KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

ASTA ŽOLYNAITĖ

WEB SERVISŲ AUTOMATINIŲ TESTŲ GENERAVIMO ALGORITMŲ ANALIZĖ IR TYRIMAS ATLIEKANT MUTACINĮ TESTAVIMĄ

Magistro baigiamasis darbas

Darbo vadovas
lekt. dr. Š. Packevičius

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

ASTA ŽOLYNAITĖ

WEB SERVISŲ AUTOMATINIŲ TESTŲ GENERAVIMO ALGORITMŲ ANALIZĖ IR TYRIMAS ATLIEKANT MUTACINĮ TESTAVIMĄ

Magistro baigiamasis darbas

Darbo vadovas
lekt. dr. Š. Packevičius

Recenzentas
doc. dr. D. Barisas

KAUNAS, 2013

AUTENTIŠKUMO PATVIRTINIMAS

Patvirtinu, kad įteikiamas baigiamasis darbas „WEB servisų automatinių testų generavimo algoritmų analizė ir tyrimas atliekant mutacinį testavimą“:

1. Autoriaus atliktas savarankiškai, jame nėra pateikta kitų autorių medžiagos kaip savos, nenurodant tikrojo šaltinio.
2. Nebuvo to paties autoriaus pristatytas ir gintas kitoje mokymo įstaigoje Lietuvoje ar užsienyje.
3. Nepateikia nuorodų į kitus darbus, jeigu jų medžiaga nėra naudota darbe.
4. Pateikia visą naudotos literatūros sąrašą.

_____

(studento vardas, pavardė) (data) (parašas)

SANTRAUKA

Šiame magistro darbe pateikiama automatinių testų generavimas XML WEB servisams, naudojantis WSDL failais.

Pirma dalis aprašo algoritmų tyrimą ir būdus generuoti automatinius testus XML WEB servisams. Buvo sukurtas papildinys Visual Studio programinės įrangos kūrimo įrankiui. Papildinys, pasinaudojant WSDL specifikacija, analizuoja ir atvaizduoja WEB serviso struktūrą medyje. Vartotojai, naudojantis išanalizuotais duomenimis, gali lengvai įvesti testavimo duomenis ir generuoti automatinius vienetų testus. Taip pat sistema leidžia vartotojui įvesti atsako laiką ir generuoti automatinius našumo testus. Aprašyti papildinio reikalavimai, funkcinė ir nefunkcinė specifikacija bei architektūra.

Tyrimo skyriuje yra aprašomas sukurto įrankio tyrimas. Šioje srityje buvo tiriama darbo su įrankiu efektingumas, atliekant mutacinį testavimą.

SUMMARY

In this master thesis the automated tests generation for XML WEB services by using WSDL files is presented.

First section describes a research of algorithms and ways to generate automated test for XML WEB services. For automated tests generation plug-in for Visual Studio development tool is created. Plug-in takes web service specification analyses and shows WEB service structure in tree. Users by using analyzed data user can easily enter testing data by using developed tool and generate automated unit tests. Also system let user to enter response time and generate performance tests. Created plug-in requirements, functional and non-functional specification, architecture.

In investigation section is described developed plug-in investigation. In this section were investigate the working efficiency of tool by doing mutation testing for generated automated tests.

TURINYS

1. LENTELIŲ SĄRAŠAS

2. PAVEIKSLŲ SĄRAŠAS

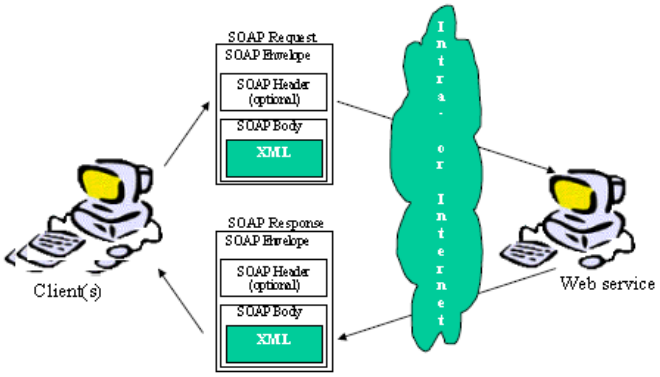
3. TERMINŲ IR SANTRUMPŲ ŽODYNAS

WSDL	– (ang. *Web Services Definition Language*) yra XML pagrindu sukurta kalba apibūdinanti WEB servisus ir kaip juos pasiekti.
SOAP	– (ang. *Simple Object Access Protocol*) yra paprastas XML pagrindu sukurtas protokolas leidžiantis keistis informaciją tarp programų.
XML	– (ang. *Extensible Markup Language*) yra bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba. Pagrindinė XML kalbos paskirtis yra užtikrinti lengvesnį duomenų keitimąsi tarp skirtingo tipo sistemų, dažniausiai sujungtų tinkle.
Juodoji dėžė	– testavimo metodas vidiniai testuojamo elemento veiksmai yra nežinomi ir testavimo atvejai ruošiami pagal specifikaciją.
DOM	– (ang. *Document Object Model*) yra platforma ir neutralios kalbos sąsaja, kuri leidžia programas ir skriptus dinamiškai pasiekti ir atnaujinti dokumento turinį, struktūrą ir stilių.
LAN	– (ang. *Local Area Network*) kompiuterių tinklas jungiantis kompiuterius ribotame plote tokiame kaip mokykla, biuras ir panašiai.
WLAN	– (ang. *Wireless Local Area Network*) jungia du ar daugiau įrenginių, naudojant belaidžio paskirstymo metodą ir paprastai į platesnį interneto ryšį per prieigos tašką.
Regresijos testavimas	– testavimo rūšis, kurios pagrindinė paskirtis – patikrinti, ar naujas kodas išlaiko ankstesnėse versijose veikiantį funkcionalumą, testuojant ne tik pasikeitusias sritis bei modulius, bet visą funkcionalumą.
Vienetų testavimas	– procedūra naudojama parodyti, jog atskiri programinio kodo vienetai veikia teisingai. Vienetas - yra pati mažiausia testuojama taikomosios programos dalis.
Transakcija	– loginis duomenų bazių darbo vienetas, veiksmų duomenų bazėje seka.
DDT	– (ang. Data Driven Testing) terminas, naudojamas kompiuterių programinės įrangos testavimo atvejams aprašyti. Testavimo duomenys (įvestys ir išvestys), nustatymai, instrukcijos yra saugojamos duomenų lentelėje, kad duotų lankstumo.

STS	– servisų testavimo specifikacija.
GUI	– grafikos priemonėmis pagrįsta sąsaja tarp žmogaus ir kompiuterio.
Mutantas	– remiasi gerai apibrėžtais mutacijos operatoriais, kurie imituoja tipines programavimo klaidas (naudojimas blogų operatorių arba kintamųjų).
Mutacinis testavimas	– tai programinės įrangos testavimo metodas, kuris remiasi programos išeities kodo nedideliu modifikavimu.

4. ĮVADAS

Šiuo metu vis labiau ir labiau tobulėja informacinės technologijos bei didėja jos spendimų paklausa. Sistemos tampa vis labiau sudėtingesnės bei į jas stengiama sudėti kuo daugiau funkcionalumo, tai priveda prie to kad neišvengiamai tenka naudoti WEB servisus ir juos testuoti. Augant sudėtingumui, auga ir reikalavimai. Klientai nori stabilios, patikimos ir atitinkančios aukštus reikalavimus, veikiančios sistemos. Tai padidina pasitikėjimą tarp kliento ir paslaugų tiekėjo bei padeda įvertinti daugybė savybių, tokių kaip patvarumas, patikimumas ir panašiai. Didžiausia problema, kad WEB servisas neturi vartotojo sąsajos. Todėl dažniausia, atliekant testavimą, WEB servisų testavimas yra automatizuotas. WEB servisai naudoja XMP ir SOAP protokolą (Pav. 1), kad keistis struktūrizuota informacija.



Pav. 1. WEB serviso veikimo principinė schema. [1]

Pats WEB servisų testavimas yra ganėtinai sunkus, nes WEB servisai yra paskirstytos sistemos su daugybė funkcionalumo ir įtraukia daugybė standartinių protokolų. Didelėse sistemose naudojama, įtraukiama ir sistemos vykdymo metu pajungiama daugybė servisų. WEB servisų vartotojui yra pasiekiamos WSDL specifikacijos, taigi juodosios dėžės testavimas yra galimas, nes besikeičiančių duomenimis WEB servisų projektavimo ir įgyvendinimo detalės nėra pasiekiamos.

Projektas paskirtis yra sukurti testavimo įrankį, kuris palengvins WEB servisų testavimą. Kuriamas įrankis bus naudojamas kaip Visual Studio papildinys. Jis remiantis WSDL servisų specifikacijomis automatiškai sugeneruos automatinius vieneto testus pagal vartotojo įvestus testinius duomenis. Kadangi nėra galimybės ištestuoti WEB servisus naudojantis vartotojo sąsaja, šis testavimo įrankis teiks daug naudos. Taip pat tai palengvins WEB servisų testavimą ir padidins pasitikėjimą tarp kliento ir užsakovo. Šis įrankis apims pagrindinius testavimo metodus tokius kaip funkcijų testavimas, duomenų kontrolės tikrinimas, našumo testavimas. Kadangi

sugeneruoti testai bus saugojami failuose, WEB servisų išgautus testus bus galimybė pernaudoti daugiau nei vieną kartą, todėl galimas regresijos testavimas.

5. PROBLEMINĖS SRITIES ANALIZĖ

5.1. Temos aktualumas ir tikslingumas

Sudėtingumus ir programinės įrangos dydis dramatiškai auga. Dėl šių priežasčių artėjama programinės įrangos sudėtingumo krizės. Tobulėjantys WEB servisai skatina specifikacija pagrįstą bendradarbiavimą ir keitimąsi duomenimis atviros aplinkos programinei įrangai. WEB servisai yra technologija naudojama į paslaugas orientuotoje kompiuterijoje, kuri pasižymi standartais pagrįstu mechanizmu ir atvirų platformų integracija tarp paskirstytų autonominių komponentų. WEB servisų kokybė yra pagrindinė problema, kuriant WEB servisais pagrįstą programinę įrangą, todėl testavimas reikalingas įvertinti funkcionalumo veikimo teisingumą, našumą ir patikimumą tiek individualiems tiek sudėtiniams WEB servisams.

