Kauno technologijos universitetas

Informatikos fakultetas

Programinės įrangos testavimo metodika ir jos taikymas veiklos procesų valdymo sistemoje

Baigiamasis magistro studijų projektas

Brigita Baršauskaitė

Projekto autorė

doc. dr. Tomas Skersys

Vadovas

Kaunas, 2020

Kauno technologijos universitetas

Informatikos fakultetas

Programinės įrangos testavimo metodika ir jos taikymas veiklos procesų valdymo sistemoje

Baigiamasis magistro studijų projektas

Informacinių sistemų inžinerija (6211BX009)

Brigita Baršauskaitė

Projekto autorė

doc. dr. Tomas Skersys

Vadovas

prof. dr. Audrius Lopata

Recenzentas

Kaunas, 2020

Kauno technologijos universitetas

Informatikos fakultetas

Brigita Baršauskaitė

Programinės įrangos testavimo metodika ir jos taikymas veiklos procesų valdymo sistemoje

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Brigitos Baršauskaitės, baigiamasis projektas tema „Programinės įrangos testavimo metodika ir jos taikymas veiklos procesų valdymo sistemoje“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

Brigita Baršauskaitė — (vardą ir pavardę įrašyti ranka)

(parašas)

Baršauskaitė, Brigita. Programinės įrangos testavimo metodika ir jos taikymas veiklos procesų valdymo sistemoje. Magistro baigiamasis projektas / vadovas doc. dr. Tomas Skersys; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Informatikos inžinerija, technologijos mokslai.

Reikšminiai žodžiai: programinės įrangos testavimas, programinės įrangos testavimo procesas, programinės įrangos testavimo metodika, veiklos procesų valdymo sistemos, BPMN.

Kaunas, 2020. 93 p.

Santrauka

Programinės įrangos testavimas užima itin svarbią vietą visame programinės įrangos gyvavimo cikle – galutinio produkto sėkmė tiesiogiai priklauso nuo testavimo kokybės. Dėl šios priežasties itin svarbu, kad įmonės, kuriančios programinę įrangą, skirtų pakankamai dėmesio testavimo procesui. Siekiant užtikrinti testavimo kokybę vien kvalifikuotų testavimo specialistų nepakanka. Norint pasiekti aukščiausią kokybę pats testavimo procesas turi būti aiškiai apibrėžtas, realiu laiku stebimas, bei nuolatos optimizuojamas. Vieningo testavimo proceso apibrėžimas ir struktūrizavimas dažnai tampa iššūkiu įmonėms, kurios dirba su įvairių tipų projektais, vykdomais naudojant skirtingas programinės įrangos kūrimo metodikas. Kai testuotojų skyriuje nėra apibrėžto vieningo proceso, dažnai atsiranda nesusikalbėjimas tarp testuotojų grupės narių, grupės vadovui itin sudėtinga išmokyti naujus darbuotojus, stebėti testuotojų darbus, o testuotojams sunku pradėti darbus prie naujų projektų, suprasti savo atsakomybes, vietą įmonėje, bei reikiamas atlikti užduotis. Be to, dažnai testuotojai net neturėdami pakankamai kompetencijos turi kiekvienam projektui individualiai atlikti projekto aplinkybių analizę ir nuspręsti, kokius testavimo metodus naudoti. Dėl įvardintų priežasčių tiesiogiai nukenčia testavimo proceso kokybė. Įvertinus visus faktus ir aplinkybes, galima teigti, jog testavimo proceso struktūrizavimo ir apibendrintos metodikos sudarymo galimybių tyrimas yra aktualus, bei reikalingas.

Šio darbo tikslas yra prisidėti prie testavimo vykdymo ir valdymo procesų gerinimo. Tikslui pasiekti darbo metu sukurta programinės įrangos testavimo metodika, kuriai vykdyti buvo realizuota eksperimentinė sistema veiklos procesų vykdymu grindžiamoje platformoje. Sukurta metodika ir demonstracinė metodikos sistema apibrėžia apibendrintą testavimo vykdymo procesą, kurio vykdymo metu pateikiami rekomenduojami testavimo metodai. Rekomenduojami testavimo metodai atrenkami atsižvelgiant į testuotojų grupės vadovo nurodytas projekto charakteristikas. Siekiant užtikrinti vieningą testavimo procesą buvo sudaryti vykdomieji veiklos procesų modeliai ir įkelti į „Camunda BPM“ platformą. Pasirinkta realizacijos technologija suteikia galimybę realiu laiku stebėti vykdomus testavimo procesus, nesudėtingai keisti procesų modelius, užduotys automatiškai inicijuojamos pagal numatytą procesą, o tai sumažina užduočių inicijavimo uždelsimo tikimybę. Taip pat sukurti testavimo proceso modeliai suteikia galimybę sutrumpinti naujų testuotojų mokymo laiką. Sukurtos testavimo metodikos ir demonstracinės sistemos pritaikymo IT įmonėje galimybės buvo eksperimentiškai ištirtos, identifikuoti sukurtos metodikos privalumai ir trūkumai, sudarytas programinės įrangos testavimo metodikos pritaikymo rekomendacijų sąrašas, kuris tikėtina palengvins testavimo metodikos taikymo IT įmonėse procesą.

Baršauskaitė, Brigita. Testing Methodology and Its Implementation in a Business Process Management System. Master's Final Degree Project / supervisor assoc. prof. Tomas Skersys; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Informatics Engineering, Technology Science.

Keywords: Software Testing, Software Testing Process, Software Testing Methodology, Business Process Management Systems, BPMN.

Kaunas, 2020. 93 p.

Summary

Software testing plays a vital role throughout the software lifecycle – the success of the product depends directly on the quality of the testing. This is reason, why it is critical that software developing companies pay enough attention to the testing process. Qualified testing professionals alone are not enough to ensure the quality of testing. In order to achieve the highest quality, the testing process itself must be clearly defined, monitored in real time, and continuously optimized. Defining and structuring of a uniform testing process often become a challenge for companies working with different types of projects that are implemented by using different methodologies of software development. In cases when there is no uniform process defined in the tester department, there is often a lack of communication between members of the tester team, the team leader has to go to great lengths to train new employees and monitor works of testers, while testers find it difficult to start working on new projects, understand their responsibilities, as well as place in the company and tasks that must be performed Furthermore, it is often the case that testers, even without sufficient skills, have to carry out the analysis of the project circumstances on their own for each project and then decide which testing methods to use. The said reasons have a direct negative effect on the quality of the testing process. Considering all the facts and circumstances, the feasibility study of the testing process structuring, and the development of a summarized methodology is relevant and necessary.

The aim of this thesis is to contribute to the improvement of testing performance and management processes. A software testing methodology was developed during the writing of the thesis in order to achieve the thesis aim, and it was implemented by utilizing an experimental system in a platform based on the execution of activity processes. The developed methodology and the demonstration system of methodology define a summarized process of the testing execution, during the execution of which the recommended testing methods are presented. The recommended test methods are selected based on the project characteristics specified by the leader of the tester team. In order to ensure a uniform testing process, executive activity process models were developed and uploaded to the Camunda BPM platform. The technology selected for the implementation provides the ability to monitor the ongoing testing processes in real time, as well as easily alter process models, furthermore, tasks are automatically initiated according to the intended process, and this reduces the possibility of task initiation delay. The developed models of the testing process also provide an opportunity to reduce the training time of new testers. The possibilities of application of the developed testing methodology and the demonstration system in the IT company were researched experimentally; the advantages and disadvantages of the developed methodology were identified; and a list of recommendations for software testing methodology application was compiled, which is likely to facilitate the application of the testing methodology in IT companies.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Agile – manifestas, kuris apibrėžia pagrindinius svarbiausius aspektus programinės įrangos kūrimo procese. Remiantis šiuo manifestu kuriami programinės įrangos kūrimo metodai.

