KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Giedrius Korsakas

PROGRAMINĖS ĮRANGOS AUTOMATIZUOTO TESTAVIMO ĮRANKIŲ, SKIRTŲ INTERNETINĖMS SISTEMOMS TESTUOTI, GALIMYBIŲ IR PRITAIKYMO TYRIMAS

Magistro darbas

Darbo vadovas: dr. doc. Rita Butkienė

Kaunas, 2007

Turinys

Summary

Growing the number of active internet users and increasing internet popularity force modern companies replace their old internal software systems to new modern internet products. Such software systems enable possibility to use one centralized software platform from remote company offices, decrease communication and valuable data exchange costs. The systems gain the advantage over the competitors.

These systems commonly are very complex and require plenty of expensive support. Most of the systems support is expensive because of the huge number of functionality they provide. Every business logic change in the systems ensuring that any other part of software will not fail needs lots of testing and if there are no automated tests then testing requires the huge amount of human resources which are very expensive. The situation may change if developers of these systems pay more attention to automated software quality assurance process and start implementing automated testing. Having whole system covered with automated tests every business logic change becomes less cost and effective because no expensive human testing is required.

There are two categories of automated quality assurance tools: commercial and free. The commercial tools mostly are very complex and expensive. The free tools mostly are less documented and very complex to use and adopt.

The research will try to answer which automated software quality assurance tools are recommended to use with PHP platform based the “Travel agency Information system” and the other similar products.

1. Įvadas

Tobulėjančios Interneto technologijos ir augantis interneto vartotojų skaičius skatina įmones, jų viduje naudojamas informacines sistemas perkelti į visiems prieinamą internetą, tokiu būdu, supaprastinamas bendravimas tarp nutolusių filialų. Apsikeitimas informacija tampa paprastas, sutaupomas darbuotojų laikas. Tokios sistemos suteikia didelį pranašumą prieš konkurentus.

Kuriamos informacinės sistemos dažniausiai būna sudėtingos ir reikalaujančios nuolatinio palaikymo, priežiūros bei tobulinimo. Daugumos tokių sistemų priežiūra ir tobulinimas yra labai brangus procesas dėl jų sudėtingumo ir funkcijų gausos. Tokiose sistemoje, jų palaikymo etape, kiekvienas atliktas mažiausias kodo pakeitimas gali sąlygoti kitų sistemos dalių nekorektišką funkcionavimą. Problema atsiranda tuomet, kai tokios sistemos būna suprojektuotos nenumatant jų testuoti automatiniu būdu. Kiekvienas sistemos loginis pakeitimas, neturint sistemos automatizuotų testų reiškia, kad visą sistemą reikia testuoti iš naujo nuo pražios ir tai turi atlikti žmogus, o tai užima daug laiko ir reikalauja daug žmoniškųjų resursų. Turint automatinius testus sistemos kiekvieną kartą testuoti iš naujo nereikia, sistemos testavimas atliekamas automatiškai, neįsikišant žmogui.

Yra dvi kategorijos automatinio testavimo įrankių: komerciniai ir nemokami. Komercinės testavimo platformos yra sudėtingos ir brangios. Jų kaina gali viršyti netgi kuriamo produkto kainą. Nemokamų įrankių nėra daug, jie dažniausiai būna mažai dokumentuoti ir sudėtingai pritaikomi kuriamoms sistemoms testuoti.

Tyrimo metu bus siekiama nustatyti, kuriuos automatinio testavimo įrankius pasirinkti ir kaip reikėtų juos panaudoti sukurtai „Kelionių operatoriaus informacinei sistemai“ ir kitiems alternatyviems produktams, sukurtiems naudojant PHP programavimo kalbą.

2. Analitinė dalis

2.1. Sprendžiamos problemos

Sparčiai augant internetinių informacinių sistemų skaičiui ir didėjant jų sudėtingumui vis aktualesne problema tampa šių sistemų kokybės užtikrinimo proceso automatizavimas. Šiai problemai spręsti galima panaudoti automatizuoto kokybės užtikrinimo įrankius, kurie suteikia galimybę automatizuoti kuriamo produkto testavimo procesą. Įrankių, šiai problemai spręsti yra sukurta daug, bet ne visi jie yra tinkami naudoti visais atvejais ir su visomis programavimo kalbomis. Dauguma komercinių įrankių nepalaiko atviro kodo programavimo platformų, tokiu kaip PHP testavimo. Taip pat dauguma jų yra specializuoti ir pritaikyti specifinėms reikmėms, ir specifinėms programavimo kalboms, tokioms kaip Java ir C#. Kadangi nagrinėjama programavimo platforma yra PHP būtina nustatyti, kurie iš jau egzistuojančių testavimo įrankių geriausiai tinka šiai platformai testuoti bei ištirti jų pritaikymo ir panaudojimo galimybes.

2.2. Tiriamojo darbo tikslas

Tiriamojo darbo tikslas yra palyginti ir įvertinti plačiausiai paplitusių automatinio testavimo įrankių galimybes ir pritaikymą probleminei sričiai bei pateikti metodiką kaip juos pritaikyti realių projektų automatizuotam kokybės užtikrinimui. Bandymai bus atliekami su sukurta informacine sistema. Sistema sukurta naudojant „PHP5“ programavimo kalbą, o pats tyrimas yra skirtas šia kalba sukurtų sistemų automatizuotam kokybės užtikrinimui. Nustatytus testavimo panaudojimo metodus bus galima panaudoti ir kitiems projektams, sukurtiems naudojant šią programavimo kalbą, nepriklausomai nuo projekto sudėtingumo ir paskirties.

2.3. Inžinerinis darbo tikslas

Pasirinkti ir pritaikyti įvairius automatizuoto testavimo įrankius sukurtai informacinei sistemai. Nagrinėjami testavimo tipai, kuriuos būtina automatizuoti yra šie:

- Funkcinis/Priėmimo
- Integracinis
- Vienetų

- Greitaveikos
- Streso

Nustatyti testavimo metodai bus pritaikyti ir išbandyti su sukurtos informacinės sistemos moduliais.

2.4. Egzistuojantys automatinio testavimo sprendimai

Pasaulyje egzistuoja ne vienas komercinis ir nekomercinis produktas, kuris siūlo automatizuoti sistemos kokybės užtikrinimo procesą. Šiame skyriuje bus išnagrinėti ir palyginti šiuo metu egzistuojantys populiariausi sprendimai. Sistemos patogumo, dokumentacijos sudėtingumo ir suprantamumo, sistemos sudėtingumo ir kūrėjų palaikymo metrikos bus vertinamos 10 balų sistemoje (0 - labai blogai, 10 puikiai).

2.4.1. Testavimo įrankis „TestComplete“

Lentelėje pateikiamos testavimo produkto „TestComplete“ svarbiausios savybės:

Savybė	Aprašymas
Internetinė svetainė	http://www.automatedqa.com
Palaikomi testavimo tipai	• Funkcinis • Vienetų • Regresinis • Serverio apkrovimo, streso • Integracinis
Palaikomos programavimo kalbos	• .NET • Java • Visual C++ • Visual Basic • Delphi • C++ Builder
Licenzijos kaina	1000 - 7000$
Tinkamumas testuoti internetines	Taip

programas	
Sistemos išbaigtumas, patogumas	**Įvertinimas: 10** iš **10**. Sistema testavimo metu veikė stabiliai, visos bandytos funkcijos veikė be priekaištų.
Dokumentacijos sudėtingumas	**Įvertinimas: 9** iš **10**. Dokumentacija išsami, bet kai kurių dokumentacijos vietų supratimui reikalingos specializuotos žinios.
Kūrėjų palaikymas ir atnaujinimas	**Įvertinimas: 9** iš **10**. Sistemos atnaujinimai ir patobulinimai pateikiami karta per 1-3 mėnesius.
Pateikiama demonstracinė versija	Taip
Sistemos sudėtingumas	**Įvertinimas: 10** iš **10**. Sistema sudėtinga ir funkcionali.
Privalumai	Sistema sudėtinga ir pilnai išbaigta. Yra galimybė valdyti testavimo procesą automatiškai ir informuoti sistemos kūrėjus apie netikėtai atsiradusius defektus. Produktas puikiai dokumentuotas, svetainėje pateikiamas vaizdo įrašas kaip galima panaudoti sistemą.
Trūkumai	Sistema brangi ir nepritaikyta „PHP“ programavimo platformai.
Pastabos	Kadangi yra nagrinėjama PHP programavimo platforma su šiuo įrankiu būtų įmanoma testuoti tik serverio apkrovimo bei atlikti funkcinį sistemos testavimą su iš anksto numatytais duomenimis.

1 lentelė „TestComplete“ charakteristikos

2.4.2. Testavimo įrankis „Netvantage Functional Tester“

Lentelėje pateikiamos testavimo produkto „Netvantage Functional Tester“ svarbiausios savybės:

Savybė	Aprašymas
Internetinė svetainė	http://www.netvantagetech.com
Palaikomi testavimo tipai	Funkcinis
Palaikomos programavimo kalbos	Visos
Licenzijos kaina	170$
Tinkamumas testuoti internetines programas	Taip
Sistemos išbaigtumas, patogumas	**Įvertinimas: 8** iš **10**. Sistema testavimo metu veikė stabiliai, visos bandytos funkcijos veikė be priekaištų. Sistemos

	sąsaja nėra labai intuityvi.
Dokumentacijos sudėtingumas	**Įvertinimas: 5** iš **10**. Dokumentacija išsami, bet sistema ne pilnai dokumentuota.
Kūrėjų palaikymas ir atnaujinimas	**Įvertinimas: 6** iš **10**. Sistemos atnaujinimai ir patobulinimai pateikiami karta per 6 - 12 mėnesius.
Pateikiama demonstracinė versija	Taip
Sistemos sudėtingumas	**Įvertinimas: 7** iš **10**. Sistema nėra sudėtinga ir funkcionali.
Privalumai	Sistema nesudėtinga pilnai išbaigta. Siūloma kurti testus neturint programavimo patirties.
Trūkumai	Pagrindinis trūkumas yra ribotos funkcinės galimybės, palaikomas tik vienas testavimo tipas.
Pastabos	Kadangi yra nagrinėjama PHP programavimo platforma su šiuo įrankiu būtų įmanoma testuoti tik serverio apkrovimo bei atlikti funkcinį sistemos testavimą su išanksto numatytais duomenimis.

2 lentelė „Netvantage Functional Tester“ charakteristikos

2.4.3. Testavimo įrankis „Neoload“

Lentelėje pateikiamos testavimo produkto „Neoload“ svarbiausios savybės:

Savybė	Aprašymas
Internetinė svetainė	http://www.neotys.com
Palaikomi testavimo tipai	• Streso • Apkrovimo
Palaikomos programavimo kalbos	Visos
Licenzijos kaina	1000 iki 29000 Eurų
Tinkamumas testuoti internetines programas	Taip
Sistemos išbaigtumas, patogumas	**Įvertinimas: 10** iš **10**. Sistema testavimo metu veikė stabiliai, visos bandytos funkcijos veikė be priekaištų.
Dokumentacijos sudėtingumas	**Įvertinimas: 10** iš **10**. Dokumentacija išsami.
Kūrėjų palaikymas ir atnaujinimas	**Įvertinimas: 9** iš **10**. Sistemos atnaujinimai ir patobulinimai pateikiami karta per 2 mėnesius.
Pateikiama demonstracinė versija	Taip

Sistemos sudėtingumas	**Įvertinimas: 10** iš **10**. Sistema labai sudėtinga.
Privalumai	Sistema sudėtinga, galima kurti sudėtingus realistiškus elgsenos scenarijus.
Trūkumai	Pagrindinis trūkumas yra ribotos testavimo galimybės, programa skirta tik apkrovimo ir streso testavimui. Kaina neatitinka funkcionalumo.
Pastabos	Produkto licenzijos kaina priklauso nuo virtualių vartotojų skaičiaus ir svyruoja nuo 1000 iki 29000 eurų. Testavimo sistemos paskirtis patikrinti testuojamo produkto elgsena esant dideliam apkrautumui. Veikimo principas paprastas: sistemoje sukuriami vartotojų elgsenos scenarijai, vėliau įrašyti scenarijai yra atkartojami daugelio virtualių vartotoju vienu metu ir stebima testuojamos sistemos elgsena realiu metu. Testams pasibaigus vartotojui pateikiama ataskaita.