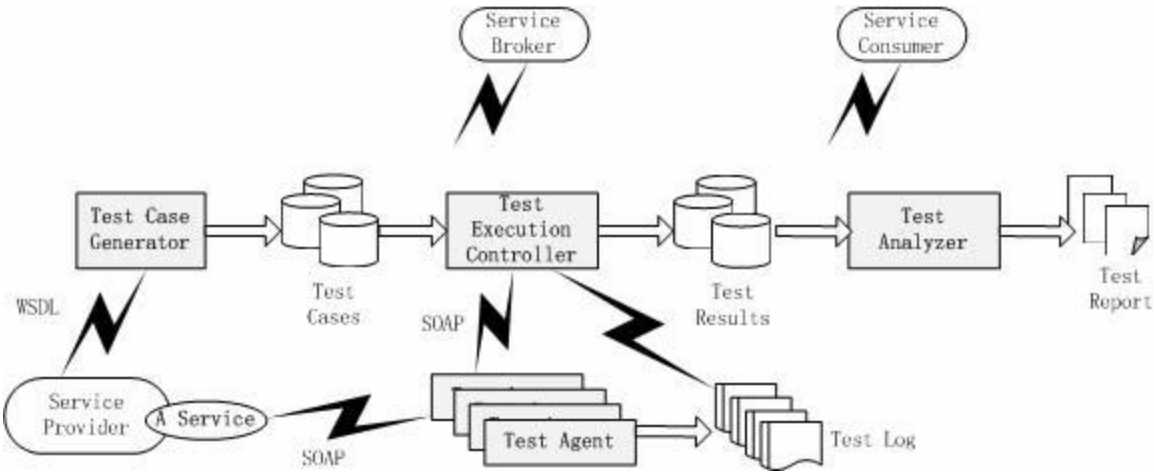
Dėl WEB paslaugų dinaminių savybių išskiriama daugybė naujų iššūkių tradicinėms testavimo technologijoms. WEB servisai yra sąveikaujantys ir integruojantys sistemos veikimo metu. Straipsnio „WSDL – Based Automatic Test Case Generation for Web Service Testing „ autoriai pabrėžė: „Norint užtikrinti WEB servisų kokybę, testavimo atvejai privalo būti generuojami automatiškai ir testavimas vykdomas, stebimas, analizuojamas vykdymo metu“ [2]. WEB servisų testavimo tikslais komunikacijos procesas yra pagrįstas specifikacija, naudojant XML struktūra pagrįsta bendru standartu.

Kitaip nei grafinės vartotojo sąsajos, WEB servisai teikiami per programuojamas sąsajas, kurios yra nematomos vartotojams. Todėl testavimo atvejai turi būti generuojami pagal standartines specifikacijas ir testavimo įrankiams reikia jas analizuoti, kad išgauti reikiamą informaciją tokią kaip: sąsajos operacijos, duomenų struktūra ir operacijų semantika. Kad naudoti ir pernaudoti testavimo atvejus atviroje aplinkoje ir per paslaugų įvertinimą, testavimo atvejai turi būti dokumentuoti naudojant XML pagrįstu standartiniu formatu.

5.2. WEB servisų testavimo proceso automatizavimas

WSDL pagrindu testavimo atvejų generavimas yra dalis iš paskirstytų paslaugų, įeinanti į WEB servisų testavimo procesą, į kurią įeina testavimo atvejų generavimas, testų valdiklis, testavimo agentas ir testavimo vertintojas. Pirma WEB serviso specifikacija paverčiama į DOM struktūros medį, skaitant WSDL failo

duomenis. Testavimo atvejai generuojami iš keturių lygmenų: testavimo duomenų generavimo, individualių testavimo operacijų generavimo, operacijų srautų generavimo ir testų specifikacijos generavimo. Testavimo duomenys yra generuojami, analizuojant XML struktūra pagrįstus failus, išskiriant duomenų tipus. Testai kiekvienai operacijai yra generuojami analizuojant susijusius parametrus. WEB servisas gali susidėti iš daugybė operacijų. Operacijų srautai yra generuojami pagal operacijų priklausomybės analizę. Apibrėžiami trys priklausomybės tipai: įvesties priklausomybė, išvesties priklausomybė, ir įvesties/išvesties priklausomybė. Galų gale, sugeneruoti testavimo atvejai yra įrašomi į XML struktūra pagrįstus testavimo failus, vadinamus servisų testavimo specifikacija.



Pav. 2. Bendrinio WEB servisų automatinio testavimo proceso schema. [2]

Pav. 2 atvaizduoja bendrinę WEB servisų automatinio testavimo proceso struktūrą:

- Testavimo atvejų generatorius (*ang.* Test Case Generator) paima WEB servisų specifikaciją, išsaugotą WSDL failuose, ir automatiškai generuoja testavimo atvejus. Sugeneruoti testavimo atvejai saugojami centrinėje duomenų bazėje.
- Testavimo vykdymo valdiklis (*ang.* Test Execution Controller) valdo testų vykdymą paskirstytose aplinkose. Jis nuskaito testavimo duomenis iš centrinės duomenų bazės, priskiria juos paskirstytiems testavimo agentams, stebi testų vykdymą ir surenka testavimo rezultatus.
- Testavimo agentai (*ang.* Test Agent) yra išskirstyti LAN ir WLAN srityje. Agentas yra tarpinis serveris, kuris atlieka nuotolinį testavimą paskirstytuose WEB servisuose su specifiniais panaudojimo atvejais ir testavimo duomenimis.
- Testų analizuotojas (*ang.* Test Analyzer) analizuoja testų rezultatus,

įvertina paslaugų kokybę ir pateikia testavimo ataskaitas.

Visos dalys, įskaitant paslaugų tiekėją (*ang.* Service Provider), paslaugų agentą (*ang.* Service Broker) ir paslaugų vartotoją (*ang.* Service Consumer) gali prieiti prie duomenų bazės skirtingomis teisėmis.

Ši bendrinė struktūra gali būti taikoma skirtingais lygmenimis, tokiais kaip:

- L1: Individualių atominio WEB serviso operacijų testavimas;
- L2: Kombinacijos atominio WEB serviso operacijų testavimas;
- L3: Individualių operacijų WEB servisų kombinacijos testavimas;
- L4: WEB servisų kombinacijos operacijų kombinacijos testavimas.

Kadangi testavimo atvejų generavimas yra ganėtinai sudėtingas, jis aprašomas sekančiame skyriuje (žr. 5.2.1. Testavimo atvejų generavimas).

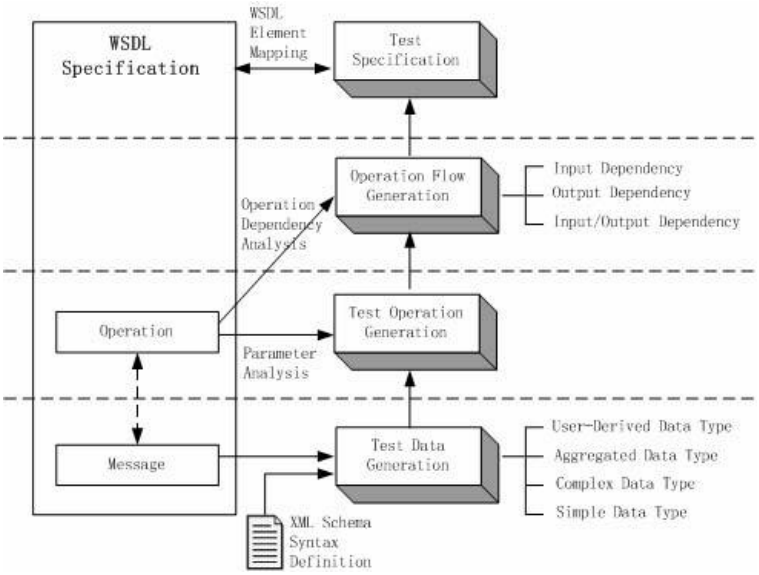
5.2.1. Testavimo atvejų generavimas

Naudojantis WSDL failo struktūra, testavimo atvejai yra generuojami keturiais lygmenimis:

1) Testavimo duomenys generuojami iš XML struktūra aprašytų failų aprašytų duomenų. Yra gvildenami keturi duomenų tipai: paprastas duomenų tipas, aggreguotas duomenų tipas, struktūrinis duomenų tipas ir kliento apibrėžtas duomenų tipas.
2) Testavimo operacijos yra generuojamos, atliekant susijusių parametrų analizę.
3) Operacijų srautai generuojami ištestuoti seką operacijų, atliekant operacijų priklausomybės analizę. Yra apibrėžti trys priklausomybės tipai: įvesties priklausomybė, išvesties priklausomybė, ir įvesties/išvesties priklausomybė.

Testų aprašymo generavimas užkoduoja testavimo atvejus XML failuose. Keliaujant lygmenimis, daugybė testavimo atvejų generuojama, kad ištestuoti WEB servisus, įskaitant individualias operacijas kaip ir operacijų kompoziciją iš neigiamos ir teigiamos perspektyvos.

Testavimo atvejų generavimo proceso schema atvaizduojama Pav. 3.



Pav. 3. Testavimo atvejų generavimas pagal WSDL. [2]

5.2.2. Testavimo duomenų generavimas

Naudojantis serviso specifikacija išgautas turinys yra susiejamas su operacijomis ir analizuojamas, kad sugeneruoti testavimo atvejus. Testavimo atvejai generuojami, atliekant duomenų analizę. XML struktūros sintaksė apibrėžia du duomenų tipus: paprastą ir struktūrinį. Skirtingi duomenų tipai gali būti sujungiami, kad sudaryti atskirą duomenų tipą. Taip pat klientas gali apibrėžti savo duomenų tipą.

Kiekvienas paprastas duomenų tipas yra susiejamas su rinkiniu požymių, kurie charakterizuoja tam tikrus duomenų tipų aspektus. Išskiriama dvejų tipų aspektai: semantiškai esminiai abstrakčių duomenų tipų, tokie kaip ribotumas, kintamųjų tvarka, kardinalumas, vienodumas, ir suvaržyti ir ne esminiai aspektai naudojami, kad apriboti duomenų reikšmes tokiomis savybėmis kaip ilgis, struktūra, išskaičiavimas ir panašiai.

Iš anksto paruoštoje žinių bazėje kiekvienas duomenų tipas yra susiejamas su numatytuoju aspektų apibrėžimu ir rinkiniu pagrįstu testavimo strategijomis, tokiomis kaip atsitiktinės reikšmės ir reikšmių duomenų ribos.

