KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Ramunė Dundulytė
Darius Mikužaitis

Virtualaus krepšinio žaidimo modelio taikymas žaidimui vystyti ir testuoti

Magistro darbas

Darbo vadovė
prof. L. Nemuraitė

KAUNAS, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Ramunė Dundulytė
Darius Mikužaitis

Virtualaus krepšinio žaidimo modelio taikymas žaidimui vystyti ir testuoti

Magistro darbas

Recenzentas
doc. dr. N. Morkevičius
2011-05-27

Vadovė
prof. L. Nemuraitė
2011-05-27

Atliko
IFM-9/4 gr. stud.
Ramunė Dundulytė
Darius Mikužaitis
2011-05-27

KAUNAS, 2011

APPLYING VIRTUAL BASKETBALL GAME MODEL FOR ITS DEVELOPING AND TESTING

SUMMARY

Improving the process of developing online manager games is an important issue for the games' developers. However, it is not a widely analyzed subject. There are various methods and technologies that may help facilitating the process of developing the game, like: models of decision making element, technologies of software delivery and methods of testing.

There already are many articles analyzing the mentioned spheres in general. This thesis analyzes the ways of applying existing methods and technologies to the subject of developing online manager game, which is seldom studied and described.

The application of the analyzed methods is represented by the creation of the match engine's model (that is afterwards used in the process of improving the game) and two subsystems: one helping to involve users in the process of improving the game and the other allowing to quickly identify the emerged problems.

Keywords: online manager games, decision making models, Software-as-a-service, web applications testing

Turinys

1. Įvadas

Šio darbo tema yra virtualaus krepšinio žaidimo modelio taikymas žaidimui vystyti ir testuoti. Tyrimo objektas yra virtualus krepšinio žaidimas, kuriam yra svarbu nuolat keistis ir tobulėti – tam, kad žaidimo vartotojai išlaikytų susidomėjimą ir liktų žaidime. Taip pat svarbu pritraukti naujus vartotojus ir plėsti žaidimo bendruomenę. Taigi labai svarbu turėti žaidimo vystymo strategiją ir atitinkamas priemones vystymo procesui pagerinti ir palengvinti.

Šiuo metu pasaulyje egzistuoja keletas panašių žaidimų. Šio tipo žaidimų pradininkas yra futbolo žaidimas *Hattrick* [1]. Šiuo metu tai labiausiai išplėtotas ir daugiausiai vartotojų turintis *MMOMG* (*Massive Multiplayer Online Manager Game – daugelio vartotojų internetinis vadovavimo žaidimas*) tipo sportinis žaidimas [2]. Vėliau pradėjo rastis panašūs įvairių kitų sporto šakų (beisbolo, ledo ritulio, krepšinio ir kt.) žaidimai.

Yra keturi pagrindiniai elementai, kurie turi būti visuose žaidimuose: taisyklės, siužetas, pateikimas ir visų šių elementų sujungimo technologija. Kiekvienas iš šių elementų yra labai svarbus sėkmingam žaidimo gyvavimui. Siužetas dažniausiai nustatomas pačioje pradžioje, prieš pradedant įgyvendinimą, ir eigoje nekinta arba kinta nedaug. Pagrindinės žaidimo taisyklės taip pat yra nustatomos kūrimo pradžioje, tačiau eigoje (atsiradus naujiems elementams) gali būti koreguojamos ar papildomos. Pateikimas tobulinimo metu taip pat gali kisti, priklausomai nuo poreikių ir ateities planų [19].

Šiame darbe nagrinėsime egzistuojančius metodus ir technologijas, kurie gali padėti patobulinti virtualaus krepšinio žaidimo vystymo procesą. Tyrimo sritis apima žaidimų kūrimo modelius, programinės įrangos paslaugų *SaaS* (*Software as a Service*) technologijas ir jų taikymą, internetinių sistemų testavimo metodus. Pasinaudojus šių sričių modeliais ir technologijomis, žaidimo vystymo procesą galima padaryti spartesnį ir efektyvesnį. Siekiant nustatyti tinkamiausius modelius ir technologijas analizuojami moksliniai straipsniai, susijusios knygos bei internete pateikiama medžiaga.

Darbe sprendžiamos problemos, susijusios su žaidimo vystymo proceso pagerinimu. Iš pirmo žvilgsnio, žaidimo sistema atitinka programinės įrangos paslaugų (*SaaS – Software as a Service*) koncepciją, tačiau sistemoje nėra išnaudojami visi galimi šios koncepcijos privalumai. Viena iš svarbiausių žaidimo dalių yra rungtynių imitatorius, tačiau jo veikimas nėra vaizdžiai aprašytas ir norint

įgyvendinti tam tikrus pakeitimus būtina įsigilinti į programinį kodą. Tam tikro egzistuojančio sprendimų priėmimo modelio panaudojimas imitatoriui aprašyti galėtų pastebimai palengvinti jo tobulinimą. Kadangi žaidimui būtina nuolat tobulėti, neišvengiamai gali pasitaikyti klaidų. Dabartinėje sistemoje nėra priemonių, padedančių surasti įvykusias klaidas ir jas pašalinti, todėl siekiant pagerinti žaidimo vystymo procesą reiktų įvesti posistemę sutrikimų stebėjimui ir fiksavimui.

Tyrimo objektas bus analizuojamas stebint jo veikimą ir atskleidžiant sistemos esmę. Tyrimo srities analizė bus atliekama naudojant mokslinės literatūros analizės bei kitų panašių sprendimų stebėjimo metodus.

Darbo pradžioje yra egzistuojanti virtualaus krepšinio žaidimo sistema, turinti pagrindines funkcijas, reikalingas sėkmingam jos naudojimui. **Darbo tikslas** yra patobulinti virtualų krepšinio žaidimą ir palengvinti jo vystymą, sukuriant imitacinį rungtynių modelį ir įvedant papildomas posistemes.

Siekiama, kad patobulinus sistemą būtų:

- gaunama daugiau pajamų iš žaidimo – tai leistų tęsti tobulinimo darbus;
- padidintas vartotojų įtraukimas į žaidimo vystymo procesą – tai palengvintų sistemos kūrėjų darbą, padėtų rasti įvairesnių idėjų žaidimo tobulinimui;
- padidintas sistemos patikimumas – net nuolat diegiant atnaujinimus, sistema turi būti pakankamai patikima, kad esami vartotojai galėtų tęsti naudojimą.

Šiam tikslui pasiekti keliami **uždaviniai**:

- išanalizuoti sprendimų priėmimo modelius, žaidimų kūrimo technologijas ir testavimo metodus;
- sudaryti rungtynių imitatoriaus modelį;
- įvesti sutrikimų fiksavimo bei papildomų paslaugų teikimo posistemes;
- suprojektuoti sistemą patobulinsiančias posistemes;
- realizuoti ir ištestuoti sistemos patobulinimus;
- atlikti eksperimentą ir nustatyti realizuotų priemonių poveikį sistemai.

Darbo metu bus sudarytas rungtynių imitatoriaus modelis bei apibrėžtos veiklos taisyklės, kas leis formaliai aprašyti imitatoriaus veikimą ir pavaizduoti jį grafiškai. Tokio modelio sudarymas palengvins imitatoriaus testavimo ir tobulinimo procesą.

Darbo metu taip pat bus suprojektuotos ir įgyvendintos žaidimo vystymo procesui svarbios posistemės – papildomų paslaugų teikimo bei sutrikimų fiksavimo.

Naujai realizuotos posistemės (kaip ir anksčiau įgyvendintos) bus testuojamos keletu etapų: baltos dėžės testavimas, suderinamumo testavimas, juodos dėžės testavimas ir patogumo naudoti testavimas.

Įgyvendinus apsibrėžtas posistemes bus atliekami eksperimentai, kurie padės patikrinti atnaujinimų įtaką sistemai ir nustatyti ar buvo pasiekti pageidaujami rezultatai.

Yra parašyta nemažai darbų, kurie įvairiais aspektais nagrinėja darbe apsibrėžtą tyrimo sritį. Visgi dauguma darbų nagrinėja bendrą modelių, metodų ar technologijų panaudojimą. Tuo tarpu mes savo darbe ieškome būdų pritaikyti egzistuojančias technologijas būtent žaidimų vystymo sričiai, kas nėra išsamiai ištirta ir aprašyta.

1.1 lentelėje pateiktas procentinis autorių indėlis į kiekvienos darbo dalies atlikimą.

1.1 Lentelė

	Ramunė Dundulytė	Darius Mikužaitis
Analizė	60%	40%
Imitatoriaus modelis	60%	40%
Reikalavimų specifikacija	50%	50%
Projektas	50%	50%
Realizacija ir testavimas	40%	60%
Eksperimentinis tyrimas	40%	60%

Darbo struktūra.

- Pirmajame skyriuje trumpai apžvelgtas visas darbas, išskirti svarbūs elementai, pristatyti darbo tikslai ir darbų pasiskirstymas.
- Antrajame skyriuje atlikta virtualaus krepšinio žaidimo analizė, nustatytos egzistuojančios problemos ir suformuluoti tikslai bei uždaviniai. Taip pat išanalizuoti metodai ir technologijos, kurie gali būti pritaikyti iškeltiems tikslams įgyvendinti – tai modeliai sprendimų priėmimo logikai aprašyti, programinės įrangos paslaugų (*SaaS*) technologija ir internetinių programų testavimo metodai.
- Trečiajame skyriuje sudarytas ir pateiktas krepšinio rungtynių imitavimo modelis, taip pat aprašytos imitatoriaus veiklos taisyklės. Šis modelis palengvina rungtynių imitatoriaus testavimo ir tobulinimo procesą.
- Ketvirtajame skyriuje apibrėžti reikalavimai naujai kuriamoms posistemėms, sudarytas dalykinės srities modelis.
- Penktajame skyriuje sudarytas sistemos architektūros modelis bei įvedamų posistemių klasių modelis.

- Šeštajame skyriuje pateikti sistemos komponentų ir diegimo modeliai. Aprašytos realizuotos posistemės ir jų veikimas, pateikti langų pavyzdžiai. Sudarytas testavimo modelis įvedamoms posistemėms ištestuoti.
- Septintajame skyriuje suformuluotos hipotezės, kurios leidžia patikrinti įgyvendintų patobulinimų įtaką sistemai. Atlikti eksperimentai, iš kurių gauti duomenys panaudoti hipotezių tikrinimo skaičiavimams. Gauti rezultatai parodė, kad įgyvendinti patobulinimai padarė lauktą įtaką sistemai ir pagerino jos veikimą.
- Aštuntajame skyriuje pateiktas viso darbo apibendrinimas – suformuluotos išvados.
- Devintajame skyriuje pateiktas atliekant darbą panaudotų šaltinių sąrašas.

2. Analizė

2.1. Virtualaus žaidimo kūrimo tikslai ir uždaviniai

2.1.1. Virtualių žaidimų kūrimo problemos

Darbo metu bus siekiama patobulinti virtualaus krepšinio žaidimo sistemą bei palengvinti jo vystymą, sudarant sąlygas ilgalaikei vystymo strategijai. Norint pasiekti šių tikslų, pirmiausia reikia įsigilinti į tyrimo objektą – virtualų krepšinio žaidimą – bei išanalizuoti metodus, kuriuos galima taikyti.

Analizės metu apsibrėšime darbo sritį ir objektą, nustatysime uždavinius. Pirmoje analizės dalyje bus išanalizuota žaidimo esmė ir problemos, kurias reikia pašalinti. Antrojoje analizės dalyje palyginsime taikytinus metodus, išanalizuosime jų privalumus ir trūkumus, atrinksime tinkamiausius darbo uždavinių įgyvendinimui.

Baigiant analizę, suformuluosime siekiamą sprendimą bei pateiksime analizės išvadas.

Sritis. Žaidimų kūrimo modeliai, programinės įrangos paslaugų *SaaS* (*Software as a Service*) technologijos ir jų taikymas, internetinių sistemų testavimo metodai.

Objektas. Virtualus krepšinio žaidimo vystymo procesas, nuolat papildant naujomis savybėmis ir tobulinant esamas.

Problema. Sukurta virtualaus krepšinio klubo valdymo sistema iš pirmo žvilgsnio daugeliu savybių atitinka programinės įrangos paslaugų koncepciją, tačiau šios koncepcijos pritaikymo nagrinėjamai sistemai galimybės nėra išsamiai išnagrinėtos. Taip pat trūksta žaidimo imitatoriaus detalaus aprašymo ir aprašymo metodų konteksto. Reikėtų patobulinti ir kai kurias sistemos funkcijas, pavyzdžiui, žaidime trūksta operatyvios informacijos apie sistemoje pasitaikiusias klaidas (žaidimo logikos bei techninių sutrikimų), nėra galimybių valdyti vartotojų apmokestinimą ir su tuo susijusių funkcijų apribojimą. Pasirinktas žaidimo tipas reikalauja nuolatinio tobulinimo ir bendravimo su vartotojais, kurie yra įvairaus amžiaus ir patirties. Trūksta terpės, kurioje būtų galima diskutuoti su aktyviausiais ir labiausiai suinteresuotais vartotojais apie esamą situaciją bei tolesnį žaidimo vystymą.

Analizės tikslas. Išnagrinėti imitatorių kūrimo modelius ir aprašyti virtualaus krepšinio žaidimo imitatoriaus veikimą. Taip pat patobulinti virtualaus krepšinio žaidimo sistemą taip, kad ji leistų administracijai operatyviai reaguoti į sistemos darbo sutrikimus ir tobulinti žaidimo logiką bei sukurti sąlygas bendrauti su aktyviausiais nariais, kurie galėtų iškelti ir padėti spręsti žaidimo tobulinimui svarbius klausimus.

Analizės tikslas ir metodai. Mūsų darbo tikslas yra patobulinti virtualų krepšinio žaidimą bei pagerinti žaidimo vystymo procesą. Kad galėtume tai sėkmingai atlikti, turime išanalizuoti virtualų krepšinio menedžerį, kurį tobulinsime, bei tobulinimui reikalingus metodus ir metodologijas: žaidimų logikos realizavimo metodus, testavimo metodus ir *SaaS (Sofware-as-a-Service)* pritaikymą programinės įrangos kūrimui.

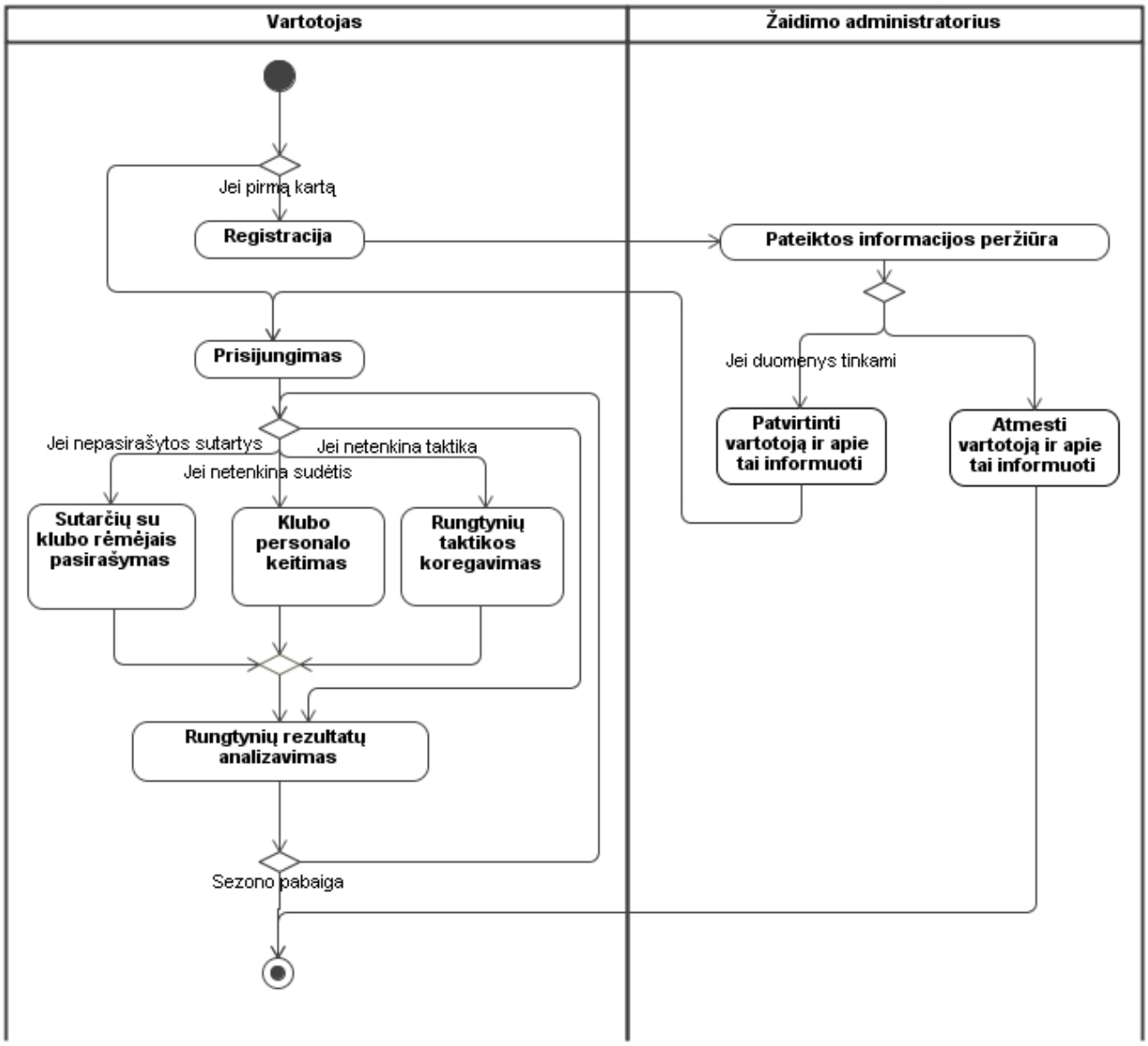
Analizės tikslas yra išnagrinėti metodus ir technologijas, kurias galima pritaikyti žaidimo vystymo problemoms spręsti, ir parinkti geriausiai tinkančius virtualaus krepšinio žaidimo tobulinimui. Taip pat išnagrinėti parinktų metodų savybes, kurias vėliau bus galima pritaikyti tobulinant sistemą.

Tyrimo objekto analizei taikysime objekto stebėjimo analizės metodą, nes jis gali geriausiai atskleisti sistemos esmę ir joje vykstančius procesus.

Esamų sprendimų analizei taikysime mokslinės literatūros analizės metodą bei kitų panašių sprendimų stebėjimo metodą. Šiais metodais galėsime palyginti jau sukurtus analogiškus sprendimus ir išanalizuoti literatūroje aprašytus teorinius metodus iškeltai problemai spręsti.

2.1.2. Virtualaus žaidimo veikimo analizė

Tobulinamame žaidime yra imituojama krepšinio klubų veikla. Kiekvienas užsiregistravęs vartotojas tampa krepšinio klubo vadovu. Jis yra atsakingas už visas klubo valdymo sritis: sutarčių su rėmėjais sudarymas, finansų planavimas, personalo paieška ir komplektavimas, žaidėjų treniravimas, taktikų nustatymas, klubo populiarinimas ir t.t.



2.1 pav. Vartotojo veiklos sezono metu procesų modelis

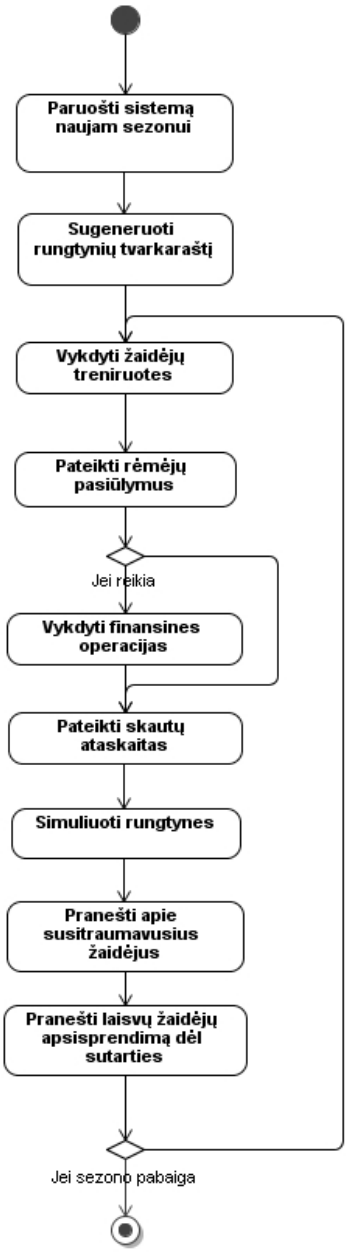
Kiekvienas klubas dalyvauja dvejose lygose – nacionalinėje ir tarptautinėje, kuriose siekia kuo geriau pasirodyti ir užimti kuo aukštesnę vietą. Nuo lygos, kurioje dalyvauja klubas, bei pasirodymo joje priklauso rėmėjų teikiamų pasiūlymų, taip pat ir klubo pajamų, dydis.

Žaidimo laikas suskirstytas į tam tikrus periodus – sezonus. Kiekvienas sezonas yra tos pačios struktūros (vyksta numatyti įvykiai, tuo pačiu metu prasideda ir baigiasi skirtingi tarpsniai, leidžiami/draudžiami tam tikri veiksmai ir pan.) ir pasibaigus vienam sezonui prasideda kitas. Žaidimo schemą iš vartotojo požiūrio galima panagrinėti vieno sezono laikotarpiu. 2.1 pav. pavaizduota apibendrinta žaidimo vartotojų veikla sezono metu.

Pirmą kartą žaidime apsilankęs vartotojas turi prisiregistruoti. Jo pateiktą informaciją peržiūri žaidimo administratorius ir nusprendžia, ar galima patvirtinti vartotoją. Jeigu vartotojas patvirtinamas, jam išsiunčiamas pranešimas, kad jau galima prisijungti prie žaidimo. Tapus žaidimo vartotoju reikia susipažinti su gauta komanda bei paruošti ją varžyboms. Pirmiausia reikia susirasti rėmėjus, kurie klubui teiks nuolatines pajamas. Jeigu komanda dar neturi sudariusi sutarčių su rėmėjais, reikia jas pasirašyti. Taip pat svarbu nustatyti klubą atstovaujančių žaidėjų potencialą ir nuspręsti ar reikia ką nors keisti. Kiekvienoms rungtynėms gali būti nurodoma treneriams rekomenduojama naudoti taktika. Rungtynėms įvykus, jų rezultatas yra analizuojamas ir daromos išvados: ar reikia keisti personalą, ar reikia keisti taktiką. Iki sezono pabaigos stengiamasi surasti optimalią sudėtį bei taktiką ir pasiekti kuo geresnius rezultatus.

Nepriklausomai nuo vartotojo veiksmų, sistema visada atlieka tam tikrus procesus. Panagrinėsime sistemos veiklos procesus sezono metu (2.2 pav.).

Prieš prasidedant sezonui reikia paruošti sistemą (pasendinti žaidėjus, įrašyti praeito sezono statistiką ir pan.) bei sugeneruoti būsimų rungtynių tvarkaraštį. Kiti veiksmai yra vykdomi kiekvieną sezono dieną. Yra imituojamos žaidėjų treniruotės, pateikiami rėmėjų pasiūlymai (toms komandoms, kurios jų dar neturi). Jeigu yra atitinkama diena, vykdomos finansinės operacijos – iš turimo klubo biudžeto nuskaičiuojamos algos personalui, pridedamos rėmėjų lėšos, atliekami kiti veiksmai. Nuolat yra pateikiamos skautų ataskaitos, imituojamos rungtynės, pranešama vartotojams apie tą dieną traumas gavusius žaidėjus bei laisvų žaidėjų apsisprendimus dėl kontrakto pasirinkimo (siūliusiems sutartis klubams). Šie veiksmai kartojami kiekvieną dieną iki sezono pabaigos.



2.2 pav. Sistemos veiklos sezono metu procesų modelis

2.1.3. Virtualaus krepšinio žaidimo vartotojų analizė

Sistemos vartotojų skaičius yra didelis – jų gali būti nuo kelių iki keliasdešimt tūkstančių. Vartotojai yra įvairaus amžiaus (10-60 m.), išsilavinimo ir įvairių tautybių žmonės. Visus vartotojus vienija tik bendras pomėgis – krepšinis.

Sistemos vartotojai yra keleto tipų:

- paprasti vartotojai – atsakingi tik už savo komandos valdymą;
- moderatoriai – turi tas pačias teises kaip ir paprasti vartotojai, tačiau šalia komandos valdymo dar gali atlikti dalį administravimo veiksmų;

- administratoriai – turi visas teises tiek komandos valdyme, tiek žaidimo administravime.

Atlikdami šį darbą realizuosime sistemos darbo sutrikimų bei papildomų paslaugų įsigijimo posistemes (2.1 lentelė).

2.1 Lentelė

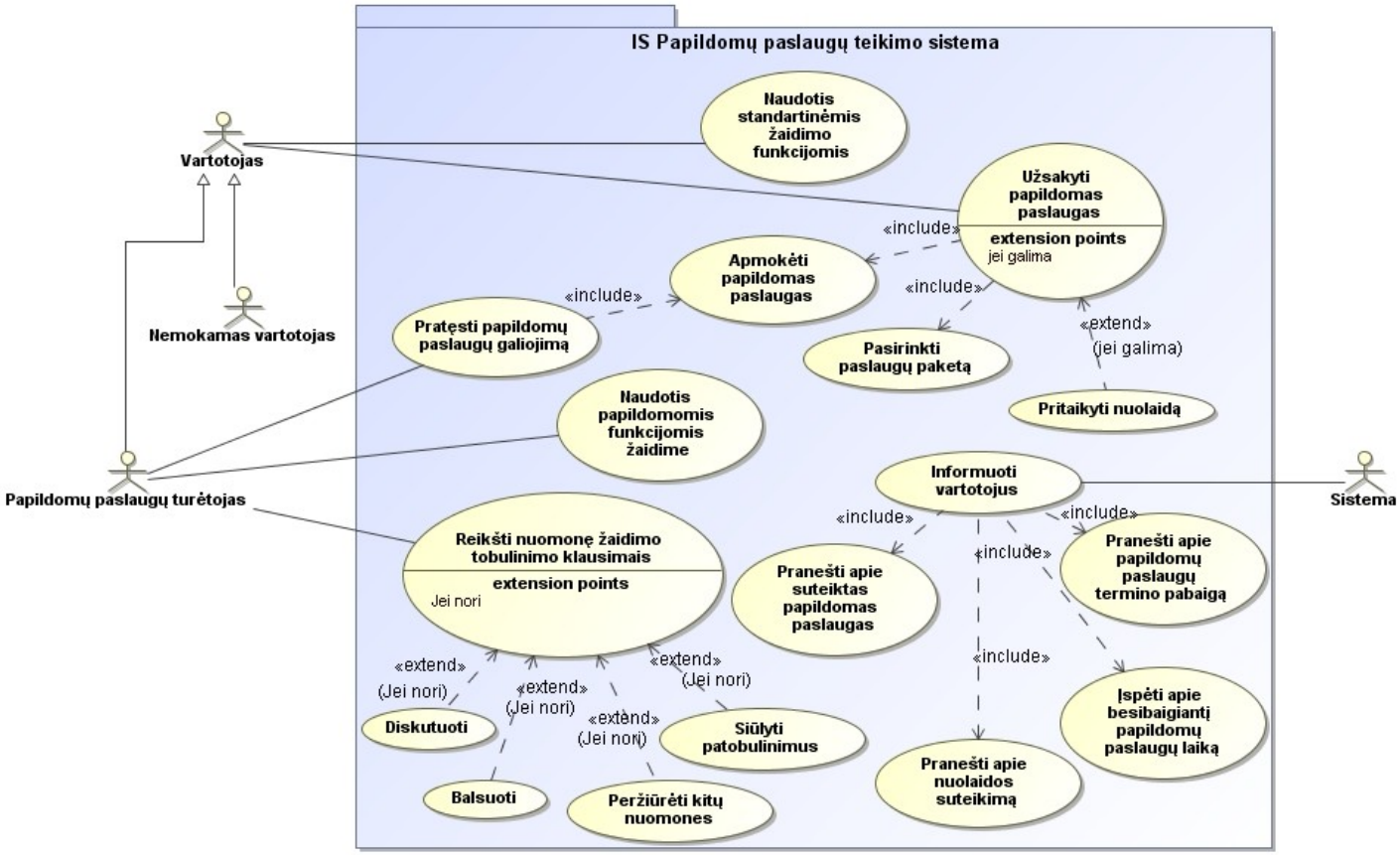
Problema	**Dabartinė situacija**	**Sprendimas**
Susidarius specifinėms aplinkybėms sistemoje pasitaiko loginio bei techninio pobūdžio sutrikimų	Apie pastebėtas klaidas vartotojai gali pranešti administracijai tik asmeninėmis žinutėmis, tačiau jos dažnai būna nepakankamai informatyvios	Galimybė pranešti apie pastebėtas klaidas – klaidų stebėjimo ir fiksavimo posistemėje
Standartiniuose failuose pateikiama informacija apie klaidas yra nepatogi skaityti ir reikalauja daug laiko	Failuose yra kaupiami standartiniai duomenys apie sistemoje pasitaikiusias klaidas, tačiau reguliariai juos skaityti ir ieškoti reikiamos informacijos yra nepatogu, užima daug laiko	Patogus klaidų atvaizdavimas – klaidų stebėjimo ir fiksavimo posistemėje
Rungtynių imitatorius yra nuolat tobulinamas. Būtina užtikrinti, kad jis visada veiktų tinkamai	Atnaujinus rungtynių imitatorių, sudėtinga jį ištestuoti, nes kiekvienos rungtynės yra skirtingos ir klaidos gali pasitaikyti tik keletoje rungtynių. Testuojant atnaujinimus, programuotojai negali peržiūrėti visų rungtynių ir atrasti visų nekorektiško veikimo atvejų	Keleto imitatorių naudojimas, imitatoriaus testavimas – klaidų stebėjimo ir fiksavimo posistemėje
Trūksta sistemos, kuri padėtų atrinkti aktyviausius vartotojus ir sudaryti sąlygas jiems bendrauti bei padėti spręsti svarbius žaidimo vystymui klausimus	Visi vartotojai reiškia savo nuomonę įvairiais klausimais. Kadangi jų amžius ir patyrimas yra labai skirtingas, didelė dalis diskusijų neteikia jokios naudos	Papildomų paslaugų naudotojų bendravimo aplinka – papildomų paslaugų teikimo posistemėje

Vartotojai nori gauti papildomų paslaugų	Papildomos paslaugos nėra įvestos į sistemą	Papildomos paslaugos, jas užsisakiusiems vartotojams – papildomų paslaugų teikimo posistemėje
Nėra sistemos, kuri priimtų mokestį iš vartotojų ir automatiškai suteiktų atitinkamas paslaugas	Nėra būdo apmokėti užsakytas paslaugas	Apmokėjimas už papildomas paslaugas – papildomų paslaugų teikimo posistemėje

Darbo metu bus ieškoma sprendimų, kurie padėtų efektyviai išspręsti 2.1 lentelėje išvardintas problemas. Tyrimo metu bus sukurti tokie artefaktai:

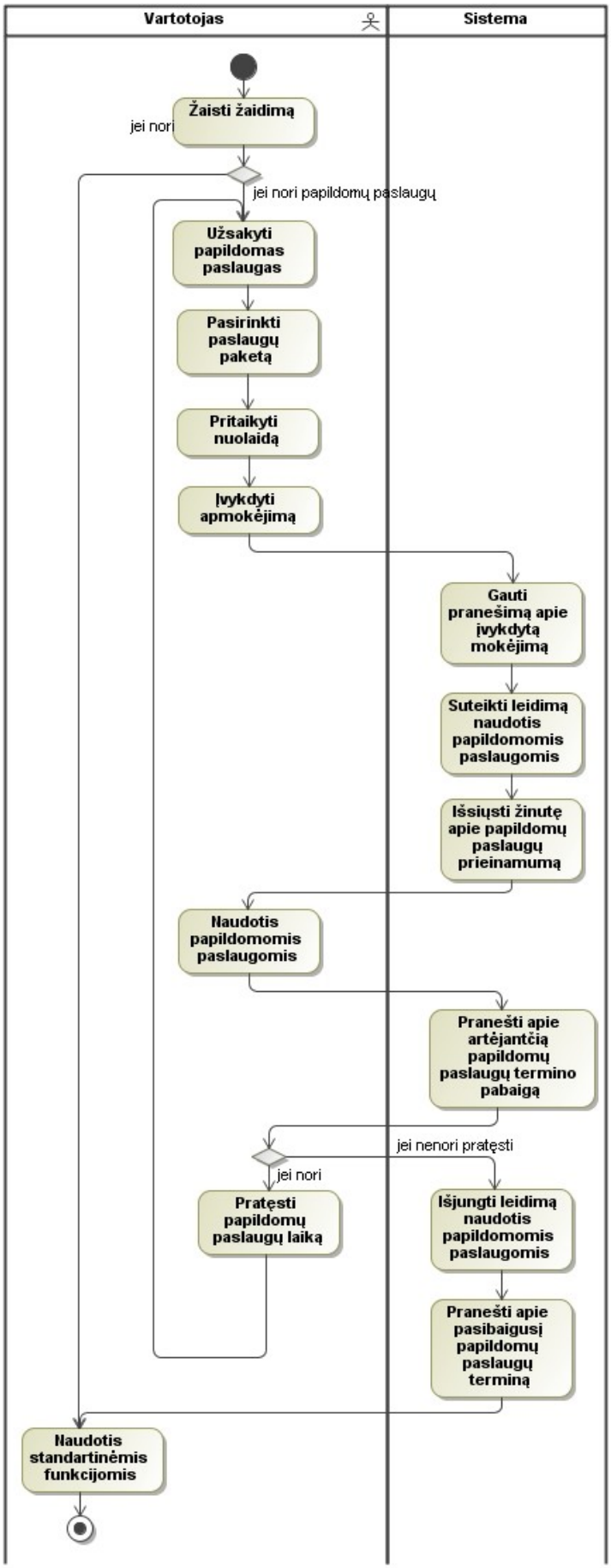
- Aktyviausių vartotojų atrinkimo, papildomų paslaugų paketų valdymo bei apmokėjimo sistema. Ją sudarys papildomų paslaugų paketai, užsakytų paslaugų apmokėjimas bei užsakytų ir apmokėtų paslaugų teikimas. Papildomų paslaugų turėtojai galės bendrauti tarpusavy ir su administracija specialiai jiems skirtoje aplinkoje.
- Informavimo apie sutrikimus sistema. Sistema suteiks patiems vartotojams galimybę pranešti apie pastebėtas logines ar technines klaidas. Taip pat bus automatiškai registruojamos sistemoje pasitaikiusios loginės ir techninės klaidos bei pateikiamos administracijai. Administracija galės peržiūrėti įvairių tipų klaidas.
- Rungtynių imitatoriaus testavimo sistema. Sistemoje bus galima turėti dvi imitatoriaus versijas rungtynių vykdymui. Viena (stabili ir ištestuota) – skirta oficialioms varžyboms, kita (patobulinta ir testuojama) – testavimo rungtynėms. Visi vartotojai galės testuoti naują imitatorių ir pranešti apie pastebėtas klaidas. Kai testuojamas imitatorius yra ištestuotas ir patikimas, jis pradedamas naudoti oficialioms rungtynėms.