3 lentelė „Neoload“ charakteristikos

2.4.4. Testavimo įrankis „Webserver Stress Tool“

Lentelėje pateikiamos testavimo produkto „Webserver Stress Tool“ svarbiausios savybės:

Savybė	Aprašymas
Internetinė svetainė	http://www.paessler.com/
Palaikomi testavimo tipai	• Streso • Apkrovimo
Palaikomos programavimo kalbos	Visos
Licenzijos kaina	200 iki 2000 Eurų
Tinkamumas testuoti internetines programas	Taip
Sistemos išbaigtumas, patogumas	**Įvertinimas: 10** iš **10**. Sistema testavimo metu veikė stabiliai, visos bandytos funkcijos veikė be priekaištų.
Dokumentacijos sudėtingumas	**Įvertinimas: 9** iš **10**. Dokumentacija nėra labai detali.
Kūrėjų palaikymas ir atnaujinimas	**Įvertinimas: 9** iš **10**. Sistemos atnaujinimai ir patobulinimai pateikiami karta per 2-3 mėnesius.

Pateikiama demonstracinė versija	Taip
Sistemos sudėtingumas	**Įvertinimas: 9** iš **10**. Sistema nėra labai sudėtinga.
Privalumai	Sistema sudėtinga, galima kurti sudėtingus realistiškus elgsenos scenarijus. Palaikoma specializuota programavimo kalba scenarijams kurti. Puikiai tinka testuoti sistemos elgsena didelio apkrovimo metu.
Trūkumai	Pagrindinis trūkumas yra ribotos testavimo galimybės, programa skirta tik apkrovimo ir streso testavimui.
Pastabos	Produkto licenzijos kaina priklauso nuo virtualių vartotojų skaičiaus ir svyruoja nuo 200 iki 2000 eurų. Testavimo sistemos paskirtis patikrinti testuojamo produkto elgsena esant dideliam apkrautumui. Sistema sukuria virtualius vartotojus, kurie kiekvienas gali turėti savo elgsenos scenarijų, virtualus vartotojai naudoja testuojamąją sistemą, o testavimo platforma registruoja sistemos būsenas testų metu ir vėliau pasibaigus testui pateikia vartotojui detalią ataskaitą.

4 lentelė „Webserver stress tool“ charakteristikos

2.4.5. Testavimo įrankis „PHPUnit“ ir „JUnit“

Lentelėje pateikiamos testavimo produktų „PHPUnit“ ir JUnit“ svarbiausios savybės:

Savybė	Aprašymas
Internetinė svetainė	http://www.phpunit.de/ http://www.junit.org/
Palaikomi testavimo tipai	• Vienetų testavimas • Integracinis testavimas • Išimčių testavimas (exceptions) • Greitaveikos testavimas • Išvedimo testavimas • Testų generavimas • Kodo padengimo ataskaitos generavimas

	• Integracija su Selenium RC
Palaikomos programavimo kalbos	PHPUnit – PHP, JUnit - Java
Licenzijos kaina	Nemokamos
Tinkamumas testuoti internetines programas	Tinka testuoti progamas parašytas PHP ir Java kalbomis.
Sistemos išbaigtumas, patogumas	**Įvertinimas: 10** iš **10**. Sistema testavimo metu veikė stabiliai, visos bandytos funkcijos veikė be priekaištų.
Dokumentacijos sudėtingumas	**Įvertinimas: 7** iš **10**. Dokumentacija reikalaujanti papildomų techninių žinių.
Kūrėjų palaikymas ir atnaujinimas	**Įvertinimas: 8** iš **10**. Atnaujinimai reguliarūs. Nėra komercinio palaikymo.
Pateikiama demonstracinė versija	Produktas nemokamas, pateikiama pilna versija.
Sistemos sudėtingumas	**Įvertinimas: 10** iš **10**. Sistema sudėtinga ir funkcionali.
Privalumai	Bibliotekos sudėtinga, turinčios daug funkcinių galimybių, bet tuo pačiu ir paprastos naudoti, puikiai tinka parašyto kodo testavimui žemiausiame lygyje. PHPUnit galima pritaikyti testuoti PHP.
Trūkumai	Nėra komercinio palaikymo. Naudotojas pats atsakingas už bibliotekų panaudojimą.
Pastabos	PHPUnit yra PHP kalbos biblioteka skirta būtent šios kalbos testavimui. JUnit turi tokias pat galimybes tik pritaikyta testuoti Java programavimo kalba parašytas programas.

5 lentelė „PHPUnit“ ir „JUnit“ charakteristikos

2.4.6. Testavimo įrankis „SimpleTest“

Lentelėje pateikiamos testavimo produkto „SimpleTest“ svarbiausios savybės:

Savybė	Aprašymas
Internetinė svetainė	http://simpletest.sourceforge.net/
Palaikomi testavimo tipai	• Vienetų testavimas • Integracinis testavimas • Išimčių testavimas (exceptions)

	• Greitaveikos testavimas • Išvedimo testavimas
Palaikomos programavimo kalbos	PHP
Licenzijos kaina	Nemokamos
Tinkamumas testuoti internetines programas	Taip
Sistemos išbaigtumas, patogumas	**Įvertinimas: 10** iš **10**. Sistema testavimo metu veikė stabiliai, visos bandytos funkcijos veikė be priekaištų.
Dokumentacijos sudėtingumas	**Įvertinimas: 7** iš **10**. Dokumentuotas veikimas, reikalingos specializuotos techninės žinios norint viska suprasti ir panaudoti.
Kūrėjų palaikymas ir atnaujinimas	**Įvertinimas: 7** iš **10**. Atnaujinimai nėra dažni ir reguliarūs. Nėra komercinio palaikymo.
Pateikiama demonstracinė versija	Produktas nemokamas, pateikiama pilna versija.
Sistemos sudėtingumas	**Įvertinimas: 10** iš **10**. Sistema sudėtinga ir funkcionali.
Privalumai	Biblioteka sudėtinga, turi daug funkcinių galimybių, paprastos naudoti, puikiai tinka parašyto kodo testavimui žemiausiame lygyje.
Trūkumai	Nėra komercinio palaikymo. Naudotojas pats atsakingas už bibliotekų panaudojimą.
Pastabos	SimpleTest galima naudoti kaip alternatyva PHPUnit. Taip pat kaip ir PHPUNIT ši testavimo platforma yra nemokama ir yra atviro kodo.

6 lentelė „SimpleTest“ charakteristikos

2.4.7. Testavimo įrankis „Selenium RC“

Lentelėje pateikiamos testavimo produkto „Selenium RC“ svarbiausios savybės:

Savybė	Aprašymas
Internetinė svetainė	http://www.openqa.org/selenium/
Palaikomi testavimo tipai	Grafinės aplinkos funkcinis testavimas
Palaikomos programavimo kalbos	Visos
Licenzijos kaina	Nemokamos

Tinkamumas testuoti internetines programas	Taip
Sistemos išbaigtumas, patogumas	**Įvertinimas: 10** iš **10**. Sistema testavimo metu veikė stabiliai, visos bandytos funkcijos veikė be priekaištų.
Dokumentacijos sudėtingumas	**Įvertinimas: 9** iš **10**. Pilnai dokumentuotas veikimas, bet reikalingos papildomos specializuotos techninės žinios norint viska suprasti ir panaudoti.
Kūrėjų palaikymas ir atnaujinimas	**Įvertinimas: 9** iš **10**. Atnaujinimai dažni reguliarūs, produkto funkcionalumas pastoviai plečiamas. Nėra komercinio palaikymo.
Pateikiama demonstracinė versija	Produktas nemokamas, pateikiama pilna versija.
Sistemos sudėtingumas	**Įvertinimas: 10** iš **10**. Sistema sudėtinga ir funkcionali.
Privalumai	Biblioteka sudėtinga, turi daug funkcinių galimybių, paprasta naudoti, puikiai tinka priėmimo funkciniam testavimui.
Trūkumai	Nėra komercinio palaikymo. Naudotojas pats atsakingas už priemonės panaudojimą ir jos veikimo kokybę.
Pastabos	Nemokamas atviro kodo įrankis sukurtas atlikti priėmimo ir funkcinį testavimą. „Selenium“ sistemos pagrindą sudaro „Javascript“, kuris užkraunamas interneto naršyklėje ir valdo testuojamą sistemą. Yra keletas skirtingų „Selenium“ įrankio modifikacijų: „Selenium IDE“, „Selenium RC“. • Selenium IDE veikia kaip Firefox interneto naršyklės įskiepis, testuotojas įrašo veiksmus, kuriuos vėliau sistema atkartoją ir informuoja testuotoją apie testo metu gautus rezultatus. • Selenium RC veikia kaip specializuotas virtualus Java serveris, kuris užkrauna „Selenium“ branduolį į pasirinktą naršyklę ir ją valdo. Java serverį galima valdyti siunčiant jam specialias komandas iš bet kokios visiškai atskiros programos. PHPUnit platforma savyje turi Selenium RC serverio valdymo modulį. Selenium testavimo aplinka veikia įvairiose platformose ir palaiko daugeli naršyklių. Palaikomos naršyklės: Windows OS: • Internet Explorer 6.0 ir 7.0 • Firefox 0.8 iki 2.0 • Mozilla Suite 1.6+, 1.7+ • Seamonkey 1.0

	• Opera 8 & 9 Mac OS X: • Safari 2.0.4+ • Firefox 0.8 iki 2.0 • Camino 1.0a1 • Mozilla Suite 1.6+, 1.7+ • Seamonkey 1.0 Linux: • Firefox 0.8 to 2.0 • Mozilla Suite 1.6+, 1.7+ • Konqueror • Opera 8 & 9

7 lentelė „Selenium RC“ charakteristikos

2.4.8. Testavimo įrankių palyginimas

Pateikiamas nagrinėjamų automatinio testavimo įrankių svarbiausių savybių palyginimas:

	Automatizuoto testavimo sistemų palyginimas							
Testavimo platformų savybės	Savybė	TestComplete	Netvantage Functional Tester	NeoLoad	Webserver Stress Tool	PHPUnit ir JUnit	Simpletest	Selenium
	Palaikomi testavimo tipai							
	Funkcinis	Taip*	Taip	Ne	Ne	Ne	Dalinai	Taip
	Integracinis	Taip*	Ne	Ne	Ne	Taip	Taip	Ne
	Vienetų	Taip*	Ne	Ne	Ne	Taip	Taip	Ne
	Greitaveikos	Taip	Ne	Taip	Taip	Taip	Taip	Ne
	Streso	Taip	Ne	Taip	Taip	Ne	Ne	Ne
	Kitos savybės							
	Tinkamumas testuoti internetines programas	Taip	Taip	Taip	Taip	Taip, labiau tinka kodui	Taip, labiau tinka kodui	Taip
	Sistemos išbaigtumas, patogumas	10/10	8/10	10/10	10/10	10/10	10/10	10/10
	Dokumentacijos sudėtingumas	9/10	5/10	10/10	9/10	7/10	7/10	9/10

	Kūrėjų palaikymas ir atnaujinimas	9/10	6/10	9/10	9/10	8/10	7/10	9/10
	Pateikiama demonstracinė versija	Taip	Taip	Taip	Taip	Pilna versija	Pilna versija	Pilna versija
	Sistemos sudėtingumas	10/10	7/10	10/10	9/10	10/10	10/10	10/10
	Bentras įvertinimas	**9,5**	**6,5**	**9,7**	**9,25**	**8,75**	**8,5**	**9,5**
	Kaina	7000$	170$	1000 iki 29000 Eurų	200 iki 2000 Eurų	0	0	0

8 lentelė Testavimo platformų charakteristikų palyginimas

* Netinka nagrinėjamai informacinei sistemai, nes nepalaiko PHP programavimo kalbos.

Atlikus detalesnę kiekvieno šių įrankių analizę paaiškėjo, kad sukurtai informacinei sistemai testuoti žemiausiu lygiu (vienetų ir integraciniai testai) egzistuoja dvi priemonės: „PHPUnit“ ir „SimpleTest“ bibliotekos. Abi jos turi panašias automatizavimo testavimo galimybes ir yra atviro kodo testavimo platformos. PHPUnit turi keletą pranašumų prieš SimpleTest:

- Kodo padengimo testais ataskaitų generavimas;
- Selenium RC integruotas palaikymas;
- Dažnas atnaujinimas;
- Stabili versija.

PHPUnit sugeba sugeneruoti kodo padengimo testais ataskaitą ir turi integruotą bendravimo su Selenium RC serveriu modulį. Bibliotekos atnaujinta versija pasirodo keletą kartų per mėnesį, jos kūrėjai ją pastoviai papildo naujomis galimybėmis, palengvinančiomis testavimo procesą. Rinkoje yra stabili testavimo sistemos versija.

Šiuo metu rinkoje nėra nei vieno, išskyrus Selenium RC, nemokamo ir stabiliai veikiančio produkto, kuris būtų skirtas automatiniam priėmimo/funkciniam sistemos testavimui ir būtų nesunkiai integruojamas ir valdomas panaudojant PHP kalbos galimybes. Dėl šios priežasties funkciniam/priėmimo testavimui geriausia būtu pasirinkti Selenium įrankį.