BPM (angl. *Business Process Model*) – veiklos procesų modeliavimas.

BPMN (angl. *Business Process Model and Notation*) – veiklos procesų modeliavimas ir žymėjimas.

DMN (angl. *Decision Model and Notation*) – sprendimo priėmimo modeliavimas ir žymėjimas.

Funkcinis reikalavimas (FR) – sistemos funkcionalumui keliami reikalavimai, kurių aprašymas atspindi, ką ir kaip sistema turi daryti.

IS – informacinė sistema.

ISTQB (angl. *International Software Testing Qualifications Board*) – tarptautinė sistemų testavimo kvalifikacijostarnyba.

IEEE (angl. *Institute of Electrical and Electronics Engineers Standards Association*) – elektros ir elektronikos inžinierių standartų asociacijos institutas.

Nefunkcinis reikalavimas (NFR) – įvairūs apribojimai keliami sistemos funkcionalumui, kurie gali būti susiję su sistemos greitaveika, naudojamais resursais ir panašiai.

PĮ – programinė įranga.

PROD – produkcinės aplinkos (aplinka, kurioje dirba klientai) santrumpa.

RUP (angl. *Rational Unified Process*) – kartotinumu grindžiama programinės įrangos kūrimo metodika.

REST API – sąsaja, kuri apibrėžia galimas funkcijas, pasiekiamas naudojant užklausas ir gautus serverio atsakymus iš per HTTP protokolą [2].

Scrum – programinės įrangos kūrimo metodas, kuris remiasi *Agile* manifestu.

SQA (angl. *Software Quality Assurance*) – sistemos kokybės užtikrinimo procesas.

TC (angl. *Test Case*) – testavimo scenarijus.

TDD (angl. *Test-Driven Development*) – testavimu grindžiamas programinės įrangos kūrimas.

UML (angl. *Unified Modelling Language*) – unifikuota modeliavimo kalba.

Įvadas

Baigiamasis magistro darbas yra skirtas studijų programai „Informacinių sistemų inžinerija“. Šio dokumento įvade identifikuojamos egzistuojančios problemos programinės įrangos testavimo procese, iškeliamas darbo tikslas ir įvardijami konkretūs uždaviniai, kuriuos įvykdžius bus pasiektas užsibrėžtas tikslas. Taip pat šiame skyriuje aptariama dokumento struktūra, įvardijama, kokią reikšmę turi atlikto darbo rezultatai.

Darbo problematika ir aktualumas

Tiriamąjį darbą nuspręsta atlikti, nes buvo pastebėta, kad programinės įrangos testavimo procesas dažnai yra neefektyvus ir nestruktūrizuotas. Testuojant programinę įrangą dažnai įmonės neturi aiškiai apibrėžto proceso, kurį būtų galima stebėti realiu laiku. Taip pat kiekvienam projektui individualiai yra kuriamos ir inicijuojamos užduotys, atrenkami testavimo metodai rankiniu būdu. Be to, naudojami universalūs įrankiai, kurie nepadengia viso veiklos proceso ir dėl realizacinių technologijų įrankių palaikymas reikalauja daug resursų.

Darbo tikslas ir uždaviniai

Tyrimo tikslas – pagerinti programinės įrangos testavimo vykdymo ir valdymo procesus. Darbui iškelti uždaviniai:

1. atlikti programinės įrangos testavimo proceso analizę;
2. išanalizuoti esamus programinės įrangos testavimo metodus;
3. sudaryti naują metodiką, skirtą programinės įrangos testavimo vykdymui ir stebėjimui;
4. suprojektuoti, realizuoti ir ištestuoti eksperimentinę sistemą, skirtą sukurtos metodikos demonstravimui ir tyrimui;
5. eksperimentiškai ištirti sukurtą metodiką;
6. apibendrinti tyrimo rezultatus.

Atlikus visus apibrėžtus uždavinius bus pasiektas darbo tikslas.

Darbo rezultatai ir jų svarba

Tiriamojo darbo metu bus išanalizuoti egzistuojantys programinės įrangos testavimo metodai, identifikuotos taikymo galimybės, bei probleminės sritys. Analizės metu surinkta informacija ir įžvalgos bus naudojamos kaip pagrindas priimant sprendimus metodikos formavimo etape. Galutinis ir pagrindinis darbo rezultatas – sukurta ir detaliai aprašyta testavimo metodika, bei eksperimentinė testavimo metodikos taikymo sistema, sprendžianti neefektyvaus ir nestruktūrizuoto testavimo proceso problemą.

Darbo struktūra

Dokumentas sudarytas iš 12 skyrių, 5-iuose iš jų saugoma bendra darbo informacija: naudojami terminai ir santrumpos, literatūra, dokumentą papildantys duomenys (priedai), nuorodos į naudojamus paveikslėlius, lenteles. Įvado skyrius skirtas supažindinimui su darbo specifika, tikslais, bei uždaviniais. Likę skyriai priskiriami dėstomajai darbo daliai: probleminės srities analizės skyrius, sprendimo projekto skyrius, sprendimo realizacijos projekto skyrius, sprendimo testavimo ir eksperimento skyriai.

1. Probleminės srities analizė

Šiame skyriuje apibrėžiamas probleminės srities analizės tikslas, identifikuojamas tyrimo objektas, sritis, bei tyrimo problema. Taip pat atliekama detali tyrimo objekto naudotojų ir esamų problemos sprendimų analizė. Be to, tyrimo objekto analizės skyriuje pateikiamas siekiamo sprendimo apibrėžimas ir galiausiai suformuluojamos atliktos tyrimo objekto analizės išvados.

1.1. Analizės tikslas

Tyrimo objekto analizė atliekama siekiant identifikuoti pagrindines veiklas ir vaikinius procesus, kurie yra vykdomi programinės įrangos testavimo vykdymo ir valdymo procesų metu, bei suprasti ir aprašyti analizuojamų procesų kontekstą. Taip pat siekiama išanalizuoti egzistuojančius PĮ testavimo metodus, identifikuoti jų silpnąsias ir stipriąsias vietas. Be to, būtina išanalizuoti būsimus kuriamos metodikos naudotojus, identifikuoti jų poreikius ir savybes. Žinoma, taip pat svarbu išanalizuoti galimas realizacines technologijas, kurios padėtų maksimaliai vizualizuoti sukurtos metodikos naudą ir gerąsias savybes. Remiantis analizės etape surinkta informacija bus priimami sprendimai tolimesniuose metodikos projektavimo ir kūrimo etapuose.

1.2. Tyrimo objektas, sritis ir problema

Tiriamojo darbo esmę ir apimtį atspindi pagrindiniai elementai, kurie kryptingai analizuojami ir detalizuojami viso tyrimo metu. Esminiai tyrimo elementai yra šie: tyrimo problema, tyrimo objektas, tyrimo sritis. Visi šie elementai apsibrėžiami prieš pradedant tiriamąjį darbą, tam, kad būtų aiški tyrimo kryptis ir ribos, kurių būtų laikomasi viso tiriamojo darbo metu.

Tiriamąjį darbą nuspręsta atlikti, nes buvo pastebėta, kad programinės įrangos testavimo procesas dažnai yra neefektyvus ir nestruktūrizuotas. Testuojant programinę įrangą dažnai bandoma pritaikyti egzistuojančius metodus, kuriuose akcentuojamos konkrečios testavimo veiklos ir elementai, kuriuos būtina ištestuoti aprašomo metodo rėmuose, bet mažai dėmesio skiriama bendram testavimo procesui, jo vykdymui ir stebėjimui. Taip pat egzistuojantys metodai dažnai aprašomi nepateikiant realaus metodo pritaikymo konteksto, nėra bendros informacijos, kaip derinti įvairius metodus tarpusavyje, kad testavimo procesas būtų maksimaliai efektyvus. Dėl šių priežasčių dažnai testavimo procesas yra vykdomas nesilaikant bendros tvarkos testavimo skyriuje, testuotojų darbas neefektyvus, testuotojai įtraukiami į programinės įrangos kūrimo procesą vėlyvoje PĮ kūrimo fazėje.