2.3 pav. pateikta siekiamos sukurti papildomų paslaugų teikimo sistemos panaudojimo atvejų diagrama. Šiuo metu sistemoje nagrinėjamos paslaugos nėra. Visi vartotojai yra paprasti vartotojai ir gali naudotis tik standartinėmis žaidimo funkcijomis. 2.4 pav. pateikiama papildomų paslaugų įsigijimo veiklos diagrama.



2.3 pav. Papildomų paslaugų teikimo informacinės sistemos panaudojimo atvejų diagrama

Papildomų paslaugų paketų valdymo bei apmokėjimo sistemos įvedimas turėtų ženkliai padidinti iš žaidimo gaunamas pajamas. Taip pat bus suformuota aktyviausių ir labiausiai suinteresuotų žaidimo tobulinimu vartotojų bendruomenė. Sistemos įgyvendinimo rezultatai bus vertinami lyginant gautas pajamas prieš įgyvendinimą ir po jo, taip pat vartotojų įtraukimą į žaidimo vystymo procesą prieš papildomų paslaugų posistemės įgyvendinimą ir po jo.



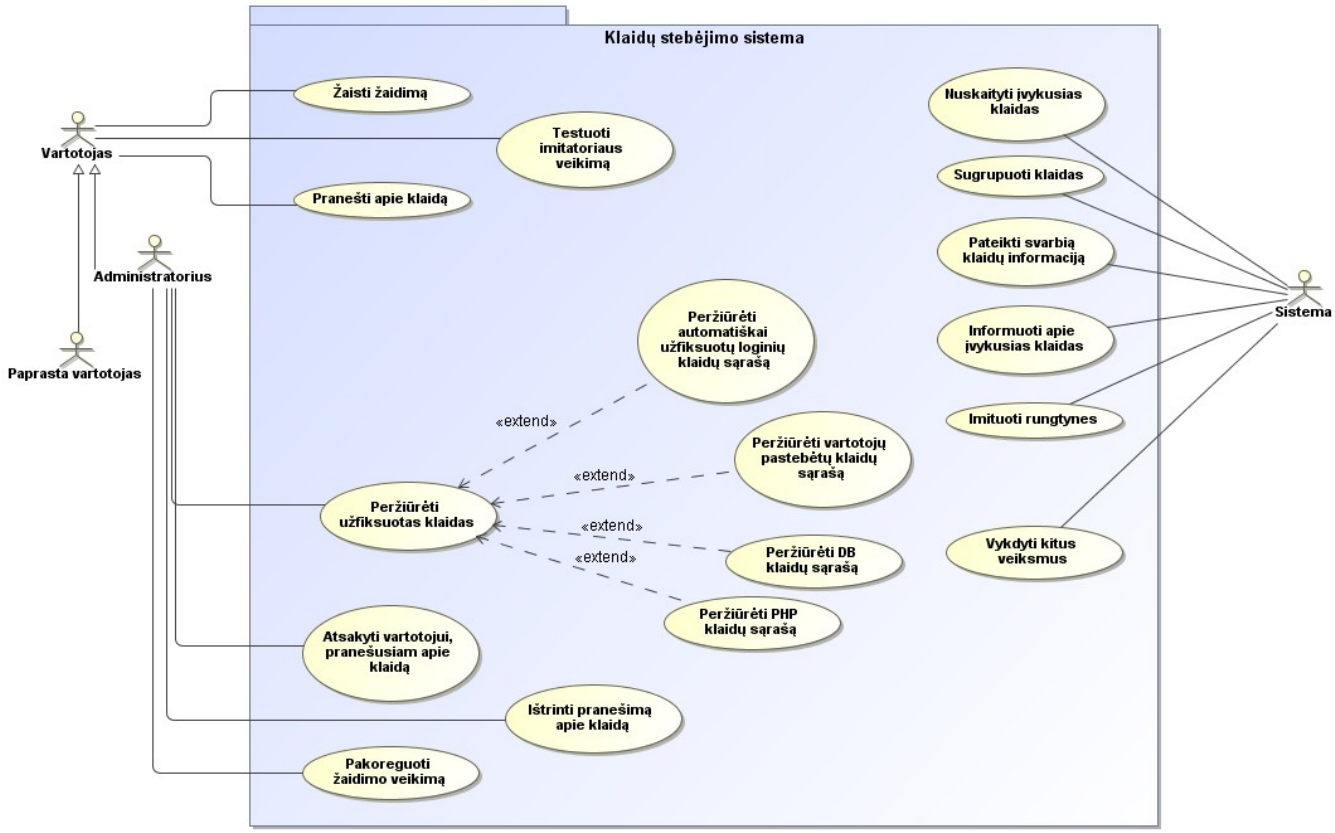
2.4 pav. Papildomų paslaugų užsakymo veiklos diagrama

2.5 pav. pateikta klaidų stebėjimo sistemos panaudojimo atvejų diagrama. Siekiama klaidų stebėjimo sistema bus sudaryta iš keleto dalių:

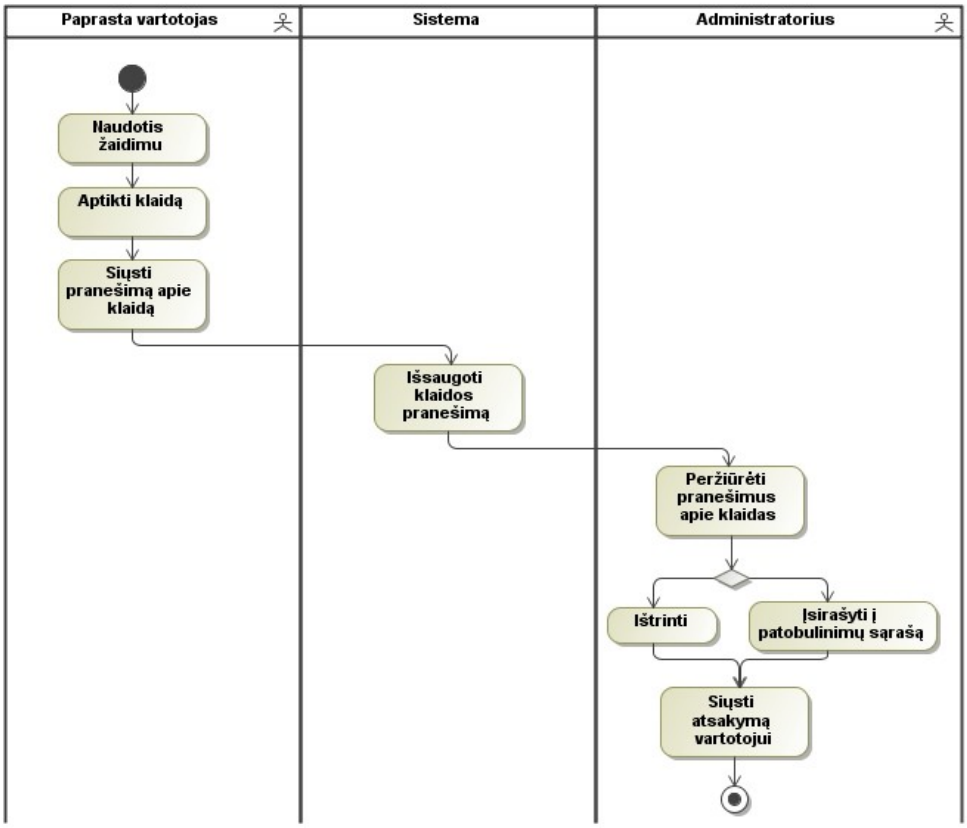
- Loginės klaidos
- Techninės klaidos (PHP ir duomenų bazės)
- Vartotojų pastebėtos klaidos
- Rungtynių imitatoriaus klaidos

Aptiktos klaidos bus pateikiamos administratoriui, kuris galės klaidas tvarkyti – peržiūrėti, atsakyti, ištrinti. Technines sistemos klaidas bus bandoma atskirti ir surūšiuoti automatiškai pagal pateiktas taisykles. Šiuo metu klaidų stebėjimas virtualiame krepšinio žaidime nėra įgyvendintas. 2.6 pav. pavaizduota pranešimo apie pastebėtą klaidą veiklos diagrama.

Informavimo apie sutrikimus sistema padės administratoriams greičiau pastebėti ir ištaisyti sistemoje pasitaikiusias klaidas. Šio patobulinimo rezultatas bus vertinamas lyginant sistemos patikimumą prieš sutrikimų fiksavimo sistemos įvedimą ir po jo.



2.5 pav. Klaidų stebėjimo sistemos panaudojimo atvejų diagrama



2.6 pav. Pranešimo apie pastebėtą klaidą veiklos diagrama

2.2. Virtualių žaidimo modelių ir realizavimo metodų analizė

2.2.1. Virtualių žaidimų kūrimo modelių analizė

Pagrindinė ir sunkiausiai įgyvendinama žaidimų kūrimo dalis yra algoritmas, kuris imituoja veikėjų mąstymą, elgesį. T.y. kūrėjai turi numatyti visus galimus variantus, juos logiškai susieti ir atitinkamai aprašyti. Remdamiesi korektišku imitatoriaus veikimo modeliu, programuotojai gali realizuoti jo veikimą.

Priklausomai nuo žaidimo tipo, imitatoriai dažniausiai naudojami trimis atvejais:

- Veikėjų judėjimui valdyti (pasiekti tam tikrą tikslą, išvengti kliūčių);
- Veikėjui priimti sprendimus (iš galimų variantų pasirinkti, kurį reikia atlikti);
- Veikėjui mąstyti strategiškai (naudojamas kai yra tam tikra komanda – numatomi įvairūs variantai, atsižvelgiama į kitus narius).

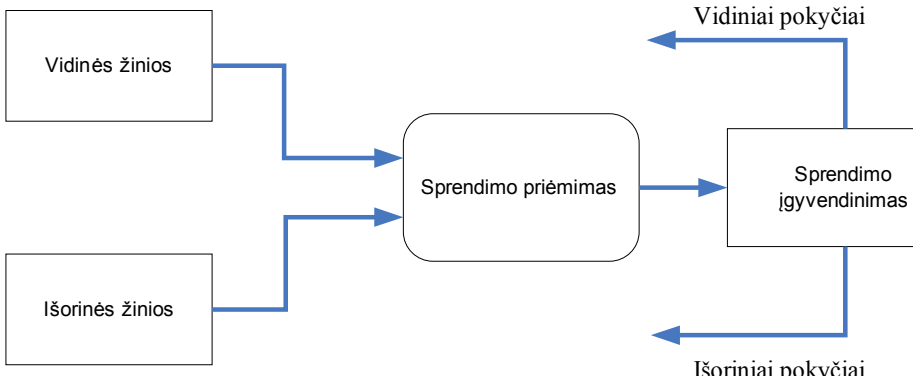
Kuriant imitatorių, jis gali būti modeliuojamas „iš viršaus į apačią“ arba „iš apačios į viršų“. T.y. „iš viršaus į apačią“ atveju, pirmiausia gali būti nustatoma bendra tendencija, o vėliau pagal ją modeliuojami atskirų veikėjų veiksmai (pvz., pagal paros metą pirmiausia nustatomas eismo intensyvumas gatvėje, o po to pagal tai

modeliuojamas mašinų ir pėsčiųjų elgesys). „Iš apačios į viršų“ atveju, atvirkščiai, pirmiausia nustatomas atskirų veikėjų elgesys, o sudėjus visus veikėjus į tą pačią aplinką, gaunasi bendras vaizdas.

Kad galėtų padaryti sprendimą, veikėjas turi turėti tam tikras žinias apie aplinką. Šios žinios būna vidinės ir išorinės. Išorinės žinios – tai gali būti jo pozicija, artimiausia jam aplinka ir pan. Vidinė aplinka – tai pačio veikėjo būsena, tikslas ir pan.

Atliktas veiksmas taip pat gali nustatyti pakeitimus išorinėje aplinkoje arba vidinėje. Vidiniai pakeitimai gali būti pasikeitusi veikėjo būsena, nuomonė ar tikslas. Išoriniai pakeitimai – kažkaip paveiktas kitas veikėjas ar kitas aplinkos objektas.

2.7 pav. pateikta sprendimo priėmimo ir jo poveikio schema.



2.7 pav. Sprendimo priėmimo schema [18, 303 p.]

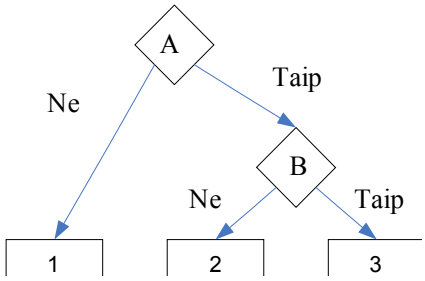
Sistemos elgesio aprašymui patogu naudoti formalias specifikacijas. Tai padeda ne tik aprašyti pageidaujamą veikimą, bet ir palengvina sistemos tobulinimą ateityje. Sprendimo priėmimas gali būti aprašomas įvairiais metodais, priklausomai nuo patogumo naudoti ir poreikių. [18], [19]

Sprendimų medžiai (angl. *Decision trees*) yra pakankamai greitai ir paprastai įgyvendinami. Paprastai nėra sudėtinga juos nagrinėti ir suprasti, nors yra ir sudėtingesnių šio metodo variantų, kurie leidžia modeliuoti ganėtinai sudėtingas situacijas.

Sprendimų medis prasideda nuo šakninio elemento. Kiekviename žingsnyje pasirenkamas tinkamas kelias ir einama prie kito elemento, kol galų gale gaunamas galutinis atsakymas.

Sprendimų medžio pavyzdys pavaizduotas 2.8 pav. A ir B yra sąlygos (klausimai), o 1, 2, 3 – priimtas sprendimas. Dažniausiai sąlygos yra paprastos ir patikrina vieną galimybę (sąlygose nėra IR, ARBA elementų).

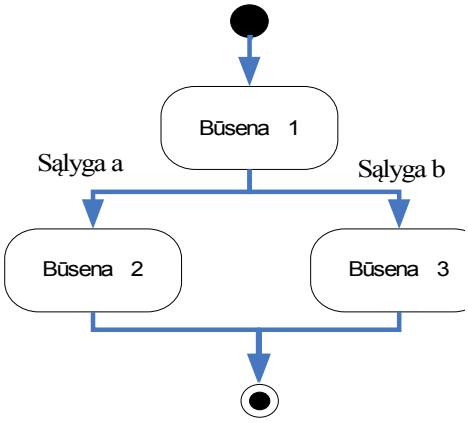
Tokia struktūra yra lengvai įgyvendinama, todėl dažnai naudojama nesudėtingiems sprendimams priimti. Visgi, naudojant šį metodą gali atsirasti klaidų dėl vykdymo metu pasikeitusių sąlygų. Pvz., sprendimo priėmimo metu sąlyga A galėjo pasikeisti ir priimtas galutinis sprendimas jau nebebus teisingas.



2.8 pav. Sprendimų medis

Būsenų mašinos (angl. *State Machines*). Taikant šį modelį, veikėjas visuomet yra tam tikroje būsenoje. Kiekvienai būsenai yra priskirti tam tikri veiksmai, kuriuos veikėjas visada atlieka būdamas toje būsenoje.

Būsenų mašinos modelį sudaro pradžios taškas, būsenos, ryšiai tarp būsenų bei pabaigos taškas. Norint pereiti iš vienos būsenos į kitą, turi būti tenkinamos tam tikros nustatytos sąlygos. Tik esant šioms sąlygoms galima pereiti iš vienos būsenos į kitą. Būsenų mašinos pavyzdys pavaizduotas 2.9 pav. Objektas gali įgauti būsenas: 1, 2 arba 3. Kad būtų pereita iš būsenos 1 į būseną 2, turi būti tenkinama sąlyga a, o iš būsenos 1 į būseną 3 – turi būti tenkinama sąlyga b.

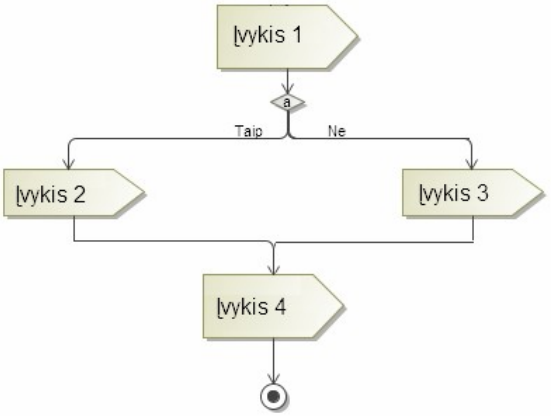


2.9 pav. Būsenų mašina

Būsenų mašinos modelis yra ganėtinai abstraktus ir pagal tą patį modelį galima padaryti daug skirtingų realizacijų. Taigi būsenų mašinos modelis yra labai lankstus, tačiau dėl to realizuojant gali tekti iškvietinėti daug funkcijų, kas gali reikalauti nemažai vykdymo laiko. [18]

Įvykių modelis (angl. *Event model*) atvaizduoja įvykių eiliškumą – koks įvykis gali įvykti ir po ko. Įvykus vienam įvykiui pagal iš anksto numatytas sąlygas parenkamas kitas įvykis.

2.10 pav. pateiktas įvykių modelio pavyzdys. Įvykus įvykiui *1*, tikrinama sąlyga *a*, kuris įvykis turi būti vykdomas toliau (2 arba 3). Pasibaigus bet kuriam iš šių įvykių vykdomas įvykis *4* ir pasiekiama modeliuojamo elemento pabaiga.



2.10 pav. Įvykių modelis

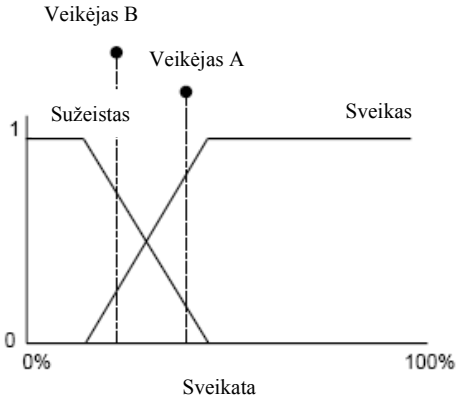
Įvykių modelis yra šiek tiek panašus į būsenų mašiną. Kurį modelį patogiau vartoti pasirenkama pagal modeliuojamo objekto logiką – ar paprasčiau aprašyti įvykiais, ar būsenomis.

Miglotoji logika (angl. *Fuzzy Logic*).

Aukščiau aptarti metodai nagrinėja du variantus – arba yra tenkinamos tam tikros sąlygos, arba ne. Tuo tarpu miglotoji logika nagrinėja ir tarpinius variantus, pavyzdžiui, veikėjas gali būti ne tik pavargęs arba nepavargęs, bet ir kažkiek pavargęs.

Naudojant tradicinę logiką, objektą galima aprašyti predikatais (savybėmis). Tam tiesiog reikia žinoti visų savybių reikšmes – ar yra, ar nėra. Pavyzdžiui, veikėjas gali būti pavargęs arba ne, sužeistas arba ne ir t.t. Visų objekto savybių aibė su reikšmėmis priklauso arba nepriklauso pilnai aprašo objektą.

Tuo tarpu miglotoje logikoje įvedamas predikato įvertinimas tam tikra reikšme. Pavyzdžiui, veikėjas gali būti 70% pavargęs ir 5% sužeistas. Taigi miglotoje logikoje savybės gali ne tik priklausyti ar nepriklausyti objekto aibei, bet ir dalinai priklausyti (2.11 pav.) Vienos savybės gali priklausyti labiau nei kitos. Šios objektą aprašančios aibės yra vadinamos miglotosiomis aibėmis (angl. *fuzzy sets*).



2.11 pav. Miglotosios logikos predikatų galimos reikšmės

Miglotąją logiką galima panaudoti sprendimų priėmimui. Pavyzdžiui, turime mašiną, kuri dideliu greičiu artėja prie sienos. Šiuo atveju aišku, kad mašina turi būti stabdoma. Priešingu atveju, jei priekyje nieko nėra, mašina turėtų didinti greitį. Tokie atvejai yra paprastai išsprendžiami pasitelkiant sprendimų medį ar būsenų mašiną. Tačiau vis tiek lieka neaišku, ką daryti tarpiniais variantais – kada ir kaip stipriai reikia pradėti spausti stabdžio ar greičio pedalą. Remiantis miglotąja logika, galima įvertinti tokius tarpinius variantus ir priimti tam tikrą sprendimą.

Šio metodo trūkumas, kad jis reikalauja daug resursų ir negali būti naudojamas su dideliu kiekiu duomenų. Išeitis – taikyti miglotąją logiką tik nedideliam kiekiui taisyklių.

Į tikslą orientuotas elgesys (angl. *Goal Oriented Behavior (GOB)*).

Kai yra daug veikėjų su daug skirtingų tikslų, būsenų mašinos naudojimas tampa ganėtinai sudėtingas. Tokiu atveju kur kas patogiau veikėjams priimti sprendimus, pasirenkant vieną iš galimų tikslų. Veikėjai tuo pačiu metu gali turėti keletą tikslų, kurie yra įvertinti balais. Didžiausią balą turintis tikslas turi didžiausią svorį. Veikėjai siekia tikslą įgyvendinti arba sumažinti jo vertę. Pavyzdžiui, jei veikėjas yra susižeidęs, jo tikslas yra išgyti. Visgi net esant sveikam šis tikslas vis tiek išlieka tik su maža verte (nes šiuo metu veikėjas ir yra sveikas).

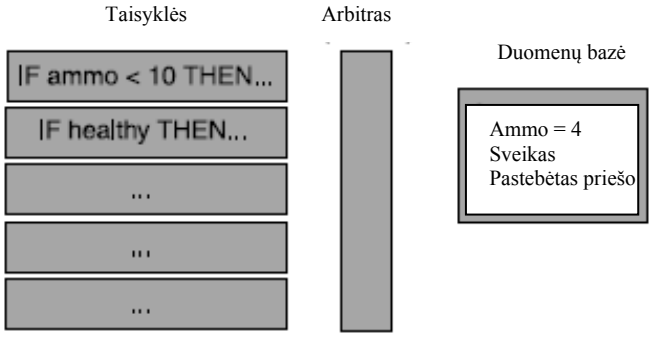
Kiekvienas veikėjas gali atlikti tam tikrus veiksmus. Kiekvienas veiksmas daro tam tikrą įtaką tikslams. Pavyzdžiui, vaistai gali pagerinti sveikatą. Nuo jų rūšies priklauso kaip stipriai pagerės sveikata. Tas pats veiksmas gali daryti įtaką ir kitam tikslui, pavyzdžiui mieguistumui. Tam tikri vaistai gali padidinti mieguistumo reikšmę.

Taip pat šiame metode svarbi veiksmo vykdymo trukmė. Reikia įvertinti, ar naudingiau atlikti ilgesnės trukmės veiksmą su didesniu poveikiu tikslui, ar trumpiau trunkantį, bet mažesnio poveikio.

Į tikslą orientuotas metodas atsižvelgia į visų veiksmų rezultatus. Kiekvienas veiksmas daro tam tikrą įtaką tikslams, o priimant sprendimą ši informacija yra naudojama nustatyti, kokią naudą kiekvienas veiksmas atneš.

Šis metodas yra nesunkiai realizuojamas, tačiau realizacija dažniausiai nebūna optimali.

Taisyklėmis paremtas metodas (angl. *Rule Based*) stipriai susietas su duomenų bazėje saugomomis reikšmėmis. Tai vienas iš seniausiai pradėtų naudoti metodų. Priimant sprendimą, remiamasi iš duomenų bazės gautomis reikšmėmis ir turimomis taisyklėmis (Jeigu... tai...). Tomis pačiomis sąlygomis gali galioti keletas taisyklių, todėl yra sukurta skirtingų algoritmų (pirmas tinkamas, mažiausiai panaudota taisyklė, labiausiai tinkanti taisyklė ir kt.) pagal kuriuos parenkama taisyklė, kuri bus naudojama (2.12 pav.)



2.12 pav. Taisyklėmis paremto metodo schema

Šis metodas tinkamas naudoti esant labai sudėtingiems sprendimų priėmimo algoritmams. [18]

Sprendimų priėmimų modelių palyginimas pateikiamas 2.2 lentelėje

2.2 Lentelė

Metodas	**Sudėtinga įgyvendinti**	**Tinka sudėtingoms sistemoms**	**Reikalauja resursų**	**Lengvai suprantamas**	**Apibrėžta realizacija**
Sprendimų medis	-	-	-	+	+
Būsenų mašinos	-	+	-	+	-
Įvykių modelis	-	+	-	+	-
Miglotoji logika	+/-	+/-	+	-	-
Į tikslą orientuotas elgesys	-	-	+/-	-	-
Taisyklėmis paremtas metodas	-	+	-	-	+

Išanalizavus ir palyginus sprendimų priėmimo modeliavimo metodus nustatyta, kad patogiausias metodas žaidimo logikai atvaizduoti yra būsenų mašina arba įvykių modelis. Šie metodai atitinka žaidimo logikos modeliavimui būtinus kriterijus: yra tinkami sudėtingoms sistemoms realizuoti, leidžia suprantamai sumodeliuoti logiką ir neapriboja realizavimo būdų.

2.2.2. Programinės įrangos paslaugų koncepcijos analizė

Šiuo metu žaidimų kūrimui pasaulyje yra dažniausiai naudojamos dvi technologijos:

- Programinės įrangos paslaugų technologija (*SaaS*)
- Įprasta programinė įranga, įrašoma į kiekvieną kompiuterį atskirai

Egzistuojančius sprendimus analizuosime lygindami jų tinkamumą spręsti iškeltoms problemoms: papildomų paslaugų apmokestinimui ir valdymui bei operatyvios informacijos apie pasitaikiusias klaidas surinkimui. Sprendimus lyginsime atsižvelgdami į jų efektyvumą, pritaikomumą ir kainą.

Abi aukščiau minėtos technologijos yra plačiai paplitę žaidimų sektoriuje. *MMOG* (*massive multiplayer online game* – masinis daugelio vartotojų internetinis žaidimas) tipo žaidimai yra kuriami naudojant tiek *SaaS* metodą (pvz., *Hattrick*, *Travian*, *eRepublik*), tiek ir platinant instaliuojamą kompiuteryje programinę įrangą (pvz., *Half-life*, *Lineage*, *FIFA 08*).

Panagrinėsime abi technologijas, naudojamas žaidimų kūrimui ir platinimui:

- **Programinės įranga kaip paslauga (*Software as a Service*) [3], [5], *SaaS***

Sukūrimo metai: 2001

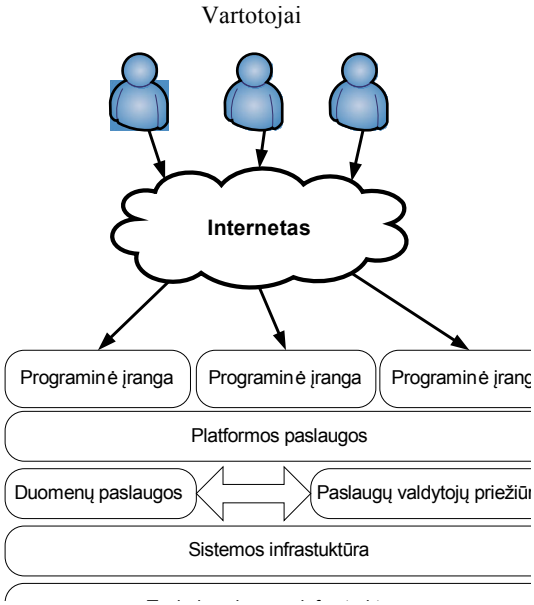
Autoriai: Software & Information Industry Association

***Informacijos šaltinis*:** http://www.whatissaas.net/

***Paskirtis*:** centralizuotai teikti, valdyti ir apmokestinti programinę įrangą

***Problemos, kurias leidžia spręsti*:** prieinamumas (iš visur, kur yra interneto ryšys), atnaujinimas (naujinimus reikia įdiegti tik į serverį ir visi vartotojai iš karto naudojasi nauja versija), apmokestinimas (vartotojams galima suteikti arba atimti teisę naudotis (papildomomis) paslaugomis pagal apmokėjimo būseną)

***Esmės aprašas*:** *SaaS* – tai programinės įrangos teikimo technologija, kai programa yra laikoma tiekėjo serveryje, o vartotojai gali ją pasiekti iš bet kur. Tiekėjai gali bet kada atnaujinti programą, leisti arba uždrausti vartotojui naudotis programa pagal apmokėjimus. Viename serveryje gali būti laikomos kelios skirtingos ar susijusios paslaugos. *SaaS* architektūra pateikiama 2.13 pav.



2.13 pav. *SaaS* architektūra [6, 98 p.]

- **Įrašoma (instaliuojama) programinė įranga [4]**

Sukūrimo metai: ~1950

***Informacijos šaltinis*:** http://www.softwarehistory.org

***Paskirtis*:** kurti ir platinti programinę įrangą

***Problemos, kurias leidžia spręsti*:** programos panaudojamumas (iš vieno kompiuterio galima naudotis bet kada, net kai nėra interneto ryšio), interaktyvumas (galima sukurti gerą grafiką ir bendravimą su vartotoju), greitis (turint reikalavimus atitinkančią įrangą, programą galima naudoti be jokių užtrukimų)

***Esmės aprašas*:** Gamintojas kuria produktą, kurį galima įrašyti į kompiuterį, ir išplatina. Vartotojas nusiperka produktą, įrašo į savo kompiuterį ir tame pačiame kompiuteryje gali visada juo naudotis.