Išanalizavus WSDL failą, analizuotojas išgauna duomenis ir sukuria atvejus pagal jų duomenų tipą, kad gauti ir įrašyti požymių reikšmes su duomenų pavyzdžiais. Kiekvienas požymis yra susiejamas atitinkamu aspektų generuotoju, kad sugeneruoti duomenis pagal išgautas duomenų tipo savybes. Pagrindinio algoritmo žingsniai:

1. Prisijungiama prie centrinės duomenų bazės, kad išgauti duomenų tipų apibrėžimus.
2. Kiekvienam duomenų tipo aspektui, jei jis nėra apibrėžtas atskiras atvejis,

pasiimama numatytoji reikšmė iš duomenų bazės; kitu atveju, pasiimama iš anksto apibrėžtą reikšmę;

3. Taikyti aspektų generatorių sekoje, kad generuoti testavimo duomenis pagal apibrėžtus aspektus.

Struktūriniai duomenų tipai apibrėžia kompoziciją paprasto ir/arba sudėtinio duomenų tipo. Tai apibrėžia masyvą, aibę, „pasirinkimą“, įrašą ir visus santykius tarp duomenų tipų narių. Kad generuoti duomenis struktūriniam duomenų tipui, generuotojas rekursyviai analizuoja duomenų struktūrą kol jis pasiekia paprastą duomenų tipą. Pagrindiniai algoritmo žingsniai:

1. Išanalizuoti struktūrinio duomenų tipo hierarchinio medžio struktūrą.
2. Išanalizuoti medį ir kiekvieną medžio mazgą:

2.1. Jei tai paprastas duomenų tipas, atlikti paprasto duomenų tipo analizę.

2.2. Jei tai sekos struktūros mazgas, generuoti rinkinį testavimo atvejų priklausomai nuo vaikinių mazgų rikiavimo tvarkos.

2.3. Jei tai „pasirinkimo“ struktūros mazgas, generuoti rinkinį testavimo atvejų priklausomai nuo vaikinio mazgo.

2.4. Jei tai bendrinės struktūros mazgas, generuoti testavimo atvejų rinkinį priklausomai nuo atsitiktinės vaikinių mazgų kombinacijos.

Duomenų tipai taip pat gali būti sujungiami, kad sukurti apibendrintų arba rinkiniu duomenų tipų tokių kaip *List* arba *Union*. Panašiai kaip sudėtinių tipų analizei, skirtingi algoritmai yra sukurti, kad generuoti testavimo atvejų rinkinius pagal santykių apribojimus tarp duomenų tipo narių. Paslaugų tiekėjas taip pat apibrėžia savo specifinius paslaugoms duomenų tipus tokius kaip kliento išgautas duomenų tipas, kurie yra apibrėžti iš standartinių duomenų tipų. Kliento apibrėžtas duomenų tipas yra pirma analizuojamas paimant žinias apie standartinio duomenų tipą. Specifinės išvesties savybės yra analizuojamos vykdymo metu atvejis po atvejo.

5.2.3. Operacijų srauto generavimas atliekant priklausomybės analizę

Servisai gali susidėti iš daugybė operacijų. Pagal sugeneruotus testavimo duomenis, daugybė testavimo atvejų gali būtų generuojama operacijai, pritaikant duomenų rinkinio parametrus. Atliekant operacijos priklausomybės analizę, testavimo atvejų rinkiniai taip pat gali būti generuojami, kad ištestuoti sudėtingus operacijų srautus.

WEB serviso viduje operacijos gali būti viena nuo kitos priklausomos. WSDL specifikacija yra nagrinėjama keturiais aspektais: įvesties – išvesties

priklausomybė, vykdymo sekos, vykdymo kartu sekos ir funkcijų hierarchijos aprašymas. Įvesties – išvesties priklausomybė identifikuoja asociacijas tarp sąsajų, kurios gali padėti sumažinti testavimo atvejų skaičių regresijos testavimui. Vykdymo sekos užregistruoja duomenų ir kontrolės srautus tarp servisų kol einamoji seka užfiksuoja sutampančią daugybės servisų transakcijų elgseną. Aprašymų seka gali būti naudojama testavimo keliui rasti. Hierarchinis funkcionalumo aprašymas aprašo sąsajos funkcines specifikacijas hierarchinėj struktūroj, kuri gali palengvinti automatinį funkcionalumo testavimą.

Yra apibrėžti trys tipai priklausomybių: įvesties priklausomybė (*ID*), išvesties/įvesties priklausomybė (*OID*) ir išvesties priklausomybė (*OD*). Dvi operacijos turi įvesties priklausomybę, jei jos dalinasi rinkiniu įvesties žinučių. Dvi operacijos turi išvesties priklausomybę, jei jos dalinasi rinkiniu išvesties žinučių. Mes naudosime *Input (op,)* kad apsibrėžti rinkinį įvesties žinučių operacijos *op* ir *Output* (op_1) rinkinį išvesties žinučių, kad apsibrėžti srautus:

- ✓ **Apibrėžimas 1** Operacijos yra priklausomos pagal įvestį, jeigu $Input(op_1) \cap Input(op_2) \neq \emptyset$, tai op_1 ir op_2 turi įvesties priklausomybę $ID(op_1, op_2)$.
- ✓ **Apibrėžimas 2** Operacijos yra priklausomos pagal išvestį, jeigu $Output(op_1) \cap Output(op_2) \neq \emptyset$, tai op_1 ir op_2 turi išvesties priklausomybę $OD(op_1, op_2)$.
- ✓ **Apibrėžimas 3** Operacijos yra priklausomos pagal įvestį/išvestį, jeigu $Input(op_1) \cap Output(op_2) \neq \emptyset$, tai op_1 yra priklausoma pagal įvestį/išvestį operacijai op_2 $IOD(op_1, op_2)$. op_1 pirminė op_2 operacija ir operacija op_1 yra priklausoma nuo operacijos op_2.

IOD tenkina tranzityvumo savybes ir apibrėžia operacijų seką. Pirma vykdomos operacijos nuo kurių priklauso kitos, o po to vykdoma pirminės. Dėl to, pirma yra atliekamas rekursyvus IOD analizės procesas ir sukuriama priklausomybės diagrama.

Išvesties analizė gali padėti suskirstyti atskiras operacijas į skirtingas grupes. Dvi operacijos, priklausančios viena nuo kitos pagal išvestį, padalinamos į skirtingas grupes. Operacijos skirtingose grupėse yra nesugrupuojamos, ir kiekviena bus toliau suskirstyta į skirtingus testavimo atvejus. Šiuo būdu, tai gali puikiai sumažinti testavimo atvejų skaičių.

Įvesties priklausomybė taip pat naudojama suskirstyti priklausomas operacijas į atskirą grupę. Priklausomos nuo išvesties operacijos yra padalinamos į atskiras grupes, atitinkančias skirtingus testavimo atvejus.

Algoritmas aprašo operacijų priklausomybės operacijų srauto generavimą:

1. Kiekvienai operacijai, atlikti IOD analizę ir identifikuoti rinkinį precedentų PRE ir rinkinį jų priklausomybių DEP.
2. Kiekvienai operacijai, atlikti OD precedentų PRE analizę ir atlikti ID priklausomybės analizę DEP. Atlikus šiuos veiksmus, yra sukuriama rinkinys subgrupių, kurios yra precedentų – PRE={pre_i} ir DEP ={DEP_i} priklausomybės aibės.
3. Surinkti visas operacijas *op*, kurios neturi surinktų PRE precedentų.
4. Jei DEP(*op*) nėra tuščia, sukurti rinkinį testavimo atvejų kai TC[].
5. Kiekvienam DEP_i(*op*) priklausomybėje DEP(*op*), sukurti testavimo atvejį OP[] kaip vektorių operacijų ir pridėti *op* ir kiekvienai operacijai DEP_i(*op*) į masyvą OP[].
6. Kartoti 4-5 kol DEP(*op*) yra išnagrinėtos.
7. Kartoti 3-6 kol visos operacijos neturi neišnagrinėtų PRE.

Gali likti atskirų operacijų neturinčių PRE ir DEP. Šiuo atveju, kol nėra jokios semantinės informacijos, kurią galima būti išgauti iš WSDL specifikacijos, testavimo atvejai generuojami atsitiktinai sujungiant pagrindą konkrečių padengimo kriterijų.

5.2.4. Testų specifikacija

```
<operation-name name="operation">
    <input-name name="input">
        <message-name name="message">
            <part-name    name="part">part-
        value</part-name>
        </message-name>
    </input-name>
    <output-name name="output">
        <message-name name="message">
            <part-name    name="part">part-
        value</part-name>
        </message-name>
    </output-name>
</operation-name>
```

Pav. 4. STS schemos apibrėžimas. [2]

Sugeneruoti testavimo atvejai yra užkoduojami XML struktūra vadinama Servisų Testavimo Specifikacija (STS). STS gali lengvai keistis per paskirstytus testavimo komponentus arba jungiasi prie tinklo protokolų tokių kaip SOAP testų vykdymui. STS paima WSDL failo koncepcijas tokias kaip operacijos, detalės, žinutės, įvestis ir išvestis, atspindint WDSL pobūdžio atvaizdavimo elementus ir

testuoti elementus. Schemos apibrėžimas yra atvaizduojamas Pav. 4.

5.2.5. Testų padengimo analizė

WSDL teikia pagrindą funkcinių testavimo atvejų generavimui ir automatiniam servisų funkciniam testavimui. Testavimo atvejų generavimo algoritmas analizuoja visą specifikaciją ir generuoja testavimo atvejus kiekvienai sąsajos funkcijai apibrėžtai WSDL specifikacijoje. Pagal WSDL schemą, įvairūs padengimo kriterijai gali būti taikomi valdyti ir įvertinti sugeneruotus rezultatus. Testavimo padengimas skirstomas į keturis lygmenis: dalies aprėptis, operacijos padengimas ir operacijų srautų padegimas.

