdą HTML puslapių skelbimui ir skaitymui.
Mo&St VO	Mokslo ir Studijų virtuali organizacija.
RMI (Remote methid invocation)	Standartų sistema koordinuotam Java kalba parašytų programų darbui Internete
SOAP (Simple Object Access Protocol)	Protokolas XML pranešimų keitimuisi tinkle, dažniausiai naudojant HTTP.
UDDI (Universal Description, Discovery, and Integration)	Nepriklausantis nuo platformas, XML pagrįstas Web servisų registras internete.
UML (Unified modeling language)	Unifikuota modeliavimo kalba.
VO	Virtuali organizacija
Web Service	Žiniatinklio paslaugos, tai programinės įrangos komponentai, sukurti

	tam, kad leistų kurti kintamo dydžio, laisvai susietas, nepriklausančias nuo platformos programas.
WSDL (Web Service DescriptionLanguage)	Žiniatinklio paslaugų sąsajos aprašymo kalba
WWW (World Wide Web)	Pasaulinis interneto tinklas
XML	Išplečiama žymių kalba (angl. eXtensible Markup Language)

PRIEDAS A.

SandS VO: E-Learning System Architecture for Grid Environment

A. Jukniute, G. Paulikas
Kaunas University of Technology, Lithuania
Aiste.Jukniute@ktu.lt, *Giedrius.Paulikas@ktu.lt*

Abstract
Network oriented e-learning systems meet common challenges of modern distributed systems and need a secure and effective way to manage and reuse shared resources in the constantly changing heterogeneous environments. Web service based Service Oriented Architectures and Grid software provide a technological and infrastructural background for advanced realization of the systems supporting dynamic Virtual Organizations (VO). The material of this paper is drawn from the experience gained during the implementation of a prototype for the Science and Studies (SandS) VO system on Globus Toolkit of Grid middleware. The paper presents the architectural framework of SandS VO focusing on security issues and dynamic nature of the distributed learning environment which is illustrated by the first fully implemented service of this e-learning system – the Assessment Service.

1. Introduction

In recent years, e-learning has gained the great popularity as a convenient way of providing teaching methods and sharing knowledge all over the world. With the evolving development of network technologies the amount of information and services available on the Internet continues to rise. No doubt this tendency will last in the future. But currently e-learning systems are reaching their limits due to the limitations in resource sharing, management, scalability and integration in the dynamic environment [1].

At present, Service Oriented Architecture (SOA) is being promoted as the next evolutionary step in software architecture. Traditional architectures provide statically organized content, distributed, but not connected resources; on the contrary, SOA allows sharing of information and resources including distributed resource integration. In fact, SOA is the set of services, which are autonomous, discoverable and communicate among themselves. The communication of services is independent, loosely coupled and is defined using standard description language. SOA and service-orientation are implementation independent paradigms that can by realized using any suitable technology [5]. Today SOA Web services are commonly based on open standards (WSDL, SOAP, UUDI).

Though many organizations have adopted e-learning, the software and hardware facilities they use vary greatly. This causes major difficulties in sharing teaching resources. Web service techniques enable the integration of different information systems within grids and solve this problem [12].

While the Grid is often thought of in terms of providing a distributed system of high-performance compute resources, this is only one aspect of successful use of Grid computing [10]. We can consider distributed computing resources as services, delivered by different organizations, which are used for enriching and improving e-learning environment. Grid technologies can satisfy the needs of effective e-learning system – dynamic resource and user management, resource sharing, integration and discovery, security among different, multi-institutional administrative domains.

With the Globus Toolkit (GT4) move to the Open Grid Standards Infrastructure (OGSI) and finally to Web Services Resource Framework (WSRF), Grid has matured to its latest development stage, adopting a service-oriented approach. WSRF defines conventions for managing state so that applications can reliably share changing information. In combination with WS-Notification and other WS-* standards, the result is to make grid resources accessible within a web service architecture. GT4 for grid computing emerged as a

middleware that addresses the common issues of web service based distributed systems, providing the building blocks for constructing the secure distributed systems for VOs.