Vienas iš svarbiausių darbų, siekiant pasiruošti tiriamajam darbui – identifikuoti tyrimo objektą, kuriame egzistuoja įvardyta problema. Tai ypač svarbu, siekiant, kad tyrimas būtų atliktas kryptingai, neapimant detalių, kurios nėra susijusios su tyrimo objektu. Šio tyrimo objektas yra programinės įrangos testavimo procesas, kurio detali analizė pateikiama 1.3 skyriuje.

Šio darbo tyrimo sritis – programinės įrangos testavimo metodikos, esamos sistemos, bei galimos realizacinės technologijos. Tyrimo apimtyje planuojama išanalizuoti egzistuojančius testavimo metodus (1.4), bei galimus įrankius, skirtus vykdyti ir valdyti programinės įrangos testavimo procesą. Detali įrankių analizė pateikiama 1.6 skyriuje. Be to, siekiama išanalizuoti technologijas (1.7), kuriomis būtų galima realizuoti eksperimentinę sistemą, kuri veiktų pagal sumodeliuotą testavimo procesą, grindžiamą tiriamajame darbe sukurta metodika.

1.3. Programinės įrangos testavimo proceso analizė

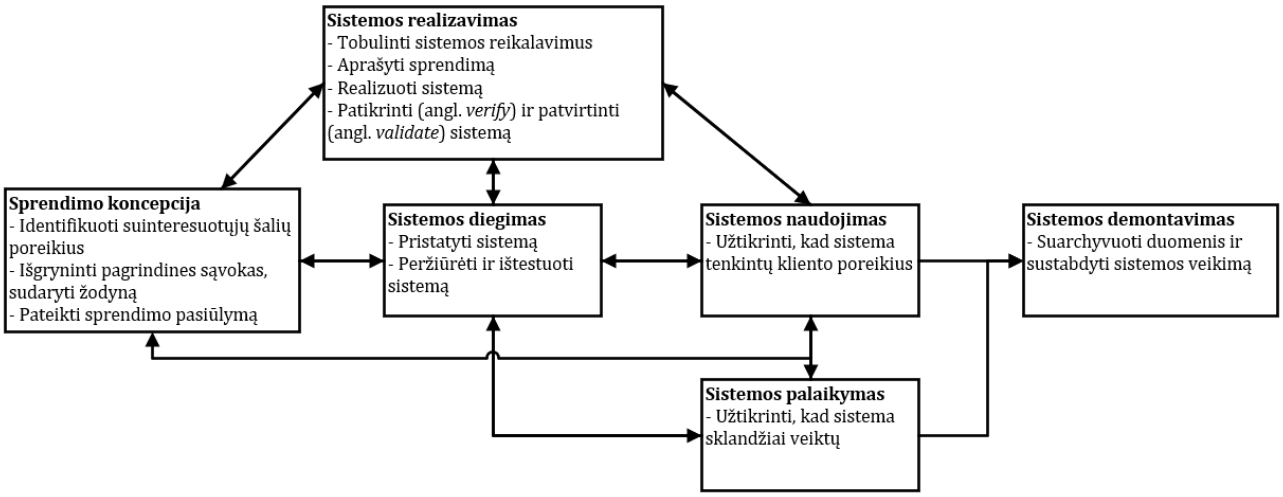
Tyrimo objektas – programinės įrangos testavimo procesas – tiesiogiai priklauso nuo pasirinktos PĮ kūrimo metodikos, todėl tyrimo objekto analizė pradedama nuo programinės įrangos kūrimo proceso analizės, apibrėžiama testavimo vykdymo proceso samprata ir pagrindinės sąvokos. Toliau identifikuojama testavimo proceso vieta bendrame PĮ gyvavimo cikle. Taip pat pateikiamos veiklos procesais grindžiamų sistemų pritaikymo galimybės testavimo procese.

1.3.1. Testavimo procesas programinės įrangos gyvavimo cikle

Prieš pradedant analizuoti testavimo procesą ir jo vaidmenį programinės įrangos gyvavimo cikle, būtina apibrėžti pagrindines sąvokas:

- **Programinė įranga** (angl. *software*) – visuma, kuri apima kompiuterinę programą, programinį kodą, procedūras, dokumentaciją ir visus sistemos veikimui reikalingus duomenis [1].
- **Programinės įrangos gyvavimo ciklas** (angl. *software life cycle*) – sistemingas procesas, apimantis visus PĮ kūrimo, bei palaikymo etapus (1 pav.). Pagal ISO/IEC/IEEE 12207 standartą programinės įrangos gyvavimo ciklas apima šias procesų grupes [1]:
    - **Sutarties/susitarimo procesai** (angl. *agreement processes*) **–** apima sistemos įsigijimo, bei tiekimo procesus.
    - **Organizacinius projektus įgalinantys procesai** (angl. *organizational project-enabling processes*) – apima gyvavimo ciklo modelio, infrastruktūros, portfelio, žmogiškųjų išteklių, kokybės, bei žinių valdymo procesus.
    - **Techniniai valdymo procesai** (angl. *technical management processes*) – apima projekto planavimo, vertinimo, kontrolės, kokybės užtikrinimo, bei sprendimų priėmimo, rizikos, konfigūracijos, informacijos valdymo procesus.
    - **Techniniai procesai** (angl. *technical processes*) – apima veiklos analizės, suinteresuotų šalių poreikių išgryninimo, reikalavimų aprašymo, sistemos analizės, architektūros sudarymo, sistemos projektavimo, PĮ realizavimo, integravimo, testavimo, tvirtinimo, operacijų stebėjimo, palaikymo, bei sistemos demontavimo procesus.

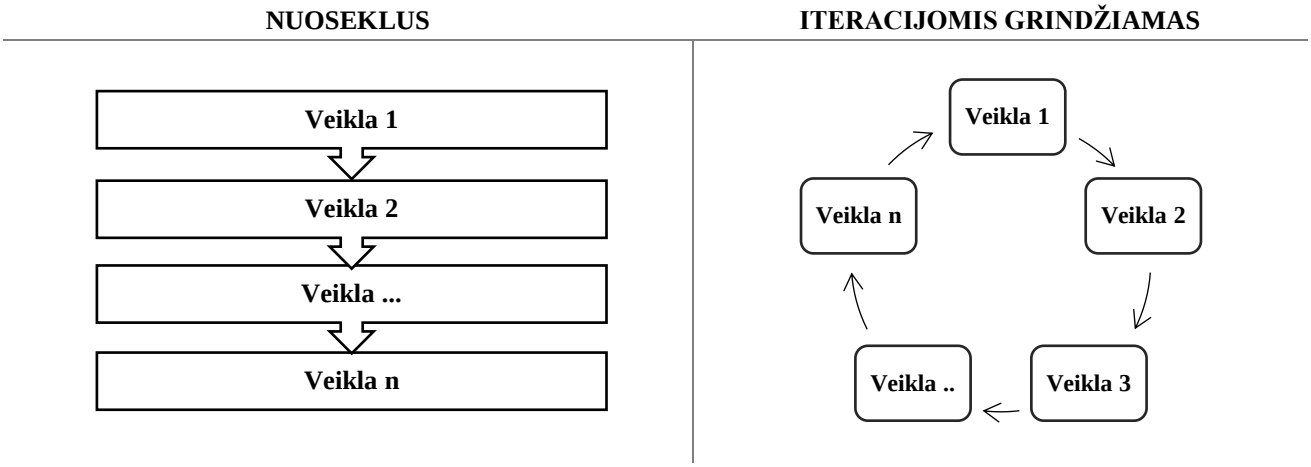
Remiantis ISO/IEC TS 24748-1 standartu [2] programinės įrangos gyvavimo ciklas yra sudarytas iš šešių etapų (1 pav.): koncepcija (angl. *concept*), realizavimas (angl. *development*), produkcinė aplinka (angl. *production*), sistemos naudojimas (angl. *utilization*), palaikymas (angl. *support*) ir sistemos išjungimas (angl. *retirement*).