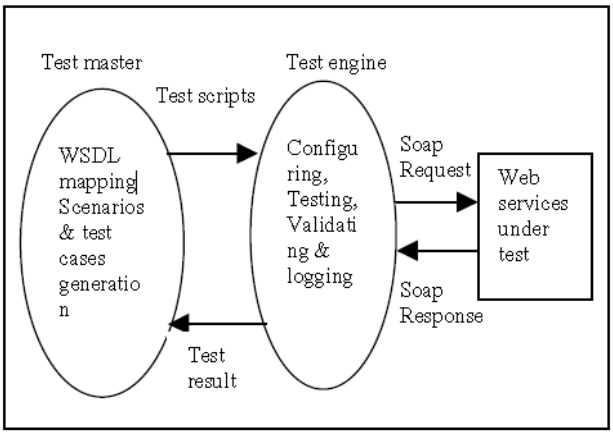
- Dalys yra paskirstytos į operacijos parametrus. Dalinio padengimo kriterijus apibrėžia taisykles testavimo atvejų duomenis kiekvienam parametrui. Pavyzdžiui, kiekviena dalis turi būti padengta bent vienu teigiamu ir neigiamu testavimo atveju arba kiekviena lygiaverčiai klasei dalis duomenų turi būtų padengta mažiausiai vienu testavimo atveju.
- Įvesties žinutė specifikuoja operacijos aprašymą ir įvestis apibrėžia laukiamus testavimo rezultatus. Žinučių padengimo kriterijus reguliuoja testavimo išbaigtumą ryšium su įvesties/išvesties sritimi. Pavyzdžiui, kiekviena įvesties žinutė turi būti padengta bent vienu teigiamu ir neigiamu testavimo atveju ir kiekviena lygiagreti klasė išvesties žinutė turi būti padengta bent vienu testavimo atveju.
- Operacijų padengimas apibrėžia mastu funkcijų kurias galima būtų padengti. Pavyzdžiui, kiekviena operacija turi būti padengta mažiausia vienu teigiamu ir vienu neigiamu testavimo atveju.
- Operacijų srautų padengimas apibrėžia kaip vykdymo keliai turi būti padengti. Pavyzdžiui, pagal priklausomybės analizę, kiekvienas kelias turi būti padengtas mažiausiai vienu teigiamu ir vienu neigiamu testavimo atveju.

Kiekvienai operacijai, testavimo atvejai generuojami pagal šiuos aspektus: duomenų apribojimo analizė ir duomenų apribojimo analizė, atsitiktinių patikimų reikšmių įvertinimas neperžengiant ribų, atsitiktinių nepatikimų reikšmių įvertinimas peržengiant ribas, operacijų priklausomybės analizė, operacijų srautų analizė.

5.3. Egzistuojantys sprendimai

5.3.1. COYOTE

Coyote yra objektinio testavimo karkasas (*ang.* framework) skirta greitai ištestuoti WEB servisus. Coyote susideda iš dvejų dalių: testavimo pagrindo (*ang.* test master) ir testavimo variklio (*ang.* test engine) (Pav. 5). Testavimo pagrindas leidžia testuotojams nurodyti testavimo scenarijus ir atvejus, naudojantis daugybė analizių tokių kaip priklausomybės analizė, išbaigtumas ir panašiai, bei konvertuoti WSDL specifikacijas į testavimo scenarijus. Testavimo variklis sąveikauja su WEB servisu per testą ir teikia vykdymo informaciją. Testavimo struktūra įtraukia pagrindines sąvokas tokias kaip pokyčių struktūrą ir struktūros modelius iš objektinio programų struktūrų, kad santykinai būtų lengva keisti testavimo scenarijus arba atvejus. Coyote palaiko testų vykdymą ir testavimo atvejų ir scenarijų organizavimą. Dėl paskirstytų sistemų pobūdžio, Coyote sutelkia dėmesį į integracijos testavimą vietoj modulių testavimo.



Pav. 5. Coyote struktūra. [3]

Testavimo variklis skaito testų scenarijus, pateiktus testavimo pagrindo modulio, ir vykdo tiksliniam WEB servisui, tai taip pat registruoja vykdymo duomenis ir siunčia testavimo rezultatus atgal testavimo pagrindo moduliui. Kitais žodžiais, testavimo variklis veikia kaip projektavimo struktūros tarpininkas ir tarpinis serveris.

Testų vykdymas įtraukia tris fazes:

- **Konfigūracija**: Nustato prisijungimą prie WEB serviso, konfigūruoja testavimo duomenis su informacija, aprašoma žymoje `<scenario_configuration>`.
- **Testai:** Generuoja SOAP užklausos žinutes, sąveikauja su tam tikru serviso metodu, naudojantis tam tikrais įvesties parametrais apibrėžtais

žymoje <testcases> ir <scenario_path>.

- **Vertinimas ir registravimas:** Tikrina ir įvertina testavimo rezultatus SOAP atsakymo žinutėje, palyginant laukiamas išvestis apibrėžtas testavimo scenarijuose ir registruoja įvertintus rezultatus.

Testavimo bazė gali būti konfigūruojama dviem būdais: WEB servisų palaikančių paskirstytą testavimą, kur skirtingi testavimo varikliai gali bendradarbiauti, kad atlikti vienos užduoties testavimą ir dalintis skirtinga informacija per visą testavimo procesą.

Coyote taip pat teikia palaikymą regresijos testavimui. Kai testai buvo sugeneruoti, regresijos testavimas gali būti atliekamas bet kuriuo metu. Kad visko pertestavimas ir atrankinio testavimo strategijos būtų įtraukiamos į bazę.

5.3.2. QuickCheck

Kaip visi testavimo įrankiai QuickCheck susideda iš testavimo duomenų generuotojo, testų vykdytojo ir analizuotojo. QuickCheck testavimo duomenis generuoja iš WSDL specifikacijos failų. Testavimo atvejų generavimą atlieka atsitiktine tvarka, generuodamas kartu ir testavimo duomenis. QuickCheck atsitiktinai įvertina tam tikro tipo savybes ir registruoja visus rastus skaičiavimus. Šis įrankis automatiškai testuoja servisus naudojantis atsitiktines įvesties reikšmes ir leidžia apibrėžti tam tikrų reikšmių generavimą. „Automated Web – Service Testing with QuickCheck“ straipsnio autoriaus nuomone: “Šios produkto savybės ruošia leidžia ruošti geresnius aprašymus nei automatinis testavimo atvejų generavimas, nes jie padengia bendrinius atvejus su daugiau nei vienu pavyzdžiu“ [4] .

Šis įrankis pateikia daugybė naudingų kombinacijų testavimo metodų, tokių kaip sąlyginės savybės, testavimo atvejų klasifikavimas ir testavimo duomenų surinkimas. QuickCheck generuoja didelį kiekį testų ir lengvai vykdo juos. Su QuickCheck pagalba, mes gali ištestuoti visas įvesties savybes iki duoto gylio.

Kiekvieną kart veikiant QuickCheck yra galimybė, kad bus generuojami nauji testavimo atvejai. Tuo atveju, jei QuickCheck yra pajungiamas, einant laikui mes galėtume tikėtis rasti vis daugiau klaidų. Be to, QuickCheck gali sumažinti nepasisekusių atvejų skaičių, analizuojant gautus rezultatus tam tikru aspektu ir registruojant juos.

5.4. Analizės išvados

Atlikus analizę, buvo išskirtas funkcionalumas ir metodai problemai spręsti.

Buvo apžvelgti egzistuojantys sprendimai ir išskirta jų esminės savybės bei skirtumai.

Abu nagrinėti sprendimai yra testavimo karkasai (*ang.* framework), tačiau šiuo metu yra daug populiarių testavimo karkasų, kuriuos moka daugelis programuoti mokančių testuotojų ar programuotojų. Paminėti sprendimai pačių testavimo atvejų negeneruoja. QuickCheck generuoja automatiškai testavimo duomenis, naudojantis WSDL duomenimis. Iš vienos pusės tai yra gerai, nes kas kart paleidus testus, gausi naujus duomenis, bet yra didelis minusas, nes kartais vienas duomenų rinkinys būna esminis ir su juo gaunamas visai kitas rezultatas. Esamų sprendimų palyginimą galite pamatyti apačioje esančioje lentelėje (žr. Lentelė 1).

Lentelė 1. QuickCheck ir Coyote įrankių palyginimas.

	COYOTE	QuickCheck	Visual Studio papildinys
GUI	-	-	+
Klasių generavimas	-	-	+
Vieneto testų generavimas	-	-	+
Našumo testų generavimas	-	-	+
Testinių duomenų generavimas	-	+	-
Regresijos testavimas	+	+	+
STS	-	+	+

Būtų labai patogu turėti didelio programų kūrimo įrankio papildinį, bet ne naują sistemą. Taip programuotojas ar testuotojas galėtų laisvai ir analizuoti gautus rezultatus, ir naudoti sugeneruotą kodą vienu metu.

Kadangi testavimo metu nėra priėjimo prie kodo, padengimo analizė nebus atliekama.

6. VISUAL STUDIO PAPILDINIO PROJEKTAS

6.1. Sistemos paskirtis

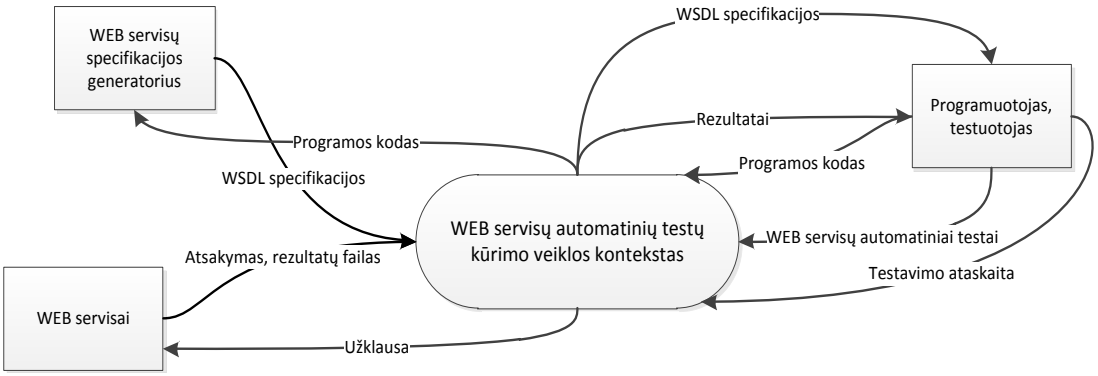
Sistemos paskirtis yra automatizuoti vieneto testų, skirtų WEB servisų testų testavimui, rašymą, generuojant juos iš WSDL failų. Kadangi nėra galimybės pilnai ištestuoti WEB servisus iš vartotojo sąsajos, tai išspręs pagrindinę WEB servisų testavimo problemą. Didžiausi sistemos pliusai yra sutapyti laiko resursai ir palaikoma WEB servisų kokybė. Ji padės generuoti testus pagal norimus kriterijus, duomenis

saugojant pasirinkto formato failuose arba tiesiogiai į programinį kodą. Sugeneruotus testus yra galimybė leisti daug kartų, todėl nereikės vargti pertestuojant servisus per naują, norint užtikrinti ar po pakeitimų neatsirado naujų klaidų.

6.2. Funkciniai reikalavimai

6.2.1. Veiklos kontekstas

Apačioje pateikiama veiklos kontekstinė diagrama (žr. Pav. 6).



Pav. 6. Veiklos kontekstinė diagrama.