Sistemos streso testavimui rekomenduojama pasirinkti „Webserver Stress Tool“, nes ši sistema yra pigesnė nei konkurentų, funkcionali bei lengvai panaudojama. Ši sistema, baigus testavimo procesą, pateikia išsamią ataskaitą. Iš pateiktos ataskaitos nesunkiai galima spręsti, kurias testuojamos sistemos vietas reikia optimizuoti, koks didžiausias lygiagrečių vartotojų skaičius gali dirbti su sistema, kaip kito sistemos resursų naudojimas didėjant lygiagrečių vartotojų skaičiui ir kita naudinga informacija.

2.5. Išvados

Atlikta detali automatizuoto testavimo įrankių analizė parodė, kad geriausia priemonė funkciniam/priėmimo testavimui yra Selenium RC, vienetų ir integraciniams testams rašyti labiausiai tinka PHPUnit įrankis. Šie testavimo įrankiai yra nemokami, funkcionalūs ir pilnai galintys užtikrinti tęstinį automatinį sistemų, sukurtų naudojant PHP programavimo kalbą, testavimą. Pagrindinė problema yra daug specializuotų žinių reikalaujantis ir sudėtingas šių įrankių panaudojimas, apjungimas į vieną visumą ir pritaikymas testuoti konkrečią sistemą.

Streso testavimui galima pasirinkti „Webserver Stress Tool“, nes ši sistema yra pigesnė, lengvai panaudojama ir funkcijų bei pateikiamų ataskaitų gausa nenusileidžianti brangesniems konkurentų produktams. Šios priemonės dažnai naudoti nėra būtinybės, dėl pačio testavimo specifikos. Kadangi „Webserver stress tool“ yra komercinis įrankis, turintis grafinę aplinką ir detalų vartotojo vadovą, šios priemonės panaudojimo ir pritaikymo galimybės nebus nagrinėjamos.

3. Automatizuoto testavimo įrankių PHPunit ir Selenium RC panaudojimo metodika

3.1. Nagrinėjama sritis ir metodikos paskirtis

Šiame skyriuje detaliai pateikiama metodika, kuria rekomenduojama remtis, norint automatizuoti kuriamos sistemos testavimo procesą naudojantis „PHPUnit“ ir „SELENIUM RC“ priemonių rinkiniu. „Selenium RC“ įrankiu bus atliekamas sistemos funkcinis testavimas, panaudojant Microsoft Internet Explorer naršyklę. Vienetų ir integracinis testavimas bus atliekamas naudojantis „PHPUnit“ priemonių rinkiniu, taip pat „PHPUnit“ bus panaudotas kaip „Selenium“ įrankio valdymo biblioteka PHP programavimo kalbai.

3.2. Testavimo procesą sudarančios dalys

Testavimo procesą sudaro šios dalys:

- Testavimo platformų paruošimas:
  - PHPUNIT paruošimas;
  - Selenium RC paruošimas.
- Specifikuotų testinių scenarijų analizė ir testų sudarymas:
  - Pradinių testinių duomenų paruošimas pagal testuojamą scenarijų;
  - Testuojamo scenarijaus numatytų atlikti veiksmų įgyvendinimas.
- Testų vykdymas.

3.3. PHPUnit platformos paruošimas

„PHPUnit“ veikimui užtikrinti testiniame serveryje privalo būti įdiegta „PHP5“ programavimo aplinka. Pati „PHPUnit“ platforma nereikalauja sudėtingų diegimo procedūrų, nes ji pateikiama bibliotekos pavidalu. „PHPUnit“ gali būti panaudota atskirai, kaip testuojamos sistemos dalis arba kaip testavimo platformos dalis. „PHPUnit“ biblioteką galima atsisiųsti adresu http://www.phpunit.de. „PHPUnit“ biblioteka nėra patogi naudotis norint automatizuoti visą testavimo procesą. Biblioteka neturi galimybės automatiškai aptikti sukurtų testų, juos reikia rankiniu būdu prijungti prie testavimo sistemos, o tai reikalauja papildomų žinių. Taip pat neturi

savyje bibliotekų pradiniams duomenims užkrauti ir pašalinti. Viso testavimo proceso supaprastinimui buvo sukurta papildoma programinė įranga, kuri automatiškai suranda sukurtus testus, juos prijungia prie „PHPUnit“ testavimo platformos ir juos įvykdo. Naudojant šia įrangą jokių papildomų žinių ir paruošimo darbų nereikia, pakanka įvykdyti „phpunit.bat“ komandą komandinėje eilutėje ir peržiūrėti gautus rezultatus. Ši įranga pateikiama kaip magistrinio darbo priedas.

3.4. Selenium RC platformos paruošimas

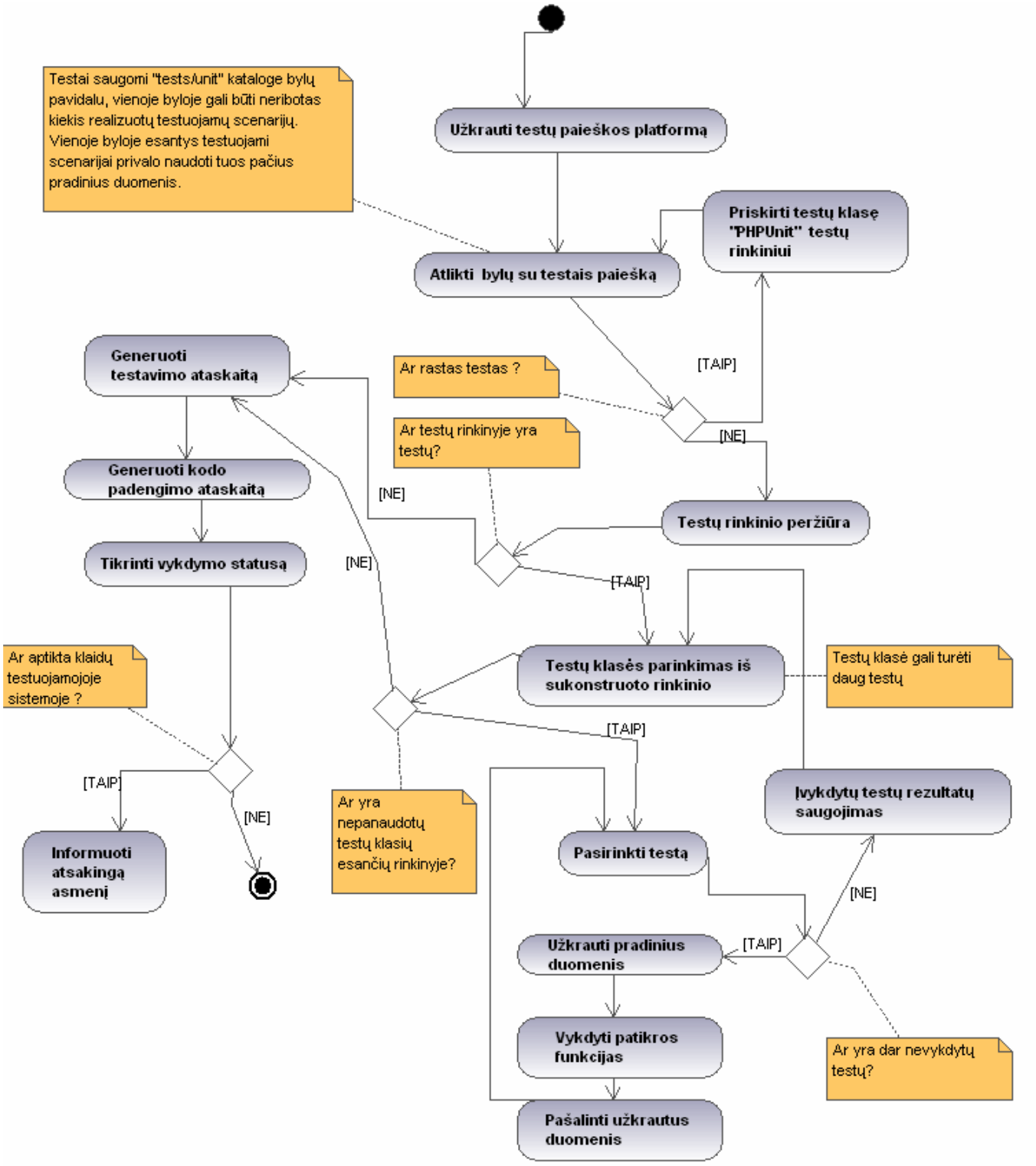
Selenium RC įrankio veikimui užtikrinti, testiniame serveryje privalo būti įdiegta Java 5 arba aukštesnė Java aplinkos versija. Naujausia Selenium įrankio versiją atsisiųsti galima adresu http://www.openqa.org/selenium-rc/download.action. „Selenium RC“ panaudojimas testavimo procese nenaudojant specialių bibliotekų yra painus ir sudėtingas. Dėl šios priežasties testavimo procesas ir „Selenium RC“ funkcinis testavimas bus vykdomas naudojantis PHPUnit biblioteką, kuri valdys „Selenium RC“ serverį. Visas testavimo procesas bus valdomas naudojant tą pačią, studijų metu sukurtą programinę įrangą kaip ir PHPUnit atveju. Selenium RC testų paleidimas atliekamas įvykdžius „phpselenium.bat“ komandą komandinėje eilutėje.

3.5. Testavimo proceso vykdymas

Pagrindiniai vienos iteracijos testavimą sudarantys žingsniai yra šie:

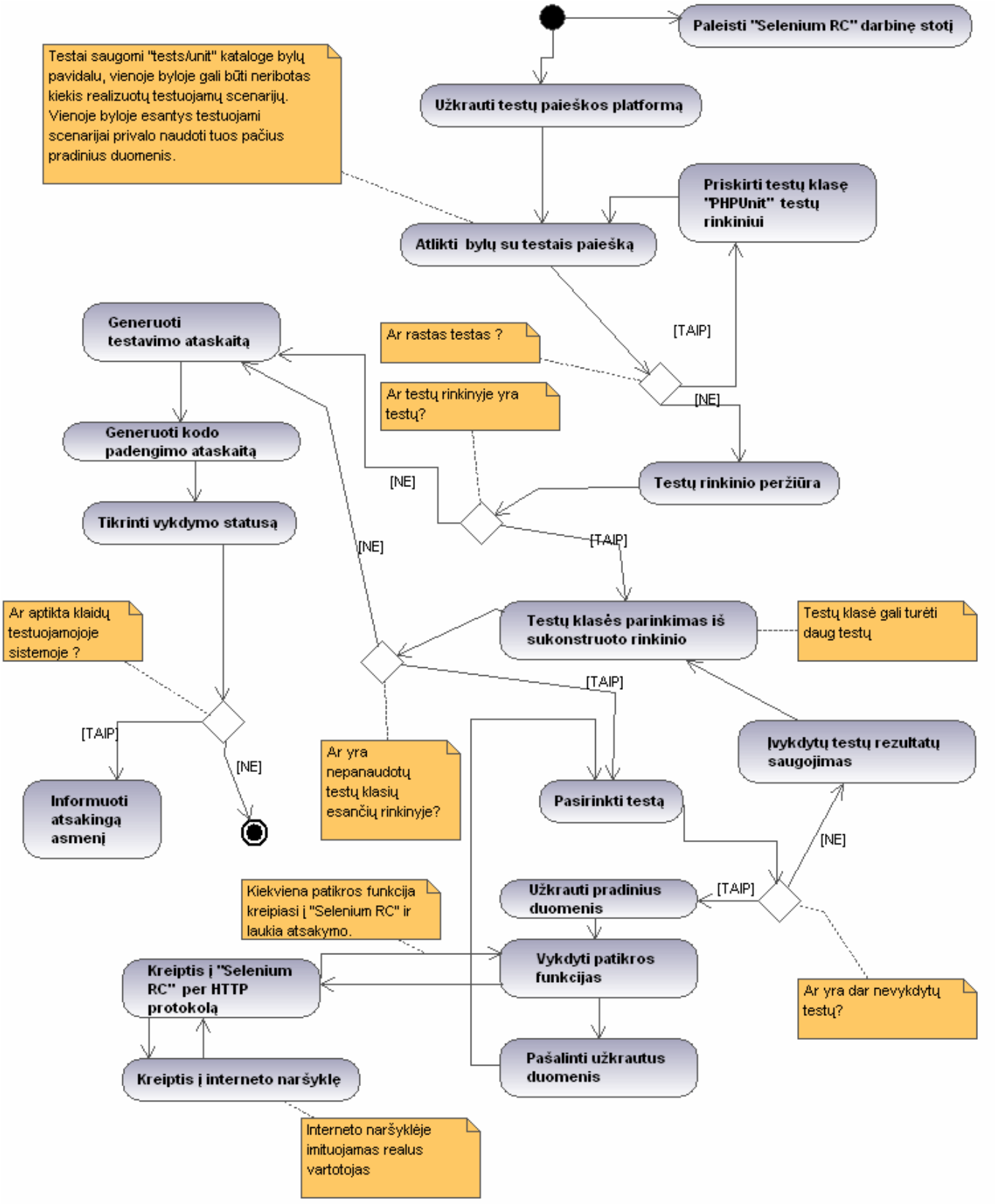
- Specializuotų pradinių testinių duomenų užkrovimas;
- Testų vykdymas;
- Užkrautų testinių duomenų pašalinimas;
- Testų vykdymo metu gautų rezultatų išsaugojimas.