2. Virtual Organization

In a broad sense of informatics a **virtual organization** (VO) is defined as one or more organizations interconnected and augmented by means of modern information and communication technologies. The driving force of virtualization includes the rapid change of the market, the availability of advanced technology and the need to increase efficiency. Various dimensions of an organization are apt to virtualize: location, interfaces and boundaries, processes, structures and products/services [7].

The more precise way to describe a VO is achieved by identifying the goals and of connecting the concerned organizations and individuals into the single entity and the means that glue separate organizational pieces together. I. Foster et al. [6] specifies the VO as a set of individuals and/or institutions defined by sharing rules aimed at coordinating the resource sharing and problem solving in dynamic multi-institutional environment. Sharing implies direct access to various computational resources: processing cycles, data, software, devices, etc. This requires the virtualization of accessed resources and full control over the process of providing and consuming them. The rules of sharing are based on the security of communication among partners participating in the VO system. Authentification and successive authorization of subjects engaged in pursuing common goals is essential to control the processes in the VO (scheduling, load management, data replication, information services, etc.).

The broad scale of emerging VOs and heterogeneity of participating entities foster a new wave of technologies for VO implementation. The loosely coupled service based architectures based on standard internet protocols ensure the level of interoperability that is crucial to fulfill these requirements. An equally important component of modern VO is the infrastructure that facilitates the implementation of VO system by supplying common building blocks. Architectural and some technical issues are addressed by SOA, while infrastructural aspects are encountered in **grid** computing. Merging the features of both innovative computing models allows to separate applications from services (SOA) and both applications and services from underlying infrastructure and system resources (grid) [8]. The combined solution empowers the effective implementation of dynamic distributed systems for VOs, improving availability, reliability and scalability of resource utilization when pursuing common VO goals.

3. System Architecture of Science and Studies VO

The current trends of ubiquitous dynamic e-learning VOs leverage the development of open network environments with systems of distributed learning resources to support them. The ability to successfully employ multiple resources of heterogeneous VO members can be ensured by relying on open standards of communication, namely Web services. Other features of such distributed software architectures are usually encountered in enterprise systems and involve high levels of security, control and dynamism of distributed resources.

EU funded research and implementation SPD (Single Programming Document) project at Kaunas University of Technology is aimed at improving the quality of human resources by creating the **Science and Studies** (SandS) VO based on grid technologies. The technological result of this project will consist of the implementation of distributed e-learning system that will be deployed in campus computing nodes and populated with sample learning resources. SandS VO system implementation will benefit both the potential users of the e-learning material of VO and as a reference for future developers of similar distributed environments. As project approaches its midpoint, the design of the system framework is

finished and some pilot learning services are partially implemented. This paper presents the overall architecture of the VO system and elaborates on main challenges met during system design phase.