6.2.2. Veiklos padalinimas

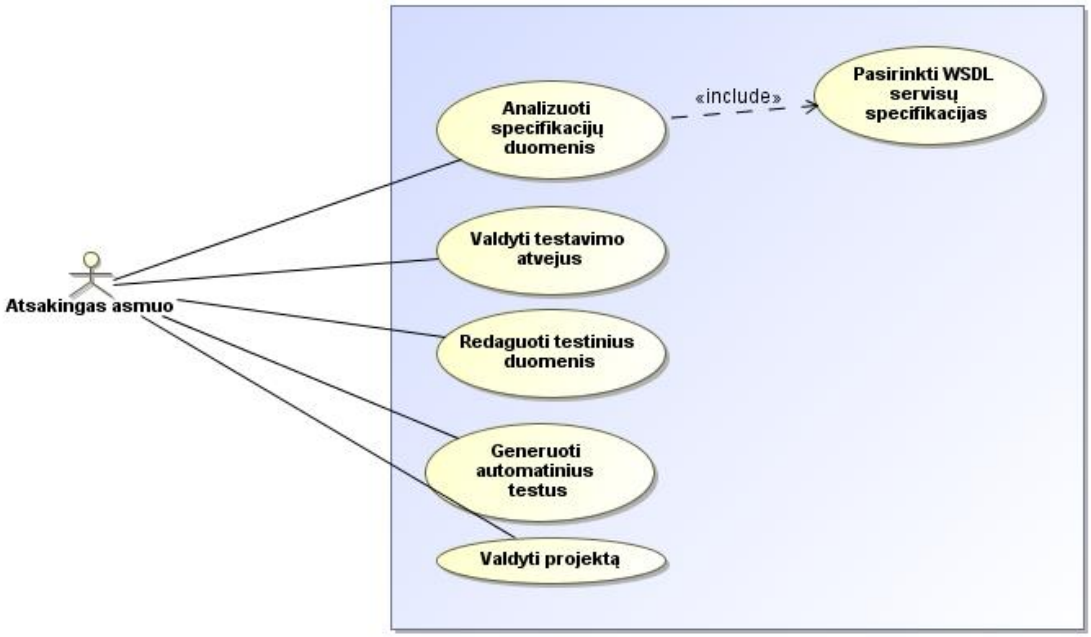
Apačioje esančioje lentelėje aprašom veiklos įvykių sąrašas (žr. Lentelė 2).

Lentelė 2. Veikos įvykių sąrašas.

Eil. nr.	Įvykio pavadinimas	Įeinantys/išeinantys informacijos srautai
1	Programos kodo pateikimas	Programos kodas (*in*)
2	WEB servisų specifikacijos generavimas	Programos kodas (*out*) WSDL specifikacijos (*in*)
3	WEB servisų automatinių testų programavimas	WSDL specifikacijos (*out*) WEB servisų automatiniai testai (*in*)
4	Automatinių testų paleidimas	Užklausa (*out*) Atsakymas, rezultatų failas (*in*)
5	Testavimo ataskaitų parengimas	Rezultatai (*out*) Testavimo ataskaita (*in*)

6.2.3. Panaudojimo atvejų diagrama

Apačioje pateikiama panaudojimo atvejų diagrama (žr. Pav. 7).



Pav. 7. Panaudojimo atvejų diagrama.

Lentelė 3. PA „Analizuoti specifikacijų duomenis“ aprašas.

Eil. nr.	***1***
Panaudos atvejis	***Analizuoti specifikacijų duomenis***
Tikslas	Pateikti WSDL failų analizės rezultatus
Aktoriai	Atsakingas asmuo
Ryšys su kitais PA	Pasirinkti WSDL servisų specifikacijas
Nefunkciniai reikalavimai	Analizės vykdymo laikas vienam failui turi būti ribotas
Prieš-sąlyga	Nėra
Sužadinimo sąlyga	Meniu punkto paspaudimas
Po-sąlyga	Pateikti WSDL specifikacijos analizės rezultatai
Pagrindinis scenarijus	Pasirenkama meniu juostoje vykdyti WSDL failų analizę; Įvedami projekto duomenys; Vykdomas PA „Pasirinkti WSDL servisų specifikacijas“; Patvirtinama WSDL failų analizavimas.
Alternatyvus scenarijus	Neteisingai užpildomi duomenys; Patikrinami suvesti duomenys; Paprašoma vartotojo ištaisyti klaidas.

Lentelė 4. PA „Pasirinkti WSDL specifikacijas“ aprašas.

Eil. nr.	***2***
Panaudos atvejis	***Pasirinkti WSDL specifikacijas***
Tikslas	Pasirinkti vieną arba daugiau WSDL specifikaciją, duomenų analizei
Aktoriai	Atsakingas asmuo
Ryšys su kitais PA	Nėra
Nefunkciniai reikalavimai	Nėra
Prieš-sąlyga	Atidaryta naujo projekto kūrimo forma
Sužadinimo sąlyga	Mygtuko, skirto pasirinkti failus paspaudimas
Po-sąlyga	Pasirinkta WSDL specifikacija
Pagrindinis scenarijus	Paspaudžiama mygtukas, skirtas pasirinkti failus; Pasirenkama failai; Patvirtinama failo pasirinkimas
Alternatyvus scenarijus	Nėra

Lentelė 5. PA „Valdyti testavimo atvejus“ aprašas.

Eil. nr.	***3***
Panaudos atvejis	***Valdyti testavimo atvejus***
Tikslas	Testavimo atvejų valdymas
Aktoriai	Atsakingas asmuo
Ryšys su kitais PA	Nėra
Nefunkciniai reikalavimai	Nėra
Prieš-sąlyga	Atlikta WSDL failo analizė
Sužadinimo sąlyga	Pasirenkama norima operacija
Po-sąlyga	Atlikti veiksmai su testavimo atvejais

Pagrindinis scenarijus	1. Kurti naują testavimo atvejį; 1.1. Pasirinkti norimą operaciją; 1.2. Įvedama duomenys; 1.3. Patvirtinama naujo testavimo atvejo kūrimas; 2. Trinti testavimo atvejį; 2.1. Pasirenkama norimas testavimo atvejis; 2.2. Paspaudžiama mygtukas, skirtas trinti testavimo atvejį; 2.3. Patvirtinama testavimo atvejo trynimas; 3. Redaguoti testavimo atvejį; 3.1. Pasirenkama norimas testavimo atvejis; 3.2. Pakeičiama norimi duomenys; 3.3. Patvirtinama testavimo atvejo duomenų pakeitimai.
Alternatyvus scenarijus	1. Įvedus netinkamus duomenis, išmetama pranešimas apie neteisingų duomenų įvedimą. Leidžiama vartotojui pataisyti klaidas; 2. Atmetus testavimo atvejo trynimą, testavimo atvejis neištrinamas; 3. Įvedus netinkamus duomenis, išmetama pranešimas apie neteisingų duomenų įvedimą. Leidžiama vartotojui pataisyti klaidas.

Lentelė 6. PA „Redaguoti testinius duomenis“ aprašas.

Eil. nr.	***4***
Panaudos atvejis	***Redaguoti testinius duomenis***
Tikslas	Testavimo atvejo duomenų pateikimas
Aktoriai	Atsakingas asmuo
Ryšys su kitais PA	Nėra
Nefunkciniai reikalavimai	Nėra
Prieš-sąlyga	Sukurta testavimo atvejis
Sužadinimo sąlyga	Atsidaroma parametrų redagavimo langas

Po-sąlyga	Pakeisti testavimo atvejo duomenys
Pagrindinis scenarijus	Atsidaromas parametrų redagavimo langas; Suvedami norimi duomenys; Išsaugomi duomenys;
Alternatyvus scenarijus	Nėra

Lentelė 7. PA „Generuoti automatinius testus“ aprašas.

Eil. nr.	***5***
Panaudos atvejis	***Generuoti automatinius testus***
Tikslas	Automatinių testų generavimas
Aktoriai	Atsakingas asmuo
Ryšys su kitais PA	Nėra
Nefunkciniai reikalavimai	Nėra
Prieš-sąlyga	Atlikta WSDL duomenų failo analizė
Sužadinimo sąlyga	Pasirenkama generuoti automatinius testus
Po-sąlyga	Sugeneruota WEB servisų automatiniai testai
Pagrindinis scenarijus	Pasirenkama generuoti automatinius testus; Suvedama norimi duomenys; Patvirtinama automatinių testų generavimas
Alternatyvus scenarijus	Nėra

Lentelė 8. PA „Valdyti projektą“ aprašas.

Eil. nr.	***6***
Panaudos atvejis	***Valdyti projektą***
Tikslas	Projekto duomenų valdymas
Aktoriai	Atsakingas asmuo
Ryšys su kitais PA	Nėra
Nefunkciniai reikalavimai	Nėra
Prieš-sąlyga	Nėra
Sužadinimo sąlyga	Pasirenkama norima operacija
Po-sąlyga	Atlikti norimi veiksmai su projektu

Pagrindinis scenarijus	1. Kurti projektą (žr. į PA nr. 1); 2. Trinti projektą: 2.1. Atidaromas norimas projektas; 2.2. Pasirenkama trinti projektą; 2.3. Patvirtinamas projekto trynimas; 3. Redaguoti projektą: 3.1. Atidaromas norimas projektas; 3.2. Pasirenkama redaguoti projektą; 3.3. Pakeičiama projekto duomenys; 3.4. Išsaugojami duomenys.
Alternatyvus scenarijus	1. Įvedus netinkamus duomenis, išmetamas klaidos pranešimas ir vartotojui leidžiama keisti duomenis; 2. Nepatvirtinus projekto trynimą, projektas neištrinamas; 3. Įvedus netinkamus duomenis, išmetamas klaidos pranešimas ir vartotojui leidžiama keisti duomenis.

6.3. Nefunkciniai reikalavimai

6.3.1. Reikalavimai sistemos išvaizdai

Bendri reikalavimai vartotojo sąsajai:

- ✓ Lengvai skaitoma sąsaja;
- ✓ Paprastas naudojimas;
- ✓ Neryškios spalvos.

6.3.2. Reikalavimai panaudojimui

Bendri reikalavimai panaudojimui;

- ✓ Bendra formų struktūra (pvz.: klaidų pranešimai tuoj pačioj vietoj visose formose);
- ✓ Paprastas naudojimas;
- ✓ Lengvas funkcionalumo perpratimas pažengusiam vartotojui;
- ✓ Anglų kalba.

6.3.3. Reikalavimai vykdymo charakteristikoms

Bendri reikalavimai vykdymo charakteristikoms:

- ✓ Tranksacijų greitis – ne ilgiau kaip 300 ms.