„PHPUnit“ testavimo proceso eigos diagrama:



1 pav. Testavimo proceso diagrama „PHPUnit“

„Selenium RC“ testavimo proceso eigos diagrama:



2 pav. Testavimo proceso diagrama „Selenium“

Atliekant „PHPUnit“ ir „Selenium RC“ kiek galima paprastesnio pritaikymo realiam panaudojimui galimybių tyrimą, buvo sukurta testavimą palengvinanti programinė įranga, kuri apjungia „PHPUnit“ ir „Selenium RC“ įrankius. Taip pat ši programinė įranga automatizuoja visą testavimo proceso valdymą, randa sukurtus testus, juos vykto, grupuoja rezultatus, pateikia bendrą ataskaitą, informuoja atsakingą asmenį elektroniniu paštu apie testuojamojoje sistemoje rastus defektus. Testavimo proceso diagramose pavaizduotas šios programinės veikimo modelis „PHPUnit“ ir „Selenium RC“ atvejais.

Startavus šiai sistemai, iš nustatytų katalogų yra surenkami ir įvykdomi visi testai.

Sėkmingai pasibaigus testavimo procesui testavimo procesą reikia pradėti iš naujo, kai tik sistemos kodas yra pakeičiamas. Jei testavimo metu sistema pastebi defektus testuojamojoje sistemos dalyje, apie juos sistema informuoti testuotoją ar kitą atsakingą asmenį. Testavimo metu aptikti defektai privalo būti pašalinti.

3.6. Testo sudedamosios dalys

Nepriklausomai nuo testo tipo, kiekvienas testas realizuoja tam tikros programinės įrangos dalies veikimo teisingumo patikrinimą. Pagrindinės bet kurio tipo testo sudedamosios dalys yra šios:

- Specializuoti pradiniai testiniai duomenys;
- Patikros funkcijų rinkinys.

3.6.1. Specializuotų testinių duomenų paruošimas ir galimi formatai

Specializuotų testinių duomenų paruošimas yra labai svarbus etapas testų kūrimo etape. Nuo testinių duomenų paruošimo kokybės priklauso testo, naudosiančio šiuos duomenis, kokybė ir sugebėjimas aptikti testuojamos sistemos defektus. Vieną kartą paruošti duomenys vėliau bus pakartotinai panaudoti tūkstančius kartų, siekiant aptikti sistemos defektus. Dėl šios priežasties duomenų kokybė yra labai svarbi. Specializuoti pradiniai duomenys turi būti ruošiami atsižvelgiant į konkretų testuojamą atvejį ir jo specifikaciją.

Reikalavimai testiniams duomenims:

- Turi atitikti realia testuojamą sistemos būseną ;

- Turi būti sudaryti taip, kad būtų įmanoma pilnai atlikti visus testo scenarijuje numatytus veiksmus ir pasiekti visas numatomas testuoti sistemos būsenas;
- Testinių duomenų saugojimo formatas pasirenkamas laisvai, priklausomai nuo situacijos.

Specializuotų duomenų formatas priklauso nuo to, kokios bibliotekos bus naudojamos duomenims užkrauti testo vykdymo metu. Taip pat duomenų formatas priklauso nuo testuojamos sistemos savybių, testų tipų ir sudėtingumo. PHP programavimo kalboje dažniausia yra naudojami XML, YAML arba paprastas tekstinis formatas saugoti testinius pradinius duomenis.

Jei testas testuoja tik kažkokios vienos funkcijos įėjimus ir išėjimus, tuomet nebūtina testinių duomenų atskirti nuo pačių testų.

Jei sistema yra sudėtinga ir prieš atliekant testavimą būtina užpildyti duomenų bazę(s) duomenimis, tokiu atveju duomenų užkrovimas turėtų būti atskirtas nuo testo ir duomenys saugomi vienu iš pasirinktų formatu.

3.6.2. Patikros funkcijų rinkinys

Patį testą sudaro patikros funkcijų rinkinys, skirtas tam tikros sistemos dalies atitikimo numatytam funkcionavimui patikrinti. Vieną testą gali sudaryti daug patikros funkcijų, tikrinančių ar testuojamos sistemos funkcionalumas atitinka tai kas yra numatyta specifikacijoje. Testo tikslas yra atsakyti į klausimą ar testuojama sistemos dalis tikrai veikia taip kaip turi veikti, o jei neveikia tai kuri vieta tiksliai neveikia, taip kaip tikimasi.

3.7. Papildoma programinė įranga palengvinanti testavimą

Atliekant daugelio programinių sistemų testavimą yra nepraktiška kaskart iš naujo stengtis išsiaiškinti, kaip panaudoti PHPUnit ir Selenium RC bibliotekas. Testus rašantis žmogus turėtų rūpintis kaip sukurti kokybišką testą, o ne testavimo proceso valdymo ypatumais. Labai nepatogu sukūrus testą ieškoti atsakymo į klausimą kaip jį rankiniu būdu prijungti prie testavimo sistemos. Pradinių duomenų užkrovimas ir pašalinimas taip pat turėtų būti kuo paprastesnis. Testuotojas turėtų rūpintis tik pradinių duomenų sudarymu ir testinių scenarijų realizacija, o ne testavimo platformų panaudojimu ir pritaikymu testavimo procesui.

Būtent dėl šių priežasčių buvo sukurta speciali programinė įranga, kuri apjungė PHPUnit ir Selenium RC bibliotekas. Ši programinė įranga:

- supaprastino testų vykdymo ir paleidimo procesą;
- supaprastino duomenų bazės panaudojimą;
- automatizavo testų surinkimą ir vykdymą;
- supaprastino pradinių duomenų įkėlimą ir pašalinimą.

PHPUnit ir Selenium testai vykdomi atskirai. Kad sistema veiktų, PHPUnit testai turi būti patalpinami į „UNIT“ katalogą, esanti programinės įrangos pagrindiniame kataloge. Selenium testai talpinami į „FUNCTIONAL“ katalogą. Testai paleidžiami iš komandinės eilutės įvykdžius komandą „phpunit.bat“ arba „phpselenium.bat“ . Testai pateikiami eksperimentinėje dalyje.

4. Eksperimentinė dalis

4.1. Testuojama informacinė sistema

Eksperimentiniam testavimui internetinė kelionių operatoriaus informacinė sistema nėra pasirinkta atsitiktinai. Šios sistemos užsakovas yra UAB „Guliverio kelionės“ tarptautinių kelionių organizatorius. Sistema buvo suprojektuota, sukurta ir įdiegta pas užsakovą magistratūros studijų metu. Šiuo metu sistemą naudoja kelionių operatoriaus partneriai ir darbuotojai visoje Lietuvoje. Nuolatos vykdomi sistemos priežiūros ir palaikymo darbai bei plečiamos funkcinės galimybės. Sistema sukurta panaudojant PHP programavimo kalbą ir Mysql duomenų bazės valdymo sistemą. Produkto apimtis yra didelė, kodą sudaro apie 33000 eilučių bei bibliotekų. Vis aktualesnė problema tampa šios sistemos kokybės užtikrinimas pritaikant ją prie nuolatos besikeičiančių reikalavimų. Sistemą sudaro daugiau nei 150 klasių, kurios yra tarpusavyje susijusios viena su kita. Atlikus pakeitimą, ar išplėtus funkcines galimybes vienoje kodo vietoje neturint automatizuotų testų, labai sudėtinga nustatyti ar atliktas pakeitimas nesugriaus kitos sistemos vietos. Tokiu atveju yra vienintelė išeitis - viską testuoti rankomis. Testavimas rankomis yra daug laiko užimantis ir neturintis išliekamosios vertės procesas, kurį būtina automatizuoti.

4.1.1. Informacinės sistemos paskirtis

Sukurta internetinė informacinė sistema, kurią galima nesunkiai pritaikyti daugeliui šiuo metu rinkoje esančių kelionių operatorių, sukurtas ir pritaikytas realiam panaudojimui sistemos branduolys leidžiantis nesunkiai kurti ir palaikyti kelionių operatorių, bei kitos paskirties internetines informacines sistemas veikiančias moduliniu pagrindu.

4.1.2. Sistemos vartotojo tikslai

Produktą buvo numatoma naudoti kaip realaus kelionių operatoriaus administracinę sistemą ir elektroninę parduotuvę. Kadangi sistema numatyta naudoti realiai, yra pageidaujama, kad produkto sąsaja būtu lengvai suprantama tiek firmos darbuotojams, tiek elektroninės parduotuvės klientams.

Kitos produkto paskirtys bei su jomis susiję vartotojo reikalavimai:

- Nesudėtingas išplečiamumas ir pritaikymas pagal kliento keliamus reikalavimus.

Šiam tikslui įgyvendinti reikia, kad programinė įranga būtų sukurta pasirenkant tokį modelį, kuris leistų kuriamą produktą bet kuriuo laiko momentu nesudėtingai papildyti naujais moduliais arba pašalinti jau nebenaudojamus modulius. Sistema turi būti sudaryta iš bazinės sistemos ir papildomų modulių.

Pagrindiniai bendri reikalavimai programinei įrangai:

1. Programinė įranga turi veikti stabiliai ir visuomet pasiekiama;
2. Programinė įrangos vartotojo sąsaja turi būti draugiška;
3. Programa turi taupyti darbuotojų laika, optimizuojant jų darbą;
4. Programinė įranga turi būti lengvai įdiegiama ir reikalauti kuo mažiau palaikymo;
5. Programinė įranga turi būti lengvai išplečiama.

Minimalūs, konkretūs programos reikalavimai – kokybiškas, greitas ir stabilus darbas.

4.1.3. Sistemos tikslai

Sistemos tikslai:

- automatizuoti kelionių organizavimo procesą. Kelionių organizavimo proceso automatizavimas pagerins teikiamų paslaugų kokybę, nes sumažės rutininio darbo kiekis ir darbuotojai galės daugiau laiko skirti klientų aptarnavimui bei naujų paslaugų tiekimui.
- Automatizuoti kelionių ir kitų paslaugų pateikimą kelionių operatoriaus svetainėje. Tai leis padidinti kelionių operatoriaus parduodamų kelionių skaičių ir tokiu būdu padidinti gaunamą pelną.
- Suteikti kelionių operatoriaus partneriams galimybę prisijungti prie rezervacinės sistemos, atlikti paslaugų užsakymus bei realiu laiku valdyti juos. Tai padarys kelionių operatorių patrauklesniu esamiems ir potencialiems partneriams.
- Kaupti informaciją apie darbuotojų ir partnerių atliktus pardavimus. Remiantis šia informacija darbuotojai ir partneriai bus skatinami įvairiomis premijomis.
- Kaupti klientų kontaktinę informaciją. Taip pat kaupti informaciją apie klientų užsisakytas keliones. Remiantis šia informacija pastoviems klientams bus taikomos specialios nuolaidos. Tokiu būdu bus pagerinta teikiamų paslaugų kokybė.
- Leisti prisijungti ir dirbti su sistema iš kelionių operatoriaus filialų Lietuvoje ir užsienyje.

Sistemoje naudojamu panaudos atvejų diagrama:

3 pav. Panaudos atvejų diagrama

4.1.4. Informacinės sistemos funkcijos

Sukurta programinė įranga yra skirta pagerinti kelionių operatoriaus parduodamų kelionių valdymą ir pateikimą galutiniam vartotojui.
Sistemą sudaro 2 dalys:

- Administracinė dalis;
- Elektroninė parduotuvė.

Elektroninės parduotuvės moduliai:

- Kelionių paieškos;
- Kelionių pirkimo;
- Registracijos;
- Prisijungimo;
- Pagalbos;
- Užsakymų valdymo.

Administracinės dalies moduliai:

- Ataskaitų;
- Rezervacijų;
- Kelionių valdymo;
- Vartotojų;
- Pagalbos.

Sistema susideda iš dviejų pagrindinių dalių: elektroninės parduotuvės ir administracinės dalies. Elektroninėje parduotuvėje pateikiama informacija apie kelionių operatoriaus teikiamas paslaugas, parduodamas keliones, viešbučius. Vartotojas gali ieškoti kelionių pagal įvairius parametrus. Taip pat suteikiama galimybė užsisakyti kelionę internetu. Elektroninėje parduotuvėje pateikiama informacija gauta iš administracinės dalies.