1 pav. Programinės įrangos gyvavimo ciklo modelis, atspindintis galimas progresijas ir etapų tikslus [2]

Kiekvienas etapas gyvavimo cikle turi savo tikslus, bei atspindi programinės įrangos būsenos kitimą, bendrą programos vystymąsi. Testavimo procesas yra viena iš PĮ gyvavimo ciklo (1 pav.**)** sudedamųjų dalių, todėl testavimo veiklų vykdymo eiliškumas ir svarba tiesiogiai priklauso nuo pasirinktos PĮ kūrimo metodikos, kuri apibrėžia PĮ gyvavimo ciklo etapų vykdymo eiliškumą, bei veiklas, kurios yra vykdomos kiekvieno etapo metu. Vieno unikalaus metodo, kuris užtikrintų 100% sėkmingą projekto įvykdymą, nėra. Kuriant programinę įrangą, metodas ar jų kombinacija dažniausiai yra pasirenkami atsižvelgiant į pačio projekto, kliento, dalykinės srities specifiką ir kitas projekto savybes. Šio tyrimo apimtyje detaliai analizuoti programinės įrangos kūrimo proceso nebūtina, nes tyrimas orientuotas tik į vieną PĮ kūrimo dedamąją – testavimo procesą. Dėl šios priežasties pakanka apibrėžti tik pagrindinius programinės įrangos kūrimo proceso principus.

Sistemų realizavimo metodai pagal veiklų vykdymo eiliškumą yra skirstomi į nuoseklius ir iteracijomis grindžiamus metodus [3] (2 pav.).



2 pav. Sistemų kūrimo metodų tipai

Nuosekliuose metoduose (pvz., „Krioklys“, „V modelis“) veiklos yra vykdomos nuosekliai viena po kitos, negalima pereiti iš vykdomos veiklos į prieš tai buvusią. Nuosekliuose metoduose kiekvienas etapas yra vykdomas pilna apimtimi, t. y. jeigu einamu laiko momentu vykdomas analizės etapas, tuomet analizuojami visi programinei įrangai keliami reikalavimai. Perėjus į sekantį etapą grįžti į

ankstesnį – nėra galimybės. Dėl šios priežasties nuosekliu etapų vykdymu grindžiami metodai nėra lankstūs ir sunkiai suderinami su kintančiais sistemos reikalavimais [4].

Metoduose, kurie yra grindžiami iteracijomis, galutinis produktas kuriamas per kelias iteracijas. Vienos iteracijos metu sukuriama galutinio produkto dalis, kuri turi savo reikalavimus, reikalavimų specifikaciją ir yra realizuota, bei ištestuota. Įvykdžius keletą iteracijų sukuriamas galutinis, pilnai išbaigtas produktas. Iteracijomis grindžiami metodai (pvz., „RUP“, „Scrum“) yra modernesni, pasižymi greitesniu ir kokybiškesniu sistemos kūrimo procesu [5].

Pagrindiniai tiriamajame darbe analizuojami PĮ kūrimo metodai: „V“ modelis, „RUP“, „Scrum“.

Testavimo procesas nuosekliuose programinės įrangos kūrimo metoduose

Pagrindiniai nuoseklūs programinės įrangos metodai: „Krioklys“ (angl. *Waterfall*) ir „V“ modelis. Šie metodai yra patys pirmieji ir vis dar sėkmingai naudojami, kai sistemai keliami reikalavimai yra visiškai aiškūs, stabilūs ir kardinaliai nesikeičia realizuojant programinę įrangą [4]. „Krioklio“ metodas yra pats pirmasis ir jis turi keletą atmainų. Viena iš atmainų yra „V“ modelis, kuris puikiai tinka atvaizduoti, kaip PĮ kūrimo veiklos gali būti susiejamos su testavimo veiklomis. Dėl šios priežasties tiriamajame darbe nuspręsta analizuoti būtent „V“ modelį.

„V“ modelis vizualiai atspindi, kaip testavimo veiklos gali būti tiesiogiai susiejamos su kitomis PĮ kūrimo veiklomis (3 pav.).

3 pav. „V“ modeliu grindžiamas programinės įrangos kūrimas [6]

„V“ modelyje visos veiklos yra vykdomos nuosekliai. Dėl šios priežasties „V“ modelis ir yra laikomas „Krioklio“ modelio atmaina.

Naudotojų reikalavimų surinkimas ir dokumentavimas

Remianti pradinės naudotojų reikalavimų išgavimo fazės metu surinktais rezultatais, turi būti sukurtas detalios analizės dokumentas. Šiame dokumente aprašomos visos pagrindinės veiklos, kurios atsispindi ir techninėje sistemos specifikacijoje. Be to, detalios analizės dokumentas suformuoja pagrindą priėmimo testavimo struktūrai ir reikalavimams. Dėl šios priežasties sistemų analitikas privalo aprašyti visus surinktus reikalavimus taip detaliai, kad testuotojai galėtų ruošti testavimo scenarijus.

Pagal knygą „Software Testing Practice: Test Management“ [7], „V“ modelyje naudotojų reikalavimų surinkimo fazė tiesiogiai siejasi su **priėmimo testavimu** už kurį dažnai yra atsakingas užsakovas. Šį testavimą vykdo klientas, kai sistemos kūrėjų komanda baigia vidinį sistemos testavimą ir perduoda klientui jau ištestuotą sistemos versiją. Pagrindinis šio testavimo tikslas – nustatyti, ar sistema tenkina kliento poreikius, ar atitinka apibrėžtus funkcinius ir nefunkcinius reikalavimus. Šiame etape jau nėra koncentruojamasi į klaidų paiešką.

Funkcinių reikalavimų surinkimas ir dokumentavimas

Šiame etape surinkti reikalavimai transformuojami į funkcijas ir komponentų tarpusavio sąveikas sistemoje. Paruošti dokumentai naudingi sistemos testuotojams ruošiant funkcinių testų testavimo scenarijus.

Pagal knygą „Software Testing Practice: Test Management“ [7], „V“ modelyje funkcinių reikalavimų surinkimo fazė tiesiogiai siejasi su **funkciniu sistemos testavimu**, kurio metu koncentruojamasi ties visu sistemos funkcionalumu, kuris apibrėžtas sistemos specifikacijoje. Šis testavimas itin svarbus, todėl testavimą būtina atlikti aplinkoje, kurioje yra duomenys, atitinkantys gamybinės aplinkos duomenis. Tokiu atveju yra sumažinama rizika, kad po diegimo į produkcinę aplinką iškils nenumatytų incidentų. Testavimo apimtyje svarbu ištestuoti šiuos elementus: apibrėžtos rizikos, susijusios su reikalavimais, veiklos procesai, panaudojimo atvejai, sistemos resursai, komunikacija su operacine sistema.

Sistemos architektūros sudarymas

Architektūros sudarymo etape sistema suskaidoma į valdomus atskirus komponentus, apibrėžiama grafinė komponentų vartotojo sąsaja. Šie darbai atliekami atsižvelgiant į sistemos komponentų tarpusavio komunikaciją. Visa surinkta ir specifikuota informacija naudinga sistemos integraciniam testavimui, kurio tikslas patikrinti, kaip sistemos komponentai komunikuoja tarpusavyje.