6.3.4. Reikalavimai veikimo sąlygoms

Norint, kad sistema veiktų stabiliai, vertinant sistemos veikimo būtinas charakteristikas reikia pridėti resursus, kuriuos naudoja nuolat veikiančios programos. Būtina prieš diegiant sistemą turėti įsidiegus Visual Studio 2008 ar vėlesnę versiją.

6.3.5. Reikalavimai sistemos peržiūrai

Ateityje planuojama galimybė testavimo atvejus skirstyti pagal testavimo rinkinius, ne pagal operacijas, laiko sąnaudos dar nepamatuotos. Ši dalis planuojama įgyvendinti sistemos tolimesniam vystymo laikotarpiu.

6.3.6. Reikalavimai saugumui

Kadangi sistema nenaudoja interneto, paties vartotojo asmeninių duomenų bei bus diegiama lokaliai pas kiekvieną vartotoją, saugumo reikalavimai nekeliami.

6.3.7. Politiniai reikalavimai

Nėra.

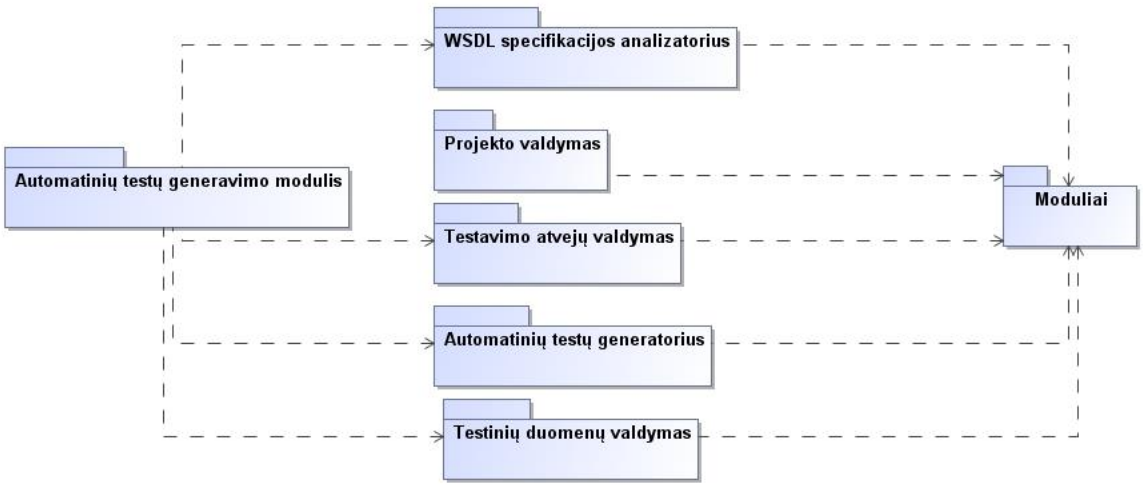
6.3.8. Teisiniai reikalavimai

Nėra.

6.4. Sistemos architektūra

6.4.1. Paketų diagrama

Sistema suskaidyta į penkis paketus aukščiausiame lygyje (žr. Pav. 8) :

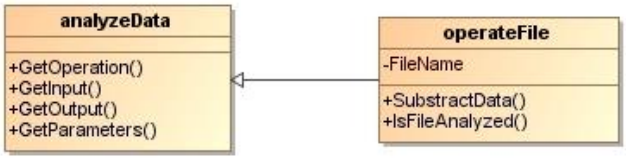


Pav. 8. Paketų diagrama.

6.4.2. Paketų detalizavimas

6.4.2.1. Paketas WSDL specifikacijos analizatorius

Šis paketas yra atsakingas už WSDL failo duomenų analizę ir išgavimą. Paketo diagrama pateikta apačioje (žr. Pav. 9).



Pav. 9. WDSL specifikacijos analizatoriaus paketo diagrama.

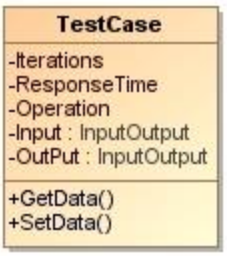
6.4.2.2. Paketas projekto valdymas

Paketas yra atsakingas už projekto duomenų valdymą. Paketo diagrama pateikta apačioje (žr. Pav. 10).

Pav. 10. Projekto valdymo paketo diagrama.

6.4.2.3. Paketas testavimo atvejų valdymas

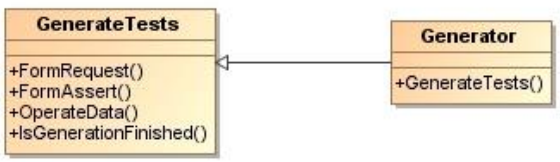
Paketas skirtas valdyti testavimo duomenis, tokius kaip iteracijų skaičius, atsako laikas ir pan. Paketo diagrama pateikta apačioje (žr. Pav. 11).



Pav. 11. Testavimo atvejų valdymo paketo diagrama.

6.4.2.4. Paketas automatinių testų generatorius

Paketo paskirtis generuoti automatinius testus iš pateiktų vartotojo bei analizuotojo duomenų. Paketo diagrama pateikta apačioje (žr. Pav. 12).



Pav. 12. Automatinių testų generatoriaus paketo diagrama.

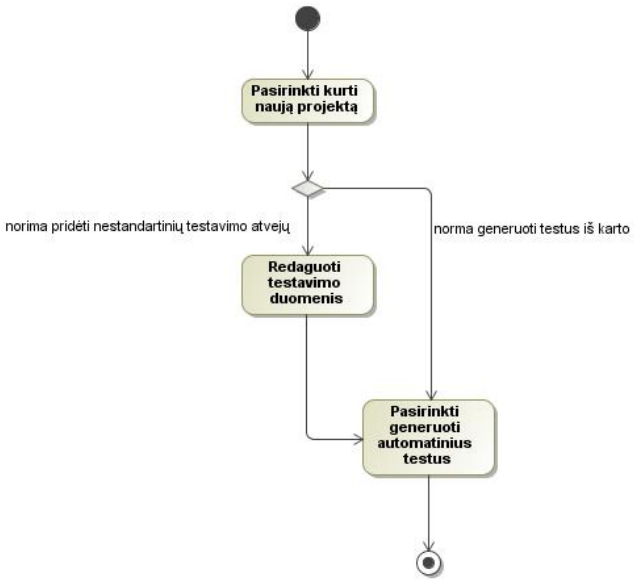
6.4.2.5. Paketas testinių duomenų valdymas

Paketas yra atsakingas už testinių duomenų (užklausos ir atsakymo) duomenų valdymą. Paketo diagrama pateikta apačioje (žr. Pav. 13).

Pav. 13. Testavimo duomenų valdymo paketo diagrama.

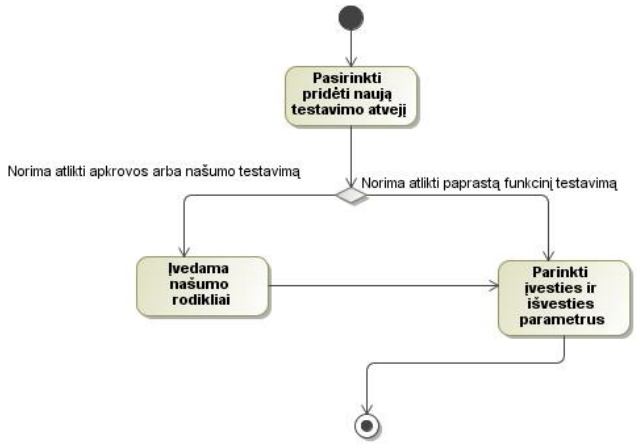
6.4.3. Veiklos diagramos

Veiklos diagramose pateikta bendrinė automatinių testų generavimo sekų diagrama ir detalizuojama „Redaguoti testavimo duomenis“ šaka. Bendrinę testų generavimo veiklos diagramą galite pamatyti apačioje (žr. Pav. 14).



Pav. 14. Bendrinė automatinių testų generavimo veiklos diagrama.

Sekančioje diagramoje yra detalizuojama šaka „Redaguoti testavimo duomenis“. Tai gali būti vykdoma iteracijomis t.y. pridedama tiek testavimo atvejų, kiek norisi iki tol kol pamatome, kad visi testavimo atvejai yra padengti.



Pav. 15. Detalizuota šakos „Redaguoti testavimo atvejus“ veiklos diagrama.

6.5. Atlikti patobulinimai

Šiuo metu sistemoje atlikti patobulinimai:

- ✓ Ištaisytos klaidos;
- ✓ Pertvarkytas kodas, naudojantis Resharper kodo analizės įrankiu;
- ✓ Pakeista rezultato tikrinimo generavimas;
- ✓ Sukurta galimybė generuoti testus ir paruoštiems testavimo atvejams, ir operacijoms (pateikiant žinutes, kurioms nėra priskirta joki parametrai);
- ✓ Sistemoje prie testavimo atvejo įvedus atsako laiką, sugeneruotas testavimo atvejis tikrins ne tik rezultato teisingumą, bet ir ar rezultatas gautas per mažesnį nei tikimąsi maksimaliai.

6.6. Siūlomi patobulinimai

Ateityje būtų naudinga praplėsti sistemą papildomom funkcijom, kurios palengvintų testavimo procesą. Šiuo metu sistema neturi galimybės atlikti operacijų priklausomybės analizę, automatinį testinių duomenų generavimą ir testavimo padengimo analizę. Šios funkcijos būtų labai naudingos, nes padeda suprasti kokios funkcijos eina viena po kitos, sumažinti testavimo atvejų skaičių, gauti detalę padengimo analizę bei kiekvieną kartą nereikia rūpintis dėl įvedamų ir gaunamų duomenų.

Šiuo metu gali būti sukurta testavimo atvejų bei sugeneruota automatinių testų, kurie testuoja tas pačias operacijas. Būtų optimaliau vykdyti tik tuos atvejus, kurie vykdo testavimo operaciją vieną kartą testuojant pagal atitinkamą aspektą.

Padengimo analizė kaip kiekvienam programuojančiam testuotojui gali duoti puikių rezultatų. Analizės duomenys puikiai tiktų sekti statistikai bei matyti atsakingo asmens už WEB servisų testavimą progresą. Kadangi sistema testavimo metu neturi

galimybės prieiti prie programinio kodo, ši dalis nėra įgyvendinta. Būtų galima apsvarstyti alternatyvias galimybes atlikti padengimo analizę.