Užsakymų informacija peržiūrima firmos darbuotojų, prisijungus prie administracinės dalies, kurioje taip pat kelionių operatorius gali kurti ir administruoti keliones, viešbučius, laikus,

kainas, priimti arba atmesti rezervacijas.

Produktas sukurtas kaip vientisa sistema ir įdiegta į serverį. Sistema ateityje bus daugiausiai naudojama, kaip kelionių operatoriaus ir jo partnerių efektyvi priemonė paprastam tarpusavio bendradarbiavimui parduodant keliones.

4.1.5. Bendri apribojimai sukurtai sistemai

Sistema turi veikti UNIX, WINDOWS IR LINUX operacinių sistemų aplinkose be papildomų pakeitimų.

Minimalios serverio kompiuterio charakteristikos, kuriame veiks produktas:

- Pelė, klaviatūra;
- Operatyviosios atminties talpa 2 GB;
- Procesoriaus greitis 3 Ghz;
- 10 GB kietojo disko vietos programai ir duomenų bazės duomenims saugoti;
- Apache Web serveris;
- Mysql serveris;
- 1Mbps interneto ryšio išeinantysis kanalas.

Minimalios kliento kompiuterio charakteristikos:

- Pelė, klaviatūra;
- Monitorius (raiška 1024x768);
- Operatyviosios atminties talpa 32MB;
- Procesoriaus greitis 200 MHZ;
- Windows, Unix operacinė sistema;
- Interneto naršyklė.

Pagrindinis dėmesys turėtų būti skiriamas serverio procesoriaus greičiui, atminties kiekiui ir interneto kanalo pralaidumui, nes vienu metu prie sistemos jungsis ir ja naudosis daug vartotojų (firmos darbuotojai, partneriai, klientai).

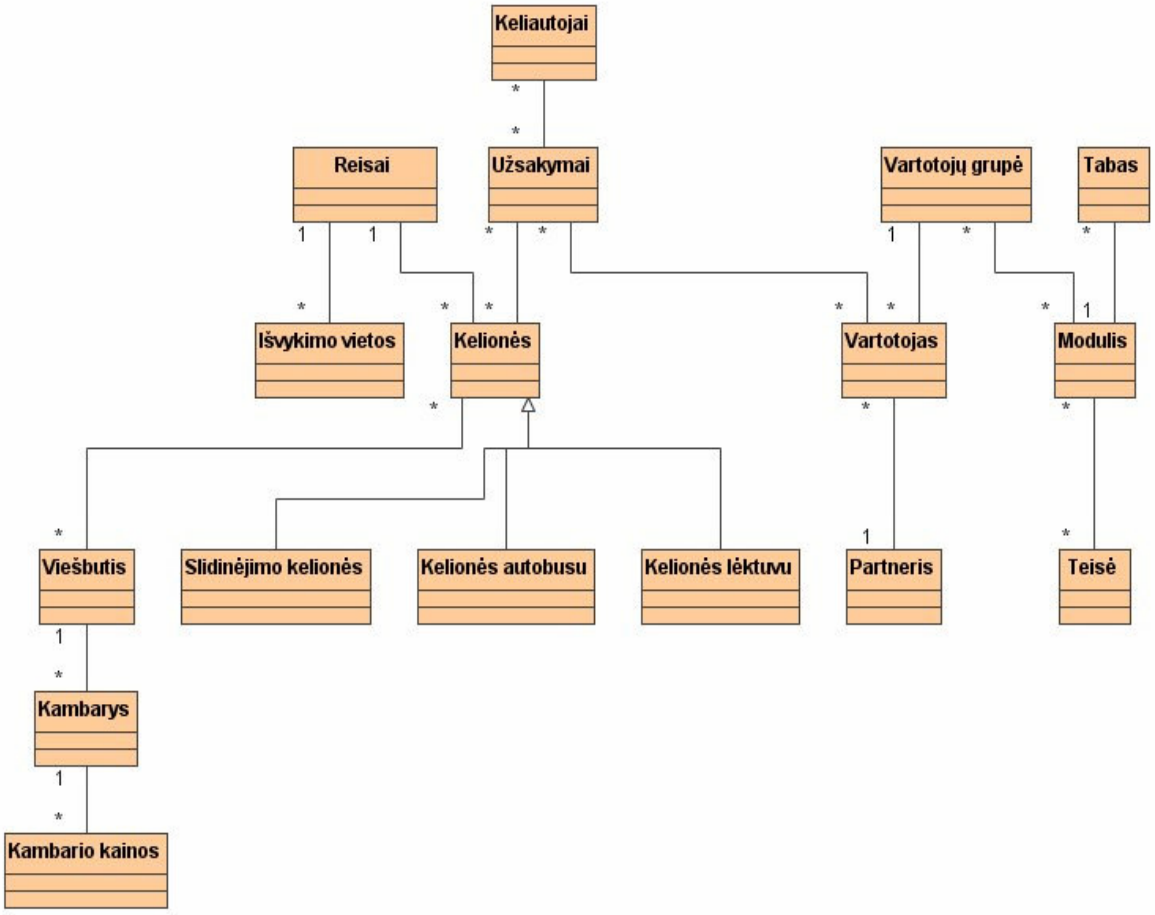
Kūrimui naudojama programinė įranga:

- Microsoft Windows XP;
- Microsoft Office 2003;
- Magic Draw UML 9.0;

- Eclipse IDE + phpEclipse;
- Linux Server OS (WEB, FTP, Samba server);
- MySQL serveris;
- SVN serveris;
- Zend Studio (serveris ir klientas).

4.1.6. Sistemos objektai ir jų ryšiai

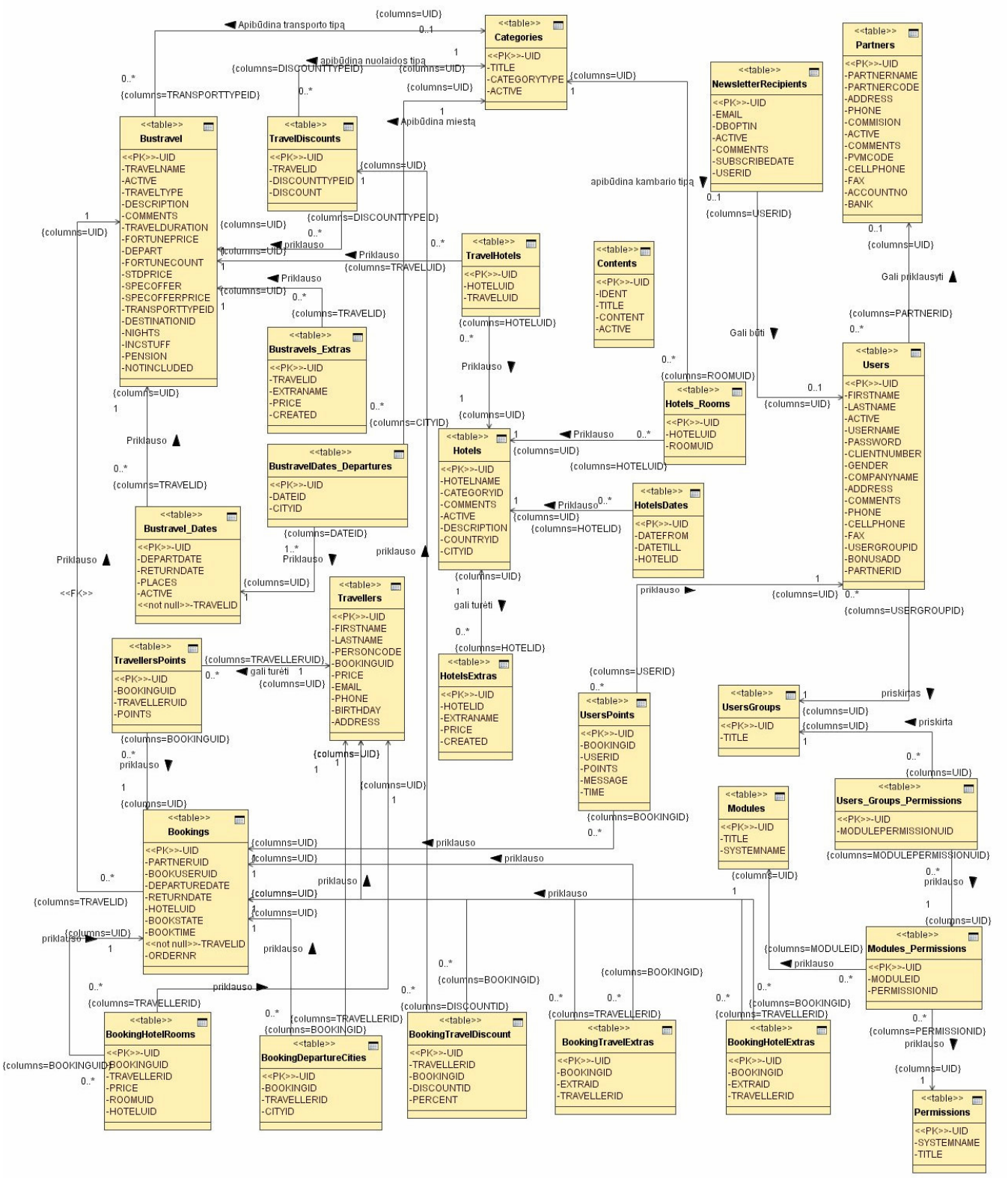
Preliminari sistemos objektų bei jų ryšių diagrama:



4 pav. Sistemos objektai ir sąryšiai

4.1.7. Duomenų vaizdas

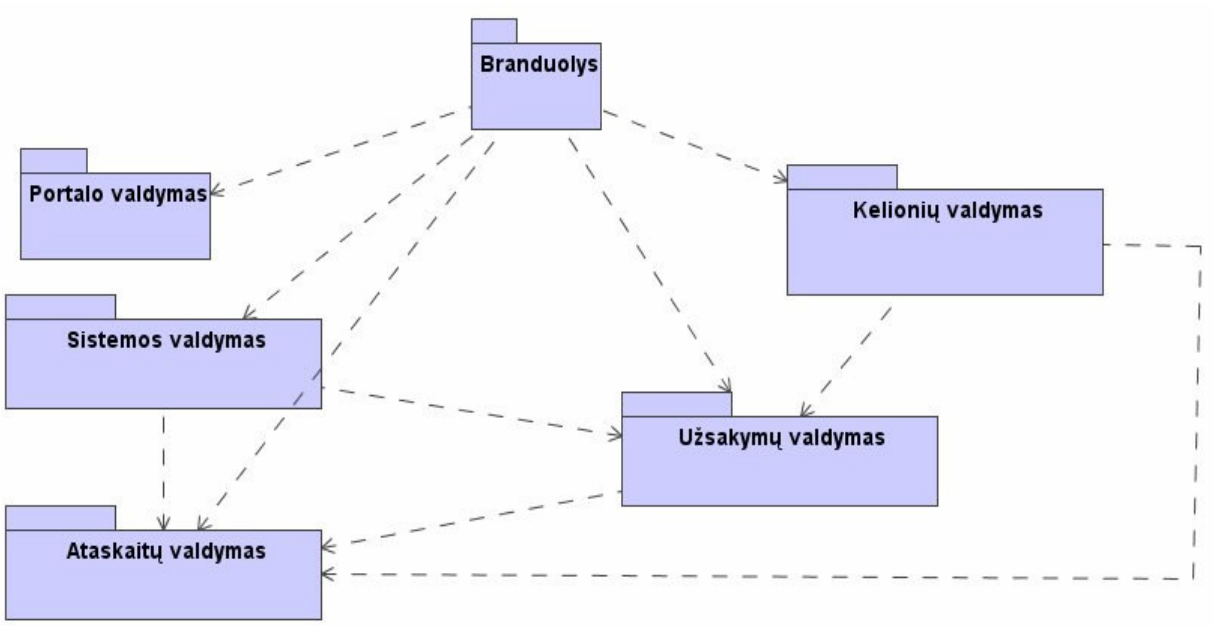
Duomenų vaizdas pateiktas UML DDL diagrama



5 pav. Duomenų vaizdas

4.1.8. Sistemos išskaidymas į loginius paketus

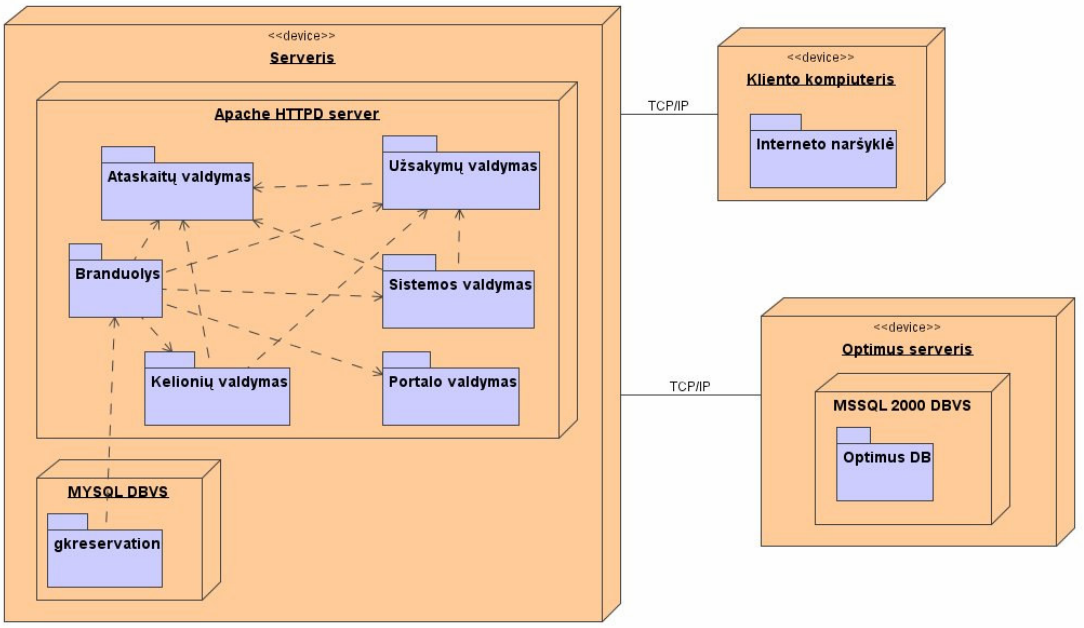
Pateikiamas sistemos išskaidymas į loginius paketus ir juos sudarančias klases. Išskaidymas į paketus pasirinktas pagal sistemos atliekamas funkcijas.