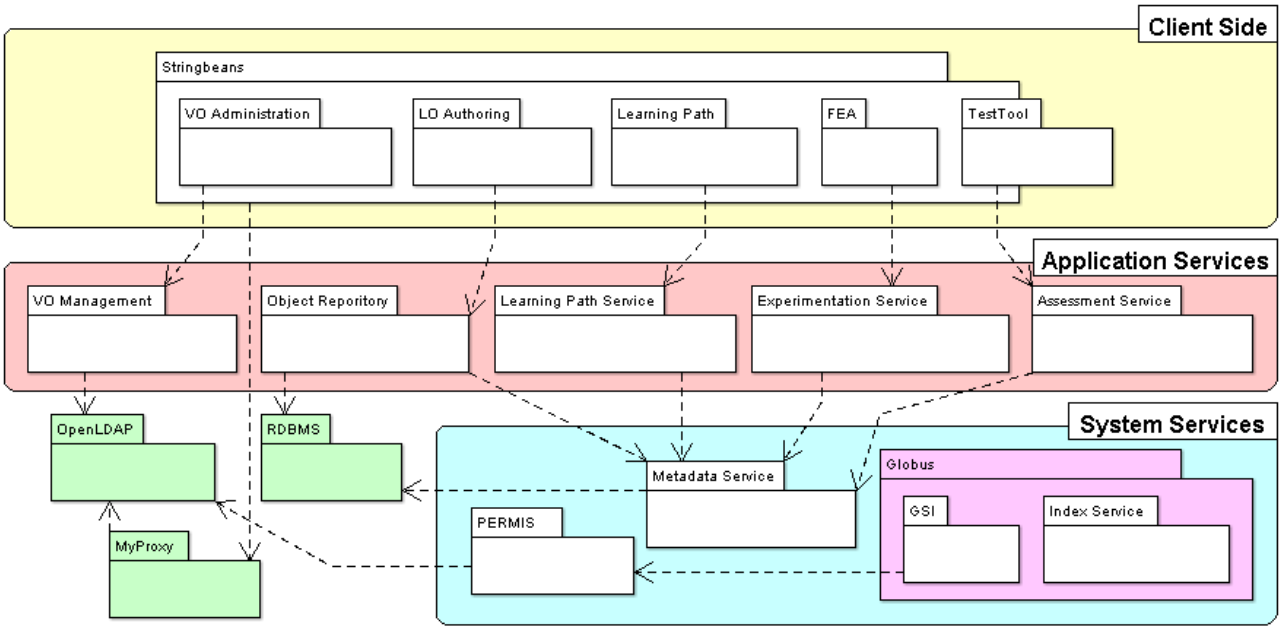


FIGURE 1: SandS VO architecture

The system architecture of the SandS VO encompasses three conceptual layers:

System services and backend software. The core of this distributed system consists of Globus grid middleware and supporting system services. SandS VO uses just a subset of default Globus services, basically **Globus Security Infrastructure** (GSI) [11] and **Index Service**. GSI ensures the required level of security while Index Service provides a service registry that glues the dispersed learning services into solid organization. These systemic Globus services are mandatory for all SandS VO educational services.

The system layer also has two non-Globus services, supplementing the functionality of GSI and Index Service. **PERMIS** authorization service is used to control access to learning resources across VO and is tightly coupled with authentification duties of GSI. **Metadata Service** is developed exclusively for this project and assists in accumulating, publishing and searching of meta information about the static contents (**Learning Objects**, or LO) and services scattered over SandS VO deployment nodes.

System layer backend servers provide the data storage utilities for information about VO members as well as for the data of some application layer services. SQL **relational database managements system** (RDBMS) fulfills the needs of simple and effective storage engine encountered in Metadata Service and other services from the application layer. **OpenLDAP** and **MyProxy** servers keep authentification and authorization information of the VO members in the form of X.509 certificates along with personal profiles of VO users.

Application services expose the educational resources of SandS VO to their users. These services are created from scratch (Learning Path Service), reengineered from existing learning software to make them grid-aware (Assessment Service) or provided as wrappers over proprietary systems (Experimentation Service) to accommodate the needs of the distributed e-learning environment. **Assessment Service** has the most advanced realization stage so it is used as a sample case of grid service implementation further in the article. **Learning Path Service** allows to create the knowledge acquisition scenarios by composing the new services from already available service instances and follow the constructed paths to pursue individual learning goals.

The other two services, **VO Management** and **Object Repository**, support the backbone of SandS VO by providing convenient access to VO member information for administrators and LO storage space for authors of educational material.

- **Client Side.** Considering the demand to enable the easy user access to VO resources from any networked machine the SandS VO project chose a web based portal for its UI representation. Distributed nature of VO system and employment of web service technologies led to the choice of **Stringbeans** JSR-168 portal with WSRP support [4]. The GUI portions of SandS VO e-learning subsystems are implemented as Java portlets that deliver the functionality of the corresponding grid services to the user. In cases like **TestTool** portlet when sophisticated GUI can't be implemented using plain HTML markup the external program is started from the portlet. The separate security scheme of Stringbeans portal requires synchronizing the information of authentification and authorization of client side and system layer security domains (more on this in the following section).