Pagal knygą „Software Testing Practice: Test Management“ [7], „V“ modelyje sistemos architektūros sudarymo fazė tiesiogiai siejasi su **integracijų testavimu**. Šio testavimo metu testuojamos integracijos tarp komponentų, komunikacija su skirtingomis sistemos dalimis: operacine sistema, failų sistema, technine įranga. Taip pat testuojamos sąsajos su kitomis sistemomis, jei tokios yra. Šiame etape sistemų testuotojai koncentruojasi tik į integraciją, neanalizuoja dviejų susijusių komponentų vidinio funkcionalumo.

Sistemos projektavimas

Šio etapo metu sudaroma detali sistemos specifikacija, kuri apibrėžia ne tik sistemos elgseną, vidinę struktūrą, bet ir darbų sekas, atspindinčias komponentų tarpusavio sąveikas. Ši informacija naudojama programuojant sistemą ir ruošiant testavimo scenarijus, skirtus ištestuoti atskirus sistemos komponentus.

Pagal knygą „Software Testing Practice: Test Management“ [7], „V“ modelyje sistemos projektavimo fazė tiesiogiai siejasi su **komponentų testavimo scenarijų kūrimu.** Šio tipo testavimo scenarijai yra skirti ištestuoti pavienius sistemos komponentus (objektus, klases, modulius) nepriklausomai nuo viso sistemos gyvavimo ciklo.

Sistemos programavimas

Šio etapo metu realizuojamas sistemos programinis kodas. Ankstesniuose etapuose surinkta informacija naudojama programuojant sistemą ir ruošiant testavimo scenarijus, skirtus ištestuoti atskirus sistemos komponentus.

Pagal knygą „Software Testing Practice: Test Management“ [7], „V“ modelyje programavimo fazė tiesiogiai siejama su **komponentų testavimu**, kurio metu testuojami pavieniai sistemos komponentai (objektai, klasės, moduliai) nepriklausomai nuo viso sistemos gyvavimo ciklo. Šio tipo testavimas dažnai vykdomas naudojant pagalbinę programinę įrangą, skurtą „vienetų“ testavimui (angl. *Unit test*). Komponentų testavimo metu dažnai įtraukiamas ne tik testuotojas, bet ir programuotojas, kuris rašė programinį kodą.

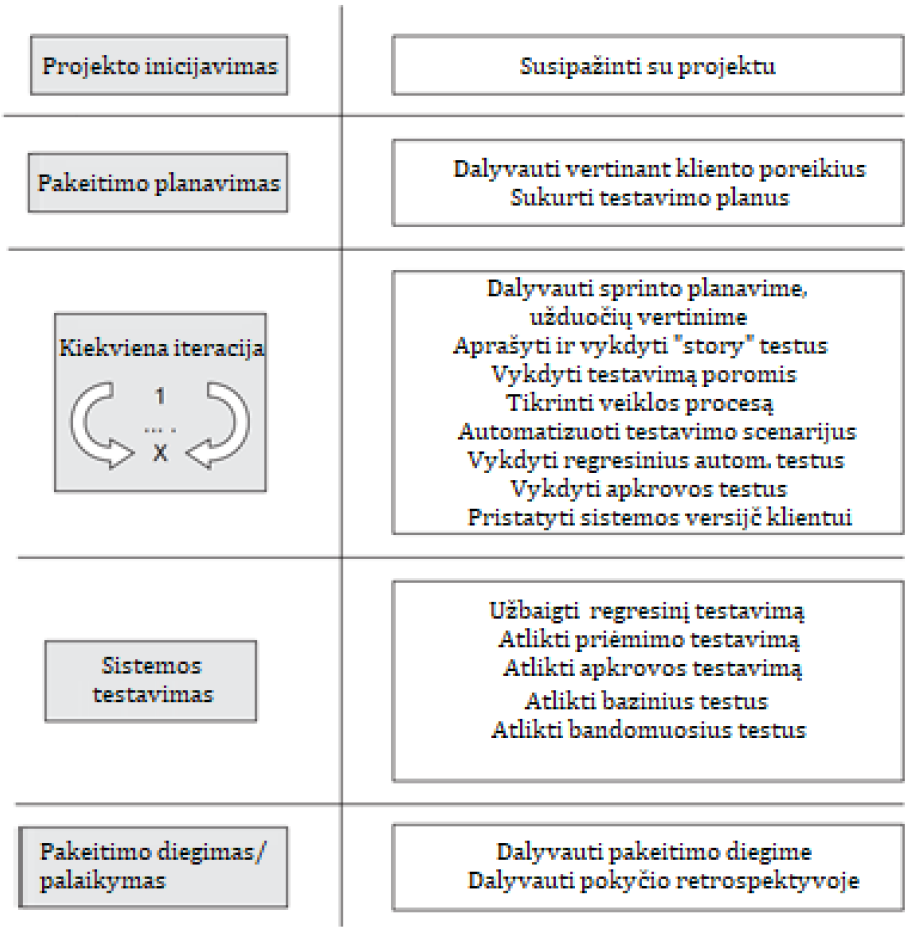
Pagal „V“ modelį kiekviena projekto kūrimo fazė turi turėti atitinkamą testavimo fazę.Ttam pritaria ir knygoje „Software Testing Practice: Test Management“ [7] apibrėžti principai, kurie teigia, jog testavimas bus kokybiškas, jeigu:

- kiekviena projekto realizavimo veikla turės atitinkamą testavimo veiklą;
- kiekvienas testavimo etapas turės konkretų tikslą;
- testuotojai kiek įmanoma bus įtraukti į visas projekto vykdymo veiklas, kuriose aptariami reikalavimai, kliento poreikiai, programavimo problemos ir pan.

Apibendrinant, galima teigti, kad testavimo proceso veiklos „V“ modeliu grindžiamuose projektuose yra glaudžiai susijusios su kitomis sistemos kūrimo veiklomis ir tai užtikrina visapusį sistemos kūrimo veiklų, bei jų atributų ištestavimą.

Testavimo procesas *Agile* manifestu grindžiamuose metoduose

Agile manifestu grindžiamuose projektuose dažniausiai yra išskiriamos penkios pagrindinės projekto vykdymo fazės ir kiekvienoje iš jų yra vykdomos tam tikros testavimo veiklos [8]: projekto inicijavimas (angl. *Project initiation*), projekto realizacijos planavimas (angl. *Release planning*), iteracijų vykdymas, sistemos testavimas, diegimas į gamybinę aplinką, palaikymas. Kiekvienoje fazėje yra vykdomos tam tikros testavimo veiklos (4 pav.).



4 pav. Testavimo proceso veiklos *Agile* tipo projektuose [8]

Kiekviena programinės įrangos kūrimo fazė turi savo tikslą, apibrėžtas testavimo, analizės, programavimo ir kitas reikiamas atlikti veiklas.

Inicijavimo fazė

Šiame etape testuotojai susipažįsta su projektu, gilinasi į turimą informaciją, dažniausiai realios testavimo veiklos nevykdomos. Etapo tikslas – susipažinti su projektu.

Planavimo fazė

Šiame etape testuotojai dalyvauja naudotojų reikalavimų vertinime ir aprašyme, ruošia testavimo planus. Šios fazės apimtyje identifikuojama būsima testavimo komanda, pasirenkami ir sudiegiami testavimo įrankiai, kurie bus naudojami testavimo metu. Be to, pagal poreikį sudaromas ribotų resursų planas. Jeigu projektas turi konkrečią užbaigimo datą, tuomet apibrėžiamas terminas iki kurio turi būti pradėta diegimo fazė. Ši metodika gera tuo, kad didžioji dalis funkcionalumo ištestuojama kūrimo fazėje, todėl diegimo fazėje belieka ištestuoti bendrą veiklos procesą. Taip pat šioje fazėje turėtų būti paruošiami pirminiai priėmimo testavimo scenarijai, testavimo planas.