6 pav. Sistemos išskaidymas į loginius paketus

4.1.9. Išdėstymo vaizdas

Išdėstymo diagrama:



7 pav. Išdėstymo vaizdas

4.1.10. Informacinės sistemos apimtis

Pateikiama bendra sukurtos informacinės sistemos apimtis:

- Kodo eilučių kiekis:
  - Be bibliotekų: ~33000;
  - Su bibliotekomis: ~130000;
- Klasių: ~150;
- Vartotojo sąsajos langų: ~70.

4.2. Sistemos testavimas naudojant PHPUnit

PHPUnit vienetų testavimo eksperimentas bus atliekamas su viena iš daugelio sistemą sudarančių klasių. Pasirinkta klasė yra informacinės sistemos „Viešbučių“ modulio sudedamoji dalis. Su šia klase bus atliekamas testavimo eksperimentas. Klasės pasirinkimas didelės įtakos testavimo procesui ir naudojamiems metodams neturi. Pasirinkus kitą klasę būtų kitoks testavimo scenarijus ir pradiniai duomenys, testų sudarymas išliktų toks pats.

4.2.1. Testuojamos „HotelActions“ klasės specifikacija

Testuojama klasė yra sudedamoji testuojamos sistemos dalis.

HotelActions klasę sudaro šie metodai: :

- Duoto periodo masyvo sudarimo metodas „CreateDaysArray“ :
  Parametrai:
  - dtFrom – periodo pradžios data;
  - dtTill – periodo pabaigos data;
  - flPrice – periodo kaina;
- Viešbučio datų gavimo metodas „GetHotelDates“ :
  Parametrai:
  - hotelId – viešbučio unikalus numeris;
- Kainos skaičiavimo metodas „CalculatePrice“:
  Parametrai:
  - dateFrom - data nuo;
  - dateTill – data iki;
  - pricePerDay – nakvynes kaina.

4.2.2. „HotelActions“ klasės testavimo scenarijus

Klasės vienetų testavimas bus atliekamas PHPUnit priemonėmis. Remiantis klasės

specifikacija testavimo metu būtina patikrinti šiuos atvejus:

- „CreateDaysArray“ metodo veikimo teisingumą;
  - Patikrinti ar gražinama reikšmė yra „null“ jei paduoti pradiniai duomenys yra neteisingi.
  - Patikrinti ar metodas gražina 4 dienų masyvą, jei datų skirtumas yra 4 dienos imtinai.
  - Patikrinti ar metodas gražina 3 dienų masyvą, jei datų skirtumas yra 3 dienos imtinai.
- „GetHotelDates“ metodo veikimo teisingumą
  - Patikrinti ar metodas gražina „false“ jei viešbutis neegzistuoja.
  - Patikrinti ar gražinamos viešbučio datos, jei viešbutis egzistuoja.
- „CalculatePrice“ metodo veikimo teisingumą
  - Patikrinti ar gražinama reikšmė yra „false“ jei paduoti pradiniai duomenys yra neteisingi.
  - Patikrinti ar gražinama reikšmė yra „false“ jei kaina yra neigiama.
  - Patikrinti ar gražinama kaina yra 40, jei praleidžiamu nakvynių kiekis yra 4, o kaina 10.

4.2.3. „HotelActions“ klasės testiniai duomenys ir testas

Testiniai pradiniai duomenys sudaromi atlikus kiekvieno iš testuojamų metodų struktūros analizę. Testiniai duomenys sudaromi taip, kad būtų įvykdytos visos testuojamo metodo kodo eilutės. Dėl testuojamos klasės paprastumo testiniai duomenys saugomi pačiame teste. Testiniai duomenys užkraunami „setUp“ metode, o pašalinami „tearDown“ metode. Šiuo atveju, pašalinti duomenų nereikia.

Pateikiamas klasė kodas:

```
<?php
class HotelActions
{
    private $_db;

    function __construct($db)
    {
        $this->_db = $db;
    }
```

```
        public function createDaysArray($dtFrom, $dtTill, $flPrice)
        {
          $fromDate = strtotime($dtFrom);
          $tillDate = strtotime($dtTill);
          $dif = ($tillDate - $fromDate)/60/60/24;

          $returnValue = null;
          if ($dif >= 0)
          {
            for($i=0; $i<=$dif; $i++)
            {
                $key = date("Y_m_d", $fromDate+(86400)*$i);
                $returnValue[$key] = $flPrice;
            }
            $key = date("Y_m_d", $tillDate);
            $returnValue[$key] = $flPrice;
          }
          return $returnValue;
        }

        public function getHotelDates($hotelId)
        {
            $dates = $oConfig->oDB->GetAll("SELECT * FROM hotels_dates
                                            where HOTELID = ?",
                                                   array($hotelId));
            return $dates;
        }

        public function calculatePrice($dateFrom, $dateTill, $pricePerDay)
        {
            $da = $this->createDaysArray($dateFrom, $dateTill, $pricePerDay);
            if (!$da || count($da) == 0){
                error_log('Viesbutis neturi kainu');
                return false;
            }
            else
            {
                $price = 0;
                foreach($da as $current){
                   $price += $current;
                }
            }
            if ($price > 0)
                return $price;
            else
                return false;
        }
    }
```

Pateikiamas testo kodas:

```
<?php
/**
 *  Itraukiama testuojama klase
 */
require(dirname(dirname(dirname(__FILE__))).'/tc/core/hotelactions.php');

/**
 * HotelsActions klases testai
 */
class HotelActionsClassTest extends PHPUnit_Framework_TestCase
```

```
{
    private $_fixtures; // pradiniai testavimui skirti duomenys
    private $_ha;

    protected function setUp()
    {
        $testHelper = TestHelper::getInstance();
        $db = $testHelper->getDbConnection();
        $this->_ha = new HotelActions($db);

        // pradiniu duomenu ir laukiamu rezultatu uzkrovimas
        $this->_fixtures['data'][0]=array('dtFrom'=>'2008-02-25',
                                    'dtTill'=>'2008-01-25', 'price' => 10);
        $this->_fixtures['data'][1]=array('dtFrom'=>'2008-01-01',
                                     'dtTill'=>'2008-01-04', 'price' => 200);
        $this->_fixtures['data'][2]=array('dtFrom'=>'2008-01-04',
                                     'dtTill'=>'2008-01-06', 'price' => 170);

        $this->_fixtures['expectedResults'][0] = null;
        $this->_fixtures['expectedResults'][1] = array(
                                                    '2008_01_01' => 200,
                                                    '2008_01_02' => 200,
                                                    '2008_01_03' => 200,
                                                    '2008_01_04' => 200,
                                                  );
        $this->_fixtures['expectedResults'][2] = 3;

    }

    protected function tearDown()
    {
       // cia vykdomas testiniu duomenu pasalinimas
    }


    public function test_createDaysArray_return_null_when_input_data_is_incorrect()
    {
      $from  = $this->_fixtures['data'][0]['dtFrom'];
      $till  = $this->_fixtures['data'][0]['dtTill'];
      $price = $this->_fixtures['data'][0]['price'];
      $expect = $this->_fixtures['expectedResults'][0];
      $this->assertEquals($expect, $this->_ha->createDaysArray($from, $till, $price));
    }


    public function test_createDaysArray_return_4_days_array()
    {
       $from  = $this->_fixtures['data'][1]['dtFrom'];
       $till  = $this->_fixtures['data'][1]['dtTill'];
       $price = $this->_fixtures['data'][1]['price'];
       $expect = $this->_fixtures['expectedResults'][1];
       $this->assertEquals($expect, $this->_ha->createDaysArray($from, $till, $price));
    }


    public function test_createDaysArray_return_3_days_array()
    {
      $from  = $this->_fixtures['data'][2]['dtFrom'];
      $till  = $this->_fixtures['data'][2]['dtTill'];
      $price = $this->_fixtures['data'][2]['price'];
      $expect = $this->_fixtures['expectedResults'][2];
      $this->assertTrue($expect == count($this->_ha->createDaysArray($from, $till,
$price)));
    }
```

```
    public function test_calculatePrices_return_false_when_wrong_parameters()
    {
         $from  = $this->_fixtures['data'][0]['dtFrom'];
         $till  = $this->_fixtures['data'][0]['dtTill'];
         $price = 0;
         $expect = $this->_fixtures['expectedResults'][2];
         $this->assertFalse($this->_ha->calculatePrice($from, $till, $price));
    }

    public function test_calculatePrices_return_false_when_price_is_negative()
    {
         $from  = $this->_fixtures['data'][1]['dtFrom'];
         $till  = $this->_fixtures['data'][1]['dtTill'];
         $price = -10;
         $expect = $this->_fixtures['expectedResults'][2];
         $this->assertFalse($this->_ha->calculatePrice($from, $till, $price));
    }

    public function test_calculatePrices_return_price()
    {
         $from  = $this->_fixtures['data'][1]['dtFrom'];
         $till  = $this->_fixtures['data'][1]['dtTill'];
         $price = 10;
         $expect = $this->_fixtures['expectedResults'][2];
         $this->assertEquals($this->_ha->calculatePrice($from, $till, $price),40);
    }
}
```

Testuojamą klasę sudaro trys vidiniai metodai. Eksperimentiniai testais atlikti su dviem metodais. Trečias metodas lieka netestuotas ir tai atsispindės „kodo padengimo testais“ („code coverage“) ataskaitoje, kuria testavimo aplinka sugeneruos automatiškai.

Testavimo procesas valdomas anksčiau paminėtos palengvinančios testavimą programinės įrangos.

4.2.4. Atlikto testavimo rezultatai

Pasibaigus testavimui komandinėje eilutėje gaunamas bendras rezultatas, kiek testų pavyko kiek nepavyko. Nepavykus vienam iš testų testavimo pabaigoje tarp rezultatų pateikiama detali informacija, kurioje vietoje įvyko klaida ir kokia yra jos priežastis. Nagrinėjamu atveju buvo vykdomi 6 testai. Visi testai pasibaigė be klaidų.

Pateikiami testuotos klasės rezultatai:

```
> PHPUNIT.bat
  Loading test class: HotelActionsClassTest
  PHPUnit 3.0.0 by Sebastian Bergmann.
  ......
  Time: 00:01
  OK (6 tests)
```

Vykdytų testų informacija XML formatu:

```
<?xml version="1.0" encoding="UTF-8"?>
<testsuites>
  <testsuite name="" tests="6" failures="0" errors="0" time="0.060449123382568">
    <testsuite name="HotelActionsClassTest" tests="6" failures="0" errors="0" time="0.060449123382568">
      <testcase name="test_createDaysArray_return_null_when_input_data_is_incorrect" class="HotelActionsClassTest" time="0.043200969696045"/>
      <testcase name="test_createDaysArray_return_4_days_array" class="HotelActionsClassTest" time="0.0041759014129639"/>
      <testcase name="test_createDaysArray_return_3_days_array" class="HotelActionsClassTest" time="0.0029160976409912"/>
      <testcase name="test_calculatePrices_return_false_when_wrong_parameters" class="HotelActionsClassTest" time="0.0031960010528564"/>
      <testcase name="test_calculatePrices_return_false_when_price_is_negative" class="HotelActionsClassTest" time="0.0036871433258057"/>
      <testcase name="test_calculatePrices_return_price" class="HotelActionsClassTest" time="0.0032730102539063"/>
    </testsuite>
  </testsuite>
</testsuites>
```

4.2.5. Kodo padengimo testais ataskaita

Testavimo metu „PHPUnit“, analizuoja vykdomas testuojamo kodo eilutes ir pateikia išsamią ataskaitą, kuri parodo kurios testuojamo kodo eilutės buvo vykdytos testų metu. Norint užtikrinti programinės įrangos kokybišką veikimą rekomenduojama, kad vienetų ir integraciniais testais būtų padengta nuo 70% iki 100% viso sistemą sudarančio kodo.