Kadangi testinių duomenų generavimas yra labai trivialus dalykas, ši funkcija nėra realizuota. Sunku sužinoti koks galimas rezultatas įvedus atitinkamą įvestį. Dažniausia įvestis ir išvestis labai skiriasi, todėl įrankis turėtų atlikti išvesties ir įvesties priklausomybės analizę, paleidžiant funkcijas su atsitiktinėmis įvestimis ir registruojant rezultatus. Iš to gauti priklausomybes bei po to generuoti testinius duomenis, išmetant testinius duomenis, kurie testuoja funkcijas tais pačiais aspektais.

Taip pat galima būtų pritaikyti DDT metodologiją, įrankį pritaikyti paskirstytų servisų testavimui ir testus generuoti dviem aspektais. Dabar sugeneruoti WEB servisų testai veikia, iškviečiant užklausas ir gaunant jų rezultatus, galima servisus testuoti naudojantis nuorodomis (*ang.* reference), kurios padarys kodą lengviau skaitomu ir šiek tiek pasunkins patį kodo generavimą.

7. TYRIMAS ATLIEKANT MUTACINĮ TESTAVIMĄ

Mutacinis testavimas yra puikus būdas ištirti ar sugeneruot testai duoda naudą, kurios tikimąsi. Bus testuojamos du servisai su skirtingomis apimtimis. Automatiniai testai, mutaciniam testavimui, buvo generuojami su sukurtu testavimo įrankiu, suvedus testinius duomenis į sistemą.

Lentelė 9. Kodo metrikos.

	LOC	MI	DIT	Operacijų skaičius	Ciklomatinis sudėtingumas
1.	85	90	3	1	26
2.	168	52	3	25	36

Kaip maitosi pateiktose kodo metrikose, pirmas servisas yra daug mažesnis, bet su sudėtingomis operacijomis, antras turi daug operacijų, bet yra pagal apimtis nesudėtingas (t.y. turintis nesudėtingas operacijas).

Lentelė 10. Metrikų aprašymas.

Metrika	Aprašymas
LOC	Programos kodo eilučių skaičius.

Metrika	Aprašymas
MI	Prižiūrimomo indeksas arba Programų sistemos priežiūrai reikalingų pastangų matas. Prižiūrimumo indeksas gali būti skaičiuojamas tiek visai sistemai (tuomet imamos vidutinės parametrų reikšmės), tiek atskiriems jos moduliams.
DIT	Ilgiausio kelio iki šakninės klasės ilgis. Aukštesnė DIT reikšmė reiškia sudėtingesnę sistemą, kurioje gali būti daugiau klaidų.
Ciklomatinis sudėtingumas	Nepriklausomų kelių programos kodo grafe skaičius. Ciklomatinio sudėtingumo variantas yra **esminis sudėtingumas** – tai ciklomatinis sudėtingumas skaičiuojamas programoje pakeitus visus struktūrizuotus sakinius paprastu sakiniu.

7.1. Pirmas servisas

Primas servisas buvo parsiųstas iš C# Corner puslapio. Jis skirtas meilės procentui apskaičiuoti, pagal partnerių vardus. Servisas nagrinėja abu vardus paraidžiui ir radus balsę, didinamas indeksas. Kiekviena balsė turi savo reikšmę kuria didinamas indeksas. Šis servisas turi daug sąlygų ir palyginus su tuo, kad jis susidaro iš vienos operacijos, yra sudėtingas. Paleidus pirmus testus (prieš pradedant atlikti mutacinį testavimą), buvo ištaisytos kelios klaidos, kurios trukdė vykdyti bet kokį testavimą.

Lentelė 11. Pirmo serviso mutacinio testavimo rezultatai.

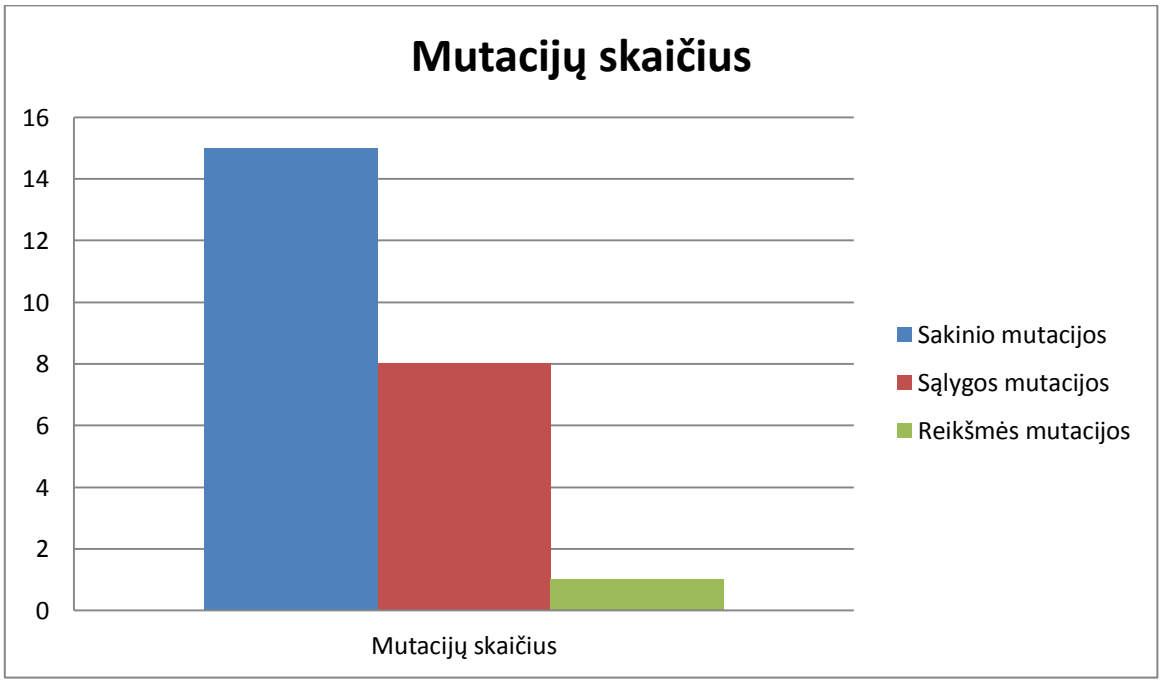
Testai	Testų paruošimo laikas	LOC	Mutantų skaičius	Operacijų sk.	Rasta klaidų
12	20 min	85	24	1	66,67%

Lentelė 12. Pirmo serviso mutacinio testavimo rezultatai pagal mutacijos tipą.

	Sąlygos mutacijos	Sakinio mutacijos	Reikšmės mutacijos
Mutacijų skaičius	8	15	1
Aptikta mutantų	100 %	53,33%	100 %

Iš gautų duomenų po pirmo serviso mutacijos testavimo matosi, kad sąlygos ir reikšmės mutacijos atveju buvo sugauti visi mutantai, bet sakinio mutacijų buvo

sugauta tik apie pusė. Taip buvo todėl, nes testavimo duomenys nebuvo tinkamai parinkti ir kai kurios sąlygos tiesiog nevykdomos bet kokiu atveju t.y. nėra galimybės patekti į atitinkamą sąlygą. Reiktų pakeisti testavimo duomenų rinkinį ir testus paleisti per naują.



Pav. 16. Pirmas servisas. Mutacijų skaičius.

7.2. Antras servisas

Antras servisas yra virtualus mokslinis skaičiuoklis. Jis buvo suprogramuotas sujungus kelis servisus ir pridėjus kelias papildomas funkcijas. Tai nesudėtingas, bet didesnės apimties servisas.

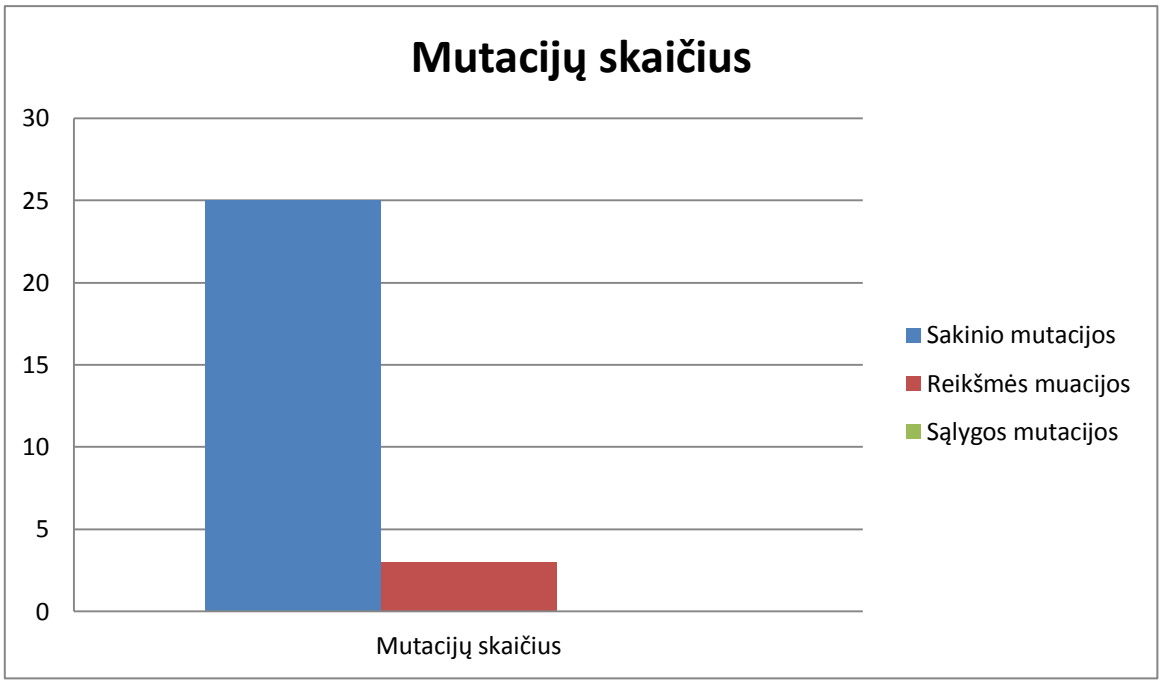
Lentelė 13. Antro serviso mutacinio testavimo rezultatai.

Testai	Testų paruošimo laikas	LOC	Mutantų skaičius	Operacijų sk.	Rasta klaidų
68	70 min	168	28	25	100 %

Lentelė 14. Antro serviso mutacinio testavimo rezultatai pagal mutacijos tipą.