Kūrimo fazė

Kūrimo etape testuotojai dalyvauja darbų planavime, užduočių apimties vertinime. Taip pat etapo metu yra kuriami ir vykdomi naudotojų reikalavimų testai, sudaromos testavimo poros pvz., du testuotojai arba vienas testuotojas ir programuotojas, vykdomas testavimo atvejų automatizavimas,

paleidžiami automatizuoti regresiniai testavimo atvejai, vykdomi apkrovos testai ir pateikiama sistemos demonstracija suinteresuotoms šalims.

Sistemos testavimo fazė

Šios fazės ribose užbaigiamas regresinis testavimas, atliekamas priėmimo testavimas. Pagal poreikį gali būti atliekamas apkrovos testavimas. Taip pat atliekami likę testai: baziniai (angl. *Smoke*), bandomieji (angl. *Mock*), apkrovos, regresiniai testai. Klientas taip pat atlieka priėmimo testavimą.

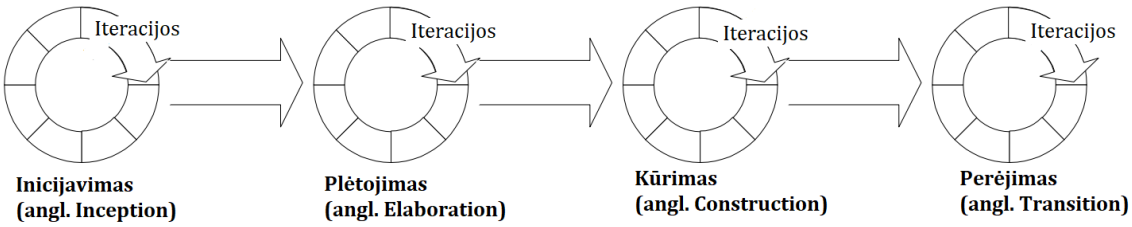
Diegimo ir palaikymo fazė

Šioje fazėje belieka ištirti užregistruotas kliento klaidas, o pataisius jas pertestuoti. Taip pat testuotojai dalyvauja produkto diegime į gamybinę aplinką ir projekto retrospektyvoje.

Pagrindiniai *Agile* manifestu grindžiami metodai: „RUP“ ir „Scrum“.

Testavimo procesas „RUP“ tipo projektuose

Straipsnyje [9] teigiama, kad PĮ kūrimo procesas turėtų pasižymėti kartotinumu. Tai reiškia, kad didelį projektą reikėtų suskaidyti į mažesnes dalis ir vykdyti jų realizavimą smulkiomis iteracijomis. Pagrindinį kartotinumu (angl. *iterative*) grindžiamą programų kūrimo principą atspindi „RUP“ [10] karkasas, kuriame pateikiamas kartotinio (angl. *iterative*) kūrimo modelis. „RUP“ metodą sudaro 4 fazės, kurios vykdomos iteracijomis (5 pav.).



5 pav. Kartotinio informacinių sistemų kūrimo fazės [11]

Viena „RUP“ iteracija pateikia tam tikrą vykdomą, ištestuotą sistemos prototipo versiją, kuri yra labiau išplėtota lyginant su prieš tai buvusios iteracijos versija. Šiame metode testavimo planas pradedamas ruošti pradinėje projekto fazėje. Visos testavimo veiklos yra struktūrizuotos, kiekviena dalis vykdoma atskiroje iteracijoje. „RUP“ metode išskiriami keturi testavimo etapai [12]: vienetų testavimas, integracijų testavimas, sistemos testavimas, priėmimo testavimas.

Be to, „RUP“ metodas apima skirtingus testavimo tipus: palyginimo testavimas (angl. *benchmark test*), konfigūracijos testavimas, funkcinis testavimas, diegimo testavimas, integracinis testavimas, našumo testavimas, maksimalios apkrovos testavimas (angl. *stress test*), regresinis testavimas. Pagrindinės testavimo veiklos išskiriamos „RUP“ metodikoje [12]:

- Testavimo planavimas – sukuriamas testavimo planas, kuriame pateikiama informacija, kokie reikalavimai bus testuojami, kokių tipų testavimai bus vykdomi ir testavimui reikalingi resursai, testavimo užduotys.
- Testavimo kūrimas – aprašomi konkretūs testavimo atvejai ir procedūros, kurios apibrėžtos testavimo plane ir bus vykdomos testavimo metu.
- Testavimo įgyvendinimas – realizuojami testavimo scenarijai, jų vykdomieji failai.

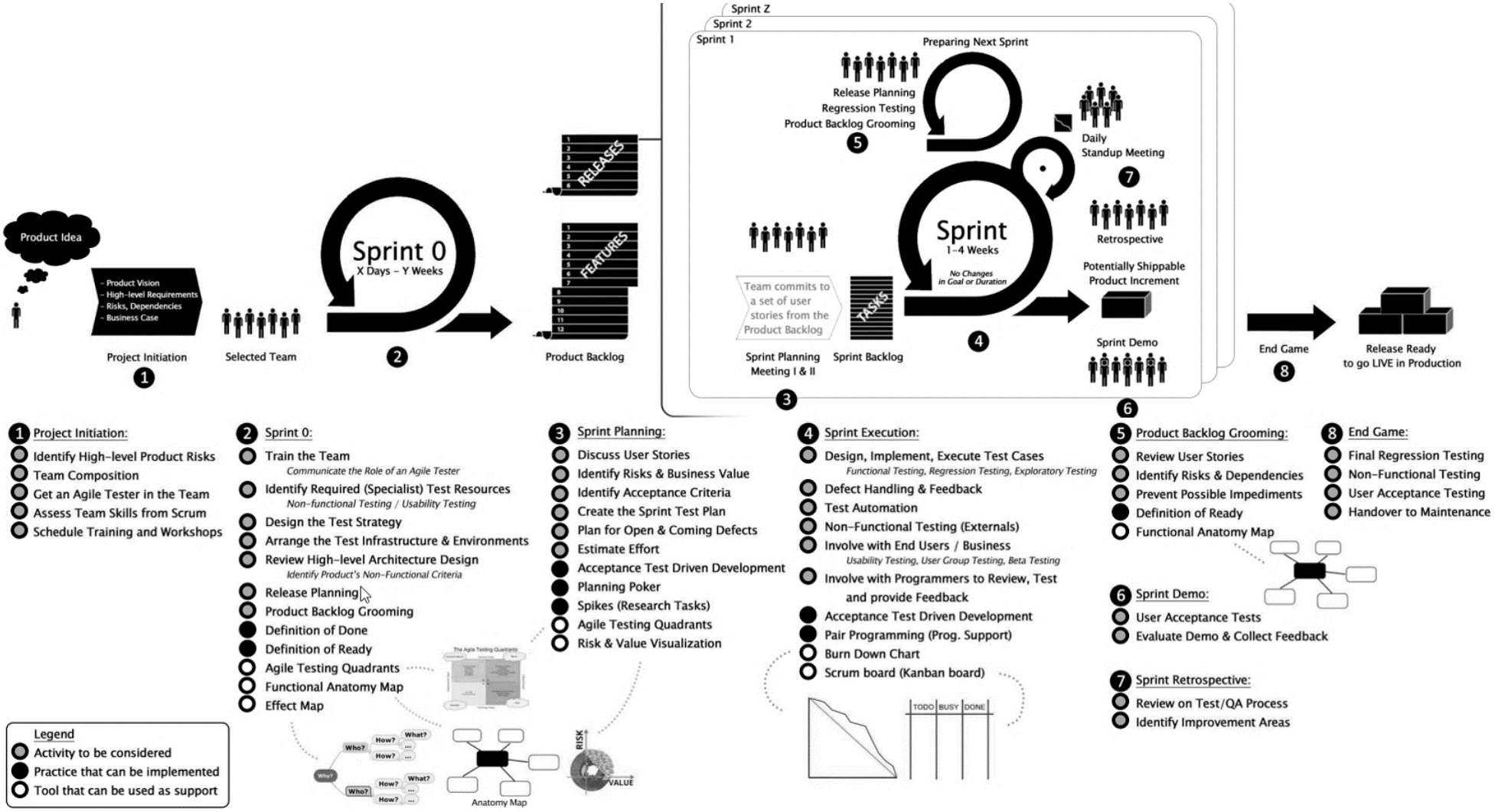
- Testavimo vykdymas integracijų lygyje – vykdomi automatiniai testai ir rankiniai, skirti patikrinti sistemos komponentų integracijas.
- Testavimo vykdymas sistemos lygyje – vykdomi automatiniai testai ir rankiniu būdu patikrinamas pagrindinis sistemos funkcionalumas.
- Testavimo vertinimas – vertinami atlikto testavimo rezultatai, paruošiamos pakeitimų užsakymo formos. Sukuriami matavimai, skirti įvertinti produkto (programinio kodo, reikalavimų padengimo, klaidų valdymo, testavimo vykdymo) kokybę.