Programinės įrangos paslaugų technologijos privalumai [3], [5]:

- Greitai ir paprastai pasiekiama (per interneto naršyklę);
- Lengvai atnaujinama versija;

- Nėra platinimo išlaidų (įrašinėti kompaktams);
- Paprasta valdyti vartotojų teises ir paslaugų teikimą.

Trūkumai:

- Norint naudotis sistema, būtinai reikia turėti prieigą prie interneto;
- Esant prastam interneto ryšiui, sistema veikia lėtai;

Įrašomos programinės įrangos technologijos privalumai:

- Galima naudotis net tada, kai nėra interneto ryšio;
- Galima pateikti didesnį vykdymo greitį;
- Galima sukurti interaktyvią sistemą ir pridėti įvairių grafikos elementų, kurie naudoja daug kompiuterio resursų.

Trūkumai:

- Programos atnaujinimas vyksta lėtai - reikia parduoti naują versiją visiems vartotojams;
- Reikalingos sudėtingos priemonės apsisaugoti nuo nelegalaus programos naudojimo;
- Priklausomai nuo programos, vartotojams gali reikti kompiuterio su labai geromis charakteristikomis.

2.3 Lentelė

Lyginimo kriterijai	***SaaS***	**Įrašoma programinė įranga**
Būtinas interneto ryšys	Taip	Ne
Reikia diegti papildomą programinę įrangą kompiuteryje	Ne	Taip
Galima pasiekti iš bet kurio kompiuterio	Taip	Ne
Reikia didelių kompiuterių pajėgumų	Ne	Gali reikti
Paprasta atnaujinti programą	Taip	Ne
Įdiegus vieną naują funkciją galima platinti naują versiją	Taip	Ne
Galima stebėti vartotojų reakciją į atnaujinimus ir greitai reaguoti	Taip	Ne
Vartotojų apmokestinimas	Greitas ir paprastas	Ganėtinai ilgas, sunku apsisaugoti nuo nelegalaus vartojimo
Platinimo išlaidos	Nėra	Yra
Galimybė nuomoti programinę įrangą	Taip	Ne
Lengvai kaupiama statistika	Taip	Ne
Reikia rūpintis technine įranga programai palaikyti	Taip	Ne

Iš programinės įrangos kaip paslaugos (*SaaS*) bei įrašomos programinės įrangos palyginimo rezultatų (2.3 lentelė), nustatėme, kad:

- *SaaS* pagrindiniai privalumai yra pasiekiamumas, nesudėtingas programos atnaujinimas bei vartotojų apmokestinimas;
- Įrašomos programinės įrangos – nebūtinas interneto ryšys, galima užtikrinti gerą greitį

Mūsų darbe iškeltoms problemoms spręsti būtina, kad sistema turėtų greitą grįžtamąjį ryšį su vartotojais, būtų paprastai ir greitai atnaujinama ir būtų galima nesudėtingai valdyti paslaugų suteikimą/uždraudimą vartotojams.

Atsižvelgiant į atliktą sprendimų analizę matome, kad šiuos kriterijus geriau įgyvendina programinės įrangos paslaugų technologija. Pasirinkus šį sprendimą, paslauga bus teikiama per internetą – bus nesudėtinga kontroliuoti suteikiamas paslaugas, atnaujinus sistemą ji iš karto atsinaujins visiems vartotojams ir nebus sistemos platinimo išlaidų.

Kadangi mūsų nagrinėjamai sistemai tinkamesnė yra *SaaS* technologija, ją panagrinėsime plačiau.

***SaaS* technologija** sudaro sąlygas nuomoti programinę įrangą, o ne ją pardavinėti. Tokiu būdu galima užtikrinti greitą ir nesudėtingą programos atnaujinimą iš karto visiems vartotojams. Nuomojant programą (suteikiant teises ja naudotis) yra kur kas paprasčiau kontroliuoti vartotojams teikiamas paslaugas ir sumažinama nelegalaus programinės įrangos vartojimo rizika.

SaaS technologijos bruožai

1. Programinė įranga yra laikoma nutolusiame serveryje;
2. Vartotojai gali naudotis programine įranga internetu iš bet kurios pasaulio vietos;
3. Dažniausiai vartotojams nereikia įdiegti jokių papildomų programų į savo kompiuterį – tereikia turėti interneto naršyklę [7], [11].

Programinė įranga kaip paslauga yra patogi naudotis tiek didelėms įmonėms, tiek pavieniams žmonėms. Yra įvairių modelių, kaip gali būti apmokestinami programinės įrangos vartotojai, todėl kiekvienas gali pasirinkti sau patogiausią:

- Prenumerata – imamas mėnesinis mokestis už tam tikrą vartotojų kiekį;
- Suvartojimas – mokestis paskaičiuojamas pagal naudojimosi programa kiekį. Pagal susitarimą, kiekis gali būti vertinamas įvairiais būdais – pvz., besinaudojančių kompiuterių skaičiumi;

- Paslaugos – imamas atskiras mokestis už naudojimąsi kiekviena paslauga;
- Vertė – mokestis imamas už atliktus kliento iškeltus tikslus;
- Fiksuotas mokestis – imamas iš anksto sutartas mėnesinis mokestis, teikiant iš anksto sutartas paslaugas tam tikram vartotojų skaičiui [5].

Programinės įrangos kaip paslaugos (*SaaS*) yra orientuotos į verslą arba tiesiogiai į vartotojus.

Į verslą orientuota programinė įranga yra parduodama įmonėms. Dažniausiai tai yra produktų valdymo paslaugos arba bendravimo su klientais priemonės.

Tiesiogiai į vartotojus orientuota programinė įranga yra skirta visuomenei. Tokios paslaugos gali būti apmokestinamos (pvz., mėnesiniu mokesčiu) arba teikiamos nemokamai. Jeigu vartotojai už naudojimąsi paslaugomis nėra tiesiogiai apmokestinami, pagrindinės pajamos yra gaunamos iš reklamos.

Žvelgiant iš vartotojų pusės, *SaaS* programos labai patogios, kai žadama naudotis neilgą laiko tarpą – tokiu atveju galima nusipirkti leidimą naudotis programa reikiamo ilgio periodui. Kitas privalumas – tokios programos nereikalauja vietos diske ir galingo kompiuterio.

Programos kūrėjams *SaaS* technologija suteikia galimybę gauti nuolatines pajamas iš prenumeratorių. Instaliuojamos programinės įrangos atveju, kūrėjai iš pirkėjo pajamas gauna tik vieną kartą. Parduota programinė įranga dažnai patenka į kompiuterinių nusikaltėlių (“piratų”) rankas ir dėl to stipriai sumažėja programų kūrėjų gaunamos pajamos. Naudojant *SaaS* technologiją, programų kūrėjai gali patys valdyti ir prižiūrėti priėjimą prie programos ir efektyviai apsisaugoti nuo įsilaužėlių. Programinę įrangą kaip paslaugą yra paprasta atnaujinti visiems vartotojams vienu metu, tokiu būdu panaikinant saugumo spragas ar patobulinant kitus programos elementus [7], [11].

Galimybė greitai atnaujinti programinę įrangą leidžia efektyviai užtaisyti saugumo spragas ir pasitaikiusias klaidas. Visgi tai nėra vienintelis pranašumas. *SaaS* technologija pagrįsta programinė įranga gali būti nuolat tobulinama, įvedant naujas funkcijas po vieną – taip atsiranda galimybė stebėti vartotojų reakciją į konkrečią funkciją bei greitai ištaisyti su nauja funkcija susijusias klaidas. Kūrėjams tokia galimybė suteikia pranašumų prieš instaliuojamą programinę įrangą, kur patobulinimai yra kaupiami ir vartotojams pristatomi visi kartu – su nauja versija [9].

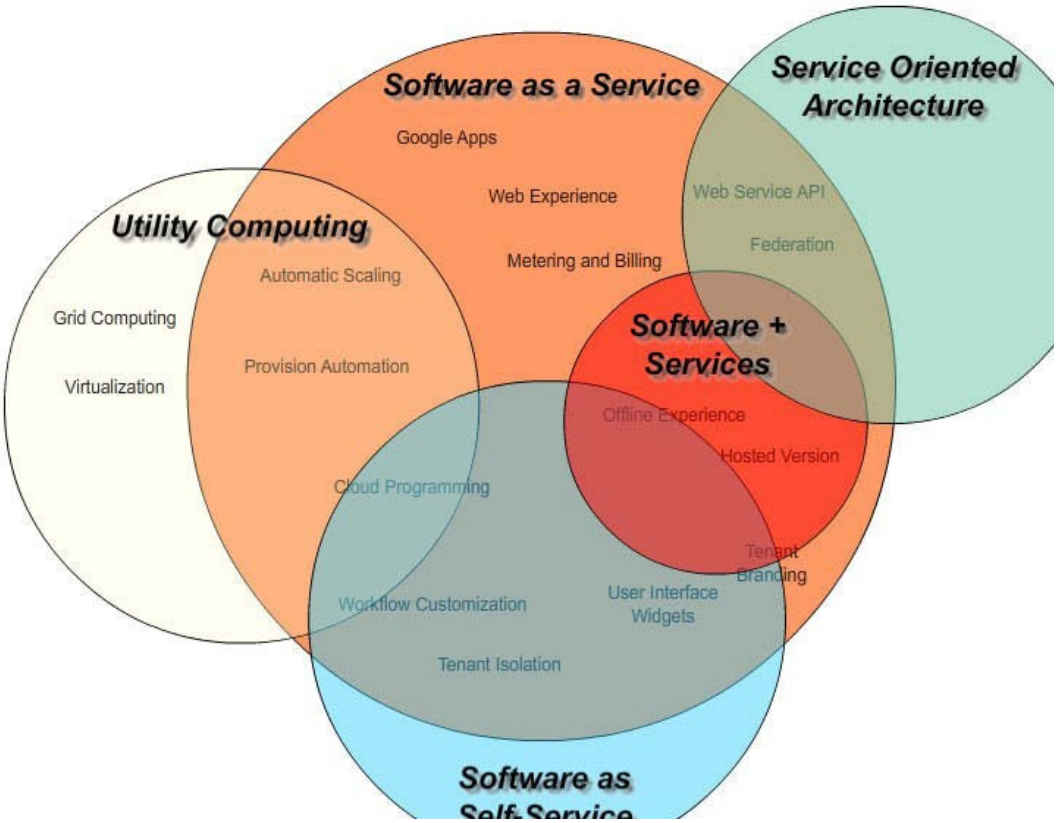
SaaS kūrėjams svarbu nuolat stebėti ir matuoti naudojimąsi programine įranga. Tam reikalingos specialios priemonės, kurios stebėtų sistemos būseną ir fiksuotų

analizei reikiamus dydžius. Toks stebėjimas leidžia kūrėjams spręsti apie turimą techninę įrangą ir daryti išvadas ar ji yra pakankamai galinga, ar reikia kažką keisti. [7]

Dar vienas svarbus dalykas *SaaS* technologijoje yra galimybė programinę įrangą pritaikyti pagal konkretaus kliento poreikius. Priklausomai nuo programinės įrangos ir klientų, gali būti keičiami įvairūs programos aspektai: išvaizda, reikiamos programos funkcijos ar prieiga prie programos. Todėl *SaaS* programinė įranga dažnai turi dvejopą sąsają – vieną skirtą vartotojams, kitą – administratoriams. [8]

Programinė įranga kaip paslauga (*SaaS*) yra viena iš internetinės kompiuterijos (*Cloud computing*) kategorijų. *Cloud computing* sąvoka apima viską, kas susiję su internete talpinamų paslaugų apdorojimu ir tiekimu. Visos *Cloud computing* technologijų galimybės yra pavaizduotos 2.14 pav. Pagrindinėmis laikomos trys:

- Infrastruktūra kaip paslauga (*Infrastructure as a service, IaaS*; taip pat vadinama *Utility Computing*) – pagal pareikalavimą teikiamas virtualus serveris su unikaliu IP adresu, talpinimo vieta ir specialiais nustatymais. Klientai naudoja tiekėjo taikomosios programos sąsają (*application program interface, API*), leidžiančią paleisti, sustabdyti ir keisti nustatymus jų virtualiame serveryje;
- Platforma kaip paslauga (*Platform as a service, PaaS*; taip pat vadinama *Service Oriented Architecture*) – programinės įrangos bei produkto tobulinimo įrankiai, laikomi tiekėjo serveriuose. Tai tam tikri karkasai, veikiantys kaip programų vykdymo aplinka ir skaičiavimų platforma. Per internetą programuotojai kuria taikomąsias programas tiekėjo platformoje.
- Programinė įranga kaip paslauga (*Software as a Service, SaaS*) – tiekėjas teikia tiek programinę, tiek techninę įrangą. Tiekėjas ir klientas bendrauja per specialią internetinę svetainę [7], [10].



2.14 pav. *Cloud computing* technologijų galimybės [7, 34 p.]

2.2.3. Interneto sistemų testavimo metodų analizė

Vienas iš jau minėtų *SaaS* privalumų – galimybė realiu laiku stebėti sistemą ir jos būseną. Tai leidžia testuoti sistemą ne tik kūrimo metu, bet ir vėliau – pasitelkiant vartotojus.

Sistemos testavimą gali sudaryti daug skirtingų aspektų:

1. Funkcionalumo testavimas (angl. *functionality*) – tai įvairių sistemos elementų veikimo patikrinimas. Jo metu nustatoma, ar sistema tikrai veikia taip kaip numatyta ir tikimasi. Testuojant remiamasi programos specifikacija. Patikrinama ar vykdymo metu nėra klaidų ar įspėjimų, ar jokios galimos įvesties reikšmės nesudaro sąlygų klaidoms atsirasti.
2. Patogumo naudotis (angl. *usability*) testavimas. Nustatoma, ar puslapiu naudotis yra patogu, ar paprasta sistemoje susigaudyti naujam vartotojui, ar sudėtingesnėse vietose yra pateikiama pagalba padedanti susigaudyti. Šis testas taip pat atkreipia dėmesį į sistemos išvaizdą – naudojamas spalvas, šriftus ir kitas formatavimo priemones. Tikrinama ar nėra vaizdinių priemonių pertekliaus, ar jos

neerzina vartotojo. Taip pat tikrinamas turinys, jo taisyklingumas ir suprantamumas.

3. Sąsajų testavimas (angl. *interface*) padeda nustatyti ar atskiri moduliai tinkamai bendrauja vienas su kitu. Tai padeda užtikrinti, kad atsiradus klaidai viename iš modulių likusių sistemos dalių darbas nebus sutrikdytas, o vartotojui bus parodytas atitinkamas pranešimas.

4. Suderinamumo testavimas (angl. *compatibility*). Testuojamas sistemos suderinamumas su įvairiais išoriniais produktais/platformomis. Internetinėms programoms ypatingai svarbus suderinamumas su įvairiomis naršyklėmis. Nors kartais sunku pasiekti, kad visos naršyklės vienodai atvaizduotų turinį, testuojant suderinamumą galima užtikrinti, kad pagrindinės naršyklės turinį atvaizduotų kiek įmanoma panašiau į pageidaujamą rezultatą.

5. Vykdymo testavimas (angl. *performance*). Tikrinama ar sistema gali atlaikyti pakankamai didelius vartotojų, užklausų srautus. Taip pat tikrinama sistemos reakcija į didelį klaidų skaičių.

6. Saugumo testavimas (angl. *security*) apima vartotojo teisių tikrinimą, serverio katalogų ir failų prieinamumą, apsisaugojimą nuo automatinių registracijų/prisijungimų bei klaidų ar bandymų apeiti apsaugas įrašymą ir saugojimą.[12] [17]

Augant internetinių programų skaičiui ir joms užimant vis didesnę rinkos dalį, šių programų testavimas taip pat įgyja vis didesnę reikšmę. Šių programų vartotojai dažnai yra iš įvairių pasaulio šalių, skirtingi patys kaip asmenys bei naudoja skirtingas priemones. Todėl tradiciniai testavimo būdai, įtraukiantys vieną ar keletą testuotojų, gali būti neefektyvūs ir negali atskleisti visų problemų, kurios gali atsirasti naudojimo metu.

Galutinių vartotojų įtraukimo į testavimo procesą privalumai:

- Kadangi testavime gali dalyvauti visi interneto vartotojai, nėra jų skaičiaus ar tipo limitų;
- Ištestuojami tokie scenarijai, kurių kūrėjai galėjo nenumatyti;
- Sutaupoma testavimui skirto laiko. Tuo pačiu metu kai sistemą testuoja galutiniai vartotojai, kūrėjai gali toliau tobulinti sistemą ar taisyti pastebėtas klaidas.

Norint, kad vartotojai galėtų testuoti sistemą, pirmiausia ją reikia atitinkamai paruošti – sukurti bendravimo priemones tarp vartotojų ir kūrėjų. Vartotojai turi turėti

galimybes patogiai pateikti informaciją apie pastebėtas klaidas ar nuomonę apie sistemos naudojamumą, o kūrėjai – galimybes stebėti vartotojų atsiliepimus ir į juos reaguoti. [13]

Kuriant internetines sistemas taip pat svarbu yra stebėti ir kaupti informaciją apie joje pasitaikiusias automatiškai užfiksuotas klaidas. Tokios klaidos gali būti:

- Vykdymo metu serveryje užfiksuotos klaidos. Jos gali sustabdyti sistemos darbą (angl. *fatal errors*) arba tik įspėti (angl. *warnings*). Šių klaidų fiksavimas ir šalinimas yra labai svarbus norint užtikrinti patikimą sistemos veikimą.
- Turinio formavimo klaidos. Pavyzdžiui *HTML* klaidos, kurios sutrikdo puslapio atvaizdavimą. Jos gali lemti, kad vartotojams rodomas iškraipytas, nevaizdus, o kartais galbūt ir nesuprantamas turinys. Tai gali turėti labai nepageidaujamų pasekmių:
  - Neteisingai suformuoti puslapiai gali turėti saugumo spragų.
  - Netinkamai suformuotas turinys gali priversti puslapį ilgiau krauti informaciją.
  - Standartų neatitinkantis turinys skirtingose naršyklėse gali būti atvaizduojamas laba skirtingai ir iškraipyti pageidaujamą vaizdą.
  - Esant netinkamai suformuotam turiniui, kai kurios naršyklės gali atvaizduoti dalį turinio, o kitą dalį tiesiog užslėpti.
  - Paieškos varikliams gali kilti sunkumų indeksuojant netinkamai suformuotus puslapius. [14]

Internetinės sistemos neturi savo atskiros sąsajos, per kurią vartotojas gali ja naudotis. Vartotojams yra būtina tarpinė priemonė – naršyklė. Tai taip pat išskiria internetines programas nuo instaliuojamos programinės įrangos. Ši tarpinė priemonė (naršyklė) dažnai gali sukelti papildomų problemų vartotojams naudojantis sistema.

Be jau minėto skirtingo turinio atvaizdavimo skirtingose naršyklėse, gali iškilti ir kitų problemų. Pavyzdžiui, kiekviena naršyklė turi "Atgal" mygtuką (angl. *"Back"*). Internetinės programinės įrangos kūrėjai privalo atsižvelgti į šią naršyklių savybę ir kiekvienu atveju numatyti, kas atsitiks jei vartotojas pasinaudos šia savybe.

Kita išskirtinė naršyklių savybė – galimybė pačiam vartotojui įrašyti internetinį adresą, o taip pat ir galimybė per jį pakeisti programai perduodamus parametrus. Kuriant programinę įrangą šis atvejis turi būti apgalvotas ir nuo jo apsisaugota.

Dar vienas tikėtinas klaidų šaltinis (naudojant interneto naršyklę) yra galimybė vienu metu atsidaryti keletą sistemos langų. Sistema turi numatyti tokius atvejus ir priimant duomenis iš vartotojo būtinai pasitikrinti, ar gauti duomenys yra būtent tai, ko pageidavo vartotojas. Pvz., vartotojas galėjo atsidaryti du langus su skirtingais pasirinkimais (pirkiniais ar kitais sistemos elementais). Apsvarstęs abi galimybes, vartotojas gali pasirinkti tiek atidarytąją anksčiau, tiek atidarytąją vėliau. Svarbu užtikrinti, kad į sistemą bus nusiųsti būtent tie duomenys, kurie atspindi vartotojo pasirinkimą (o ne paskutiniojo atidaryto lango duomenys).

Tad į testavimo planą reikalinga įtraukti ir šias savybes, kurios būdingos naršyklėms ir kurios gali sudaryti galimybes atsirasti klaidoms išskirtinai internetinėse programose. [15] [16]

2.4 lentelėje pateikiamas esamų testavimo priemonių palyginimas.

2.4 Lentelė

Lyginimo kriterijai	***Simple Test***	***Compuware***	***Selenium***	***Segue***	***AETG***	***QA Inspect***	***User Testing***
Unit testai	+	+	+	+			
Nuorodų tikrinimas	+	+	+	+			
Formų tikrinimas	+	+	+	+			
PHP klaidų tikrinimas	+	+	+	+			
Duomenų bazės klaidų tikrinimas		+			+		
Suderinamumo testavimas	+		+				
Vykdymo (*performance*) testavimas		+		+			
Saugumo testavimas						+	
Naudojimo (usability) testavimas							+

Yra sukurta įvairių testavimo priemonių, kurios gali padėti tikrinti internetui skirtas programas. Visgi mūsų realizuojamos programos atveju, nei vienas iš jau esančių įrankių negali pilnai ištestuoti sistemos, atsižvelgiant į visus reikiamus aspektus. Sistemos logiką gali testuoti tik į ją įsigilinę vartotojai, kurie puikiai supranta ką sistema daro, kas yra korektiška ir kas – ne. Kai kurios testavimo

programos siūlo testavimą, pasitelkiant įvairius interneto vartotojus, tačiau mūsų atveju tai nėra tinkamas variantas, nes reiktų pakankamai ilgo laikotarpio, kol jie įsigilintų į mūsų sistemą ir ją perprastų.

Kadangi nėra tokios testavimo programos, kuri pilnai atitiktų mūsų reikalavimus, tinkamiausias variantas yra kurti vientisą testavimo sistemą, kuri apimtų visas sistemos dalis ir galėtų atlikti visą reikalingą testavimą.

2.3. Virtualaus žaidimo tobulinimo proceso tyrimo formuluotė

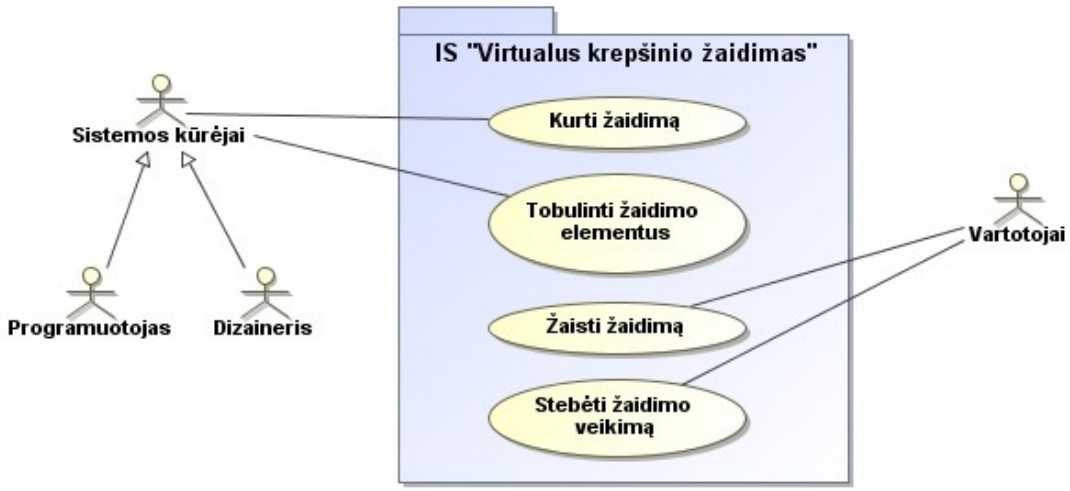
Darbo objektas – tai realiai veikiantis virtualus krepšinio žaidimas, turintis savo vartotojus. Žaidimas turi visas pagrindines funkcijas, tačiau yra nuolat tobulinamas ir vystomas, kad išlaikytų susidomėjimą ir taptų vis geresnis. Iš esmės, nėra žaidimo tobulinimo ribų, nes visuomet galima plėsti funkcionalumą.

Kad žaidimas vystytųsi sklandžiai, reikalinga ne tik ilgalaikė strategija, bet ir tam tikros priemonės, kurios padėtų palaikyti šį procesą ir užtikrintų jo tęstinumą. Todėl šio **darbo tikslas** yra patobulinti virtualų krepšinio žaidimą ir palengvinti jo vystymą, sukuriant imitacinį rungtynių modelį ir įvedant papildomas posistemes, kurios leis vystyti žaidimą iki kito tobulinimo etapo.

Šiam tikslui pasiekti iškelti **uždaviniai:**

1. išanalizuoti žaidimų imitatorių aprašymo modelius ir nustatyti patogiausią variantą virtualaus krepšinio žaidimo veikimo logikai aprašyti;
2. išanalizuoti *SaaS* koncepcijos privalumus ir nustatyti, kaip *SaaS* metodologija gali padėti patobulinti virtualų krepšinio žaidimą;
3. išanalizuoti internetinių sistemų testavimo modelius ir pritaikyti analizės duomenis žaidimo vystymo procesui;
4. sudaryti rungtynių imitatoriaus veikimo modelį;
5. suprojektuoti sistemą patobulinančias programines paslaugas;
6. realizuoti sistemos patobulinimus;
7. ištestuoti sistemos patobulinimus;
8. atlikti eksperimentą ir įvertinti tyrimo rezultatus.

2.15 pav. pavaizduota žaidimo sistema žaidimo vystymo aspektu: kūrėjai kuria ir tobulina žaidimą, o vartotojai žaidžia žaidimą ir tuo pat metu stebi kaip jis veikia – žaidimo logiką bei pasitaikančias klaidas.



2.15 pav. Virtualaus krepšinio žaidimo principas

Kaip jau buvo minėta analizės pradžioje, žaidimo vartotojų skaičius yra didelis ir jie yra labai įvairūs (tautybių, kalbų ir kitais požiūriais). Vartotojai gali turėti visiems bendrų norų ir pageidavimų (susijusių su pačia žaidimo esme), o taip pat ir tokių, kurie aktualūs tik vienai ar kelioms vartotojų grupėms (pavyzdžiui, tam tikra kalba).

Žaidimo kūrėjų tikslas yra turėti patenkintus vartotojus ir išlaikyti jų susidomėjimą žaidimu, todėl būtina atsižvelgti į vartotojų nuomonę ir turėti geras sąlygas tobulinimui.

Šiame darbe nagrinėjamos technologijos ir metodai, kuriuos galima pritaikyti, siekiant produktyvaus sistemos vystymo. Tai apima keletą aspektų:

- Programinės įrangos pateikimo modeliai;
- Testavimo metodai ir priemonės;
- Rungtynių logikos (sprendimų priėmimo) modeliavimo metodai.

Darbo metu bus realizuotos posistemės, palengvinsiančios žaidimo tobulinimą.

Papildomų paslaugų teikimo posistemės įgyvendinimas ne tik padidins pageidaujamų savybių kiekį, bet ir padės formuoti žaidėjų, besidominčių sistema ir suinteresuotų jos tobulinimu, bendruomenę. Turint suinteresuotų žaidėjų bendruomenę bus galima ja remtis sprendžiant svarbiausius klausimus ir nustatant reikalingiausius trūkstamus patobulinimus.

Klaidų stebėjimo ir fiksavimo posistemė padės greitai nustatyti ir užfiksuoti sistemos klaidas. Ši posistemė apims įvairių tipų klaidų nustatymą: loginių, *php*, duomenų bazės ir imitatoriaus. Tai palengvins naujų savybių įtraukimą į sistemą.

Užsibrėžtų tikslų įgyvendinimas bus matuojamas keletu kriterijų:

- Gaunamų pajamų iš vartotojų pokyčiu prieš papildomų paslaugų sistemos įvedimą ir po to;
- Vartotojų bendruomenės įtraukimą į žaidimo vystymo procesą prieš ir po papildomų paslaugų sistemos įvedimo;
- Sistemos patikimumą prieš sutrikimų sistemos įvedimą ir po to.

2.4. Analizės išvados

1. Išnagrinėjus modelius, tinkamus aprašyti sprendimų priėmimo logiką, buvo pasirinkti būsenų mašinų ir įvykių modeliai, kuriais patogiausia remtis, aprašant žaidimo imitatoriaus veikimą.
2. Programinės įrangos pateikimo technologijų analizė parodė, kad esama nagrinėjamo krepšinio žaidimo sistema iš esmės atitinka *SaaS* technologijos principus ir ji yra tinkamiausia tolesniam žaidimo vystymui.
3. Apžvelgus testavimo metodus ir priemones nuspręsta kurti atskirą žaidimo sutrikimų fiksavimo sistemą, pasinaudojant išnagrinėtais testavimo metodais.
4. Siekiant pagerinti žaidimą ir palengvinti jo vystymą, nuspręsta į žaidimą įvesti naujus funkcionalumus: sutrikimų fiksavimo posistemę bei papildomų paslaugų teikimo posistemę.

3. Virtualaus krepšinio žaidimo imitatoriaus modelis

3.1. Žaidimo imitatoriaus aprašymas būsenų modeliais

Šiame skyrelyje bus nagrinėjamas rungtynių imitatoriaus veikimas. Kadangi ši vieta yra viena iš esminių, svarbu, kad joje būtų kuo mažiau klaidų. Išnagrinėjus kaip veikia rungtynių imitatorius, bus lengviau rasti klaidingas vietas ir jas ištaisyti.

Aprašant rungtynių imitatoriaus veiklos taisykles naudojami šie apibrėžimai:

- R – rungtynių eiga
- e1 – create();
- e2 – vykdytiRungtynes();
- e3 – baigtisRungtyniuLaikui();
- e4 – kovotiDelPirmojoKamuolio();
- e5 – atliktiMetimaIKrepsi();
- e6 – atliktiKamuolioVaryma();
- e7 – atlktiPrasiverzima();
- e8 – atliktiKamuolioPerdavima();
- e9 – baigtisAtakosLaikui();
- e10 – baigtisKelinioLaikui();
- e11 – mestiIKrepsi();
- e12 – gautiBloka();
- e13 – nepataikytiMetimo();
- e14 – pataikytiMetima();
- e15 – atkovotiKamuoli();
- e16 – prasizengti();
- e17 – varytisKamuoli();
- e18 – suklysti();
- e19 – verztis();
- e20 – perduotiKamuoli();
- e21 – isprovokuotiPrazanga();
- e22 – nepataikytiBaudosMetimo();
- e23 – pataikytiBaudosMetima().