3. SandS VO Security Subsystem

The security framework of SandS VO distributed system is built on GSI that ensures the fundamental aspects of safe communication among members of VO and provides full control of the shared resources. Globus toolkit offers multiple security features:

- **Communication protocol security** guarantees privacy and integrity of the transmitted information and authentification of the parties involved in the communication. Globus gives a freedom of choice between the transport level and message level (SOAP) security schemes. The latter is available in Secure Message or Secure Conversation forms. SandS VO uses Secure Conversation that is more flexible than transport security, more effective than Secure Message and is the only scheme that supports credential delegation.
- X.509 PKI certificates provide the cornerstone of VO member **authentification** and are required for the full-fledged security of communication. As manual management of digital certificates by users themselves has no substantial advantages and even may create a breach in system security, PKI certificates are stored at server side and retrieved on demand. MyProxy credential management service [9] is a central repository of security credentials and ensures the retrieval of user **proxy** certificates that are obligatory for credential delegation and single sign-on to the VO.
- **Authorization** policy of the authentificated VO members enforces the sharing rules of resources that are made available to the VO partners. As most access control decisions are made by services that expose some functionality to VO users, client side authorization must be present only for credential delegation. Server side authorization is the place that concentrates the majority of access control logic to VO resources. Globus, as usual, presents several options for the given task: clients can be authorized by 1) ACL-like (Access Control List) gridmap files, 2) host credentials or 3) SAML-callouts (Security Assertion Markup Language) that delegate authorization decisions to OGSA-Authz compliant service. SandS authorization policy avoids the first two methods as they make access control administration cumbersome and limit the desirable dynamic characteristics of the distributed VO system.

  SAML-callout server side authorization enables to delegate access control to the third-party **PERMIS** service that makes authorization decisions using RBAC (Role Based Access Control) [2, 3]. Besides access resolution at run time PERMIS has the facilities to manage the authorization policies and user privileges. PERMIS authorization is built around the notion of a **role** – a user is granted one or more roles while a service exposes different subsets of its functionality to different roles. Role assignment utilizes the X.509 PMI (Permission Management Infrastructure) attribute certificates that are saved in SandS VO LDAP server. Authorization policies are expressed in XML files that are installed at the authorization decision making engine.

Arbitrary **levels of security** (container, service and resource) allow to choose the required level for SandS VO system by tightening the security measures (e.g. when client side authorization is required for credential delegation) or liberating them (e.g. for operation of system layer Index Service).

The elements of the security subsystem of SandS VO system and application service layers fit together seamlessly as they are derived from a monolithic Globus framework. A different picture is emerging at the boundary between Stringbeans user GUI authorization and authentification and the lower security layers. Stringbeans possesses an autonomous mechanism for user authentification based on internal database, JAAS (Java Authentication and Authorization Service) or LDAP. While LDAP authentification is suitable for overall SandS VO architecture, PERMIS roles are saved as attribute certificates and can't be acquired directly from LDAP server (though their certificates are stored there). This raises the need to develop an interface to service layer authorization data that will be used by Stringbeans portal to extract approximate user permissions to access services. This information is used to construct a portal view for the logged in user. In case the real access to the functionality provided by some service is denied the corresponding portlets should be closed gracefully and that is achieved by direct SAML calls to PERMIS service (see Figure 2).

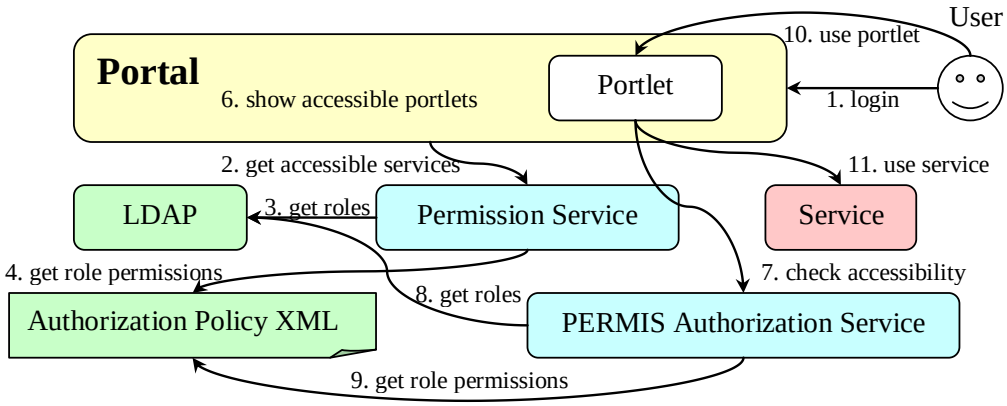


FIGURE 2: Interactions of portal UI and service layer elements for the synchronization of authorization

4. GT4 Based Assessment Service

The Assessment Service (AS) designed for the first SandS prototype is one of the first fully realized VO learning services. Having this service as a sample case for a grid service realization, the paper describes service's architectural view and implementation details.