Visos įvardintos testavimo veiklos vykdomos iteracijų metu kiekvienoje programinės įrangos kūrimo fazėje.

„RUP“ metodo modelis taip pat apibrėžia, kokie darbuotojai turėtų būti įtraukiami į testavimo veiklą: testuotojai, testavimo projektuotojai, sistemų testuotojai, našumo testuotojai, integracijų testuotojai. Pagrindiniai dokumentai yra: testavimo planas, scenarijai ir ataskaita, apkrovos testavimo planas ir atskaita, klaidų šalinimo planas.

Testavimo procesas „Scrum“ tipo projektuose

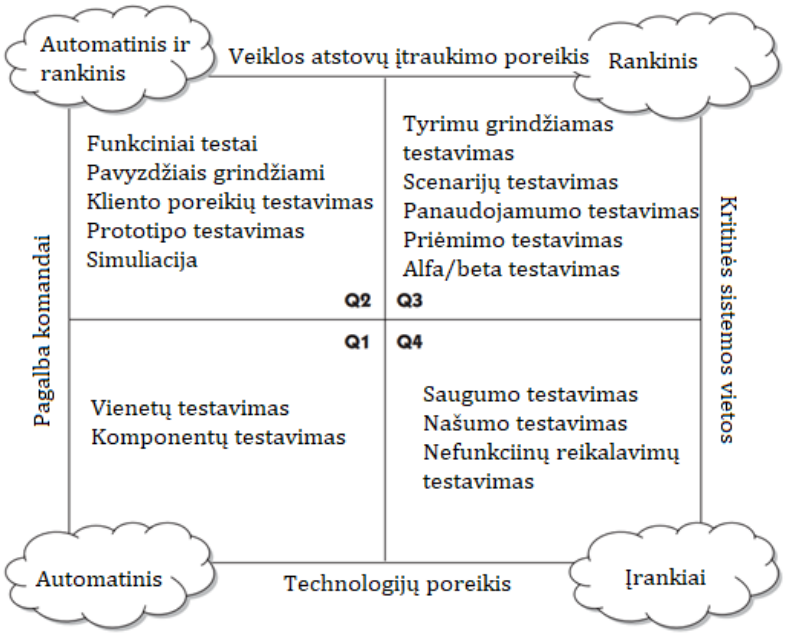
Be visų įvardintų metodų, ypač populiarus tampa *Agile* manifestu grindžiamas „Scrum“ metodas. Šis metodas orientuotas į tai, kad PĮ kūrimo procesas yra dinaminis ir jo metu keliami reikalavimai, apibrėžtas įgyvendinimo laikotarpis, ištekliai ir technologijos gali keistis. Dėl šios priežasties pats programinės įrangos kūrimo procesas turi būti kiek įmanoma lankstus ir galintis prisitaikyti prie kintančių aplinkybių. „Scrum“ metodo aprašyme [13] nėra išskirtos konkrečios testavimo veiklos, tačiau keletas mokslininkų atliko tyrimą, išanalizavo gerąsias praktikas ir sudarė „Scrum“ metodo ir testavimo veiklų karkasą, kuriame detaliai aprašomos visos rekomenduojamos testavimo veiklos (6 pav.).



6 pav. „Scrum“ ir testavimo veiklų karkasas [14]

Peržiūrėjus šaltinyje [14] sudarytą „Scrum“ metodo ir testavimo veiklų karkasą, galima išskirti pagrindinius testavimo tipus: *Agile* kvadratai, priėmimo testavimu grindžiamas programavimas, funkcinis ir nefunkcinis testavimas, regresinis testavimas, tiriamasis testavimas, priėmimo testavimas.

Tiek „RUP“, tiek „Scrum“ tipo projektuose viena iteracija apima kūrimo ir diegimo fazes, kurių metu sukuriama tam tikra ištestuota galutinio produkto versija. Iteracijos metu gali būti vykdomos įvairios testavimo veiklos, atsižvelgiant į „vartotojo poreikių“ (angl. *user story*) specifiką. 2003 metais Brian Marick ir Lisa Crispin sudarė *Agile* tipo testavimo kvadratus [15], kurie apibrėžia pagrindines galimas naudoti testavimo veiklas vienoje iteracijoje (7 pav.).



7 pav. *Agile* manifestu grindžiami testavimo veiklų kvadratai [15]

1 Kvadratas

Pirmasis kvadratas atspindi testavimu grindžiamą programų kūrimo (TDD) praktiką, kuri yra dažniausiai taikoma *Agile* tipo projektuose. Pagal šį kvadratą reikia vykdyti vienetų ir komponentų testavimus. Vienetų testavimas apima nedidelius testus, kurie skirti ištestuoti nedidelį sistemos pogrupį: objektą, metodą ar pana6iai. Komponentų testavimas jau apima didesnių sistemos elementų testavimą (pvz., klasių grupės, teikiančios tam tikrą paslaugą). Abiejų tipų testai dažniausiai yra automatizuojami naudojant įrankius, grindžiamus „Unit“ technologija. Šiuos testus dažniausiai atlieka programuotojai, kurie privalo atlikti vidinį kodo testavimą prieš atiduodant realizuotą funkcionalumą tolimesniam testavimui.

Pagrindinis šio kvadrato tikslas – testavimu grindžiamas sistemos kūrimas arba projektavimas. Ši testavimo atvejų rašymo praktika padeda programuotojams apsibrėžti ribas ir nesuprogramuoti to, kas nebuvo numatyta. Pagal sukurtus testus, programuotojai gali pasitikrinti, ar jų priimti architektūriniai sprendimai yra tinkami. Vienetų ir komponentų testai yra automatizuojami ir rašomi ta pačia kalba kaip ir testuojama sistema. Šio tipo testavimas yra skirtas vidiniam kokybės užtikrinimui, nes klientui jie tiesiog būtų neaktualūs ir dažnai nesuprantami (atsižvelgiant į kliento žinias IT srityje).