Taisyklių aprašyme naudosime šiuos žymėjimus:

$a \cdot b$ –abi operacijos būtinai vyksta viena po kitos;

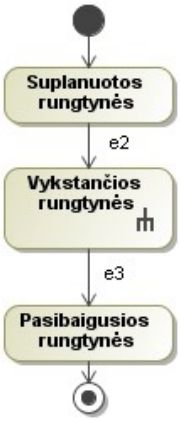
$a + b$ – ARBA operatorius (gali įvykti kažkuri viena iš operacijų);

a^* – ciklas (operacija gali būti kartojama vieną ar daugiau kartų);

$1 + a$ – neprivaloma operacija (operacija *a* gali įvykti arba ne)

3.1 pav. pavaizduota esybės Rungtynės būsenų schema bei įvykiai, kurie iššaukia būsenų kaitą. Rungtynių eiga gali būti aprašoma:

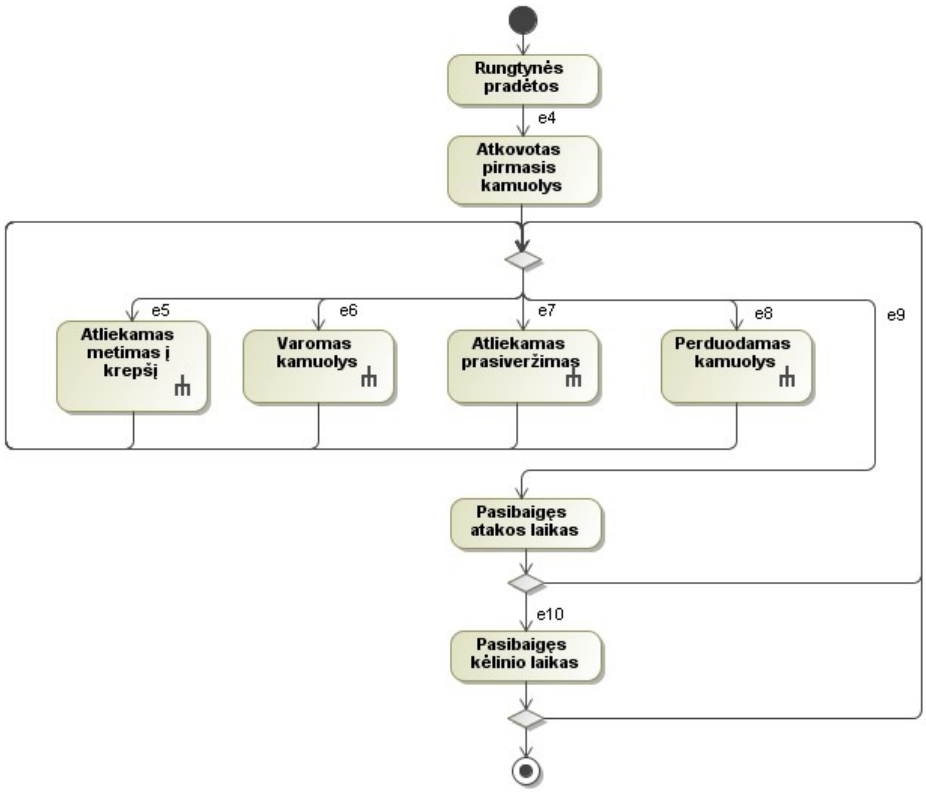
$R = e1 \cdot e2 \cdot e3$;



3.1 pav. Esybės Rungtynės būsenų modelis

3.2 pav. detaliau pavaizduotas būsenų modelis, vykstant rungtynėms.

$e2 = e4 \cdot (e5 + e6 + e7 + e8 + e9 + e9 \cdot e10)^* \cdot e9 \cdot e10$;

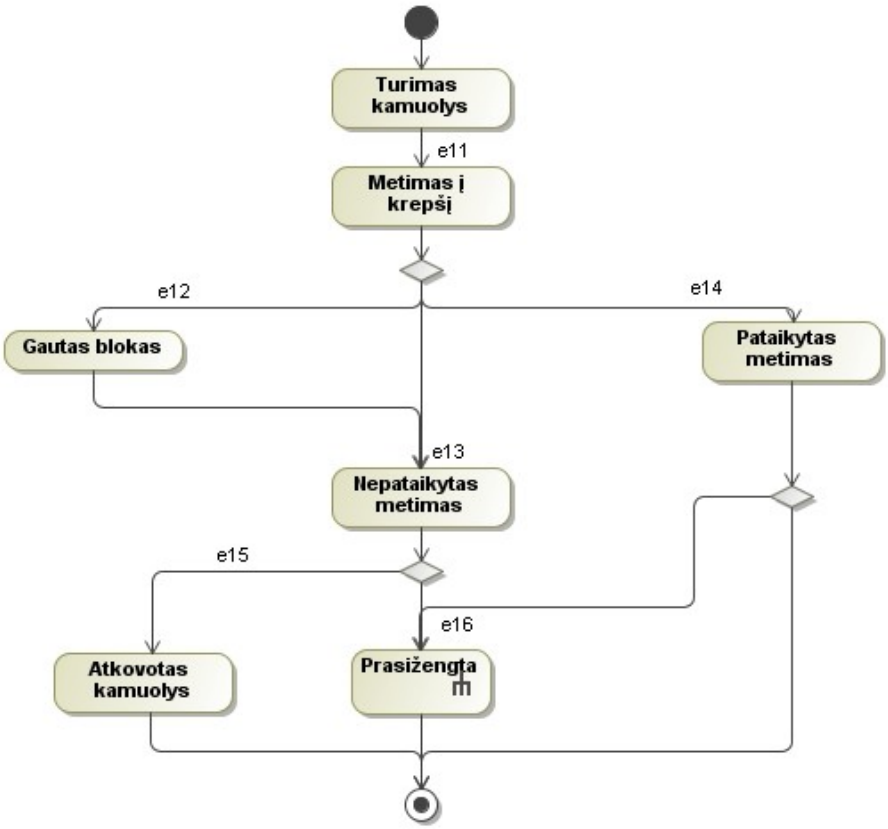


3.2 pav. „Vykstančios rungtynės" būsenų diagrama

Toliau detaliau panagrinėsime sudėtinius “Vykstančių rungtynių” elementus.

3.3 pav. pavaizduotos “Atliekamo metimo į krepšį” galimos būsenos.

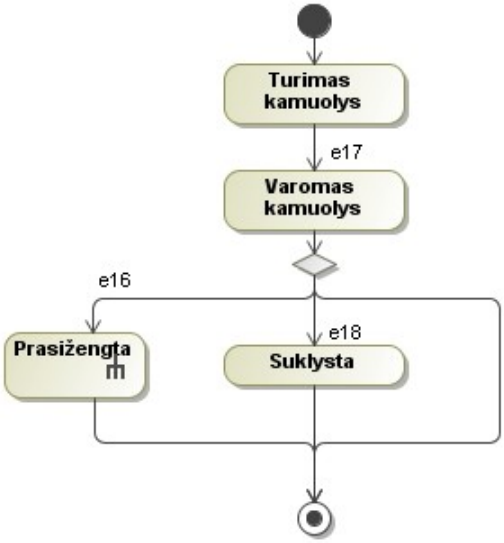
$$e5 = e11 \cdot (((1 + e12) \cdot e13 \cdot (e15 + e16)) + (e14 \cdot (1 + e16)));$$



3.3 pav. “Atliekamas metimas į krepšį“ būsenų diagrama

3.4 pav. pavaizduotos “Varomas kamuolys” galimos būsenos.

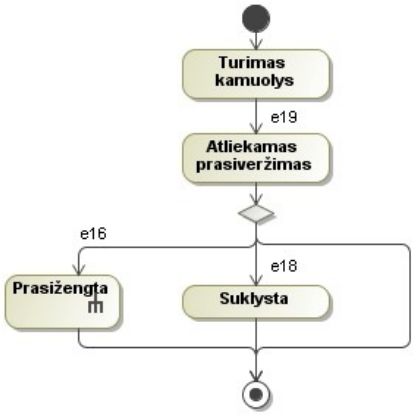
$$e6 = e17 \cdot (1 + e16 + e18);$$



3.4 pav. “Varomas kamuolys” būsenų diagrama

3.5 pav. pavaizduotos “Atliekamas prasiveržimas” galimos būsenos.

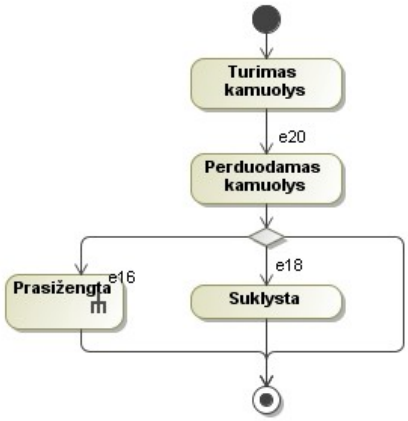
$$e7 = e19 \cdot (1 + e16 + e18)$$



3.5 pav. “Atliekamas prasiveržimas” būsenų diagrama

3.6 pav. pavaizduotos “Perduodamas kamuolys” galimos būsenos.

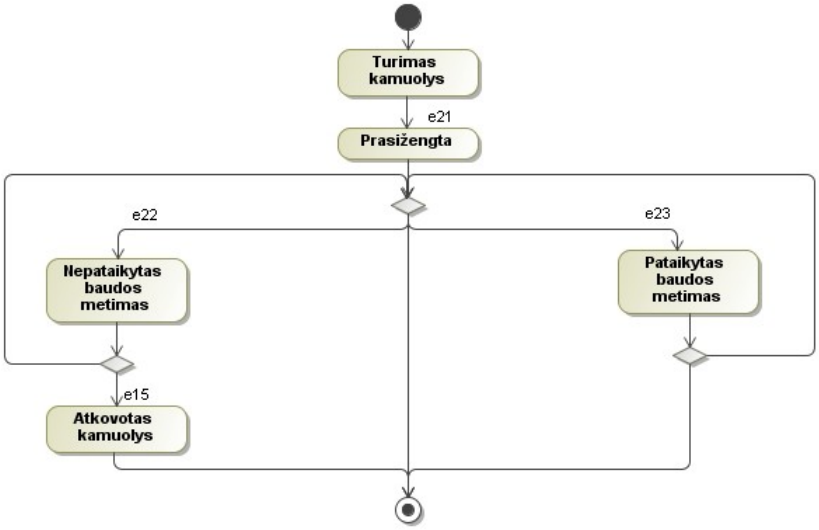
$$e8 = e20 \cdot (1 + e16 + e18)$$



3.6 pav. “Perduodamas kamuolys” būsenų diagrama

3.7 pav. pavaizduotos “Prasižengta” galimos būsenos.

$$e16 = e21 \cdot (1 + (e22 * \cdot e15) + e23 *)$$

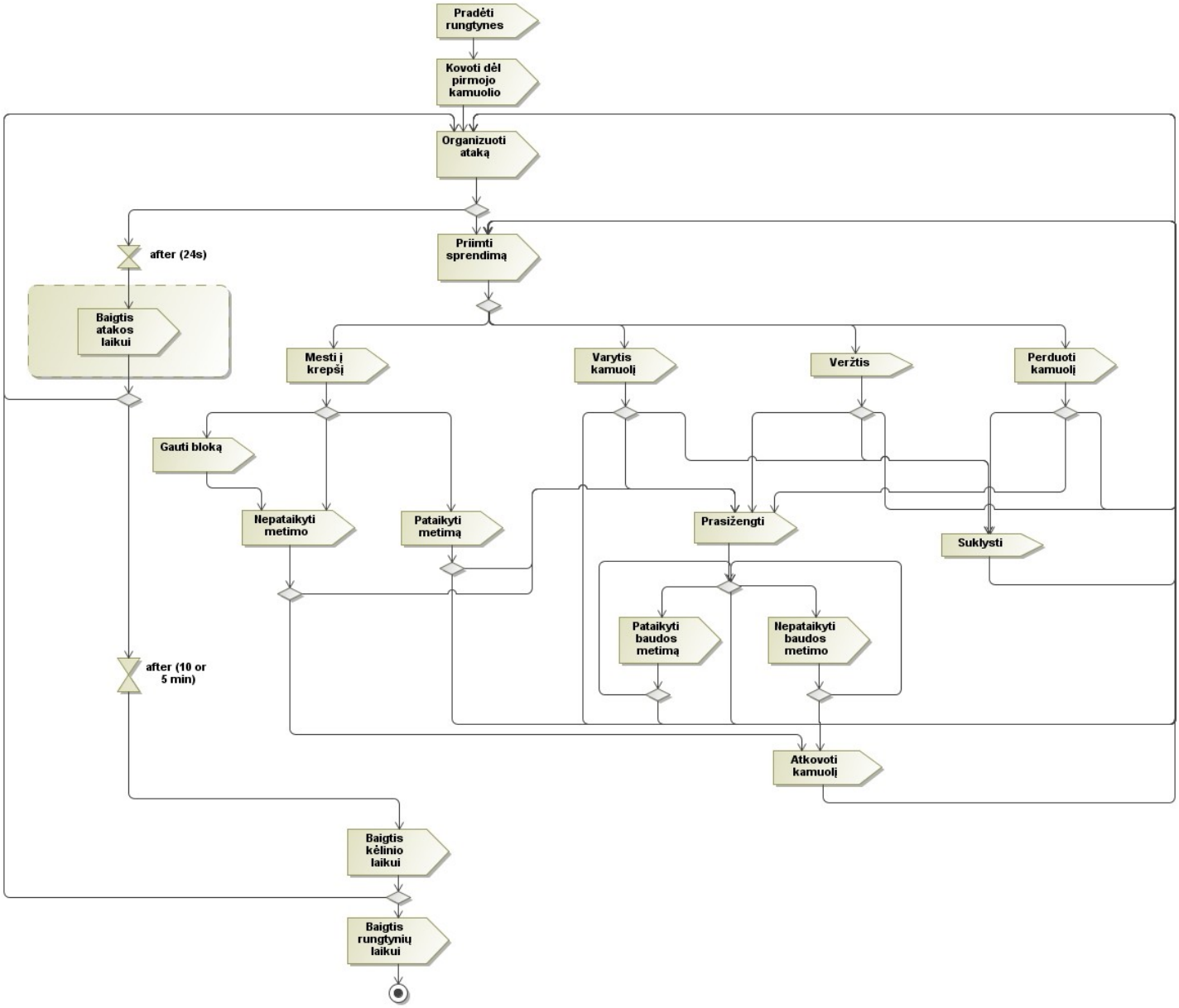


3.7 pav. “Prasižengta” būsenų diagrama

3.2. Žaidimo imitatoriaus aprašymas įvykių modeliu

Iki šiol nagrinėjome rungtynes remdamiesi būsenų modeliais. 3.8 pav. pavaizduotas vykstančių rungtynių įvykių modelis.

Tiek būsenų, tiek įvykių modeliai suteikia galimybes atskleisti rungtynių imitatoriaus veikimą, tačiau detaliau nagrinėti yra patogiau remiantis būsenų modeliais ir perėjimų iš vienos būsenos į kitą sąlygomis.



3.8 pav. Rungtynių imitatoriaus įvykių modelis

3.3. Žaidimo imitatoriaus veiklos taisyklės

3.1 skyrelyje išnagrinėjome galimas Rungtynių būsenas bei perėjimų iš vienos būsenos į kitą galimus atvejus. Šiame skyrelyje nagrinėsime taisykles, pagal kurias nusprendžiama į kurią būseną bus pereinama.

1. (*e1*) Pagal nustatytą turnyro planą yra suplanuojamos konkrečios rungtynės.
2. (*e2*) Jeigu atėjo tvarkaraštyje nurodytas laikas, pradedamos vykdyti rungtynės.
3. (*e3*) Jeigu baigėsi rungtynių laikas, jos laikomos pasibaigusiomis.
4. (*e4*) Jeigu pradedamos rungtynės, abiejų komandų centro puolėjai kovoja dėl pirmojo kamuolio.

   Jeigu

   $kovaDelPir\ moKamuolio\ (atkovojima\ s1, atkovojima\ s2, soklumas\ 1, soklumas\ 2, ugis\ 1, ugis\ 2, patirtis\ , atsitiktin\ umas) = tru$

   , kamuolys atitenka namų komandai, kitu atveju – svečių komandai.
5. (*e5*) Jeigu

   $metimasIKr\ epsi(vietaAikst\ eleje, zaidejoTak\ tika, komandineT\ aktika, gynejoPa\ \det is, atsitiktin\ umas) = true$

   , atliekamas metimas į krepšį.
6. (*e6*) Jeigu

   $\text{var}\ ymasis\ (\text{var}\ ymasis, vadovavima\ s, zaidejoTak\ tika, komandineT\ aktika, atsitiktin\ umas) = true$

   , žaidėjas varosi kamuolį.
7. (*e7*) Jeigu

   $prasiverzi\ mas\ (\text{var}\ ymasis, greitis, zaidejoTak\ tika, komandineT\ aktika, atsitiktin\ umas) = true$

   , atliekamas prasiveržimas.
8. (*e8*) Jeigu

   $perdavimas\ (zaidejoPoz\ icija, zaidejoTak\ tika, komandineT\ aktika, atsitiktin\ umas) = true$,

   perduodamas kamuolys.
9. (*e9*) Jeigu nuo atakos pradžios praėjo 24 sekundės, baigiasi atakos laikas ir naują ataką organizuoja kita komanda.
10. (*e10*) Jeigu nuo kėlinio pradžios praėjo tiek laiko, kokia yra kėlinio trukmė, kėlinys pasibaigia. Įprastiniu atveju kėlinio trukmė 10 min, o esant pratęsimui – 5 min.

11. (*e12*) Jeigu metimo į krepšį metu

$$blokuotas(p\_ugis, g\_ugis, p\_greitis, g\_greitis, p\_soklumas, g\_soklumas, g\_blokavimas,$$
$$p\_nuovargis, g\_nuovargis, p\_patirtis, g\_patirtis, atsitiktinumas) = true$$

, metimas yra blokuotas.

12. (*e13*) Jeigu metimo į krepšį metu

$$metimasTaiklus(vietaAikstelejej, p\_greitis, g\_greitis, p\_soklumas, g\_soklumas, p\_taiklumas, p\_ugis,$$
$$g\_ugis, p\_nuovargis, g\_nuovargis, p\_patirtis, g\_patirtis, atsitiktinumas) = false$$

, metimas yra netaiklus.

13. (*e14*) Jeigu metimo į krepšį metu

$$metimasTaiklus(vietaAikstelejej, p\_greitis, g\_greitis, p\_soklumas, g\_soklumas, p\_taiklumas,$$
$$p\_ugis, g\_ugis, p\_nuovargis, g\_nuovargis, p\_patirtis, g\_patirtis, atsitiktinumas) = true$$

, metimas yra taiklus.

14. (*e15*) Jeigu

$$atkovotaPuse(p\_ugis, g\_ugis, p\_atkovojimas, g\_atkovojimas, p\_soklumas, g\_soklumas,$$
$$p\_tvirtumas, g\_tvirtumas, p\_greitis, g\_greitis, p\_nuovargis, g\_nuovargis,$$
$$p\_patirtis, g\_patirtis, atsitiktinumas) = true$$

, kamuolį atkovoja besiginanti komanda. Kitu atveju – puolanti.

15. (*e16*) Jeigu

$$prazangaPrieTaiklaus(p\_zaidejoTaiklumas, g\_zaidejoTaiklumas, g\_komandinesTaiklumas, p\_nuovargis,$$
$$g\_nuovargis, p\_patirtis, g\_patirtis, atsitiktinumas) = true$$

, yra prasižengiama.

16. (*e18*) Jeigu

$$klaida(p\_varymasis, p\_perdavimas, g\_peremimas, g\_zaidejoTaiklumas, p\_nuovargis,$$
$$g\_nuovargis, p\_patirtis, g\_patirtis, atsitiktinumas) = true$$

, padaroma klaida.

17. (*e22*) Jeigu $baudaTaiklus(taiklumas, nuovargis, patirtis, atsitiktinumas) = false$, baudos metimas yra netaiklus.

18. (*e23*) Jeigu $baudaTaiklus(taiklumas, nuovargis, patirtis, atsitiktinumas) = true$, baudos metimas yra taiklus.

Šiame skyriuje aprašytas rungtynių imitatoriaus veikimas, pagal kurį realizuotas rungtynių imitavimas žaidime. Imitatoriaus realizacijai naudojamos trys klasės: VarikliukoRungtynes_class.php, VarikliukoKomanda_class.php, VarikliukoZaidejas_class.php (visų sistemos komponentų diagrama pavaizduota 6.2 pav.) Šiose klasėse *php* kalba yra aprašytos aukščiau išvardintos taisyklės, naudojamos rungtynių imitavimui.

Testuojant imitatorių gali būti nustatoma, jog kažkuris žaidimo elementas veikia nekorektiškai ir reikia atlikti tam tikrus pakeitimus. Sudarytos imitatoriaus būsenų diagramos (3.2 – 3.7 pav.) bei veiklos taisyklės palengvina imitatoriaus analizę ir pakeitimų įgyvendinimą. Nustačius reikiamus pakeitimus yra pakoreguojamos atitinkamos veiklos taisyklės ir vėl stebimas imitatoriaus veikimas.

4. Reikalavimų specifikacija

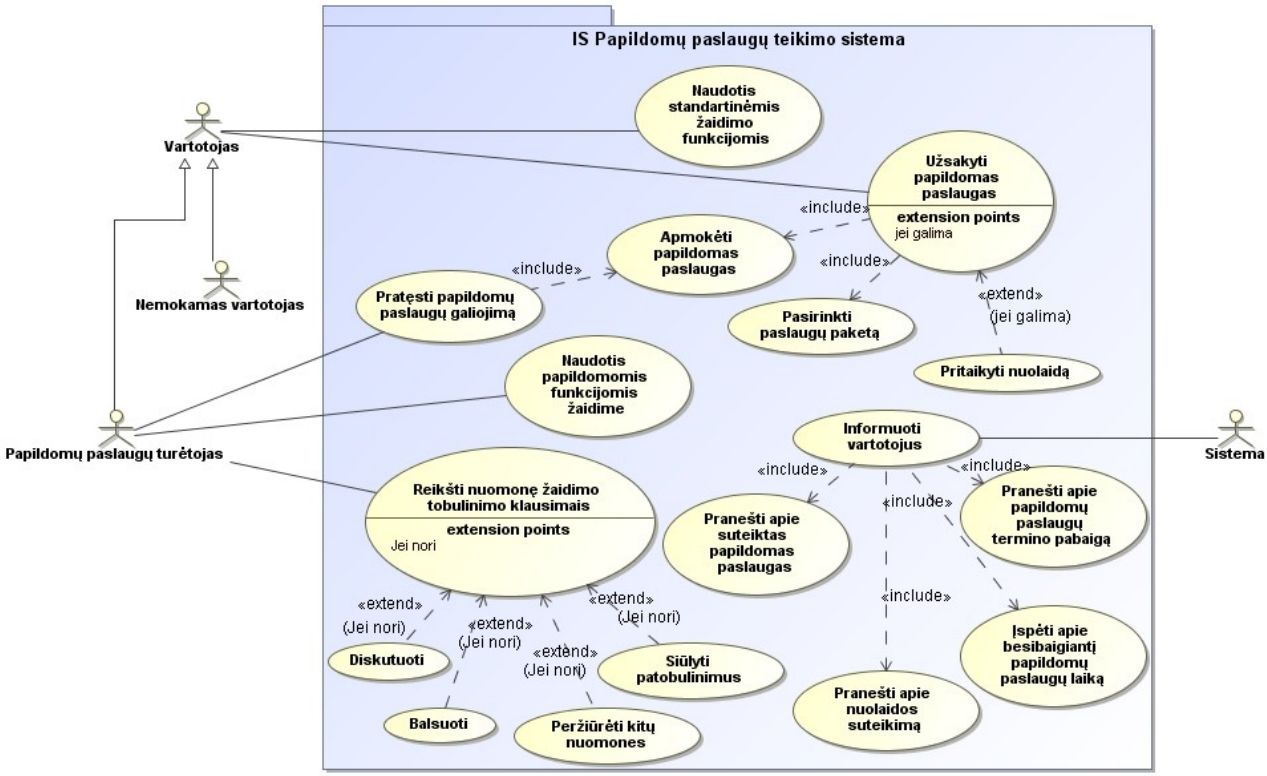
Šiame darbe nagrinėjamo virtualaus krepšinio žaidimo pagrindas jau yra sukurtas, tačiau norint patenkinti vartotojų poreikius jį būtina nuolat tobulinti. Analizės metu nustatytas papildomų posistemių poreikis: papildomų paslaugų teikimo bei klaidų stebėjimo ir fiksavimo.

Šiame skyrelyje pateikiame reikalavimų specifikacijas numatytų posistemių realizavimui.

4.1. Kuriamų krepšinio žaidimo posistemių funkciniai reikalavimai

- Papildomų paslaugų teikimo posistemė

4.1 pav. pateikta papildomų paslaugų teikimo sistemos panaudojimo atvejų diagrama.



4.1 pav. Papildomų paslaugų teikimo sistemos panaudojimo atvejai

4.1 lentelėje pateiktas PA "*Naudotis standartinėmis žaidimo funkcijomis*" aprašymas. Visi žaidimo vartotojai, neatsižvelgiant į jų tipą (paprastas, administratorius ar moderatorius), gali naudotis standartinėmis žaidimo funkcijomis – įsigyti personalą, nustatyti rungtynių taktiką, rašyti žinutes kitiems vartotojams ir t.t.

Kadangi vartotojai gali naudotis bet kuria standartine funkcija, sekų diagrama kiekvienai funkcijai skiriasi ir jos čia nepateiksime.

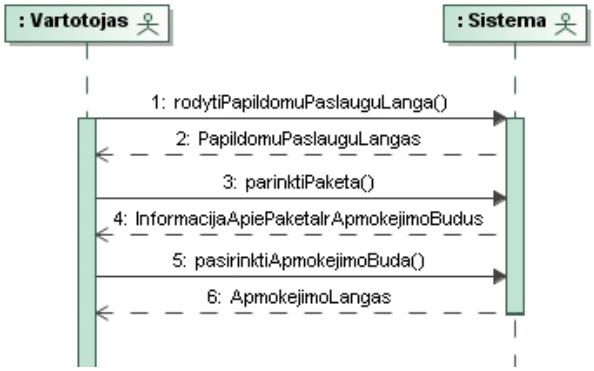
4.1 lentelė

Panaudojimo atvejis "Naudotis standartinėmis žaidimo funkcijomis"	
Aktorius	Žaidimo vartotojas
Prieš sąlyga	Žaidime yra užregistruotas vartotojas
Sužadinimo sąlyga	Vartotojas pasirenka norimą atlikti veiksmą
Susiję panaudojimo atvejai	-
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas pasirenka norimą atlikti veiksmą	1.1 Sistema parenka reikiamus duomenis ir parodo/išsaugo užsakytą informaciją.
Po sąlyga	Atliktas vartotojo norimas veiksmas.
Alternatyvūs scenarijai	
-	

4.2 lentelėje pateiktas PA "*Užsakyti papildomas paslaugas*" aprašymas, o 4.2 pav. – sekų diagrama. Vartotojas pasirenka papildomų paslaugų paketą, kurį nori įsigyti, pritaikoma nuolaida (jei ją galima suteikti vartotojui) ir įvykdomas apmokėjimas.

4.2 lentelė

Panaudojimo atvejis "Užsakyti papildomas paslaugas"	
Aktorius	Registruotas vartotojas
Prieš sąlyga	Vartotojas yra užregistravęs klubą žaidime
Sužadinimo sąlyga	Vartotojas ateina į papildomų paslaugų užsakymo puslapį
Susiję panaudojimo atvejai	*Apima:* Pasirinkti paslaugų paketą, Apmokėti papildomas paslaugas. *Išplėtimas:* Pritaikyti nuolaidą
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas ateina į papildomų paslaugų užsakymo langą	1.1 Sistema pateikia informaciją apie papildomas paslaugas
2. Vartotojas pasirenka norimą paketą ir terminą	2.1 Sistema parodo informaciją apie paketą bei informaciją apie apmokėjimo būdus 2.2 Jeigu galima, vartotojui pritaikoma nuolaida
3. Vartotojas pasirenka apmokėjimo būdą	3.1 Sistema nukreipia vartotoją į atitinkamo apmokėjimo būdo puslapį
4. Vartotojas apmoka už paslaugas	4.1 Sistema išsaugo informaciją, kad buvo įvykdytas apmokėjimas
Po sąlyga	Vartotojas yra įsigijęs papildomų paslaugų paketą
Alternatyvūs scenarijai	
4a Vartotojas apsigalvoja ir atsisako apmokėti	Sistema panaikina informaciją apie įsigijimą



4.2 pav. Panaudojimo atvejo "Užsakyti papildomas paslaugas" sekų diagrama

Panaudojimo atvejai *"Pasirinkti paslaugų paketą", "Apmokėti papildomas paslaugas"* ir *"Pritaikyti nuolaidą"* yra nesudėtingi, todėl jų atskirai neaprašinėsime. Bendras jų panaudojimas pavaizduotas 4.2 lentelėje ir 4.2 pav.

4.3 lentelėje pateiktas PA "*Pratęsti papildomų paslaugų galiojimą*" aprašymas, o šio panaudojimo atvejo sekų diagrama yra labai panaši į 4.2. Prieš pasibaigiant turimo paketo galiojimo laikui, vartotojas gali prasitęsti paketo galiojimą

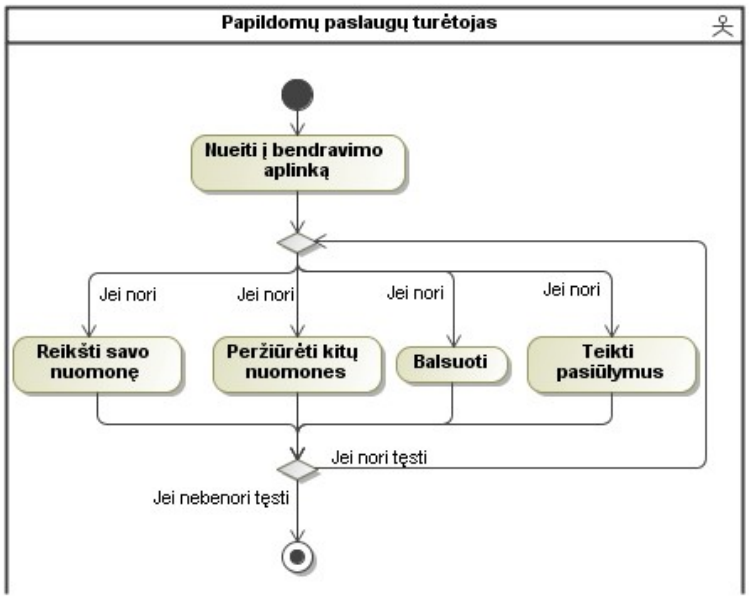
4.3 lentelė

Panaudojimo atvejis "Pratęsti papildomų paslaugų galiojimą"	
Aktorius	Vartotojas, įsigijęs papildomų paslaugų paketą
Prieš sąlyga	Vartotojas jau turi įsigijęs papildomų paslaugų paketą ir nori jį prasitęsti
Sužadinimo sąlyga	Vartotojas ateina į papildomų paslaugų užsakymo/pratęsimo puslapį
Susiję panaudojimo atvejai	*Apima:* Apmokėti papildomas paslaugas
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas ateina į papildomų paslaugų pratęsimo langą	1.1 Sistema pateikia informaciją apie turimas papildomas paslaugas bei galimą užsakymą
2. Vartotojas pasirenka terminą	2.1 Sistema parodo informaciją apie paketą bei informaciją apie apmokėjimo būdus 2.2 Jeigu galima, vartotojui pritaikoma nuolaida
3. Vartotojas pasirenka apmokėjimo būdą	3.1 Sistema nukreipia vartotoją į atitinkamo apmokėjimo būdo puslapį
4. Vartotojas apmoka už paslaugas	4.1 Sistema išsaugo informaciją, kad buvo įvykdytas apmokėjimas
Po sąlyga	Vartotojui yra pratęstas papildomų paslaugų galiojimas
Alternatyvūs scenarijai	
4a Vartotojas apsigalvoja ir atsisako apmokėti	Sistema panaikina informaciją apie įsigijimą

4.4 lentelėje pateiktas PA "*Reikšti nuomonę žaidimo tobulinimo klausimais*" aprašymas, o 4.3 pav. – veiklos diagrama. Šis panaudojimo atvejis turi tris išplėtimo taškus: jeigu vartotojas nori, gali siūlyti jo manymu reikiamus patobulinimus, balsuoti už esamus pasiūlymus bei diskutuoti apie žaidimo savybes.