Following the SandS VO vision - to provide interactive learning services with modeling capabilities, an existing application - TestTool (TT) was used for AS realization. TT is highly interactive assessment system developed in Kaunas University of Technology based on the graphical components. TT was modified and redesigned to make it grid-aware.

Further in this article the architecture and implementation details of AS are presented. Figure 3 shows the inner design of AS, identifying the communicating services and user interaction through VO portal.

When migrating to the new grid version of TestTool, two main architectural modifications were made. First, TT system users and assessment engine were separated in the service implementation relying on global VO user authentication and authorization policies. Second, the idea of test composition from distributed learning objects, hosted in different environments, led us to the solution where TT questions are not stored in AS but rather discovered through Metadata and retrieved from the Object Repository Services. Moreover, such an implementation ensures global access control and better reuse possibilities avoiding the data duplication.

The kernel of Assessment Service actually follows factory pattern. The use of factory pattern enables the dynamic creation of Grid service instances based on the Grid service

description. Testing Factory is always available static service responsible for new Testing Services instances and related resources creation. This approach gives us the possibility to create new tests on demand, considering test as a temporary service, thus having a limited life cycle.

Testing Service is one Web Service implementing TestTool logic, which can be divided in two logical parts – Tutor and Students operations. The state information of Testing Service contains some basic metadata of particular test and is kept separated in the Testing Resource. The exposed state is relevant when publishing created test instance in the global services registry – GT4 Index Service.

The user of service can have a (PERMIS) role of Student, Tutor or Researcher. Researcher is the role which encompasses both Student and Tutor capabilities when using AS. AS communicates with other VO services and in order to be properly authorized by these services, it must perform the request on user's behalf. User credential delegation is performed when the connection is made to other service which requires authorization.

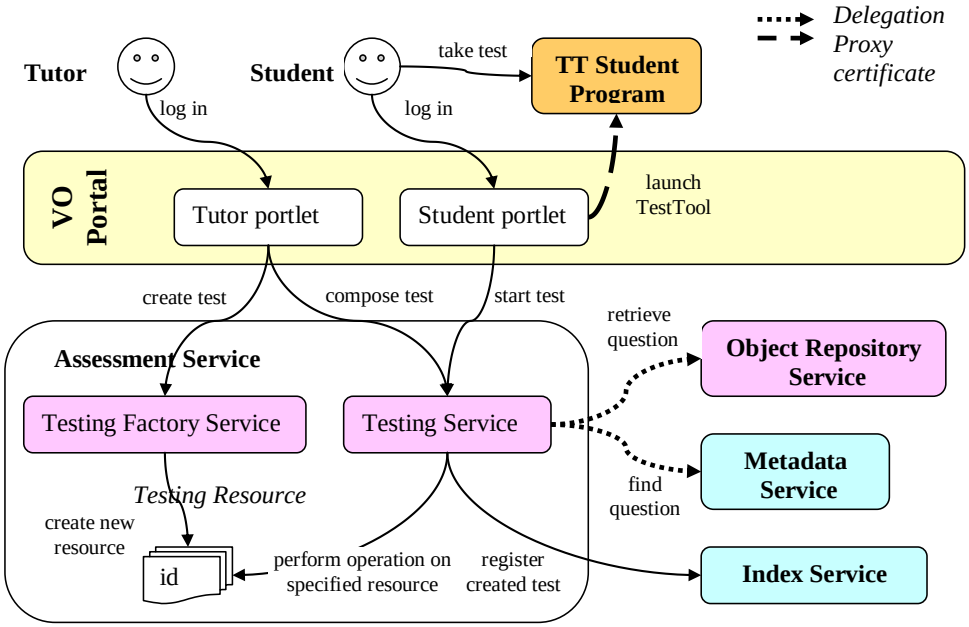


FIGURE 3: Relations between Assessment Service and other VO components

Depending on the user's role, different TT portlets are shown when logged in the portal. While having simple portlet like as a GUI for the service, the authentication and authorization are performed seamlessly as the portal already has the user proxy certificates, obtained at the time user logged in. This is the case with the Tutor portlet. The Student portlet represents much more complex scenario. TestTool graphical environment can't be implemented using HTML that's why the portlet contains the link, which launches external Java application at client's computer. Security is ensured by passing the user credentials from the portal to application at run time using secure connection.