Kodo padengimo ataskaita:

Current view: C:\Program Files\Apache Group\Apache2\htdocs\guliveriokeliones\tc\core

Date: Tue May 15 21:29:29 EEST 2007 — Executable lines: 29

Code covered: 93.10% — Executed lines: 27

Legend: Low: 0% to 35% | Medium: 35% to 70% | High: 70% to 100%

	Coverage (hide details)		
hotelactions.php		93.10%	27 / 29 lines
HotelActionsClassTest		**Lines hit**	
test_calculatePrices_return_false_when_price_is_negative		86.21%	25 / 29 lines
test_calculatePrices_return_false_when_wrong_parameters		44.83%	13 / 29 lines
test_calculatePrices_return_price		82.76%	24 / 29 lines
test_createDaysArray_return_3_days_array		55.17%	16 / 29 lines
test_createDaysArray_return_4_days_array		55.17%	16 / 29 lines
test_createDaysArray_return_null_when_input_data_is_incorrect		31.03%	9 / 29 lines

8 pav. Kodo padengimo ataskaita

Iš kodo padengimo testais ataskaitos matome, kad klasė „hotelactions.php“ yra padengta

testais 93,10%. Taip yra nes viena iš trijų funkcijų nebuvo testuota ir jos kodo eilutės nebuvo vykdytos.

Detali kodo padengimo ataskaita:

Current view: hotelactions.php
Date: Tue May 15 21:29:29 EEST 2007
Code covered: 93.10%
Executable lines: 29
Executed lines: 27
Legend: executed | not executed | dead code

```
1         : <?php
2         : class HotelActions
3         : {
4         :     private $_db;
5         :
6         :     function __construct($db)
7         :     {
8       6 :         $this->_db = $db;
9       6 :     }
10        :
11        :     public function createDaysArray($dtFrom, $dtTill, $flPrice)
12        :     {
13      6 :       $fromDate = strtotime($dtFrom);
14      6 :       $tillDate = strtotime($dtTill);
15      6 :       $dif = ($tillDate - $fromDate)/60/60/24;
16        :
17      6 :       $returnValue = null;
18      6 :       if ($dif >= 0)
19      6 :       {
20      4 :         for($i=0; $i<=$dif; $i++)
21        :         {
22      4 :             $key   = date("Y_n_d", $fromDate+(86400)*$i);
23      4 :             $returnValue[$key] = $flPrice;
24      4 :         }
25      4 :         $key = date("Y_n_d", $tillDate);
26      4 :         $returnValue[$key] = $flPrice;
27      4 :       }
28      6 :       return $returnValue;
29        :     }
30        :
31        :     public function getHotelDates($hotelId)
32        :     {
33      0 :         $dates = $oConfig->oDB->GetAll("SELECT * FROM hotels_dates where HOTELID = ?", array($hotelId));
34      0 :         return $dates;
35        :     }
36        :
37        :     public function calculatePrice($dateFrom, $dateTill, $pricePerDay)
38        :     {
39      3 :         $da = $this->createDaysArray($dateFrom, $dateTill, $pricePerDay);
40      3 :         if (!$da || count($da) == 0){
41      1 :             error_log('Viesbutis neturi kainu');
42      1 :             return false;
43        :         }
44        :         else
45        :         {
46      2 :             $price = 0;
47      2 :             foreach($da as $current){
48      2 :                 $price += $current;
49      2 :             }
50        :         }
51      2 :         if ($price > 0)
52      2 :             return $price;
53        :         else
54      1 :             return false;
55        :     }
56        : }
```

9 pav. Detalus kodo padengimas

Iš detalios kodo padengimo ataskaitos aiškiai matomos vykdytos ir nevykdytos kodo eilutės. Taip pat prie kiekvienos eilutės yra pateikta informacija, kiek kartų eilutės kodas buvo įvykdytas.

4.3. Sistemos testavimas naudojant Selenium RC

Pasinaudojant „Selenium RC“ testavimo įrankiu bus atliekamas funkcinis/priėmimo grafinės vartotojo sąsajos testavimas. Dėl šio testavimo specifikos neįmanoma gauti kodo padengimo ataskaitos. Taip yra, nes testavimas ir pati testuojama programinė įranga testų metu yra visiškai atskirta, dėl šios priežasties kodo padengimo ataskaita yra nesugeneruojama. Funkcinio/priėmimo testavimo metu išanksto numatytus veiksmus su testuojamą sistemą atlieka virtualus Selenium serverio sukurtas vartotojas. Imituojamas realaus vartotojo elgesio modelis, vykdomas numatytas darbo su sistema scenarijus. Tikrinamas vartotojo sąsajos langų komponentų egzistavimas ir funkcionavimo atitikimas specifikacijai.

4.3.1. Testuojamo „Naujienų“ modulio specifikacija

Naujienų modulis yra sistemos administracinės dalies sudedamoji dalis.

Naujienų modulį sudaro:

- Naujienų sąrašas;
- Naujo įrašo įvedimo forma;
- Sukurtų naujienų informacijos redagavimo forma;
- Naujienos pašalinimas.

Sukurtos naujienos rodomos kelionių operatoriaus internetiniame puslapyje.

4.3.2. Naujienų modulio testavimo scenarijus

Naujienų modulio grafinės vartotojo sąsajos funkcinis/priėmimo testavimas atliekamas su Selenium RC priemonėmis. Remiantis modulio specifikacija testavimo metu būtina patikrinti šiuos atvejus:

- Ar veikia naujo įrašo įvedimo forma:
    - Prisijungiama prie administracinės sistemos dalies;
    - Pasirenkamas naujienų modulis;
    - Užpildoma naujienų įvedimo forma;

    - Paspaudžiamas „Išsaugoti“ mygtukas.
- Ar sukurta naujiena atsirado sąraše:
    - Patikrinama ar sąraše po formos užpildymo ir išsaugojimo atsirado atitinkamas įrašas.
- Ar sukurta naujiena galima redaguoti:
    - Paspaudžiama naujienų sąrašo eilutė;
    - Užsikrovus naujienos duomenims, duomenys pakeičiami;
    - Forma išsaugoma;
    - Paspaudžiama naujienų sąrašo pakeisto įrašo eilutė;
    - Patikrinama ar duomenys tokie kokie turi būti..
- Ar atlikus pašalinimo veiksmą įrašas buvo pašalintas iš sistemos:
    - Spaudžiamas pašalinimo mygtukas sąraše;
    - Tikrinama ar naujiena vis dar yra ar išsitrynė.

Visi scenarijuje aprašyti veiksmai aprašomi Selenium priemonės teikiamomis galimybėmis.

Kadangi testuojamas sistemos modulis yra administracinėje dalyje, kuri yra apsaugota vartotojo vardu ir slaptažodžiu, testavimo metu Selenium RC virtualus vartotojas turės prisijungti prie sistemos su testiniu vartoto vardu ir slaptažodžiu.

4.3.3. Naujienų modulio testavimas

Naujienų modulis pasirinktas testavimo eksperimentui atsitiktinai iš visų testuojamos sistemos modulių. Modulio pasirinkimas eksperimentui įtakos neturi.

Testavimo metu Selenium sukurtas virtualus vartotojas prisijungs prie Informacinės sistemos administracinės dalies ir atliks testavimo scenarijuje numatytus veiksmus su naujienų moduliu.

Testą sudaro:

- Pradiniai duomenys;
- Testo komandų rinkinys realizuojantis scenarijų.

Pradiniai duomenys šiuo atveju yra duomenų bazės lentelių informacija, kuri užtikrina, kad prie sistemos būtų galima prisijungti testiniu vartotoju ir slaptažodžiu bei pilnai funkcionuotų naujienų sistemos modulis. Visi kiti moduliai gali ir neveikti korektiškai. Pradiniai duomenys turi

būti minimalistiniai, jų turi būti tik tiek kiek pakanka įvykdyti testuojama scenarijų.

Jei dėl kokių nors sistemos pakeitimų naujienų modulis pradėtu veikti nekorektiškai, testas tai pastebėtų ir informuotų apie nekorektišką sistemos modulio funkcionavimą.

Testavimo procesas bus valdomas tarpinės programinės įrangos palengvinančios testavimo procesą. Pasibaigus testavimui sugeneruojama vykdymo ataskaita.

4.3.4. Naujienų modulio testavimo scenarijau testas

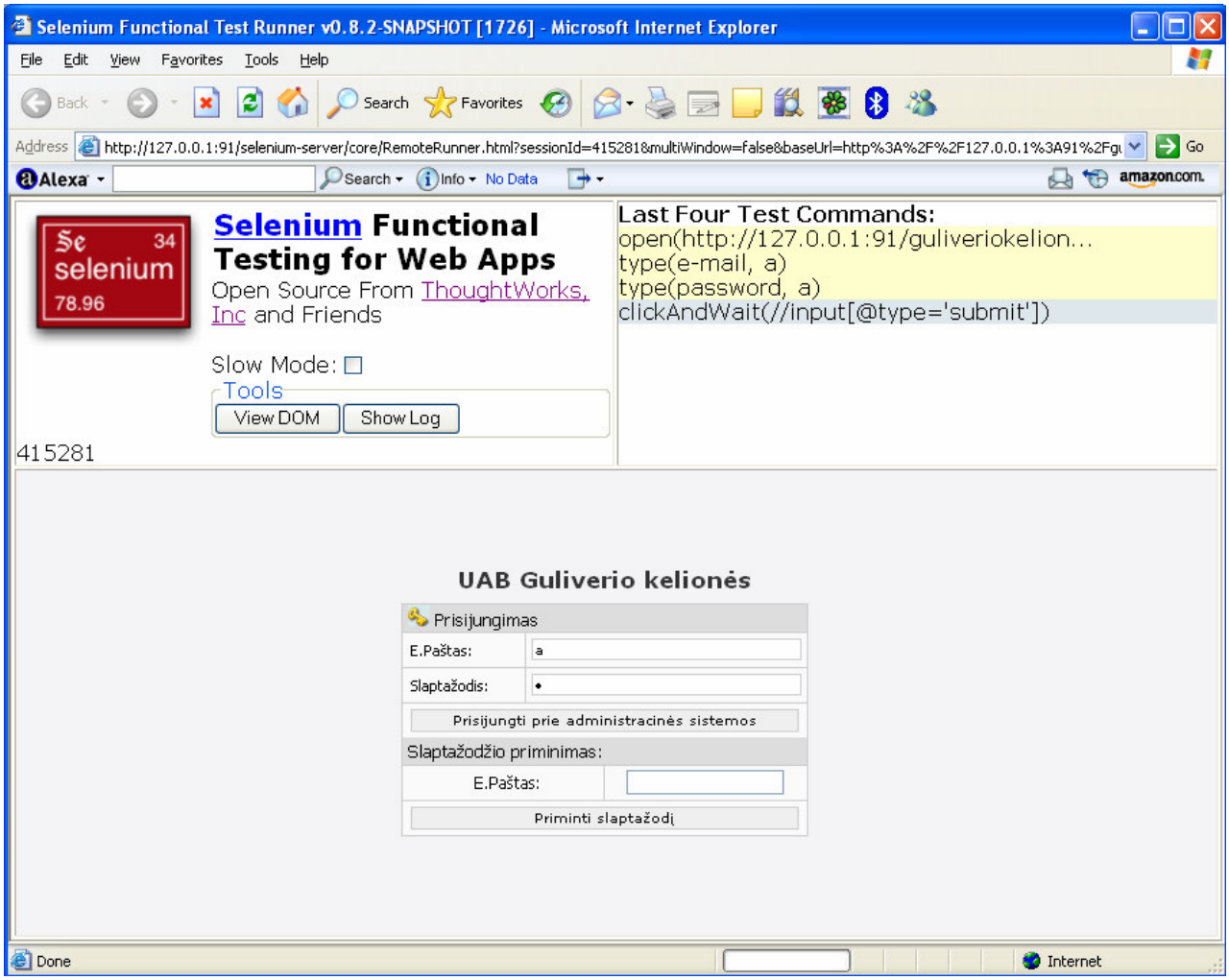
Testas pateikiamas PHP programavimo kalba:

```
require 'sqlDataLoader.php';
class TestNews extends PHPUnit_Extensions_SeleniumTestCase
{
    protected function setUp()
    {
        parent::setUp();
        $dataLoader = new SqlDataLoader();
        $dataLoader->LoadDataFile('NewsModuleTestFixtures.sql');
        $this->setBrowserUrl(ADMIN_URL);
        $this->setBrowser("*iexplore");
      }

    public function test_news_module()
    {
        $this->_login();
        $this->click("link=Naujienos");

        // sukurti nauja naujiena
        $this->click("link=Naujas įrašas");
        $this->type("//input[@name='new[title]']", 'naujiena');
        $this->click("//input[@name='new[save]']");
        $this->type("//input[@name='new[info]']", 'informacija');
        $this->click("//input[@name='new[save]']");

        // naujienos duomenu pakeitimas      ir patikrinimas ar pasikeite
        $this->click("link=datagrid_row_1");
        $this->assertText("//input[@name='new[title]']", 'naujiena');
        $this->assertText("//input[@name='new[info]']", 'informacija');
        $this->type("//input[@name='new[title]']", 'naujiena2');
        $this->type("//input[@name='new[info]']", 'informacija2');
        $this->click("//input[@name='new[save]']");
        $this->click("link=datagrid_row_1");
        $this->assertText("//input[@name='new[title]']", 'naujiena2');
        $this->assertText("//input[@name='new[info]']", 'informacija2');


        // naujienos egzistavimo sarase tikrinimas
        $this->_checkDatagridRowData(array('naujiena','*'), 1);

        // pasalinama naujiena
        $this->_deleteRow(1,2);
    }

private function _checkDatagridRowData($rowdata, $row)
    {
       $row=$row+2;// first row is 3
```

```
            $i=1;
            foreach ($rowdata as $item)
            {
              $tdLocation = ($i==1) ? '' : "[$i]";
              $this->assertText
("//body[@class='iframelist']/div/form/table/tbody/tr[$row]/td$tdLocation/div",
$item);
              $i++;
            }
         }
         private function _deleteRow($rowNumber, $deleteColumnNumber)
         {
            $rowNumber=$rowNumber+2;
            $tdLocation = ($deleteColumnNumber==1) ? '' : "[$deleteColumnNumber]";
             $this->click("//tr[$rowNumber]/td[contains(@onclick,'DeleteRecord')]");
            $this->assertConfirmationPresent("Ar tikrai norite ištrinti?");
            $this->assertConfirmation("Ar tikrai norite ištrinti?");
         }

         private function _login()
         {
            $this->open(ADMIN_URL);
            $this->type('e-mail','a');
            $this->type('password','a');
            $this->clickAndWait("//input[@type='submit']");
         }
      }
```