4.4 lentelė

Panaudojimo atvejis "Reikšti nuomonę žaidimo tobulinimo klausimais"	
Aktorius	Vartotojas, įsigijęs papildomų paslaugų paketą
Prieš sąlyga	Vartotojas jau įsigijęs papildomų paslaugų paketą
Sužadinimo sąlyga	Vartotojas ateina į aktyviausių vartotojų bendravimo puslapį
Susiję panaudojimo atvejai	*Išplėtimas:* Diskutuoti, Balsuoti, Siūlyti patobulinimus, Peržiūrėti kitų nuomones
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas ateina į bendravimo puslapį	1.1 Sistema pateikia esamų diskusijų, balsavimų sąrašą
2. Vartotojas pasirenka norimą veiksmą (rašyti savo nuomonę, balsuoti, reikšti savo pasiūlymą, peržiūrėti kitų vartotojų pareiškimus)	2.1 Sistema pateikia atitinkamą langą
3. Vartotojas atlieka pasirinktą veiksmą ir išsaugo duomenis	3.1 Sistema išsaugo vartotojo pateiktus duomenis
Po sąlyga	Vartotojo nuomonė išsaugota ir prieinama kitiems aktyviausiems nariams
Alternatyvūs scenarijai	
-	



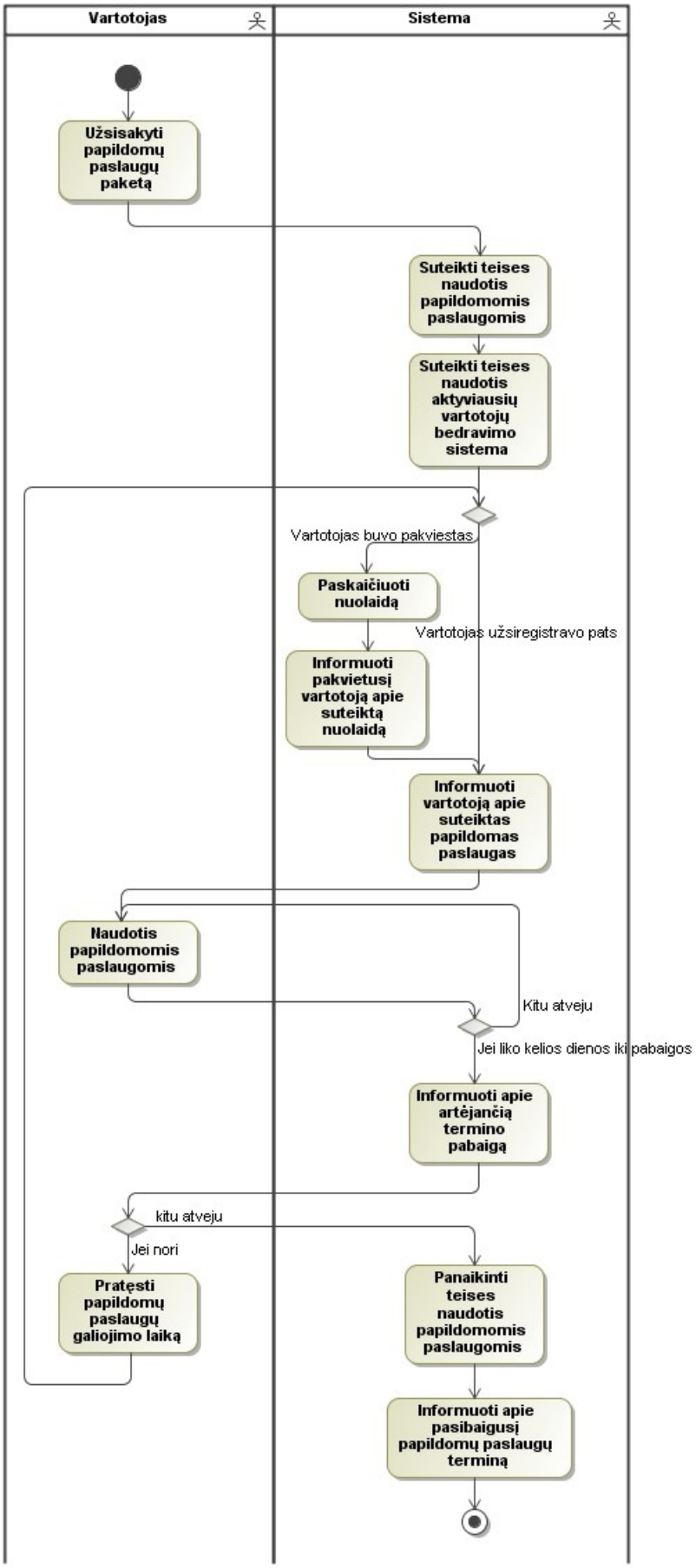
4.3 pav. Panaudojimo atvejo "Reikšti nuomonę žaidimo tobulinimo klausimais" veiklos diagrama

Panaudojimo atvejai *"Diskutuoti", „Balsuoti", „Siūlyti patobulinimus", „Peržiūrėti kitų nuomones"* yra įtraukti į panaudojimo atvejo *"Reikšti nuomonę žaidimo tobulinimo klausimais"* aprašymą (4.4 lentelė ir 4.3 pav.), todėl kiekvieno atskirai nebenagrinėsime.

45 lentelėje pateiktas PA "*Informuoti vartotojus*" aprašymas, o 4.4 pav. – veiklos diagrama. Šis panaudojimo atvejis susideda iš vartotojų informavimo apie suteiktas paslaugas, besibaigiančias ar pasibaigusias.

4.5 lentelė

Panaudojimo atvejis "Informuoti vartotojus"	
Aktorius	Sistema
Prieš sąlyga	Yra vartotojų, įsigijusių papildomų paslaugų paketus
Sužadinimo sąlyga	Įvyko įvykis, apie kurį reikia pranešti
Susiję panaudojimo atvejai	*Apima:* Pranešti apie suteiktas papildomas paslaugas, Pranešti apie nuolaidos suteikimą, Įspėti apie besibaigiantį papildomų paslaugų laiką, Pranešti apie papildomų paslaugų termino pabaigą
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas užsisako papildomų paslaugų paketą	1.1 Sistema suteikia vartotojui papildomas paslaugas bei leidimą bendrauti aktyviausių narių bendruomenėje 1.2 Išsiunčiamas pranešimas, informuojantis apie suteiktas paslaugas
Po sąlyga	Vartotojo nuomonė išsaugota ir prieinama kitiems aktyviausiems nariams
Alternatyvūs scenarijai	
1a. Baigiasi vartotojo papildomų paslaugų terminas	1.1 Sistema panaikina vartotojo teises ir išsiunčia žinutę, pranešančią apie tai
1b. Papildomų paslaugų terminas artėja į pabaigą	1.1 Sistema informuoja vartotoją apie besibaigiantį terminą ir galimas alternatyvas
1c. Vartotojo pakviestas į žaidimą klubas įsigyja papildomų paslaugų paketą	1.1 Sistema suteikia paketą įsigijusiam vartotojui 1.2 Sistema suteikia nuolaidą jį pakvietusiam vartotojui 1.3 Sistema informuoja vartotoją apie suteiktą nuolaidą

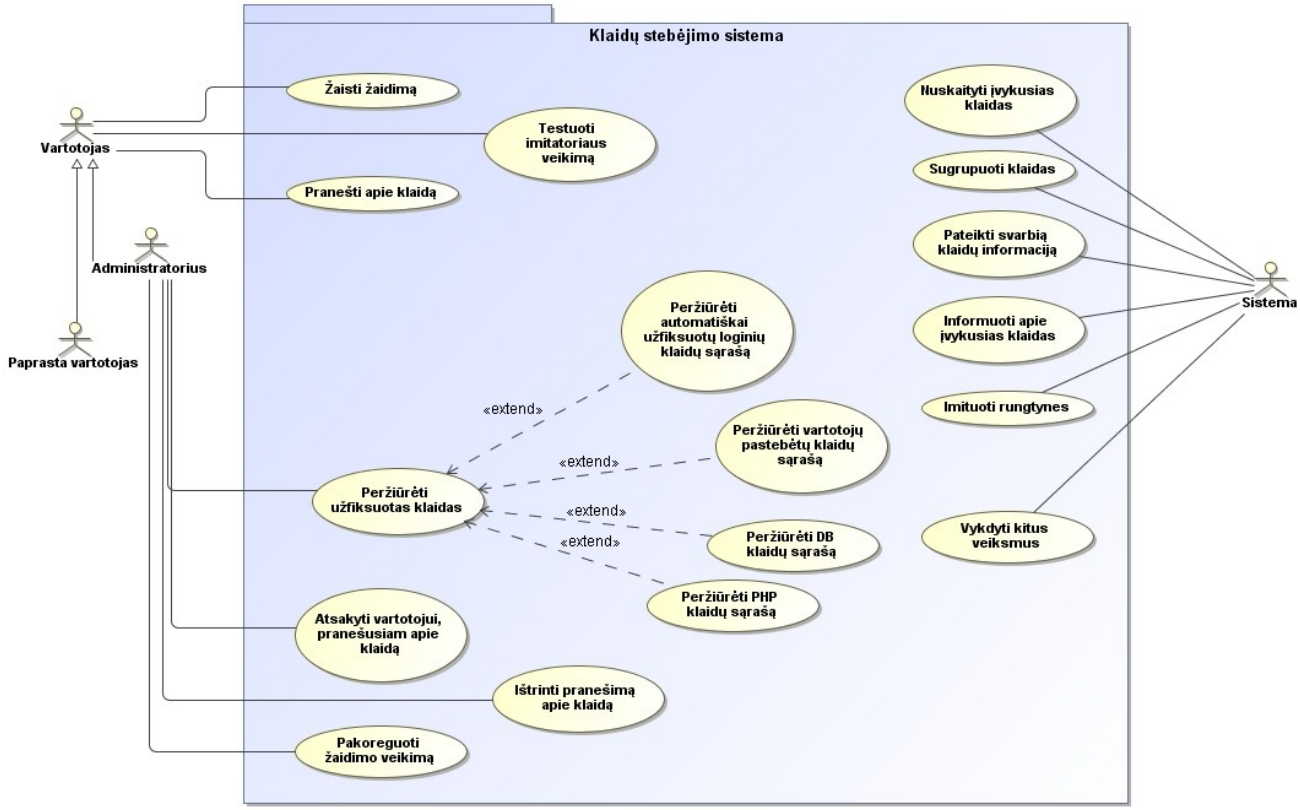


4.4 pav. Papildomų paslaugų teikimo ir vartotojų informavimo veiklos diagrama

Panaudojimo atvejai *“Pranešti apie suteiktas papildomas paslaugas“, „Pranešti apie nuolaidos suteikimą“, „Įspėti apie besibaigiantį papildomų paslaugų laiką“, „Pranešti apie papildomų paslaugų termino pabaigą“* yra aprašyti kartu su “Siųsti pranešimus” panaudojimo atveju (4.5 lentelė ir 4.4 pav.)

- Klaidų stebėjimo posistemė

4.5 pav. pateikta klaidų stebėjimo sistemos bei rungtynių imitatoriaus testavimo sistemos panaudojimo atvejų diagrama.



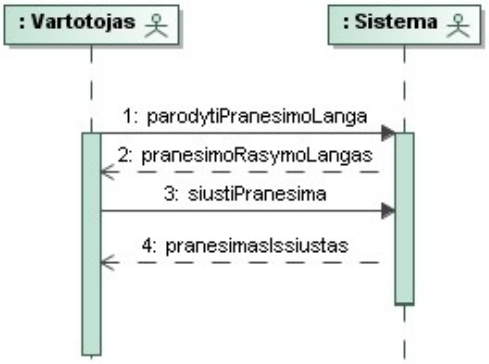
4.5 pav. Klaidų stebėjimo sistemos panaudojimo atvejai

4.6 lentelėje pateiktas PA “*Pranešti apie klaidą*” aprašymas, o 4.6 pav. – veiklos diagrama. Jeigu vartotojas mano, kad pastebėjo klaidą žaidime (loginę ar techninę), jis gali pranešti apie ją.

4.6 lentelė

Panaudojimo atvejis “Pranešti apie klaidą”	
Aktorius	Registruotas vartotojas
Prieš sąlyga	Vartotojas yra užregistravęs klubą žaidime
Sužadinimo sąlyga	Vartotojas mano, kad rado klaidą
Susiję panaudojimo atvejai	-
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas mano, kad	1.1 Sistema parodo pranešimo rašymo langą

pastebėjo klaidą ir iškviečia pranešimo rašymo langą	
2. Vartotojas parenka tipą ir parašo pranešimą	2.1 Sistema išsiunčia
Po sąlyga	Pranešimas apie klaidą išsiųstas
Alternatyvūs scenarijai	
-	

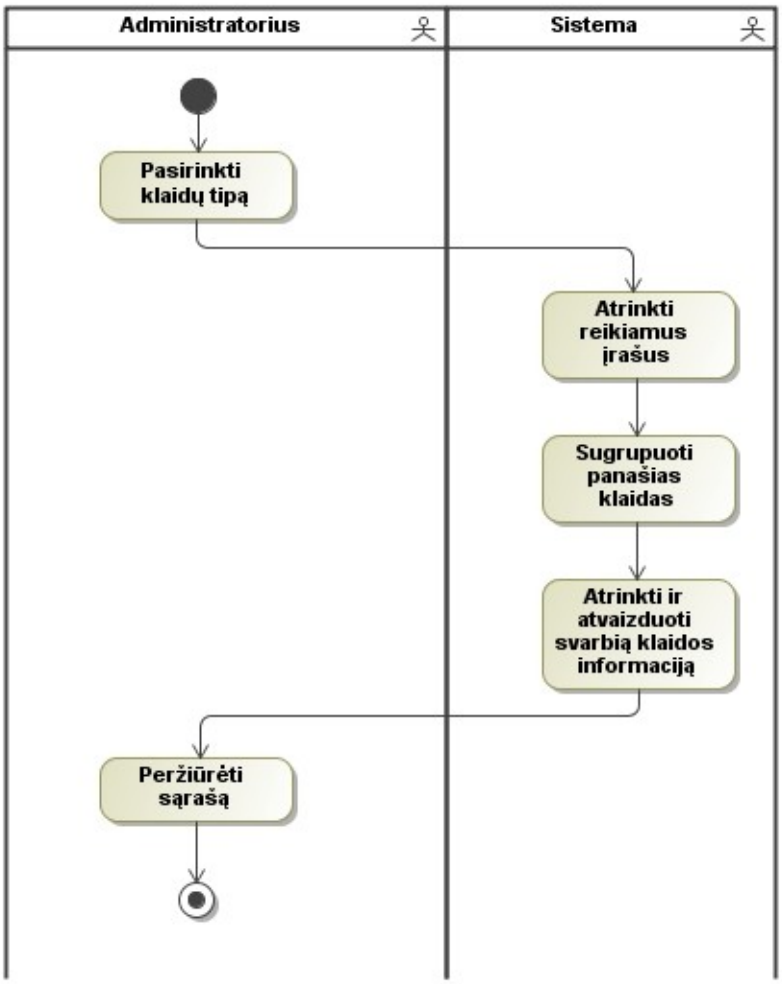


4.6 pav. Panaudojimo atvejo "Pranešti apie klaidą" sekų diagrama

4.7 lentelėje pateiktas PA "*Peržiūrėti užfiksuotas klaidas*" aprašymas, o 4.7 pav. – veiklos diagrama. Turintis atitinkamas teises administratorius gali peržiūrėti klaidų sąrašą. Panaudojimo atvejai *"Peržiūrėti automatiškai užfiksuotų loginių klaidų sąrašą", „Peržiūrėti vartotojų pastebėtų klaidų sąrašą", "Peržiūrėti DB klaidų sąrašą", „Peržiūrėti PHP klaidų sąrašą"* yra sudedamosios panaudojimo atvejo "Peržiūrėti klaidų sąrašą" dalys.

4.7 lentelė

Panaudojimo atvejis "Peržiūrėti užfiksuotas klaidas"	
Aktorius	Administratorius
Prieš sąlyga	Yra pranešimų apie klaidas sistemoje
Sužadinimo sąlyga	Administratorius ateina į klaidų sąrašo langą
Susiję panaudojimo atvejai	*Apima:* "Peržiūrėti automatiškai užfiksuotų loginių klaidų sąrašą", „Peržiūrėti vartotojų pastebėtų klaidų sąrašą", "Peržiūrėti DB klaidų sąrašą", „Peržiūrėti PHP klaidų sąrašą"
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Administratorius pasirenka norimą matyti klaidų tipą	1.1 Sistema atrenka ir atvaizduoja atitinkamus įrašus
Po sąlyga	Administratoriui pateiktas klaidų sąrašas
Alternatyvūs scenarijai	

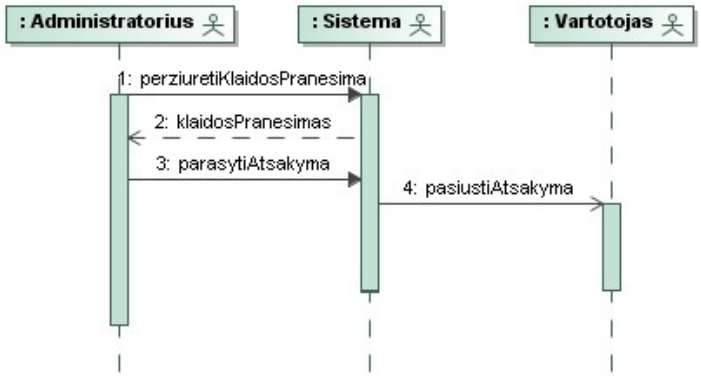


4.7 pav. Panaudojimo atvejo "Peržiūrėti klaidų sąrašą" veiklos diagrama

4.8 lentelėje pateiktas PA "*Atsakyti vartotojui, pranešusiam apie klaidą*" aprašymas, o 4.8 pav. – sekų diagrama. Peržiūrėjęs pranešimą apie klaidą, administratorius gali atsakyti apie ją pranešusiam vartotojui.

4.8 lentelė

Panaudojimo atvejis "Atsakyti vartotojui, pranešusiam apie klaidą"	
Aktorius	Administratorius
Prieš sąlyga	Vartotojas yra pranešęs apie pastebėtą klaidą
Sužadinimo sąlyga	Administratorius peržiūri klaidos pranešimą
Susiję panaudojimo atvejai	-
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Administratorius peržiūri pranešimus apie klaidas	1.1 Sistema pateikia klaidų sąrašą
2. Administratorius parašo atsakymą vartotojui	2.1 Sistema pasiunčia pranešimą vartotojui
Po sąlyga	Atsakymas vartotojui išsiųstas
Alternatyvūs scenarijai	
-	



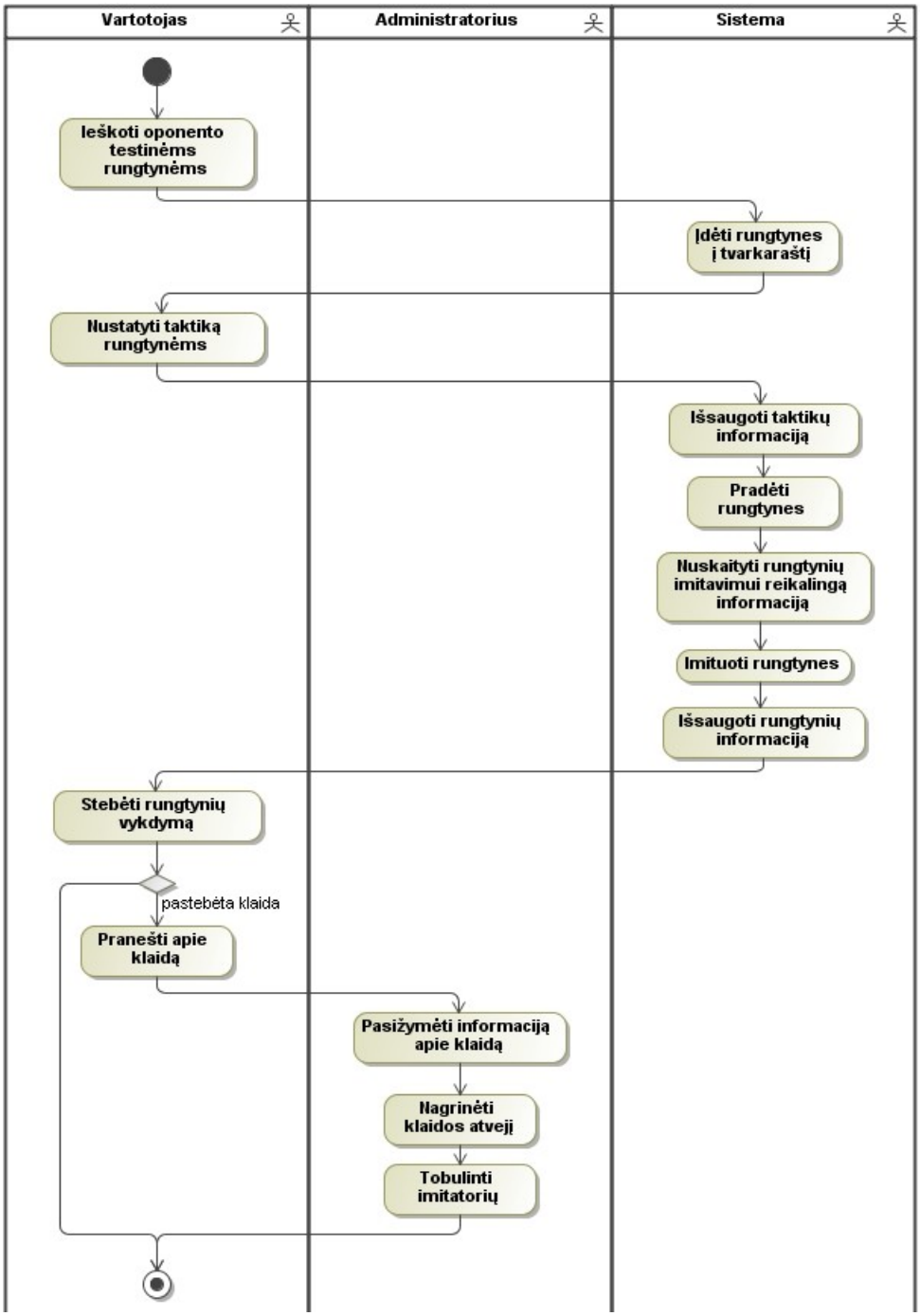
4.8 pav. Panaudojimo atvejo "Atsakyti vartotojui, pranešusiam apie klaidą" sekų diagrama

Panaudojimo atvejis "*Ištrinti pranešimą apie klaidą*" yra labai paprastas, todėl aprašymo lentelės ir sekų diagramos nepateiksime. Administratorius pasirenka "Ištrinti", o sistema ištrina pranešimą apie šią klaidą.

PA "*Testuoti imitatoriaus veikimą*" aprašymas pateiktas 4.9 lentelėje, o 4.9 pav. – veiklos diagrama. Vartotojai gali stebėti imitatoriaus veikimo teisingumą ir pranešti apie pastebėtas klaidas.

4.9 lentelė

Panaudojimo atvejis "Testuoti imitatoriaus veikimą"	
Aktorius	Žaidimo vartotojas
Prieš sąlyga	-
Sužadinimo sąlyga	Įkeliama nauja testinio imitatoriaus versija
Susiję panaudojimo atvejai	-
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas suranda priešininką testinėms draugiškoms rungtynėms	1.1 Sistema įdeda rungtynes į tvarkaraštį
2. Vartotojai nustato taktikas testinėms rungtynėms	2.1 Sistema išsaugo pateiktą informaciją
3. Pagal tvarkaraštyje numatytą laiką pradedamos rungtynės	3.1 Sistema nuskaito rungtynių imitavimui reikalingą informaciją 3.2 Sistema imituoja rungtynes 3.3 Rungtynių vykdymo informacija įrašoma į duomenų bazę
4. Stebėdamas rungtynių informaciją, vartotojas pastebi netinkamą veikimą ir praneša apie tai	
5. Administracija pasižymi pastebėjimus apie klaidas, nagrinėja ir tobulina imitatorių	
Po sąlyga	Sudarytas imitatoriaus veikimo netikslumų sąrašas
Alternatyvūs scenarijai	
4a Vartotojas klaidų nepastebi	



4.9 pav. Panaudojimo atvejo "Testuoti imitatoriaus veikimą" veiklos diagrama

4.10 lentelėje pateiktas PA "*Nustatyti įvykusias klaidas*" aprašymas. Sistema fiksuoja ir įrašo įvykusias technines klaidas.

4.10 lentelė

Panaudojimo atvejis "Nustatyti įvykusias klaidas"	
Aktorius	Sistema
Prieš sąlyga	-
Sužadinimo sąlyga	Pasitaikiusi klaida žaidime
Susiję panaudojimo atvejai	-
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***

1. Sistemoje pasitaiko klaida	1.1 Sistema užfiksuoja klaidą 1.2 Sistema išsaugo įrašą apie klaidą
Po sąlyga	Išsaugotas įrašas apie įvykusią klaidą
Alternatyvūs scenarijai	
-	

4.11 lentelėje pateiktas PA "*Sugrupuoti klaidas* " aprašymas. Sistema nustato, kokio tipo klaida įvyko.

4.11 lentelė

Panaudojimo atvejis "Sugrupuoti klaidas"	
Aktorius	Sistema
Prieš sąlyga	-
Sužadinimo sąlyga	Pasitaikiusi klaida žaidime
Susiję panaudojimo atvejai	-
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Sistemoje pasitaiko klaida	1.1 Sistema nustato, kokio tipo klaida įvyko 1.2 Sistema išsaugo informaciją apie klaidos tipą
Po sąlyga	Išsaugotas įrašas apie klaidos tipą
Alternatyvūs scenarijai	
-	

4.12 lentelėje pateiktas PA "*Pateikti svarbią klaidų informaciją* " aprašymas. Sistema nustato, kokio tipo klaida įvyko.

4.12 lentelė

Panaudojimo atvejis "Pateikti svarbią klaidų informaciją"	
Aktorius	Sistema
Prieš sąlyga	Sistemoje užfiksuota klaidų
Sužadinimo sąlyga	Vartotojas ateina į klaidų sąrašo langą
Susiję panaudojimo atvejai	-
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***
1. Vartotojas iškviečia klaidų sąrašo langą	1.1 Sistema nuskaito sistemoje pasitaikiusias ir nesutvarkytas klaidas 1.2 Sistema suformuoja langą su svarbia informacija apie klaidas
Po sąlyga	Pateiktas klaidų sąrašas su svarbia informacija
Alternatyvūs scenarijai	
-	

4.13 lentelėje pateiktas PA "*Informuoti apie įvykusias klaidas* " aprašymas. Sistema nustato, kokio tipo klaida įvyko.

4.13 lentelė

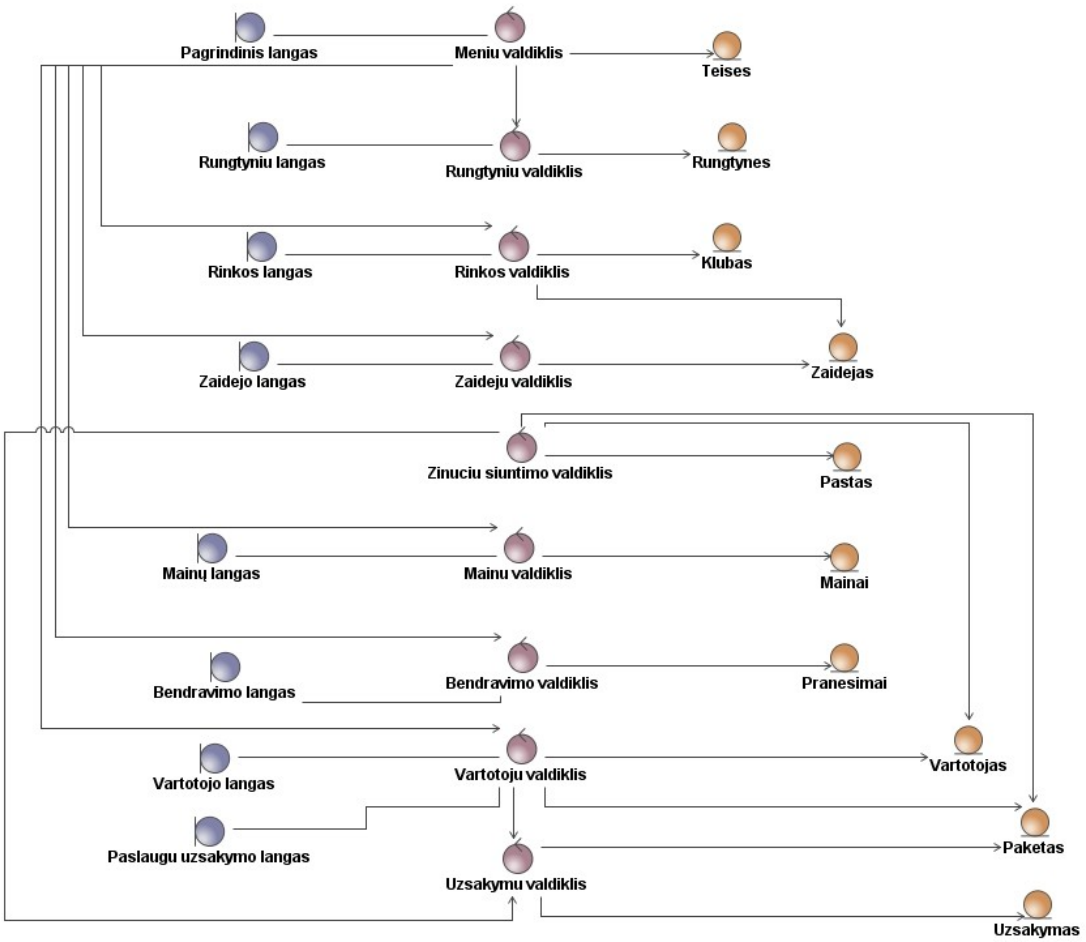
Panaudojimo atvejis "Informuoti apie įvykusias klaidas"	
Aktorius	Sistema
Prieš sąlyga	-
Sužadinimo sąlyga	Pasitaikiusi klaida žaidime
Susiję panaudojimo atvejai	-
Pagrindinis įvykių srautas	***Sistemos reakcija ir sprendimai***

1. Sistemoje pasitaiko klaida	1.1 Sistema praneša apie tai administratoriams
Po sąlyga	Administratoriai informuoti apie klaidą
Alternatyvūs scenarijai	
-	

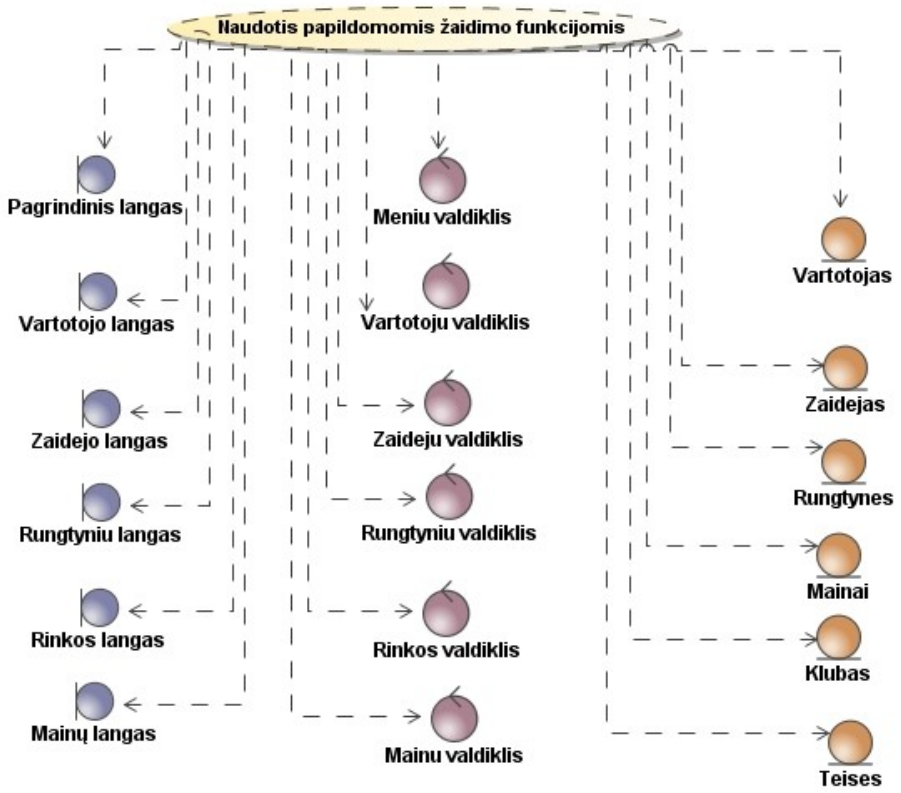
PA "*Imituoti rungtynes*" yra viena iš sudėtingiausių žaidimo dalių, kai nustatoma rungtynių eiga ir rezultatas. Imitatoriaus veikimas yra detaliai aprašytas 3-čiame šio darbo skyriuje.