References

1. Z. Abbas., M. Umer., M. Odeh., R. McClatchey , A. Ali., F. Ahmad (2005). A Semantic Grid-based E-Learning Framework (SELF). 5th IEEE International Symposium on Clust Computing and the Grid Cardiff, Wales, UK 9th – 12th May 2005, http://arxiv.org/ftp/cs/papers/0502/0502051.pdf
2. D. W. Chadwick, A. Otenko, E. Ball. Implementing Role Based Access Controls Using X.509 Attribute Certificates – the PERMIS Privilege Management Infrastructure, Security and Privacy in Advanced Networking Technologies, NATO Science Series, pp. 26-39, IOS Press, 2004.
3. D.W. Chadwick, A. Otenko. RBAC Policies in XML for X.509 Based Privilege Management, Security in the Information Society: Visions and Perspectives: IFIP TC11 17th Int. Conf. On Information Security (SEC2002), Cairo, Egypt, pp. 39-53. Kluwer Academic Publishers, May 2002.

4. W. Chokry. Stringbeans™ Installation and Setup Guide, Nabh Information System, 2006, http://www.nabh.com/doc/sb_install_guide.html.
5. T. Erl. Service-Oriented Architecture (SOA): Concepts, Technology, and Design. 2006
6. I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid, International Journal of High Performance Computer Applications, 15(3), pp. 200-222, Fall 2001.
7. D. Gould. Virtual Organization, 2000, http://www.seanet.com/~daveg/Virtual%20Organizing.pdf.
8. E. Kourpas. Grid Computing: Past, Present and Future, IBM, June 2006.
9. J. Novotny, S. Tuecke, V. Welch. An Online Credential Repository for the Grid: MyProxy, Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001, pages 104-111.
10. V. Reklaitis, K. Baniulis, T. Okamoto. Shaping e-Learning Applications for a Service Oriented Grid, Proceedings of 2nd International LEGE-WG Workshop on e-Learning and Grid Technologies : a Fundamental Challenge for Europe, Paris, France - 3rd & 4th March 2003, http://www.lege.ktu.lt/pdf/200303_paris.pdf
11. B. Sotomayor. The Globus Toolkit 4 Programmer's Tutorial, Chapter III: GT4 Security, University of Chicago, 2005, http://gdp.globus.org/gt4-tutorial/multiplehtml/pt03.html.
12. Ch. Yang,. H. Ho. An e-Learning Platform Based on Grid Architecture. 2005, http://www.iis.sinica.edu.tw/JISE/2005/200509_06.pdf.

MirS VO: E-mokymo sistemos architektūra Grid aplinkai

Santrauka

Tinklinės e-mokymo sistemos susiduria su bendrais šiuolaikinių paskirstytų sistemų kūrimo ir eksploatavimo sunkumais, joms reikalingi būdai saugiai ir efektyviai valdyti bei pakartotinai panaudoti bendrus išteklius pastoviai kintančioje nevienalytėje aplinkoje. Žiniatinklio paslaugomis paremtos į Paslaugas Orientuotos Architektūros ir Grid programinė įranga suteikia technologinį ir infrastruktūrinį pagrindą realizuoti pažangias sistemas, palaikančias dinamiškas virtualias organizacijas (VO). Šiame straipsnyje pateikiama medžiaga aprašo patirtį, įgautą kuriant bandomąją Mokslo ir Studijų (MirS) VO realizaciją Globus Toolkit tarpinės programinės įrangos bazėje. Straipsnis pristato bazinius MirS VO architektūros komponentus, daugiausia dėmesio skiriant paskirstytos mokymo sistemos saugumo ir dinamiško panaudojimo aspektams, kurie iliustruojami pirmąja išbaigta šios e-mokymo sistemos paslauga – Žinių Vertinimo Paslauga.