	Sąlygos mutacijos	Sakinio mutacijos	Reikšmės mutacijos
Mutacijų skaičius	0	25	3
Aptikta mutantų	-	100 %	100 %

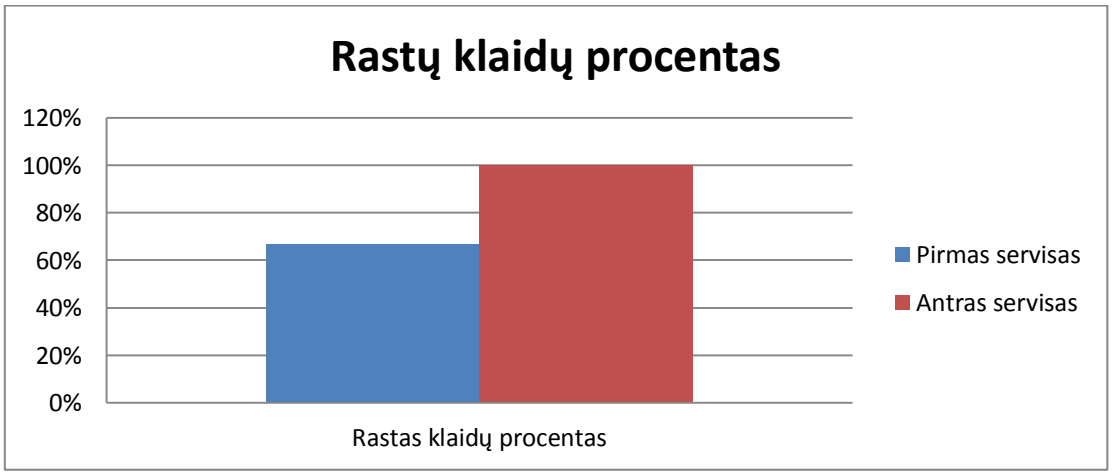
Kaip matosi iš gautų rezultatų, servisui testai buvo sugeneruoti sėkmingai ir rastos visos klaidos. Kadangi servisas turi daug operacijų ir beveik visos gražina apskaičiuotą reikšmę, neatlikdamos jokių papildomų veiksmų, didžioji dalis mutacijų buvo atlikta sakiniams ir tik kelios reikšmėms.



Pav. 17. Antras servisas. Mutacijų skaičius.

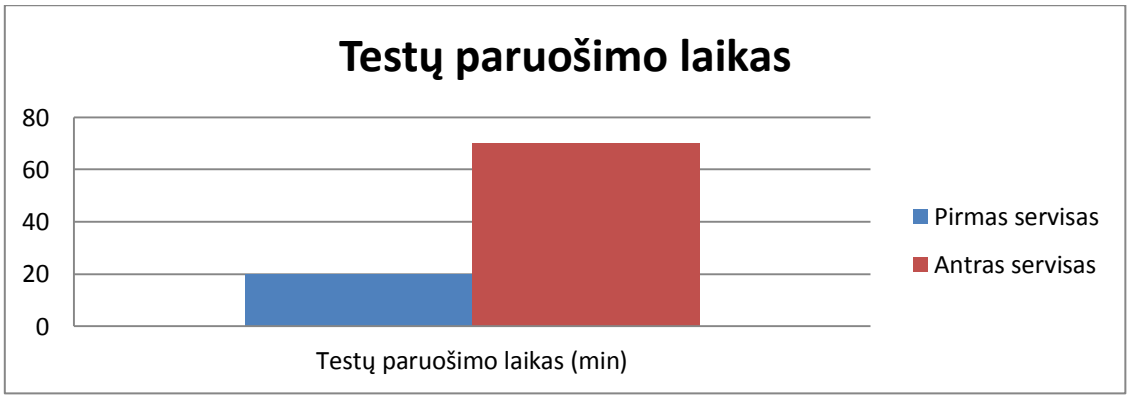
7.3. Eksperimento išvados

Buvo atlikta dviejų skirtingų servisų mutacinis testavimas ir ištirta kaip sugeneruoti testai aptinka klaidas. Testai aptiko 60 – 100 procentų mutantų (žr. Pav. 18). Tyrimo metu buvo nustatyta, kad, naudojantis sukurtu testavimo įrankiu, testavimo duomenys labiausia įtakoja klaidų aptikimą. Kuo didesnis serviso operacijų sudėtingumas, tuo testavimo atvejams yra sunkiau paruošti tinkamus testavimo duomenis.



Pav. 18. Pirmo bei antro serviso rastų klaidų procento palyginimas.

Testų rašymo greitis pasirinktiems servisams yra 20 – 70 minučių. Šis laikas kinta priklausomai nuo WEB serviso apimties ir operacijų sudėtingumo. Paprastai, testų paruošimo laikas trunka ilgiau, nes reikia suprojektuoti automatinių testų architektūrą (pasirinkti metodus kaip bus saugojami, nuskaitomi ir sulyginami duomenys), pasiruošti duomenis ir suprogramuoti testus. Šiems testams tereikėjo tik paruošti testinius duomenis ir suvesti juos į sistemą.



Pav. 19. Testų paruošimo laiko palyginimas.

Šis testavimo įrankis parodė puikius rezultatus kaip testų rašymas sutrumpėjo bei kiek klaidų gali būti rasta.

8. REZULTATŲ APIBENDRINIMAS IR IŠVADOS

1. Atlikta XML WEB servisų generavimo metodų, naudojantis WSDL failais, analizė, išanalizuota generavimo principai.
2. Palyginti bei išanalizuoti konkurenciniai produktai. Pastebėti konkurencinių sistemų privalumai ir trūkumai. Didžiausia sukurto įrankio persvara, lyginant COYOTE ir QuickCheck, yra vartotojo sąsaja ir automatinių testų generavimas pagal suvestus duomenis. Kadangi abu įrankiai yra testavimo bibliotekos arba struktūra (*ang.* framework), automatinių testų architektūrą turi bei testų kodą vartotojas turi ruoštis pats. Testavimo duomenis yra galimybė saugoti XML failuose kaip testavimo specifikaciją. Tokią galimybė turi tik QuickCheck.
3. Apibendrinus nagrinėtus produktus, suprojektuoti ir realizuoti programiniai moduliai, kurie skirti sistemos funkcionalumo, efektyvumo ir našumo pagerinimui.
4. Sistema sukurta kaip Visual Studio 2010 programų kūrimo įrankio papildinys. Dėl įrankio integracijos sugeneruotą automatinių testų kodą galima paleidinėti, analizuoti ir naudoti tame pačiame Visual Studio programų kūrimo įrankyje.
5. Sugeneruoti testai yra pateikti C# programavimo kalba ir naudoja NUnit testavimo biblioteką.
6. Eksperimento parodė, kad testavimo įrankis palengvino servisų automatinį testavimą bei pagreitino testavimo atvejų rašymą. Buvo atlikta dviejų skirtingų servisų mutacinis testavimas ir ištirta kaip sugeneruoti testai aptinka klaidas. Testai aptiko 60 – 100 procentų mutantų. Tyrimo metu buvo nustatyta, kad, naudojantis sukurtu testavimo įrankiu, testavimo duomenys labiausia įtakoja klaidų aptikimą.
7. Testų rašymo greitis pasirinktiems servisams buvo 20 – 70 minučių. Šis laikas kinta priklausomai nuo WEB serviso apimties ir operacijų sudėtingumo. Paprastai, testų paruošimo laikas trunka ilgiau, nes reikia suprojektuoti automatinių testų architektūrą (pasirinkti metodus kaip bus saugojami, nuskaitomi ir sulyginami duomenys), pasiruošti duomenis ir suprogramuoti testus. Šiems testams tereikėjo tik paruošti testinius duomenis ir suvesti juos į sistemą.
8. Testavimo įrankis buvo patobulintas bei ištaisytos esminės klaidos (6.5 skyrius).
9. Sistemos realizaciją galima praplėsti papildomomis funkcijomis bei praplėsti

įrankio galimybes. Pateikta sukurto įrankio praplėtimo galimybė (6.6 skyrelyje).

9. LITERATŪRA

[1] „Testing XML Web Services (Basic),“ 21 02 2002. [Tinkle]. Available: http://www.codeproject.com/KB/webservices/webservicestesting.aspx. [Kreiptasi 01 11 2011].

[2] W. D. W.-T. T. Y. C. Xiaoying Bai, „WSDL – Based Automatic Test Case Generation for Web Service Testing,“ *IEEE,* pp. 215-220, 2005.

[3] R. P. W. S. Z. C. W.T.Tsai, „Coyote: An XML – Based Framework for Web Services Testing,“ *IEEE,* pp. 173-174, 2002.

[4] W. F. W. Z. Y. Zhang, „Automated Web – Service Testing with QuickCheck,“ *IEEE,* pp. 173-185, 2002.

[5] I. Scheiferdecker, B. Stepein, „Automated Testing of XML/SOAP based Web Services,“ [Tinkle]. Available: http://www.site.uottawa.ca/~bernard/TestingWebServices.pdf. [Kreiptasi 07 11 2011].

[6] K. J. K. X. Yi, „A CP-nets-based Design and Verification Framework for Web Services Composition,“ *IEEE,* pp. 756-760, 2004.

[7] X. B. R. P. W. S. V. A. W. T. Tsai, „End-to-End Integration Testing Design,“ *IEEE,* pp. 166-171, 2001.

[8] X. B. R. P. K. F. L. Y. W. T. Tsai, „ Scenario-Based Modeling and Its Applications to Object-Oriented Analysis, Design, and Testing,“ *IEEE,* pp. 140-151, 2002.

[9] R. P. Y. W. C. F. D. W. W. T. Tsai, „Extending WSDL to Facilitate Web Services Testing,“ *IEEE,* pp. 171-172, 2002.

[10] N. Silver, „Web services beyond SOAP,“ 2002. [Tinkle]. Available: http://www.javaworld.com/javaworld/jw-05-2002/jw-0503-jtrix.html. [Kreiptasi 04 11 2011].

[11] S. Nakajima, „Model-Checking Verification for Reliable Web Service,“ 2002. [Tinkle]. Available: http://www.research.ibm.com/people/b/bth/OOWS2002/nakajima.pdf. [Kreiptasi 03 11 2011].

[12] S. N. a. S. Mcllaraith, „Simulation, verification and automated composition of

web services,“ 2002. [Tinkle]. Available: http://www.cs.toronto.edu/kr/papers/nar-mci-www11.pdf. [Kreiptasi 08 11 2011].

[13] W. T. T. R. P. Y. C. H. Huang, „Automated Model Checking and Testing for Composite Web Services,“ *IEEE,* pp. 756-760, 2004.

[14] S. U. ,. J. M. ,. J. K. H. Foster, „ Model – Based verification of web service compositions,“ *IEEE,* pp. 152-161, 2003.

[15] W. T. Tsai, Y. Chen, R. Paul, „Specification – Based Verification and Validation of Web Services and Service – Oriented Operating Systems,“ *IEE,* pp. 139-147, 2005.