4.2. Kuriamų krepšinio žaidimo posistemių reikalavimų analizė

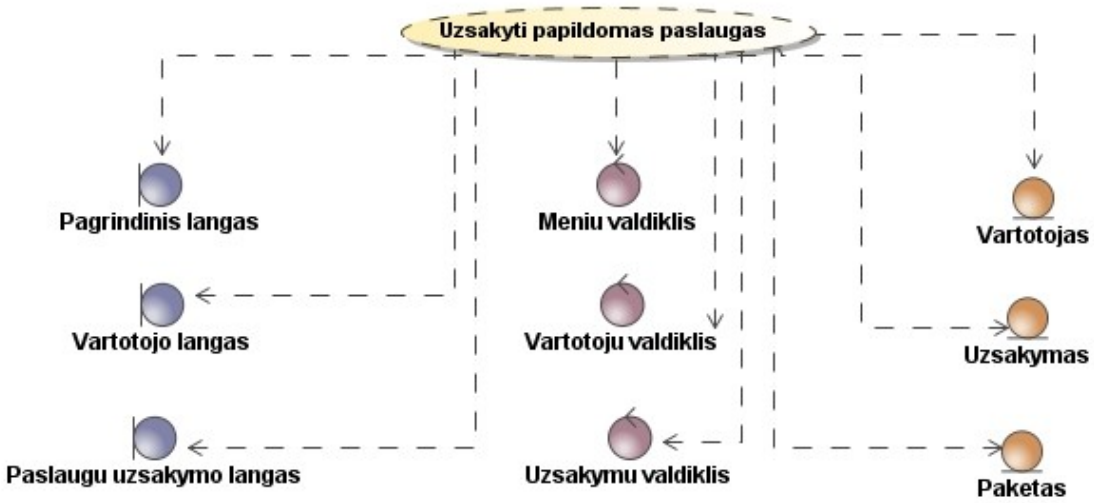
Papildomų paslaugų teikimo analizės schema pateikta 4.10 pav. 4.11-4.15 pav. pavaizduotos papildomų paslaugų sistemos panaudojimo atvejų realizacijos diagramos. Analizės diagrama nėra pateikiama PA "Naudotis standartinėmis žaidimo funkcijomis", nes ji apima visas žaidimo klases – ir nesusijusias su atliekamais patobulinimais.



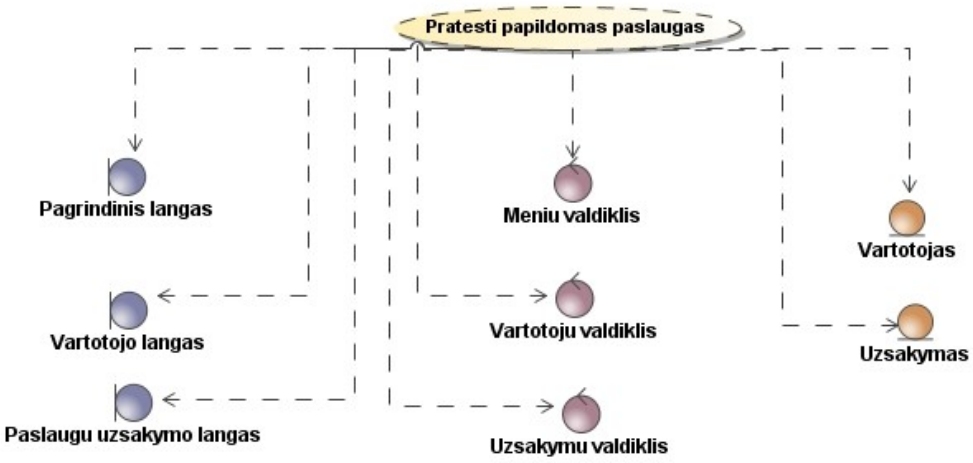
4.10 pav. Papildomų paslaugų teikimo analizės diagrama



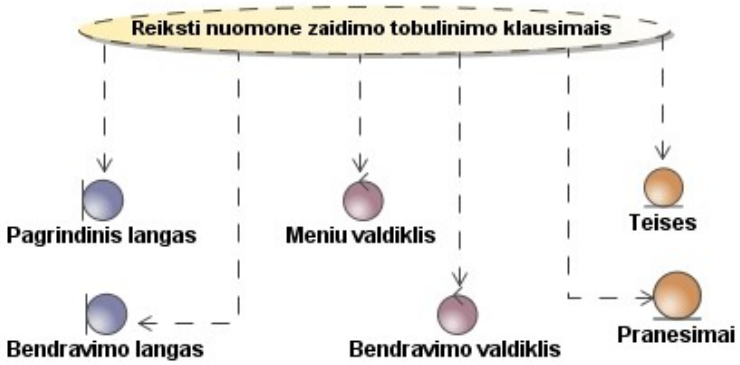
4.11 pav. PA "Naudotis papildomomis žaidimo funkcijomis" realizacijos diagrama



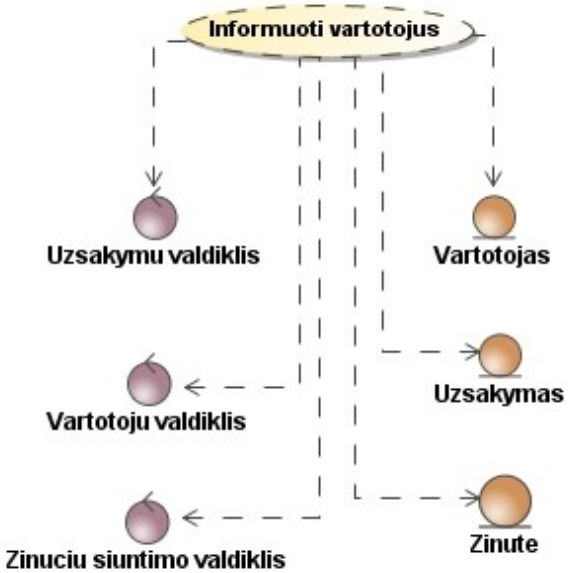
4.12 pav. PA "Užsakyti papildomas paslaugas" realizacijos diagrama



4.13 pav. PA "Pratęsti papildomas paslaugas" realizacijos diagrama



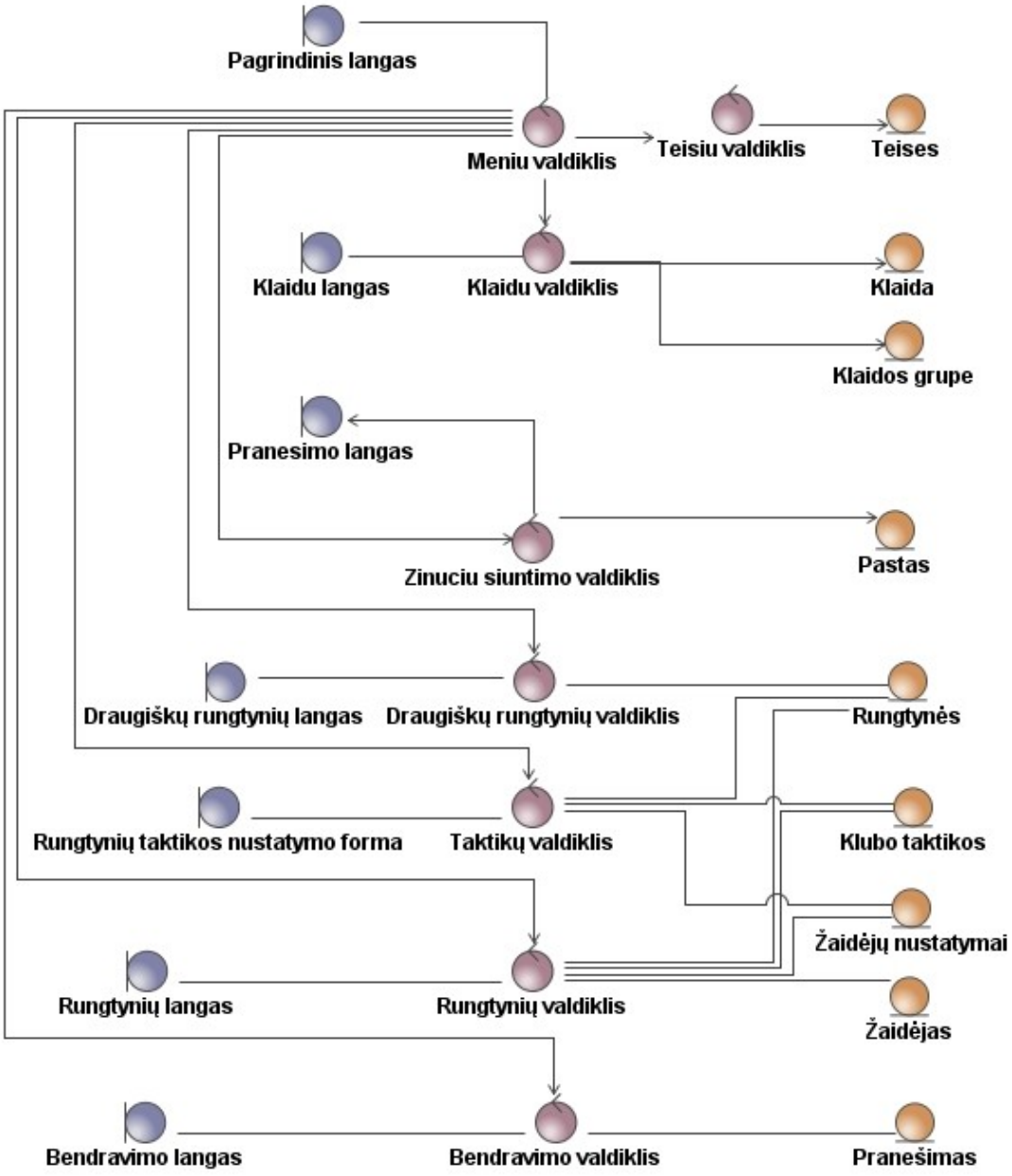
4.14 pav. PA "Reikšti nuomonę žaidimo tobulinimo klausimais" realizacijos diagrama



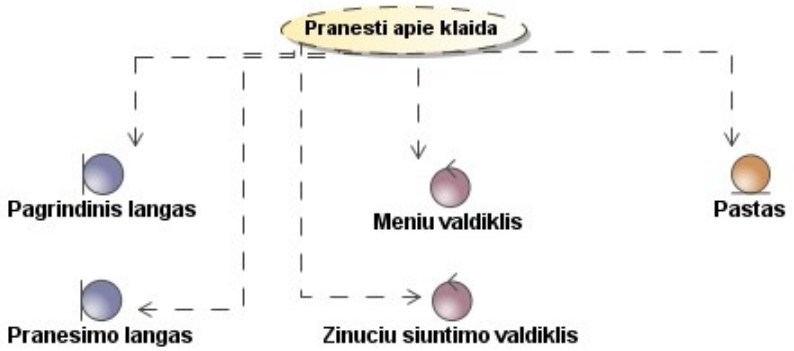
4.15 pav. PA "Informuoti vartotojus" realizacijos diagrama

Klaidų stebėjimo sistemos analizės schema pateikta 4.16 pav. 4.17-4.24 pav. pavaizduotos klaidų stebėjimo sistemos panaudojimo atvejų realizacijos diagramos.

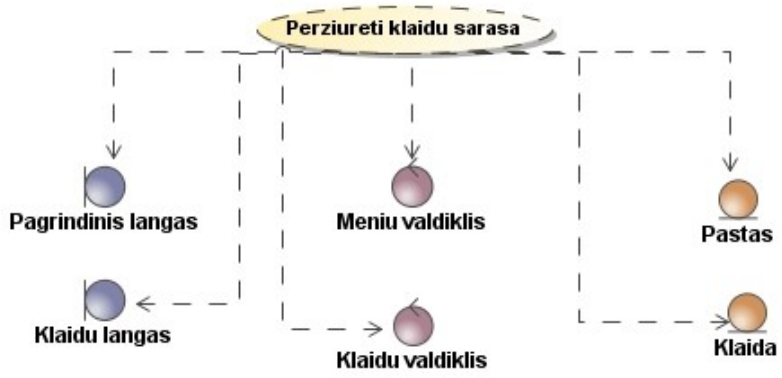
Analizės diagrama nėra pateikiama PA “Žaisti žaidimą” ir PA “Imituoti rungtynes”, nes jie nėra tiesiogiai susiję su atliekamais patobulinimais.



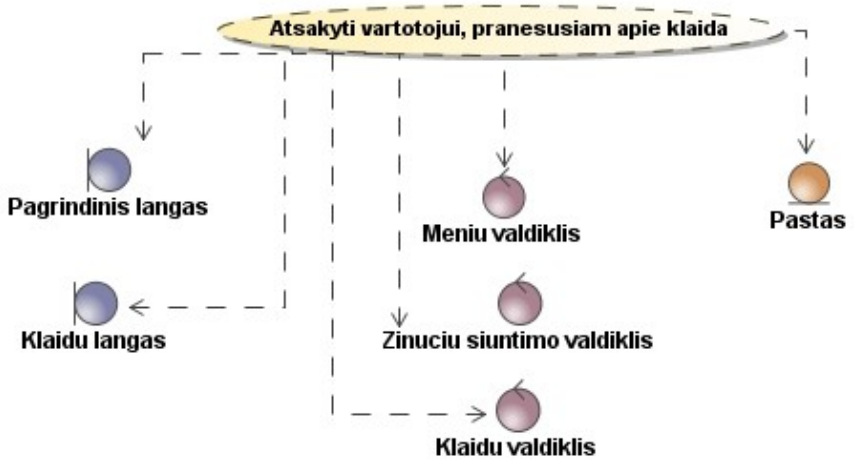
4.16 pav. Klaidų stebėjimo sistemos analizės diagrama



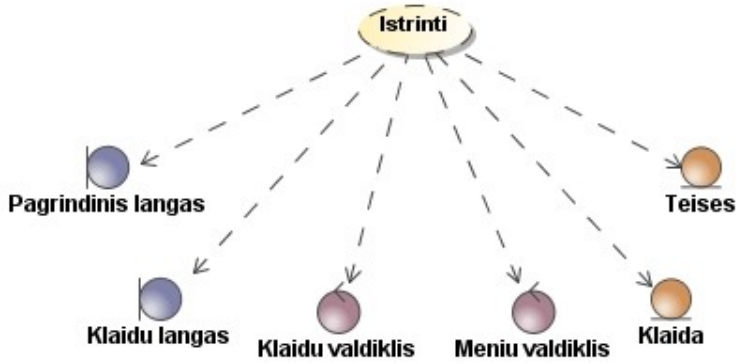
4.17 pav. PA “Pranešti apie klaidą” realizacijos diagrama



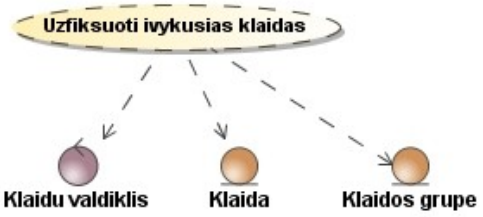
4.18 pav. PA “Peržiūrėti užfiksuotas klaidas” realizacijos diagrama



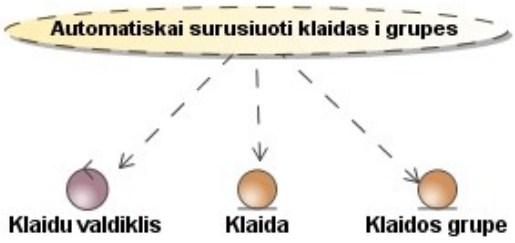
4.19 pav. PA “Atsakyti vartotojui, pranešusiam apie klaidą” realizacijos diagrama



4.20 pav. PA “Ištrinti pranešimą apie klaidą” realizacijos diagrama

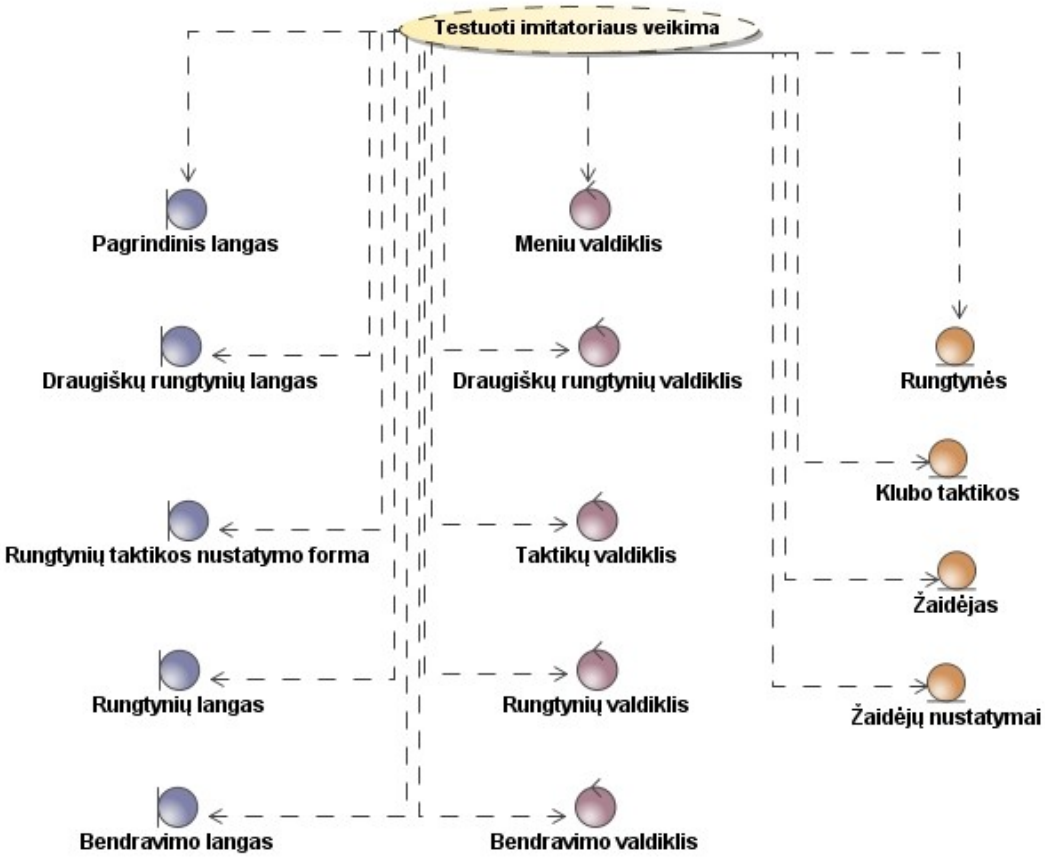


4.21 pav. PA “Nustatyti įvykusias klaidas” realizacijos diagrama



4.22 pav. PA “Sugrupuoti klaidas ” realizacijos diagrama

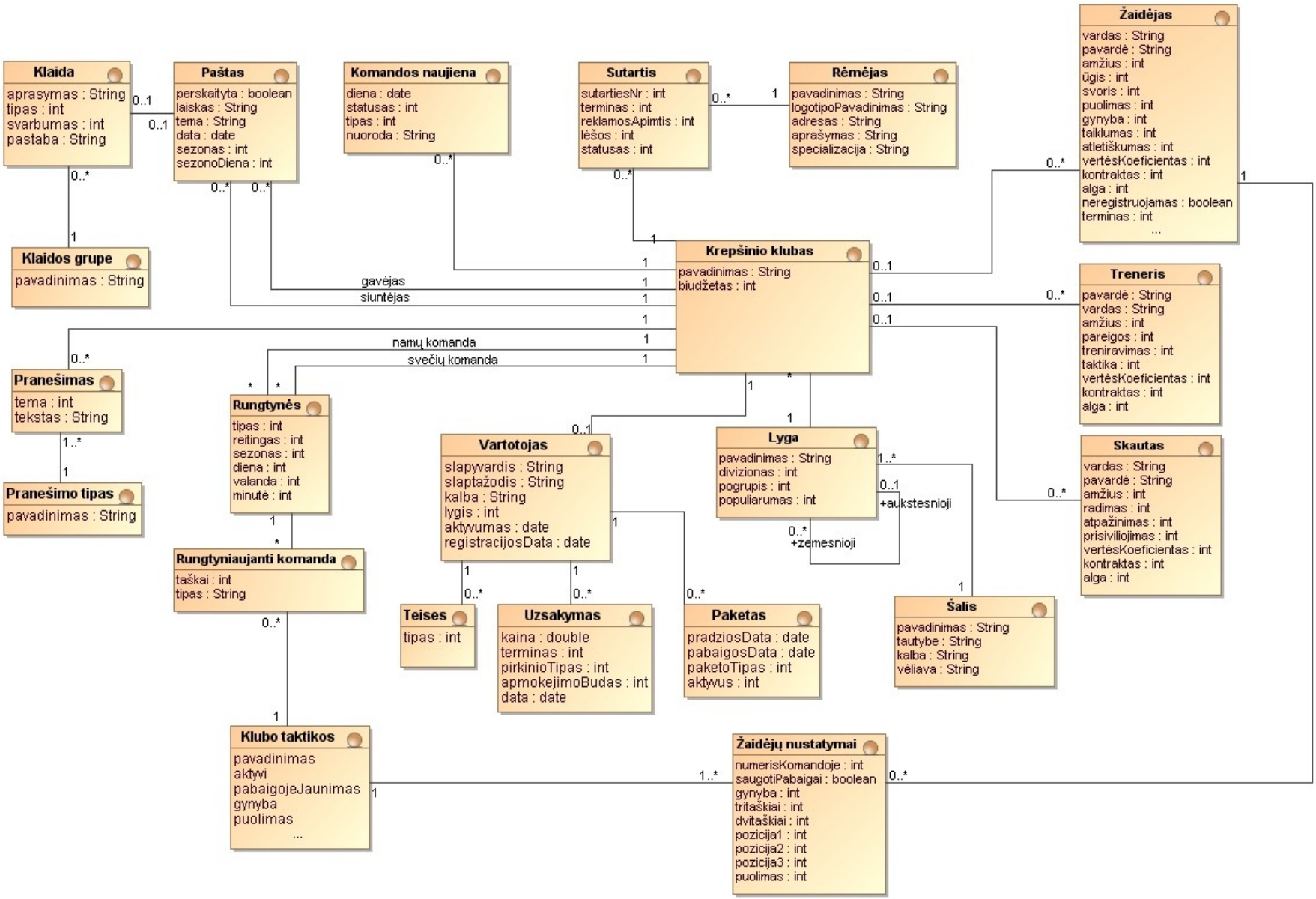
4.23 pav. PA “Informuoti apie įvykusias klaidas ” realizacijos diagrama



4.24 pav. PA “Testuoti imitatoriaus veikimą” realizacijos diagrama

4.3. Virtualaus krepšinio žaidimo dalykinės srities modelis

Dalykinės srities esybių modelis pateiktas 4.25 pav. Jame pavaizduotos pagrindinės esybės, naudojamos darbe, bei ryšiai tarp jų. Modelyje taip pat pavaizduoti esybių atributai.



4.25 pav. Dalykinės srities esybių modelis

5. Projektas

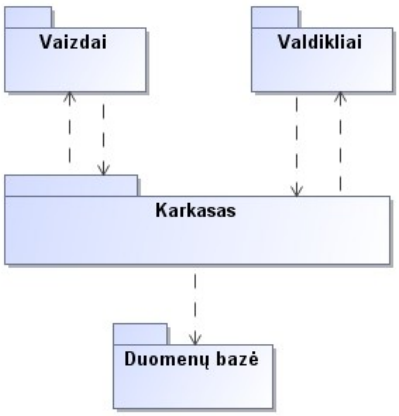
Analizės dalyje išnagrinėjome *Software-as-a-Service* (*SaaS*) metodologiją ir jos siūlomas rekomendacijas programų kūrimui, taip pat testavimo metodus ir esamas testavimo priemones. Remiantis atlikta analize buvo nuspręsta patobulinti virtualų krepšinio žaidimą pridedant dar dvi naujas posistemes.

Atlikdami šį projektą, konkrečiai nagrinėsime ir taikysime žaidimo posistemių kūrimui *SaaS* metodologijos rekomendacijas vartotojų apmokestinimo bei klaidų fiksavimo/taisymo atžvilgiu. Projektuojant klaidų fiksavimo posistemę taip pat naudosimės testavimo metodų analizės informacija.

Projekto tikslas – suprojektuoti ir realizuoti dvi papildomas virtualaus krepšinio žaidimo posistemes: papildomų paslaugų teikimo bei klaidų stebėjimo ir fiksavimo.

5.1. Virtualaus krepšinio žaidimo sistemos architektūra

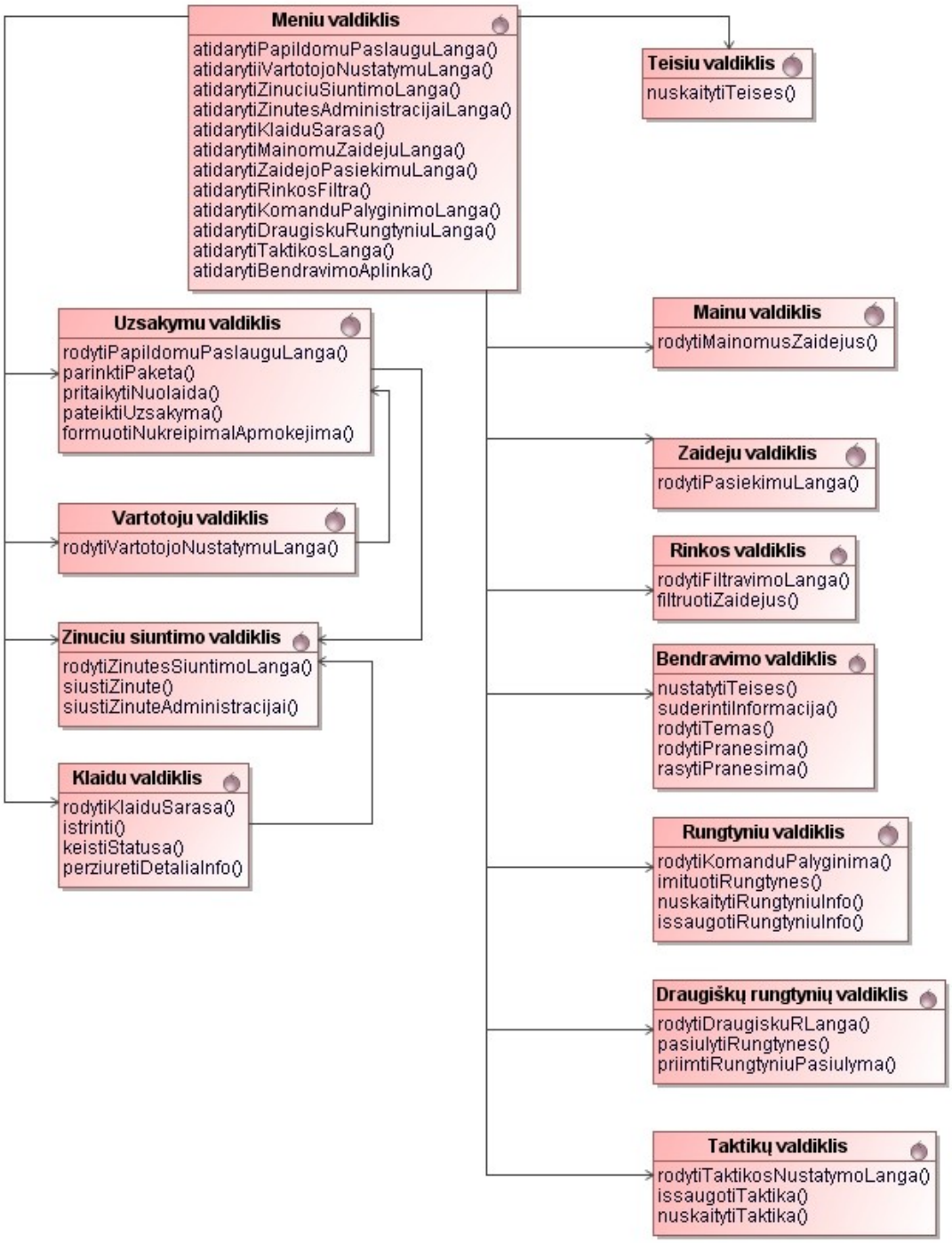
Sistemos architektūra pavaizduota 5.1 pav. Sistemą galima suskirstyti į keturias dalis: vaizdus, valdiklius, duomenų bazę bei karkasą, per kurį bendrauja likusios trys dalys. Karkase yra įgyvendintos funkcijos, palengvinančios duomenų apsikeitimą tarp atskirų sistemos dalių.



5.1 pav. Sistemos loginės architektūros diagrama

5.2. Projektuojamų žaidimo posistemių klasių modelis

5.2 pav. pateikta detali valdiklių klasių diagrama, kurioje pavaizduotos klasės bus naudojamos įvedant naujas analizės dalyje nustatytas posistemes.



5.2 Valdiklių klasių diagrama

5.1-5.13 lentelėse pateiktos kiekvienos klasės specifikacijos, kuriose pateikiami klasių metodai ir trumpi jų aprašymai.