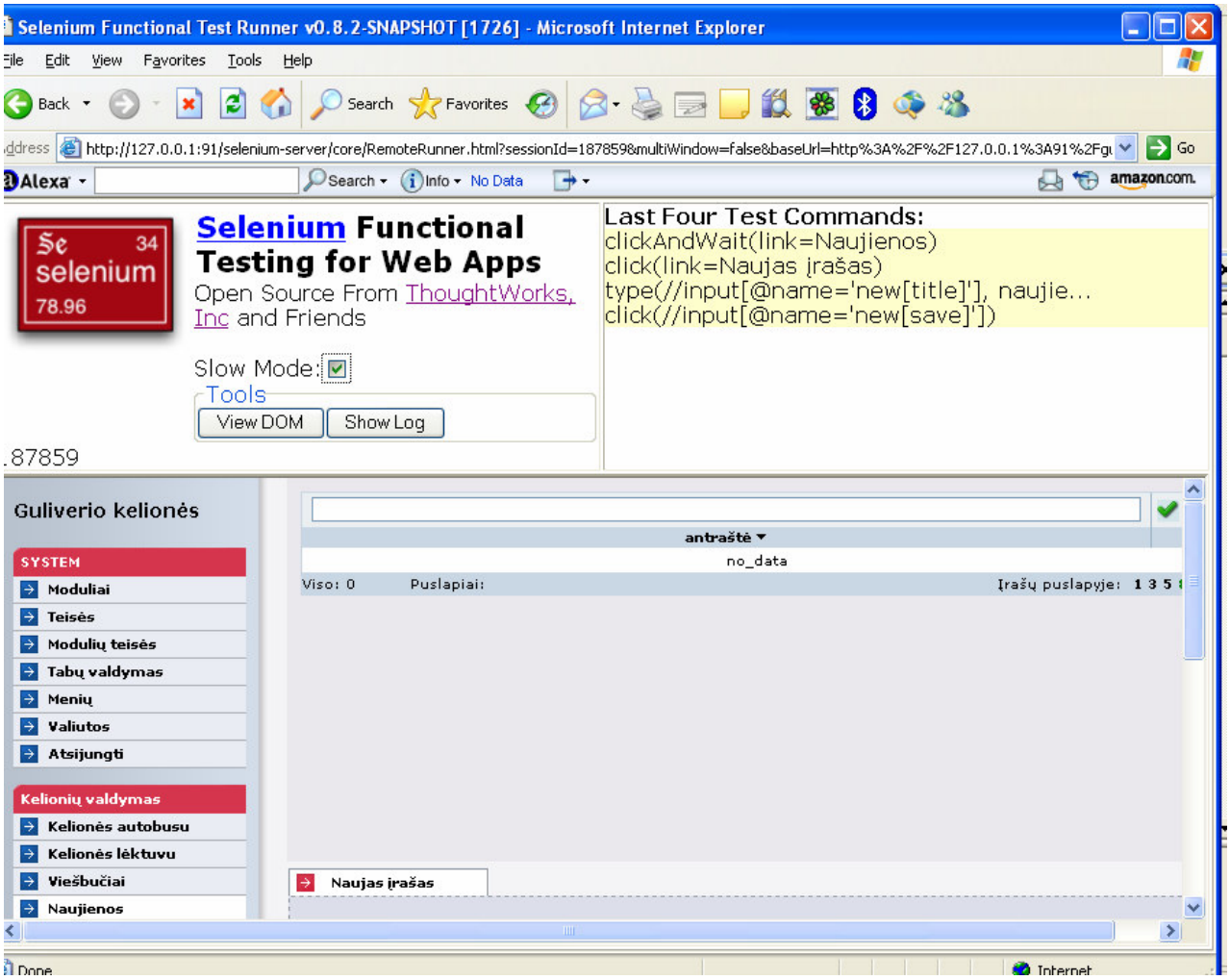
4.3.5. Testavimo etapai ir rezultatai

Naujienų modulis priklauso administracinei sistemos daliai, todėl privaloma prisijungti prieš atliekant testavimą. Pateikiamas prisijungimo prie sistemos etapas:



10 pav. Funkcinis testavimas

Naujienų modulio testavimo procesas – tikrinamas naujienų modulio veikimo teisingumas:



11 pav. Funkcinis testavimas

Pateikiami testuoto modulio rezultatai:

```
> PHPSELENIUM.bat
  Loading test class: TestNews
  PHPUnit 3.0.0 by Sebastian Bergmann.

  Preparing database data.....Done
  .......
  Removing database data....Done
  Time: 01:45
  OK (7 tests)
```

4.3.6. Testuojamos sistemos padengimas testais

Pritaikius šiame dokumente pateiktą testavimo metodiką vis didesnė dalis sukurtos Informacinės sistemos yra testuojama automatiškai.

Pateikiamas sistemos padengimas testais:

- Vienetų ir integraciniai testai (PHPUnit): 23% kodo eilučių. Tai sudaro 34 iš 150 klasių.
- Funkciniai testai: 21% vartotojo sąsajos funkcionalumo. Tai sudaro 15 iš 70 vartotojo sąsajos langų esančių sistemoje.

Testų kiekis nuolatos didėja, testavimo automatizavimas šiai sistemai, šiuo metu, yra vienas iš prioritetinių uždavinių.

5. Atlikti darbai

1. Ištirta, kurie automatinio testavimo įrankiai yra labiausiai tinkami PHP programavimo kalba sukurtoms sistemoms testuoti bei kaip juos panaudoti.
2. Sukurtas programinis produktas palengvinantis testavimo procesą.
3. Sudaryta testavimo metodika taikant PHPUnit ir Selenium RC įrankius bei papildomai sukurtą PĮ.
4. Suprojektuota ir realizuota Internetinė informacinė sistemą, kuri šiuo metu naudojama visoje Lietuvoje kelionių rezervacijoms atlikti.
5. Eksperimento metu sukurtoji IS testuota taikant pasiūlytą testavimo metodiką.

6. Išvados

Atlikus automatinių testavimo įrankių lyginamąją analizę nustatyta, kad programų sistemos realizuotoms, PHP kalba, tinkamiausi testavimo įrankiai yra PHPUnit ir Selenium. Tačiau praktiškai išbandžius šiuos įrankius paaiškėjo, kad juos naudoti kasdieniniame kokybės užtikrinimo vykdymo procese nėra paprasta ir patogu.

Šiai problemai spręsti buvo sukurta papildoma programinė įranga, kuri apjungė ir supaprastino testavimo minėtais įrankiais procesą.

Sudaryta turimų priemonių panaudojimo metodika ir atlikti automatinio testavimo bandymai su realia sistema parodė, kad sukurta programinė įranga puikiai suderinama ir efektyviau panaudoja PHPUnit ir Selenium RC testavimo priemonių teikiamas galimybes bei supaprastina testavimo procesą.

7. Literatūros sąrašas

1. Beck K. *Extreme programming explained*. USA 2000. 120psl. ISBN: 0-201-61641-6
2. Paston R. *Software testing*. USA 2005. 245psl. ISBN: 0-672-32798-8
3. James A. *Functional and Security Testing of Web Applications and Web Services* London. USA 2006. 240psl. ISBN: 0-321-36944-0
4. Lydia A. *The Web Testing Companion: The Insider's Guide to Efficient and Effective Tests.* USA 2003. 600psl. ISBN: 0-471-43021-8
5. Beizer B. *Black-Box Testing: Techniques for Functional Testing of Software and Systems USA 1995. 320psl. ISBN*: 0-471-12094-4
6. „Tools for Software Testing and Quality Assurance“. – [Žiūrėta 2007 02 01]. Prieiga per internetą: http://www.automatedqa.com
7. „Software testing, regression testing, regression testing software“. – [Žiūrėta 2007 01 20]. Prieiga per internetą: http://www.netvantagetech.com
8. „Web Application Load and Stress Testing“. – [Žiūrėta 2007 03 20]. Prieiga per internetą: http://www.neotys.com
9. „Network Monitoring Software, Network Performance Management“. – [Žiūrėta 2007 02 12]. Prieiga per internetą: http://www.paessler.com
10. „PHPUnit Pocket Guide“. – [Žiūrėta 2007 05 10]. Prieiga per internetą: http://www.phpunit.de
11. „JUnit, Testing Resources for Extreme Programming“. – [Žiūrėta 2007 04 02]. Prieiga per internetą: http://www.junit.org
12. „Unit Testing for PHP“. – [Žiūrėta 2007 05 15]. Prieiga per internetą: http://simpletest.sourceforge.net
13. „Selenium Remote Control“. – [Žiūrėta 2007 04 27]. Prieiga per internetą: http://www.openqa.org/selenium

8. Terminų žodynas

MYSQL - viena iš atviro kodo reliacinių duomenų bazių valdymo sistemų.

POSTGRESQL – nemokama objektinė-reliacinė duomenų bazių valdymo sistema.

MICROSOFT .NET – programinės įrangos kūrimo platforma.

PHP - plačiai paplitusi dinaminė interpretuojama programavimo kalba specialiai pritaikyta svetainių kūrimui.

JAVA- objektiškai orientuota programavimo kalba, 1991 metais sukurta Džeimso Goslingo ir kitų Sun Microsystems inžinierių

ASP - Microsoft technologija skirta dinamiškai generuojamų WEB puslapių kūrimui.

C# - Microsoft sukurta programavimo kalba

OOP – Objektiškai orientuotas programavimas.

OOD – Objektiškai orientuotas programų kūrimas.

XML – bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba.

HTML - tai kompiuterinė žymėjimo kalba, naudojama pateikti turinį internete.

DBVS – duomenų bazės valdymo sistema

ODBC – standartizuota taikomosios programinės įrangos (aplikacijų) programavimo sąsaja (API) prisijungimui prie duomenų bazių.

UML – (Unified Modeling Language) – modeliavimo kalba. Ši kalba naudojama sistemos modeliams sudaryti objektiškai-orientuotame projektavime.

Specifikacija – formalus sistemos funkcionalumo aprašymas.

PA - panaudojimo atvejai (Use Case)

CMS (Content Managment System) – Turinio valdymo sistema

SVN – bylų versijų kontrolės sistema