5.1 lentelė

Meniu valdiklis	
Metodas	***Paskirtis***
atidarytiPapildomuPaslauguLanga()	Inicijuoja papildomų paslaugų užsakymo lango atidarymą
atidarytiVartotojoNustatymuLanga()	Inicijuoja vartotojo nustatymų lango atidarymą
atidarytiZinuciuSiuntimoLanga()	Inicijuoja žinučių siuntimo lango atidarymą

atidarytiZinutesAdministracijaiLanga()	Inicijuoja žinutės siuntimo administracijai lango atidarymą
atidarytiKlaiduSarasa()	Inicijuoja užfiksuotų klaidų sąrašo atidarymą
atidarytiMainomuZaidejuLanga()	Inicijuoja mainomų žaidėjų lango atidarymą
atidarytiZaidejoPasiekimuLanga()	Inicijuoja žaidėjo pasiekimų lango atidarymą
atidarytiRinkosFiltra()	Inicijuoja rinkos filtro nustatymo lango atidarymą
atidarytiKomanduPalyginimoLanga()	Inicijuoja komandų palyginimo lango atidarymą
atidarytiDraugiskuRungtyniuLanga()	Inicijuoja lango draugiškoms rungtynėms surengti atidarymą
atidarytiTaktikosLanga()	Inicijuoja taktikos nustatymo lango atidarymą
atidarytiBendravimoAplinka()	Inicijuoja bendravimo aplinkos atidarymą

5.2 lentelė

Teisių valdiklis	
Metodas	***Paskirtis***
NuskaitytiTeises()	Iš duomenų bazės nuskaito, kokias vartotojas turi teises

5.3 lentelė

Uzsakymu valdiklis	
Metodas	***Paskirtis***
rodytiPapildomuPaslauguLanga()	Surenka reikiamą informaciją ir parodo papildomų paslaugų langą
parinktiPaketa()	Paruošia informaciją apie pasirinktą paketą
pritaikytiNuolaida()	Apskaičiuoja nuolaidą, kuri taikoma vartotojui
pateiktiUzsakyma()	Išsaugo užsakymo informaciją
formuotiNukreipimaIApmokejima	Suformuoja nukreipimą į apmokėjimo sistemą

5.4 lentelė

Vartotoju valdiklis	
Metodas	***Paskirtis***
rodytiVartotojoNustatymuLanga()	Parodo vartotojo nustatymų langą

5.5 lentelė

Zinuciu siuntimo valdiklis	
Metodas	***Paskirtis***
rodytiZinutesSiuntimoLanga()	Parodo žinutės siuntimo langą
siustiZinute()	Išsaugo žinutės tam tikram klubui duomenis
siustiZinuteAdministracijai()	Išsaugo žinutės administracijai duomenis

5.6 lentelė

Klaidu valdiklis	
Metodas	***Paskirtis***
rodytiKlaiduSarasa()	Parodo klaidų sąrašą
istrinti()	Ištrinti užfiksuotą klaidą
keistiStatusa()	Pakeičia ir išsaugo klaidos statusą
perziuretiDetaliaInfo()	Nuskaito ir atvaizduoja detalią informaciją apie klaidą

5.7 lentelė

Mainu valdiklis	
Metodas	***Paskirtis***
rodytiMainomusZaidejus()	Parodo komandos norimų išmainyti žaidėjų sąrašą

5.8 lentelė

Zaideju valdiklis	
Metodas	***Paskirtis***
rodytiPasiekimuLanga()	Parodo žaidėjo pasiekimų langą

5.9 lentelė

Rinkos valdiklis	
Metodas	***Paskirtis***
rodytiFiltravimoLanga()	Parodo filtravimo nustatymų langą
filtruotiZaidejus()	Pagal nustatytą filtrą atrenka žaidėjus

5.10 lentelė

Bendravimo valdiklis	
Metodas	***Paskirtis***
nustatytiTeises()	Nustato nurodytas teises bendravimo aplinkoje
suderintiInformacija()	Suvienodina informaciją žaidime ir bendravimo aplinkoje
rodytiTemas()	Parodo temų sąrašą
rodytiPranesima()	Parodo konkretų pranešimą
rasytiPranesima()	Išsaugo naują pranešimą

5.11 lentelė

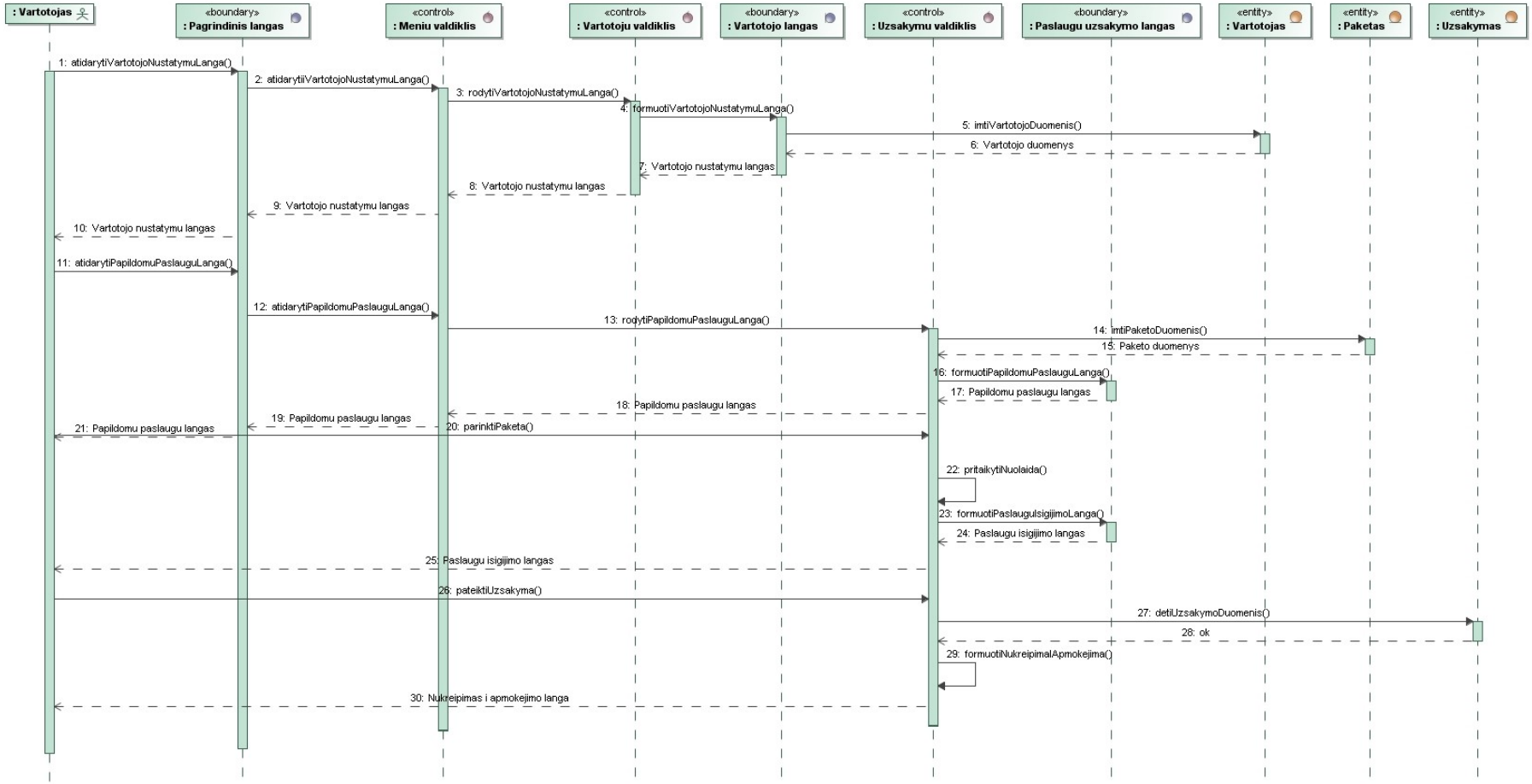
Rungtynių valdiklis	
Metodas	***Paskirtis***
rodytiKomanduPalyginima()	Parodo komandų palyginimo langą
imituotiRungtynes()	Inicijuoja rungtynių imitavimą
nuskaitytiRungtyniuInfo()	Nuskaito konkrečių rungtynių informaciją
issaugotiRungtyniuInfo()	Pasibaigus rungtynių imitavimui, išsaugo informaciją

5.12 lentelė

Draugiškų rungtynių valdiklis	
Metodas	***Paskirtis***
rodytiDraugiskuRLanga()	Parodo draugiškų rungtynių langą
pasiulytiRungtynes()	Suformuoja duomenis atvaizdavimui
priimtiRungtyniuPasiulyma()	Išsiunčia draugiškų rungtynių pasiūlymą konkrečiam klubui

5.13 lentelė

Taktiku valdiklis	
Metodas	***Paskirtis***
rodytiTaktikosNustatymoLanga()	Parodo taktikos nustatymo langą
issaugotiTaktika()	Išsaugo taktikos duomenis
nuskaitytiTaktika()	Nuskaito taktikos nustatymų duomenis



5.3 pav. Papildomų paslaugų užsakymo sekų diagrama

5.3. Kuriamos papildomų paslaugų posistemės elgsenos modelis

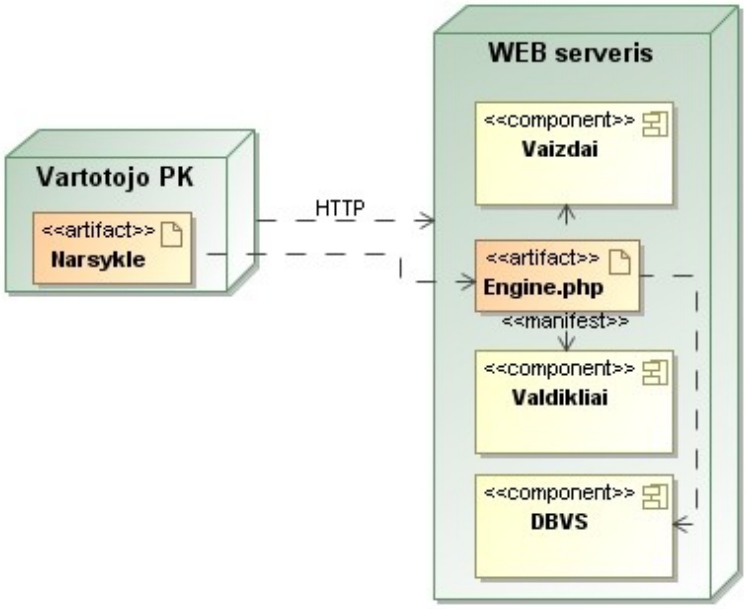
5.3 pav. pavaizduota papildomų paslaugų užsakymo sekų diagrama, vaizduojanti kaip vyksta bendravimas tarp klasių. Užsisakę papildomas paslaugas, vartotojai gali naudotis bet kuria iš jam suteiktų paslaugų.

Klaidų fiksavimo posistemės elgsenos diagramų čia nepateikiame, nes ši posistemė neturi vieno ištiso scenarijaus. Vartotojų atliekami veiksmai priklauso nuo vartotojo tipo ir poreikių. Kiekvieno panaudojimo atvejo aprašymas yra pateiktas reikalavimų specifikavimo skyriuje (4 sk.), o detaliau nagrinėti kiekvieno jų elgseną nėra svarbu, nes jų realizacija yra intuityviai numanoma.

6. Realizacija ir testavimas

6.1. Virtualaus krepšinio žaidimo komponentų ir įdiegimo diagramos

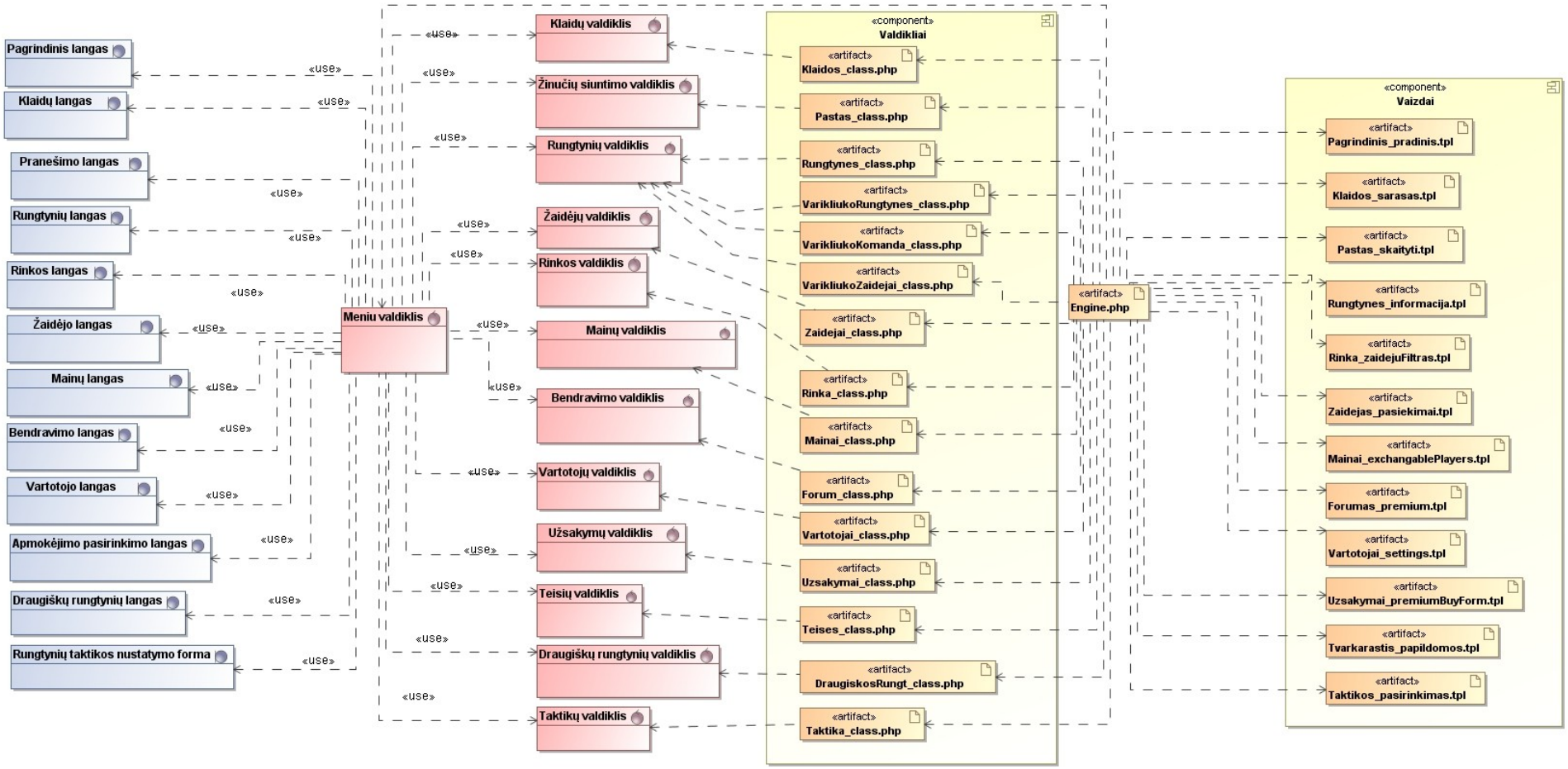
Sistemos diegimo diagrama pavaizduota 6.1 pav. Serveryje yra laikomi programos veikimui reikalingi vaizdai, valdikliai, duomenų bazė ir karkasas (Engine.php), kuris užtikrina tinkamą bendravimą tarp šių elementų.



6.1 pav. Sistemos diegimo diagrama

6.2 pav. pavaizduota sistemos komponentų diagrama.

Kiekvienas dešinėje pavaizduotas vaizdų artefaktas atitinka atvaizdavimo klases. Atvaizdavimo klasės yra išdėliotos ta pačia tvarka kaip ir jas atitinkantys artefaktai.



6.2 pav. Sistemos komponentų diagrama

6.2. Krepšinio žaidimą papildančių posistemių veikimas

- Papildomų paslaugų teikimo posistemė

Papildomų paslaugų posistemė sudaro galimybes vartotojams įsigyti teisę naudotis papildomomis paslaugomis, pavyzdžiui: senų taktikų peržiūra, žaidėjų veidų rodymas, komandos garbės lenta ir kt.

Vartotojams įsigijus papildomų paslaugų paketą, jiems suteikiamas aktyviausių vartotojų statusas. Aktyviausi žaidimo vartotojai taip pat yra ir patarėjai žaidimo vystymo klausimais. Jie gali nutarti, kurių savybių sistemai labiausiai trūksta ir kurias reikia pirmiausiai įgyvendinti ar patobulinti. Aktyviausiems vartotojams yra skirta tik jiems prieinama bendravimo aplinka, kurioje vartotojai gali diskutuoti žaidimo klausimais, teikti pasiūlymus ir pageidavimus. Patikimos bendruomenės turėjimas yra labai naudingas žaidimo tobulinimo aspektu.

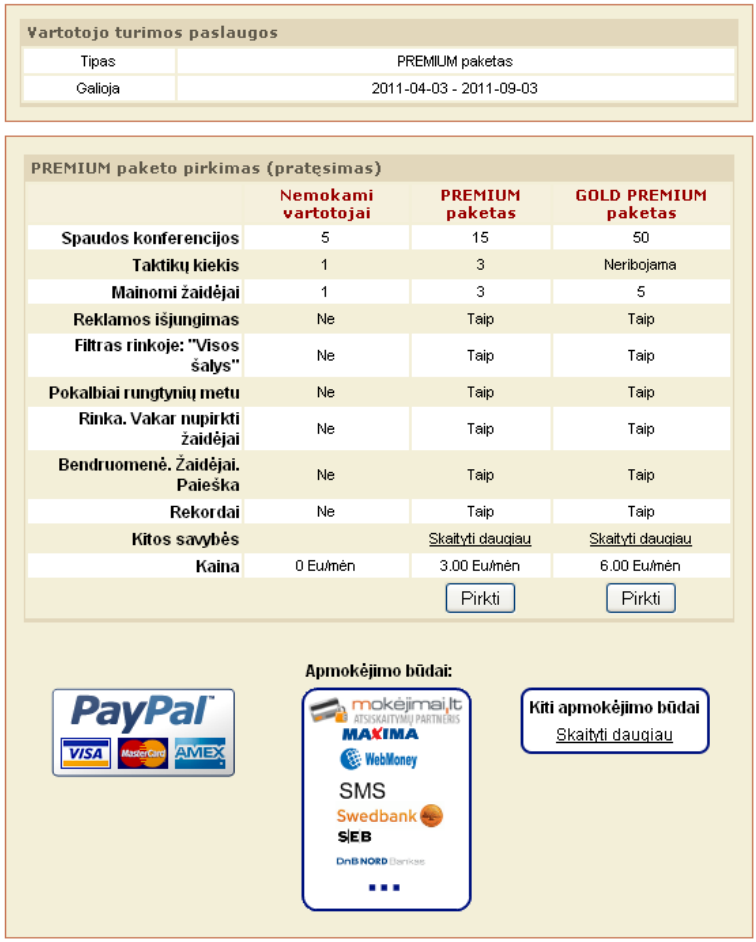
Realizuojant posistemę, buvo sukurta papildomų paslaugų užsakymo sistema. Iš keleto skirtingų tipų vartotojai gali pasirinkti norimą įsigyti paslaugą (6.3 pav.) Tuomet yra paskaičiuojama nuolaida (jeigu ją galima pritaikyti vartotojui) ir prašoma pasirinkti apmokėjimo būdą. Sistema yra suderinta su keletu apmokėjimo paslaugas teikiančių firmų. Kai tik mokėjimų tarpininkai pateikia patvirtinimą, kad apmokėjimas sėkmingai atliktas, vartotojui suteikiamas jo įsigytų paslaugų paketas, t.y. suteikiamos teisės naudotis įsigytomis paslaugomis. Pasibaigus paketo galiojimo laikotarpiui, vartotojas yra apie tai informuojamas ir gali prasitęsti paketo laikotarpį.

- Klaidų stebėjimo ir fiksavimo posistemė

Testavimo posistemės tikslas yra patikrinti ar visos realizuotos funkcijos veikia ir ar veikia tinkamai. Sistema ne tik turi atlikti suprogramuotas funkcijas, bet ir pranešti apie nepavykusius veiksmus.

Kadangi sistema, su kuria dirbama, yra sudėtinga ir nuolat tobulinama, įvairiais laiko momentais joje gali pasitaikyti įvairaus tipo klaidų:

1. duomenų bazės;
2. PHP;
3. loginės, automatiškai nustatytos sistemos;
4. vartotojų pastebėtos – loginės, atvaizdavimo ir kt.;
5. rungtynių imitavimo.



6.3 pav. Papildomų paslaugų užsakymas

Atsiradus klaidai, svarbu kuo greičiau pastebėti ir pašalinti ją, todėl klaidų registravimo sistema kaupia duomenis apie pasitaikiusias klaidas, praneša administratoriams ir informatyviai atvaizduoja klaidos duomenis.

Duomenų bazės klaidos. Šio tipo klaidos yra fiksuojamos ir saugomos atskiruose failuose. Tačiau tokia klaidų peržiūra užima daug laiko ir yra nepatogu nuolat tikrinti klaidų failus bei juose ieškoti įvykusių klaidų.

Naujai įgyvendinta duomenų bazės klaidų registravimo sistema atlieka keletą funkcijų:

- iš klaidų failų nuskaito reikiamą informaciją;
- informacija yra sugrupuojama – atrenkamos tokios pačios klaidos, nustatoma kiek kartų ir kada pasikartojo paskutinį kartą;
- reikalinga informacija atvaizduojama administratoriams;
- administratoriai gali peržiūrėti klaidas (6.4 pav.), o problemą išsprendus – ištrinti iš sąrašo.

Trinti visas klaidas

Trinti pazymetas

Trinti: ☐ **Kiek**: 1; **Data**: 2010-11-16 01:14:18 - ; **IP**: 85.255.60.183 ; **Klubo ID**: 8 ;
Klaida: 054 ** Unknown column 'statusas' in 'order clause']
ERROR : 054 ** Unknown column 'statusas' in 'order clause']
Užklausa:
SELECT Z.id, Z2S.sezonas as atejimo_sezonas, Z.pozicija, IFNULL(NZ.numeris_komandoje, 16) as numeris, 'turi' as sStatusas FROM (zaidejai as Z, salys as S) LEFT JOIN nustatymai_zaid as NZ ON (NZ.zaidejo_id = Z.id AND NZ.taktikos_id = 1755279) LEFT JOIN talentai2skautai as Z2S ON (Z2S.personalo_id = Z.id AND Z2S.talento_tipas = 1) WHERE Z.komandos_id = 8 AND S.id = Z.salis AND Z.kontraktas > 1 GROUP BY Z.id UNION SELECT Z.id, Z2S.sezonas as atejimo_sezonas, Z2S.savaite as atejimo_savaite, CONCAT(LEFT(Z.vardas, 1) , '. ', Z.pavarde) AS inicialai, CONCAT(Z.vardas, '_', Z.pavarde) AS inicialai_seo, ZP.alga, ZP.terminas - 1 as kontraktas, S.veliava, Z.vk, Z.potencialas, Z.karjeros_pabaiga, Z.amzius, Z.pavarde, Z.ugis, Z.statusas as zaidejoStatusas, Z.pozicija, IFNULL(NZ.numeris_komandoje, 16) as numeris, 'apribotas' as sStatusas FROM zaideju_pasiulymai as ZP JOIN zaidejai as Z ON Z.id = ZP.zaidejo_id JOIN salys as S ON S.id = Z.salis JOIN zaideju_pratesimas as ZPA ON (ZPA.zaidejo_id = Z.id AND ZPA.diena < 44) LEFT JOIN nustatymai_zaid as NZ ON (NZ.zaidejo_id = Z.id AND NZ.taktikos_id = 1755279) LEFT JOIN talentai2skautai as Z2S ON (Z2S.personalo_id = Z.id AND Z2S.talento_tipas = 1) WHERE ZP.komandos_id = 8 AND ZP.tipas = 2 GROUP BY Z.id ORDER BY statusas desc
BackTrace:
',) (Eil: 887)
PEAR.php DB_Error->DB_Error('-19', '1', '1024', '
',) (Eil: 557)
DB/common.php PEAR->raiseError('', '-19', '', '', '
', 'DB_Error', '1',) (Eil: 1850)
DB/mysql.php DB_common->raiseError('-19', '', '', '', '1054 ** Unknown',) (Eil: 902)
DB/mysql.php DB_mysql->mysqlRaiseError() (Eil: 332)
DB/common.php DB_mysql->simpleQuery('
',) (Eil: 1163)
DB/common.php DB_common->query('
',) (Eil: 1610)
pagalbiniai/Zaidejai_class.php DB_common->getAll('
',) (Eil: 395)
pagalbiniai/Zaidejai_class.php Zaidejai_class->getZaidejaiFinansamsAtkrintamuju('1',) (Eil: 148)
Eng.php Zaidejai_class->_finansai('statusas', 'desc', '1',) (Eil: 284)
Eng.php Eng->bandom_uzkrauti_moduli() (Eil: 175)
index.php Eng->render() (Eil: 16)

Trinti: ☐
5. Qualsiasi strumento utilizzato per nascondere l'IP e altri strumenti simili sono strettamente proibiti.
6. Quando si scambiano giocatori devono essere seguite le regole di scambio.
7. Quando si scrive un post in conferenza stampa devono essere seguite le regole di conferenze stampa.
8. Quando si scrive un post nel forum devono essere seguite le regole del forum.
9. È vietato scrivere messaggi indesiderati (spam, ad esempio, pubblicità) ad altri utenti.
10. È vietato usare frasi volgari o offensive verso altri utenti durante la comunicazione (in conferenze stampa, messaggi personali e altrove)
11. Il logo del Team non deve contenere nudità, riferimenti politici, razziali o pornografici. Il logo deve essere neutrale e non deve offendere

6.4 pav. Duomenų bazės klaidų sąrašas

PHP klaidos. Įvykus šio tipo klaidoms, informacija taip pat yra išsaugomos specialiuose failuose. Klaidų registravimo sistema:

1. nuskaito duomenis apie klaidas;
2. struktūrizuota ir vaizdžiai pateikia atitinkamą informaciją;
3. leidžia administratoriams peržiūrėti (6.5 pav.) ir ištrinti informaciją apie įvykusias klaidas.

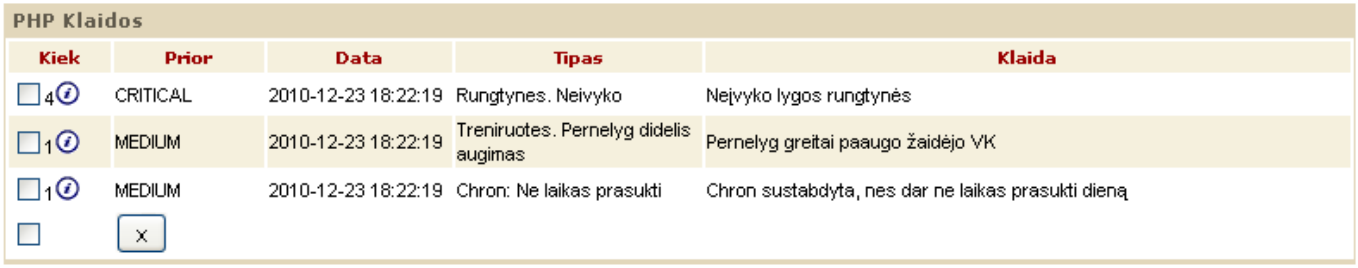
PHP Klaidos

Kiek	Data	Tipas	Klaida	Kur
☐ 2	14-Nov 14:01:05	Fatal	Call to undefined method RungtynesWap_class::getVykstanciuRungtyniuLangas()	m.bballzone.net/RungtynesWap_class.php on line 53
☐ 3	15-Nov 12:52:11	Warning	unlink(logs/db/2010-11-15_09-284ce0e10d1ddeb.log) [function.unlink]: No such file or directory	FailaiFnc_class.php on line 203
☐ 1	15-Nov 23:30:06	Fatal	Allowed memory size of 134217728 bytes exhausted (tried to allocate 42 bytes)	DB/mysql.php on line 388
☑ 8	16-Nov 01:14:18	Notice	Trying to get property of non-object	Zaidejai_class.php on line 399
☑ 8	16-Nov 01:14:18	Notice	Trying to get property of non-object	Zaidejai_class.php on line 157
☐	Trinti			

6.5 pav. PHP klaidų sąrašas

Sistemos nustatytos loginės klaidos. Šių klaidų atsiradimą lemia sistemos sudėtingumas. Sistema yra sudaryta iš daugelio glaudžiai susietų posistemių (rėmėjų, finansų, rinkos ir kt.) Nors atnaujinimai yra testuojami kūrimo metu bei pabaigus realizuoti, esant tam tikroms sąlygoms, jie gali paveikti kitas sistemos dalis ir tai gali likti nepastebėta. Todėl svarbiose sistemos dalyse įdiegiamos apsaugos, nustatančios nenumatytas situacijas (kurios normaliomis sąlygomis negali įvykti). Jeigu situacija kritinė, stabdomas tolesnis sistemos veiksmų vykdymas ir apie tai informuojamas administratorius.

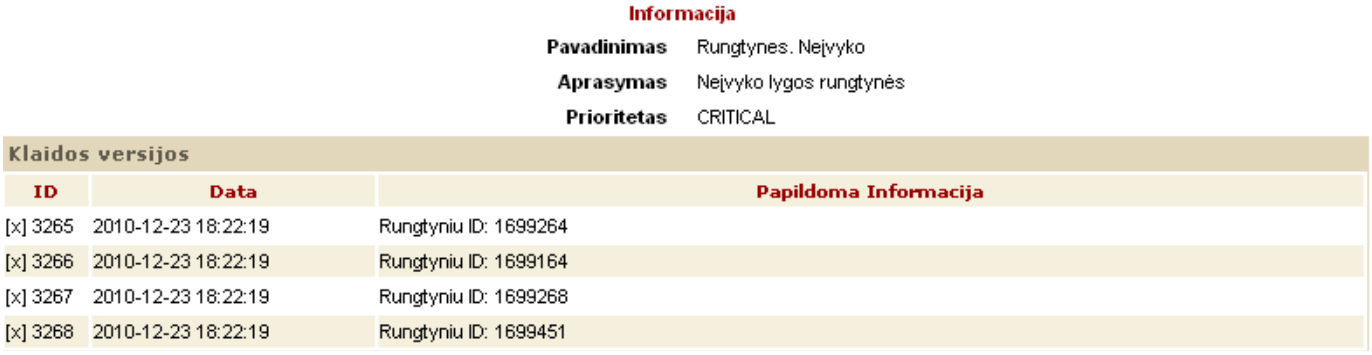
Administratoriai gali peržiūrėti sistemos užfiksuotų loginių klaidų sąrašą (6.6), smulkesnę informaciją apie konkrečią klaidą (6.7), o ištaisius klaidą – ištrinti.



PHP Klaidos

Kiek	Prior	Data	Tipas	Klaida
4	CRITICAL	2010-12-23 18:22:19	Rungtynes. Neivyko	Neįvyko lygos rungtynės
1	MEDIUM	2010-12-23 18:22:19	Treniruotes. Pernelyg didelis augimas	Pernelyg greitai paaugo žaidėjo VK
1	MEDIUM	2010-12-23 18:22:19	Chron: Ne laikas prasukti	Chron sustabdyta, nes dar ne laikas prasukti dieną
	x			

6.6 pav. Sistemos užfiksuotų loginių klaidų sąrašas



Informacija

Pavadinimas Rungtynes. Neįvyko

Aprasymas Neįvyko lygos rungtynės

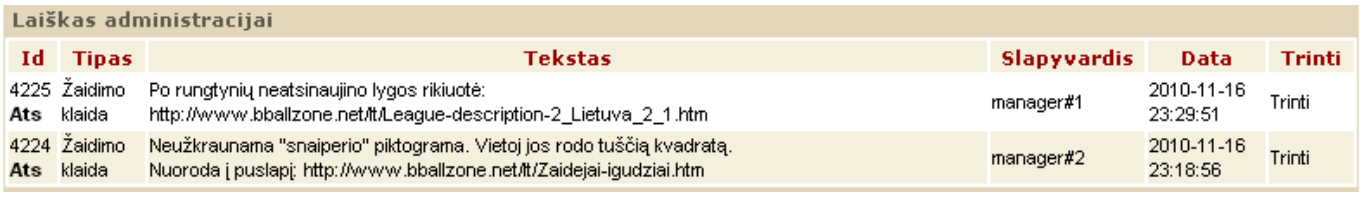
Prioritetas CRITICAL

Klaidos versijos

ID	Data	Papildoma Informacija
[x] 3265	2010-12-23 18:22:19	Rungtyniu ID: 1699264
[x] 3266	2010-12-23 18:22:19	Rungtyniu ID: 1699164
[x] 3267	2010-12-23 18:22:19	Rungtyniu ID: 1699268
[x] 3268	2010-12-23 18:22:19	Rungtyniu ID: 1699451

6.7 pav. Informacija apie konkrečią klaidą

Vartotojų pastebėtos klaidos. Vartotojų pastebėtos klaidos yra kaupiamos duomenų bazėje ir atvaizduojamos administracijai. Jei vartotojas pastebi klaidą (loginę, atvaizdavimo ar kt.), jis gali pranešti apie tai administracijai. Administracija mato visus vartotojų pranešimus (6.8 pav.) ir gali atsakyti vartotojui, pranešusiam apie klaidą.



Laiškas administracijai

Id	Tipas	Tekstas	Slapyvardis	Data	Trinti
4225 Ats	Žaidimo klaida	Po rungtynių neatsinaujino lygos rikiuotė: http://www.bballzone.net/lt/League-description-2_Lietuva_2_1.htm	manager#1	2010-11-16 23:29:51	Trinti
4224 Ats	Žaidimo klaida	Neužkraunama "snaiperio" piktograma. Vietoj jos rodo tuščią kvadratą. Nuoroda į puslapį: http://www.bballzone.net/lt/Zaidejai-igudziai.htm	manager#2	2010-11-16 23:18:56	Trinti

6.8 pav. Vartotojų pastebėtų klaidų sąrašas

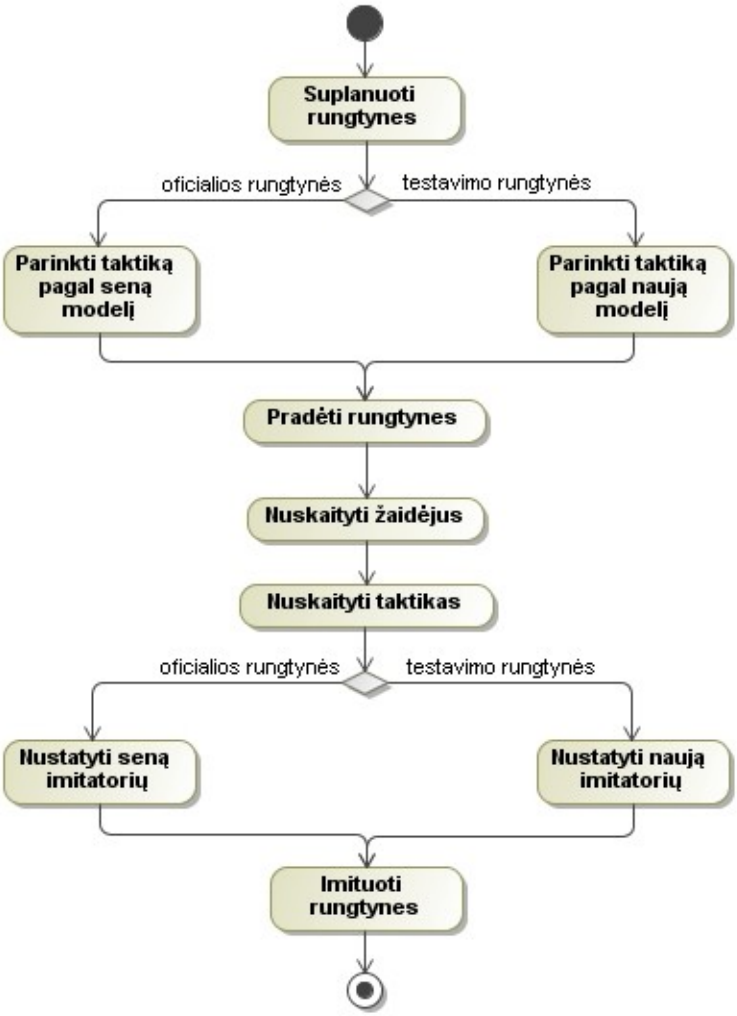
Rungtynių imitavimo testavimas. Rungtynių imitavimo posistemė yra viena iš svarbiausių ir jautriausių pasitaikiusioms klaidoms. Rungtynių imitavimas yra sudėtinga sistemos dalis, kuriai būtinas įvairiapusis testavimas.

Sistemoje nuolat vyksta didelis skaičius rungtynių, kurios yra labai skirtingos – nustatytomis taktikomis, komandų žaidėjais ir kitais parametrais. Būtina suderinti, kad rungtynių imitatorius patenkinamai veiktų visose rungtynėse – neužtenka patikrinti rezultatus keleto ar kelių dešimčių rungtynių. Vien programuotojams pilnai ištestuoti imitatoriaus veikimą yra neįmanoma, nes kiekvienas rungtynes būtina stebėti ir analizuoti.

Siekiant visuomet turėti tinkamai veikiantį rungtynių imitatorių, tačiau tuo pat metu nestabdyti jo tobulinimo, į testavimo procesą reikia įtraukti ir vartotojus. Rungtynių imitavimo testavimo sistema leidžia tuo pačiu metu sistemoje naudoti pagrindinį imitatorių ir testuoti tobulinamą imitatoriaus versiją. Dviejų imitatorių įtraukimas į sistemą pavaizduotas 6.9 pav.

Rungtynių imitavimo testavimo sistemos ypatybės:

- Tuo pačiu metu sistemoje veikia du rungtynių imitatoriai – senas ištestuotas ir naujas patobulintas.
- Ištestuotas imitatorius naudojamas visoms oficialioms rungtynėms – nacionalinio bei tarptautinio turnyrų.
- Testuojamas imitatorius naudojamas tik draugiškose testavimo rungtynėse.
- Visi vartotojai tuo pačiu metu gali žaisti oficialias rungtynes bei testuoti tobulinamą imitatorių. Vartotojai gali reikšti nuomonę apie jo kokybę, pranešti apie pastebėtas klaidas.
- Kadangi daug vartotojų testuoja imitatorių, ganėtinai greitai pastebimos klaidos ir nustatoma, ką reikia keisti.
- Kartu su vartotojais pilnai ištestavus ir ištaisius imitatoriaus klaidas, senąjį imitatorių galima pakeisti nauju.



6.9 pav. Rungtynių vykdymo veiklos diagrama, kai sistemoje yra du rungtynių imitatoriai

6.3. Įdiegtų posistemių testavimo modelis

Testavimo tikslai.

Prieš pradedant naudoti naujai kuriamas posistemes, jos turi būti testuojamos. Testavimo metu yra randamos ir ištaisomos klaidos, įvairiais aspektais patikrinamas sistemos veikimo teisingumas ir patikimumas.

Šio darbo metu buvo realizuotos dvi posistemės: papildomų paslaugų teikimo ir klaidų fiksavimo. Abiems šioms posistemėms svarbu ištestuoti, kad nebūtų realizacijos klaidų, kad sistema veiktų korektiškai – taip, kaip numatyta. Visų įmanomų situacijų patikrinti neįmanoma, todėl svarbu užtikrinti, kad klaidų nebūtų svarbiausiose posistemių dalyse, o jeigu klaidų pasitaikys – kad jos būtų greitai

ištaisytos. Įgyvendintos posistemės yra skirtingos ir testuojant jas bus akcentuojami skirtingi kriterijai.

Papildomų paslaugų posistemėje jautriausias klaidoms funkcionalumas yra apmokėjimui reikalingų duomenų išsiuntimas, priėmimas ir paslaugos suteikimas. Ypatingai svarbu užtikrinti, kad šioje vietoje klaida neįvyks, o jei dėl nenumatytų aplinkybių taip atsitiktų – kad ji bus operatyviai ištaisyta. Minėtos operacijos turi būti įvykdytos pilnai ir viena po kitos.

Klaidų stebėjimo ir fiksavimo posistemė yra skirta pagerinti viso virtualaus krepšinio sistemos testavimo sąlygoms. Ji padeda operatyviai užfiksuoti klaidas ir pašalinti jas. Pačią šią posistemę prieš pradedant naudoti taip pat reikia ištestuoti. Klaidoms jautriausias šios posistemės funkcionalumas – tai dviejų rungtynių imitatorių panaudojimas skirtingoms rungtynėms imituoti. Svarbu užtikrinti, kad oficialių rungtynių imitatorius klaidų neturėtų ir būtų pilnai ištestuotas prieš pradedant jį naudoti.

Apribojimai.

- Laikas – sistema yra nuolat tobulinama ir negalima pernelyg ilgai užtrukti prie vieno atnaujinimo. Kiekvienas atnaujinimas turi nustatytą pabaigos datą, iki kurios jį būtina įgyvendinti, nes pradės strigti kiti sistemos veiksmai.
- Darbuotojai – nėra pakankamai darbuotojų, kurie galėtų užsiimti tik viena konkrečia sritimi, t.y. nėra galimybių turėti vieną žmogų programuotoją, o kitą – testuotoją. Viena iš realizuotų posistemių (klaidų fiksavimo) turėtų padėti sumažinti šį trūkumą – prie testavimo prisidės ir žaidimo vartotojai, tad bus patikrinami įvairūs variantai ir pasitaikius klaidoms bus galima jas operatyviai pašalinti.

Testavimo strategija.

Sistemos atnaujinimų testavimas atliekamas keletu etapų.

1. *Baltos dėžės testavimas*

Posistemių realizavimo metu yra atliekami testai baltos dėžės principu. Suprasdamas kaip sistema turi veikti ir žinodamas, kaip sistema yra realizuota, programuotojas stengiasi patikrinti tikėtinas situacijas. Yra tikrinami scenarijai:

- Pagrindinė vartotojo elgsena – kaip yra tikimasi, kad vartotojas teisingai naudosis sistema. Tikrinama, ar sistema veikia korektiškai, kai vartotojas atlieka veiksmus, kurių iš jo tikimasi. Yra patikrinami visi galimi atvejai ir jų veikimo teisingumas.
- Negalimų įvesties duomenų testavimas – tikrinama, kaip sistema elgiasi, jei vartotojas parenka negalimą įvesties duomenų rinkinį. Tai apima atvejus, kai kažkuris būtinas pasirinkimas yra nenurodytas, neįvesti tam tikri būtini duomenys arba įvesti duomenys yra negalimi. Tikrinama, ar sistema atitinkamai sustabdo vykdymą ir vartotojui pateikia informatyvų pranešimą.

Testai yra atliekami programavimo metu, keletą kartų patikrinant tas pačias vietas. Pradžioje yra tikrinamas kiekvienos naujai sukurtos funkcijos veikimas, vėliau tos pačio vietos tikrinamos sujungiant viską į vieną sistemą. Iš pradžių tikrinamas kiekvienos funkcijos veikimo teisingumas, o vėliau – ar visų funkcijų sistema veikia korektiškai ir netrukdo viena kitai.

Programuotojai atlieka ir *sąsajų testavimą* – t.y. patikrina ar atskiri moduliai tinkamai bendrauja vienas su kitu, ar dėl to neatsiranda klaidų. Taip pat yra atliekamas ir *saugumo testavimas* – patikrinamos kiekvienai vartotojų grupei suteiktos teisės ir kiekvienai grupei leidžiami/neleidžiami veiksmai, teisių ir prieinamų funkcijų atitikimas. Patikrinamos apsaugos nuo galimų apėjimo būdų, jų veikimas.

Testuojant baltos dėžės principu, siekiama surasti ir ištaisyti svarbiausias klaidas – kurios gali pasitaikyti dideliam kiekiui vartotojų ir trukdytų naudoti pagrindines sistemos funkcijas. Kartais, dėl sistemos sudėtingumo (pvz., rungtynių imitavimo posistemė), neįmanoma patikrinti visų galimų situacijų, todėl šiame etape nėra pilnai užtikrinama, kad sistemoje nepasitaikys klaidų.

2. Suderinamumo testavimas

Posistemių kūrimo metu yra testuojamas jų suderinamumas su išorinėmis sistemomis/produktais.

Kaip ir kitiems internetiniams projektams, svarbu užtikrinti suderinamumą su populiariausiomis interneto naršyklėmis. Kūrimo metu bei pabaigus realizuoti posistemę, tikrinama kaip yra atvaizduojami puslapiai skirtingose naršyklėse, ar veikia visos realizuotos funkcijos.

Realizuojant papildomų paslaugų teikimo posistemę, viena iš įgyvendintų funkcijų – apmokėjimas už įsigytą papildomų paslaugų paketą. Šio funkcionalumo

įgyvendinimas apima ir sąsają su išorine sistema – apmokėjimų atlikimo tarpininku. Šis funkcionalumas yra labai jautrus klaidoms, kadangi yra susijęs su pinigais. Todėl buvo atliktas kruopštus testavimas – užsakymo pateikimo, duomenų persiuntimo tarpininkų sistemoms, duomenų iš tarpininkų sistemų priėmimo bei užsakytų paslaugų suteikimo. Buvo ištestuotas ne tik funkcijos veikimas mūsų realizuotoje sistemoje, bet ir suderinamumas su tarpininkų sistemomis (PayPal, mokejimai.lt)

3. Juodos dėžės testavimas

Realizavus atnaujinimus, jų veikimą juodosios dėžės principu testuoja vartotojai. Žaidimo vartotojai yra susipažinę su žaidimo sistema, o įdiegus atnaujinimus jiems pateikiama informacija apie atnaujinimus ir jų veikimą. Taigi vartotojai žino, kaip turi veikti sistema, tačiau jiems nesvarbu kaip ji yra realizuota.

Vartotojai išbando ir patikrina įvairias naujos posistemės galimybes. Kadangi vartotojų yra didelis skaičius ir jie turi skirtingas teises (įsigiję papildomų paslaugų paketus ir jų neturintys, moderatoriai ir paprasti vartotojai), jų poreikiai taip pat gali būti skirtingi. Taip yra ištestuojama daugelis galimų posistemės panaudojimo atvejų.

Apie pastebėtas klaidas vartotojai gali pranešti forume, o įvedus klaidų fiksavimo posistemę tam yra skirta ir atskira forma.

Šiame testavimo etape yra patikrinamas atnaujinimų funkcionalumas, identifikuojami trūkumai ir nustatomas nekorektiškas veikimas.

4. *Patogumo naudotis testavimas*

Patogumą naudotis taip pat nustato žaidimo vartotojai. Išbandydami sistemos funkcionalumą, vartotojai taip pat testuoja ir patogumą naudotis.

Jeigu dėl kažkurių funkcijų vartotojams kyla daug klausimų, daroma išvada, kad sistemoje susigaudyti nėra paprasta ir reikia keisti atvaizdavimą, išdėstymą ar pridėti daugiau informacijos. Dėl pakeitimų yra diskutuojama ir tariamasi su vartotojais, atsižvelgiama į jų pasiūlymus ir pageidavimus.

Kadangi žaidimas yra prieinamas keletu kalbų, patikrinamas posistemių vaizdas naudojant įvairias kalbas.

7. Virtualiame krepšinio žaidime atliktų patobulinimų eksperimentinis tyrimas

7.1. Eksperimento su atliktų patobulinimų rezultatais apibrėžimas

Mūsų darbo objektas – virtualus krepšinio žaidimo vystymo procesas. Žaidimas turi nuolat tobulėti, kad būtų įvairiapusiškas (trauktų visokių tokio tipo žaidimų mėgėjų dėmesį) ir išlaikytų vartotojų susidomėjimą. Žaidimas turi tobulėti ne tik naujomis savybėmis, tačiau ir finansiškai – t.y. turi kompensuoti kūrimo kaštus ir nešti pelną. Šio darbo metu buvo realizuotos dvi posistemės (papildomų paslaugų teikimo bei klaidų stebėjimo ir fiksavimo), kurios prisideda prie žaidimo tobulinimo proceso sąlygų gerinimo.

Eksperimento tikslas yra nustatyti naujai realizuotų sistemų naudą virtualiam krepšinio žaidimui ir jo vystymo procesui. Pagal nustatytus kriterijus bus lyginami sistemos parametrai prieš atnaujinimus ir po jų. Įvertinimui bus naudojami šie kriterijai:

- Iš žaidimo gaunamos pajamos;
- Vartotojų indėlis į žaidimo vystymą;
- Klaidų ištaisymo trukmė.

Kaip jau buvo minėta anksčiau, žaidimas turi būti nuolat tobulinamas, kad taptų vis įdomesnis ir išlaikytų vartotojų susidomėjimą. Didžiausią įtaką šiam procesui daro sistemos kūrėjai, įgyvendindami patobulinimus, tačiau prie to taip pat prisideda žaidimo moderatoriai (padedantys prižiūrėti žaidimą) bei paprasti vartotojai. Žaidimo vystymo procesui svarbios kelios grandys: idėjų generavimas, idėjų aptarimas ir detalizavimas, idėjų įgyvendinimas ir įdiegimas bei vartotojų informavimas apie atnaujinimus.

Įdiegus papildomų paslaugų posistemę atsirado naujas vartotojų tipas – vartotojai, kurie yra labiau susidomėję ir įsigilinę į žaidimą nei kiti vartotojai. Papildomų paslaugų paketai gali būti gaunami dviem būdais – įsigytas už nustatytą mokestį arba paskirtas už pagalbą tobulinant žaidimą. Vienu atveju žaidimas yra paremiamas finansiškai, kitu – konkrečiu žaidimo vystymui naudingu darbu.

Tyrimas bus vykdomas sistemos kūrėjų atžvilgiu, realiai naudojamoje krepšinio žaidimo aplinkoje (praktinis eksperimentas). Eksperimentas bus atliekamas dviejų magistratūros studentų ir virtualaus krepšinio žaidimo kūrėjų, taigi subjektai yra įsigilinę į tyrimo objektą ir šioje srityje sukaupę nemažą patirtį. Eksperimento metu

bus tiriama atnaujinta virtualaus krepšinio žaidimo sistema ir atnaujinimų įtaka sistemai.

Hipotezės.

Kaip jau minėta aukščiau, įvestų posistemių naudą sistemai tirsime remdamiesi trimis kriterijais, todėl kiekvienam jų suformuluosime po dvi hipotezes – patvirtinančią bei paneigiančią naudą.

Apibrėšime pažymėjimus, kurie bus naudojami šiame skyrelyje.

p_s – per mėnesį iš žaidimo gautos pajamos, prieš papildomų paslaugų posistemės įvedimą.

p_n – per mėnesį iš žaidimo gaunamos pajamos po papildomų paslaugų posistemės įvedimo.

Iškeliame hipotezes:

H_{p0} – prieš papildomų paslaugų posistemės įvedimą gaunamos mėnesinės pajamos buvo didesnės arba lygios pajamoms, gaunamoms po posistemės įvedimo:

$$H_{p0} : \overline{p}_s \succcurlyeq \overline{p}_n$$

H_{p1} – prieš posistemės įvedimą gaunamos mėnesinės pajamos buvo mažesnės nei po:

$$H_{p1} : \overline{p}_s < \overline{p}_n$$

i_s – vartotojų įtraukimo į žaidimo vystymą įvertis prieš papildomų paslaugų posistemės įvedimą.

i_n – vartotojų įtraukimo į žaidimo vystymą įvertis po papildomų paslaugų posistemės įvedimo.

H_{i0} – prieš papildomų paslaugų posistemės įvedimą vartotojų įtraukimas į žaidimo vystymo procesą buvo didesnis arba lygus esamam vartotojų įtraukimui po posistemės įvedimo:

$$H_{i0} : i_s \succcurlyeq i_n$$

H_{i1} – prieš papildomų paslaugų posistemės įvedimą vartotojų įtraukimas į žaidimo vystymą buvo mažesnis nei esamas vartotojų įtraukimas po posistemės įvedimo:

$$H_{i1} : i_s < i_n$$

k_s – sistemos patikimumas prieš klaidų stebėjimo posistemės įvedimą.

k_n – sistemos patikimumas po klaidų stebėjimo posistemės įvedimo.

H_{k0} – prieš klaidų stebėjimo posistemės įvedimą sistemos patikimumas buvo didesnis arba lygus sistemos patikimumui po posistemės įvedimo:

$$H_{k0} : k_s \geq k_n$$

H_{k1} – prieš klaidų stebėjimo posistemės įvedimą sistemos patikimumas buvo mažesnis už nukentėjusių vartotojų skaičių po posistemės įvedimo:

$$H_{k1} : k_s < k_n$$

7.2. Atliktų patobulinimų eksperimento rezultatų analizė

Iš žaidimo gaunamos pajamos.

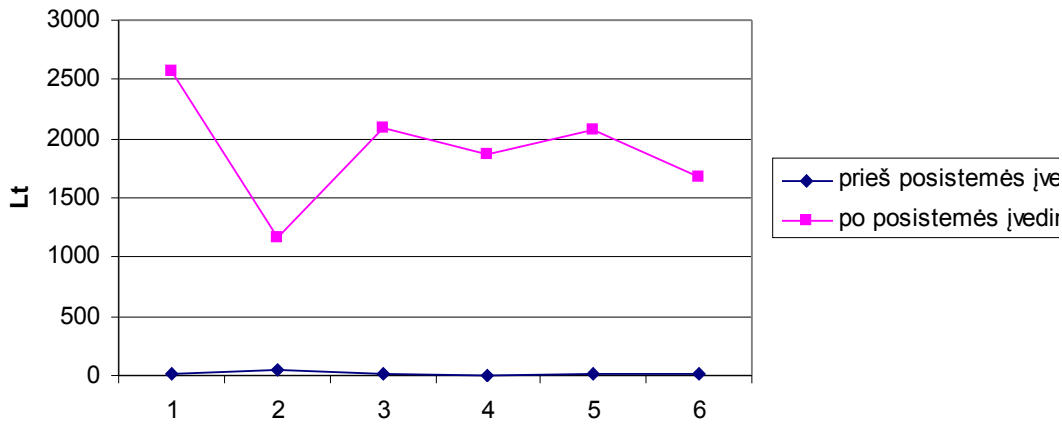
Stebint sistemą prieš papildomų paslaugų posistemės įvedimą ir po jo buvo surinkti duomenys apie pajamas iš žaidimo (litais). Buvo stebėti 6 mėnesiai prieš posistemės įvedimą ir 6 mėnesiai po įvedimo (7.1 pav.)

$p_s = \{20, 50, 20, 0, 10, 14\}$,

čia p_s – kas mėnesį gautos pajamos iš žaidimo prieš papildomų paslaugų posistemės įvedimą.

$p_n = \{2572, 1165, 2089 , 1869 , 2079, 1669\}$,

čia p_n – kas mėnesį gautos pajamos iš žaidimo po papildomų paslaugų posistemės įvedimą.



7.1 pav. Krepšinio žaidimo gautų pajamų palyginimas prieš papildomų paslaugų posistemės įvedimą ir po jo

Paskaičiuosime gautų pajamų vidurkį prieš posistemės įvedimą:

$$\overline{p}_s = \frac{1}{k_s}\sum_{j=1}^{k_s} p_{sj} = 19Lt,$$

bei gautų pajamų vidurkį po posistemės įvedimo:

$$\overline{p}_n = \frac{1}{k_n}\sum_{j=1}^{k_n} p_{nj} = 1907{,}17Lt$$

Gauname, kad $\overline{p}_s < \overline{p}_n$. Taigi galime atmesti hipotezę $H_{p0} : \overline{p}_s \geq \overline{p}_n$ ir teigti, kad papildomų paslaugų posistemės tikslas padidinti pajamas yra sėkmingai įgyvendintas.

Vartotojų įtraukimas į žaidimo vystymą.

Vartotojų įtraukimo į žaidimo vystymą įvertis bus skaičiuojamas pagal vartotojų, dalyvaujančių žaidimo tobulinimo procese, dalį. Vartotojai įsitraukę į žaidimo vystymą gali būti keleto tipų. Žemiau pateikiame pažymėjimus procentais išreikšto vartotojų kiekio pagal kiekvieną vartotojų tipą:

k – žaidimo kūrėjai, kurie įgyvendina idėjas;

m – žaidimo moderatoriai, kurie padeda prižiūrėti žaidimą bei įgyvendinti naujas idėjas;

p – papildomų paslaugų naudotojai, kurie yra įsigilinę į žaidimą, dalyvauja diskusijose dėl žaidimo tobulinimo, balsuoja dėl svarbiausių atnaujinimų;

f – forume besilankantys vartotojai, kurie diskutuoja apie dabartines žaidimo savybes bei pageidaujamas savybes;

v – vartotojai, nesilankantys forume, tačiau susipažįstantys su bendruomenės laikraštyje pateikiama informacija.

Vartotojų įtraukimo įvertis bus skaičiuojamas pagal formulę, įvertinančią kiekvieno vartotojų tipo svarbumą žaidimo tobulinimui:

$$i = 0{,}5 \cdot k + 0{,}2 \cdot m + 0{,}2 \cdot p + 0{,}08 \cdot f + 0{,}02 \cdot v$$

Kiekvieno vartotojų tipo svarbos koeficientai buvo nustatyti atsižvelgiant į ekspertų nuomonę. Žaidimo kūrėjų, tobulinančių sistemą, svarba yra labai didelė, nes be jų sistemą negalėtų tobulėti. Kitų vartotojų tipų svarbumas taip pat įvertintas koeficientais pagal jų indėlį į žaidimo tobulinimą.

Prieš papildomų paslaugų posistemės įvedimą, kiekvienam vartotojų tipui procentais išreikšta visų vartotojų dalis pateikta 7.1 lentelėje.

7.1 Lentelė

k_s	m_s	p_s	f_s	v_s
0,07%	0,17%	0%	6,67%	0%

$$i_s = 0{,}5 \cdot k_s + 0{,}2 \cdot m_s + 0{,}2 \cdot p_s + 0{,}08 \cdot f_s + 0{,}02 \cdot v_s = 0{,}601$$

čia i_s – vartotojų įtraukimo į žaidimo vystymą įvertis prieš papildomų paslaugų posistemės įvedimą.

7.2 lentelėje pateikta, kiek procentų žaidimo vartotojų sudaro kiekvienas vartotojų tipas po papildomų paslaugų posistemės įvedimo.

7.2 Lentelė

k_n	m_n	p_n	f_n	v_n
0,07%	0,17%	2,83 %	6,67%	26,67%

$$i_n = 0{,}5 \cdot k_n + 0{,}2 \cdot m_n + 0{,}2 \cdot p_n + 0{,}08 \cdot f_n + 0{,}02 \cdot v_n = 1{,}702$$

čia i_n – vartotojų įtraukimo į žaidimo vystymą įvertis po papildomų paslaugų posistemės įvedimo.

Gauname, kad $i_s < i_n$. Taigi galime atmesti hipotezę $H_{i0} : i_s \geq i_n$ ir teigti, kad įvedus papildomų paslaugų posistemę vartotojų įtraukimas į žaidimo tobulinimą padidėjo.

Sistemos patikimumas.

Sistemos patikimumą (k) vertinsime pagal formulę:

$$k = 1 - \left(0{,}5 \cdot \frac{s}{d} + 0{,}1 \cdot \frac{l}{d} + 0{,}1 \cdot \frac{r}{d} + 0{,}3 \cdot \frac{v}{b} \right),$$

čia:

s – kiek kartų per sezoną sistemos veikimą teko sustabdyti daugiau nei vienai dienai;

l – kiek kartų per sezoną žaidime vėlavo naujos dienos pradžia;

r – kritinių klaidų kiekis per sezoną;

v – sezono vidutinis kiekis vartotojų, kuriems kyla nepatogumų dėl kritinių klaidų;

b – bendras žaidimo vartotojų skaičius;

$d = 63$ – sezono ilgumas dienomis (nekintamas dydis).

Kiekvieno elemento svarbumas buvo įvertintas ekspertų ir nustatyti konkretūs svarbos koeficientai. Sistemos sustabdymas ilgiau nei keletui dienų turi didelę žalą žaidimui, todėl jo svarbos koeficientas yra didžiausias. Atitinkamai pagal svarbumą nustatyti ir kiti koeficientai.

Pateikiame statistinius stebėjimų rezultatus iki klaidų fiksavimo posistemės įvedimo ir po jo. Stebėjome 4 sezonus iki posistemės įvedimo ir 6 po įvedimo. Kadangi žaidimo kūrimo pradžioje dar tik formavosi tiek vartotojų bendruomenė, tiek pagrindinės žaidimo funkcijos, ankstesnius duomenis vertinti būtų netikslu.

Sistemos sustabdymų skaičius per sezoną (s_s – iki posistemės įvedimo, s_n – po įvedimo):

s_s = *{1, 0, 0, 1}*, $\bar{s}_s = 0{,}5$ *;*

s_n = *{0, 0, 0, 0, 0, 0}*, $\bar{s}_n = 0$.

Kartų kiekis, kai vėlavo naujos dienos pradžia (v_s – iki posistemės įvedimo, v_n – po įvedimo)

l_s = *{6, 5, 3, 4}*, $\bar{v}_s = 4{,}5$ *;*

l_n = *{2, 1, 1, 3, 0, 2}*, $\bar{v}_n = 1{,}5$.

Kritinių klaidų kiekis (r_s – iki posistemės įvedimo, r_n – po įvedimo):

r_s = *{22, 21, 18, 20}*, $\bar{r}_s = 20{,}25$ *;*

r_n = *{10, 14, 12, 8, 14, 13}*, $\bar{r}_n = 11{,}83$.

Vidutinis kiekis vartotojų, kuriems kyla tam tikrų problemų dėl kritinių klaidų (v_s – iki posistemės įvedimo, v_n – po įvedimo):

v_s = *{280, 198, 205, 150}*, $\bar{v}_s = 208{,}25$ *;*

v_n = *{110, 65, 85, 100, 135, 65}*, $\bar{v}_n = 93{,}33$.

Vidutinis vartotojų skaičius prieš posistemės įvedimą ir po jos:

b_s = *3000;*

b_n = *3700;*

Pagal turimus duomenis nustatome sistemos patikimumo įverčius:

$$k_s = 1 - \left(0{,}5 \cdot \frac{\bar{s}_s}{d} + 0{,}1 \cdot \frac{\bar{l}_s}{d} + 0{,}1 \cdot \frac{\bar{r}_s}{d} + 0{,}3 \cdot \frac{\bar{v}_s}{b_s} \right) = 0{,}93$$

čia k_s – sistemos patikimumo įvertis prieš klaidų fiksavimo posistemės įvedimą.

$$k_n = 1 - \left(0{,}5 \cdot \frac{\bar{s}_n}{d} + 0{,}1 \cdot \frac{\bar{l}_n}{d} + 0{,}1 \cdot \frac{\bar{r}_n}{d} + 0{,}3 \cdot \frac{\bar{v}_n}{b_n} \right) = 0{,}97$$

čia k_n – sistemos patikimumo įvertis po klaidų fiksavimo posistemės įvedimo.

Gauname, kad $k_s < k_n$. Taigi galime atmesti hipotezę $H_{k0} : k_s \geq k_n$ ir teigti, kad įvedus klaidų stebėjimo posistemę sistemos patikimumas padidėjo.

8. Išvados

1. Mokslinės literatūros ir virtualaus krepšinio žaidimo sistemos analizės metu buvo nustatyta, kad šiai sistemai būtinas nuolatinis tobulinimas ir kad tobulinimo procesą leistų pagerinti naujų elementų įvedimas: rungtynių imitatoriaus modelis, papildomų paslaugų teikimo posistemė bei klaidų fiksavimo posistemė.
2. Sudaryto rungtynių imitatoriaus modelio taikymas žaidimo taisyklių pakeitimų poveikiui analizuoti parodė, kad sukurtas modelis leidžia efektyviai reaguoti į po pakeitimų atsiradusias klaidas ir vartotojų reakciją bei atsekti klaidų ar neigiamo poveikio priežastis, o tai palengvina žaidimo testavimą ir taisyklių derinimą.
3. Realizuotų papildomų priemonių naudingumas buvo įvertintas atliekant trijų dalių eksperimentą realioje žaidimo veikimo aplinkoje. Eksperimentas truko 10 mėnesių ir jo rezultatai parodė, kad sukurtos priemonės artimiausiu metu leidžia vystyti žaidimą iki bus nustatytas naujų patobulinimų poreikis.
4. Remiantis pirmoje eksperimento dalyje atliktais stebėjimais, galima teigti, kad papildomos paslaugos padeda gauti iš žaidimo daugiau pajamų, o tai leidžia skirti daugiau išteklių tobulinti žaidimą toliau.
5. Antroje eksperimento dalyje atlikta vartotojų įtraukimo į žaidimo kūrimą analizė parodė, kad sistemos patobulinimai leidžia įsijungti į žaidimo vystymą didesniam skaičiui dalyvių, o tai taip pat veda prie žaidimo vystymo pagerėjimo.
6. Trečioje eksperimento dalyje buvo nustatyta, kad klaidų fiksavimo posistemė padidina žaidimo patikimumą, o tai yra teigiamas faktorius, padedantis pritraukti ir išlaikyti žaidimo dalyvius.
7. Šio darbo patirtis gali būti naudinga panašių žaidimų ir interneto svetainių kūrėjams, kurie siekia savo programinių produktų patrauklumo, tobulėjimo ir ilgesnio gyvavimo elektroninėje erdvėje.

9. Literatūra

[1] Hattrick istorija. 2008 [žiūrėta 2010-10-10] Prieiga per Internetą: http://www.hattrick.org/Help/History.aspx
[2] Games. [žiūrėta 2010-10-10] Prieiga per Internetą: http://www.ge-eu.com/games.html
[3] What is SaaS. 2009 [žiūrėta 2010-10-10] Prieiga per Internetą: http://www.whatissaas.net/
[4] An Overview of the History of the Software Industry. 2007 [žiūrėta 2009-10-10] Prieiga per Internetą: http://www.softwarehistory.org
[5] Hoch F., Kerr M. "Software as a Service: Strategic Backgrounder" Software & Information Industry Association, 2001, p.1-18
[6] Espadas J., Concha D., Molina A. "Application Development over Software-as-a-Service platforms", The Third International Conference of Software Engineering Advances, 2008, p. 98
[7] Menken I., Blokdijk G. "Cloud Computing Certification Kit Specialist Software as a Service & Web Application", 2009, p. 14-16
[8] Miete N. "Configurability in SaaS (Software as a Service) applications", ISEC'09, February 23–26, 2009, Pune, India, p. 19-26
[9] Choudhary V. "Software as a Service: Implications for Investments in Software Development", Proceedings of the 40th Hawaii International Conference on System Sciences, 2007, p. 209a
[10] Candan S., Li W., Phan T., Zhou M. "Frontiers in Information and Software as Services", Proceedings of the 2009 IEEE International Conference on Data Engineering, 2009, p. 1761-1768
[11] Jacobs D. "Enterprise Software as Service", Queue, July/August 2005, p. 37-42
[12] Web Testing: Complete guide on testing web applications. 2007 [žiūrėta 2011-03-07] Prieiga per Internetą: http://www.softwaretestinghelp.com/web-application-testing
[13] Lian Y., Wei Z., Xiaofeng D., Changzhu K., Wenbo Z., Qianxiang W., Jun Z. "Towards Call for Testing: An Application to User Acceptance Testing of Web Applications", 33rd Annual IEEE International Computer Software and Applications Conference, 2009, p. 166-171
[14] Artzi S., Kiežun A., Dolby J., Tip F., Dig D., Paradkar A., Ernst M.D. "Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Checking", IEEE Transactions on Software Engineering, July/August 2010, vol.36, no.4.
[15] Qian Z., Miao H., Zeng H. "A Practical Web Testing Model for Web Application Testing", Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, 2008, p. 434-441
[16] Licata D.R., Krishnamurthi S. "Verifying InteractiveWeb Programs", Proceedings of the 19th International Conference on Automated Software Engineering, 2004.
[17] Chillarege R. „Software Testing Best Practices", 1999, p. 1-11
[18] Millington I. "Artificial Intelligence for Games", USA, 2006.
[19] Nummenmaa T., Berki E. "Exploring Games as Formal Models", Fourth South-East European Workshop on Formal Methods, 2009, p.60-65