2 Kvadratas

Šio kvadrato testavimą taip pat atlieka sistemos kūrėjų komanda, tačiau jau aukštesniame abstrakcijos lygyje. Išskiriamos testų grupės: funkciniai testai, naudotojų pavyzdžiai, „Story“ testai, prototipas, simuliacija. Šie testai iš esmės skirti patikrinti kliento įvardytiems poreikiams, siekiant įsitikinti, ar sistema tikrai tenkina pradinius kliento reikalavimus. Šio kvadrato apimtyje testai kuriami pagal kliento įvardintus pavyzdžius iš veiklos srities, kuomet klientai detaliai paaiškina kiekvieną poreikį (angl. *user stories*). Testavimo atvejai rašomi klientui suprantamu formatu, nes šiuos testus klientai naudoja patys, tikrina, teikia pastabas ir dažnai padeda rašyti.

Antrojo ir pirmojo kvadrato testai turi duoti greitą grįžtamąjį ryšį komandai, todėl dažnai ir antrojo kvadrato testai yra automatizuojami. Jei tik yra įmanoma, stengiamasi antrojo kvadrato testus vykdyti jau su realiais duomenimis iš kliento produkcinės duomenų bazės, siekiant patikrinti kuo realesnes situacijas ir taip sutaupyti laiko tolimesniuose etapuose. Be to, šis kvadratas apima ir grafinės sąsajos testavimą, kuris nėra automatizuotas.

3 Kvadratas

Šiam kvadratui priskiriamos testavimo veiklos: scenarijų testavimas, priėmimo testavimas, tinkamumo testavimas, vidinis/tiriamasis testavimas, alfa/beta testavimas.

Scenarijų testavimas apima rankinį testavimą pagal veiklos procesą, siekiant imituoti realius procesus, kuriuos vykdys sistemos naudotojai. Automatiniai vykdomieji kodo fragmentai (angl. *scripts*) gali būti naudojami greitesniam reikiamų duomenų išgavimui ar generavimui, bet testuotojas privalo duomenis naudoti pagal veiklos logiką, siekiant užtikrinti, kad sukurtas funkcionalumas tikrai atitinka kliento poreikius.

Priėmimo testus dažniausiai atlieka užsakovas su galutiniu sistemos naudotoju. Priėmimo testavimas suteikia galimybę įvertinti, ar naujas funkcionalumas veikia korektiškai ir, ar atitinka kliento poreikius.

Tinkamumo testavimą dažniausiai atlieka grupė žmonių, kuri bando naują funkcionalumą ir pateikia atsiliepimus. Taip pat gali būti realiu laiko momentu vertinami naudotojų atliekami veiksmai dirbant su sistema siekiant įvertinti, ar sistema yra „draugiška“ naudotojo atžvilgiu.

Vidinis/tiriamasis testavimas yra svarbiausias šiame kvadrate. Šio testavimo metu testuotojas turi pasinaudoti savo kūrybiškumu. Vykdant procesus, susijusius su nauju funkcionalumu, testuotojas turi ištirti gautus rezultatus ir pagal juos sugalvoti ir ištestuoti kitus alternatyvius, bei kritiškus scenarijus, kuriuos naudotojas gali įvykdyti. Statistiškai šio testavimo metu yra randama daugiausiai klaidų.

4 Kvadratas

Šiame kvadrate tikrinami nefunkciniai reikalavimai naudojant apkrovos, apsaugos, našumo ir kitus testus. Šie testai yra kritiškai svarbūs, nes juos naudojant įvertinamos pagrindinės sistemos kokybės charakteristikos.

Agile kvadratai plačiai naudojami daugumoje *Agile* manifestu grindžiamų metodų. Dėl šios priežasties testavimo procesas *Agile* projektuose tiriamajame darbe bus analizuojamas ir projektuojamas žvelgiant iš *Agile* testavimo kvadratų perspektyvos.

Išanalizavus keletą PĮ kūrimo metodikų, galima teigti, kad vienas iš pagrindinių etapų PĮ gyvavimo cikle yra testavimo procesas ir jis yra tiesiogiai susijęs su kitomis veiklomis, todėl įtakoja viso IT projekto sėkmę. Tai tik pagrindžia, jog tyrimo objektas yra svarbus.

1.3.2. Testavimo procesas

Ankstesniame 1.3.1 skyriuje aptartas abstraktus testavimo proceso kontekstas, kuris teigia, jog testavimas tiesiogiai priklauso nuo pasirinktos programinės įrangos kūrimo metodikos. Nepaisant to, šis tiriamasis darbas yra orientuotas į testavimą, todėl būtina detaliai išanalizuoti testavimo proceso tikslus, etapus, vykdymo eigą ir stebėjimą. Atlikus šių objektų analizę, bus galima identifikuoti silpnąsias testavimo proceso vietas ir spręsti, kaip būtų galima jas patobulinti.

Testavimo proceso samprata

Testavimo procesas yra glaudžiai susijęs su sistemos kokybės rodikliais, kokybės užtikrinimo procesu. Sistemos kokybė pagal IEEE [16] yra apibrėžiama:

1. Lygis, kuriame sistema, jos komponentas ar procesas pilnai atitinka iš anksto numatytus reikalavimus.
2. Lygis, kuriame sistema, jos komponentas ar procesas pilnai atitinka kliento norus, išpildo jo lūkesčius.

Pirmasis sistemos kokybės apibrėžimas yra orientuotas į sistemos specifikaciją, kuri sukuriama sistemos projektavimo etape. Sistemos specifikacija yra sudaroma pagal kliento iškeltus reikalavimus. Pagal šį požiūrį, klaidos esančios sistemos specifikacijoje ar kliento reikalavimuose nesumažina sistemos kokybės.

Antras sistemos kokybės apibrėžimas yra orientuotas į patį klientą, jo pasitenkinimą ir lūkesčius. Pagal D. Galin knygą, laikantis antro principo testuotojas privalo papildomai įvertinti kliento poreikius, pastebėti spragas sistemos specifikacijoje, o programuotojas turi realizuoti sistemą taip, kad ji pilnai atitiktų kliento poreikius [17].

Pagal ISO 2500 standartą [18] sistemos kokybės modelis apima kliento poreikius, kurie yra susiję su sistemos našumu, tinkamumu, saugumu, prieinamumu ir kitais rodikliais.

Funkcionalumo atitikimas (angl. *functional suitability*) atspindi, ar sistemoje įgyvendintas funkcionalumas atitinka naudotojo poreikius. Apima kitas funkcionalumo metrikas: išbaigtumą, korektiškumą, tinkamumą.

Veiklos efektyvumas (angl. *performance efficiency*) parodo našumą, susijusį su naudojamų resursų kiekiu. Apima metrikas: laiko elgsena, išteklių panaudojimą, pajėgumą.

Suderinamumas (angl. *compatibility*) – rodiklis, kuris nusako, ar sistema arba jos komponentas gali dalintis informacija su kitomis sistemomis ar komponentais tuo pačiu metu dalijantis ta pačia technine ar programine įranga. Apima metrikas: sąveikos, egzistavimas.

Tinkamumas (angl. *usability*) nusako galimybę naudoti sistemą, siekiant pasiekti konkrečius tikslus ir užtikrinant, kad naudotojas bus patenkintas funkcionalumu. Apimamos metrikos: tinkamumas, mokomumas, veikimas, naudotojo klaidų išvengimas, vartotojo sąsajos estetinis vaizdas, prieinamumas.

Patikimumas (angl. *reliability*) laiko periodas per kurį sistema tam tikromis sąlygomis gali atlikti nurodytas funkcijas. Susideda iš metrikų: terminas, prieinamumas, galimybė atsistatyti, tolerancija gedimams.

Saugumas (angl. *security*) nurodo, ar sistema apsaugo informaciją ir duomenis, ar sistemoje visi naudotojai turi atitinkamus prieigų tipus ir lygius. Susideda iš metrikų: konfidencialumas, vientisumas, atskaitomybė, autentiškumas, sąžiningumas.

Priežiūra (angl. m*aintainability*) atspindi, kaip sistema gali būti modifikuojama, tobulinama arba pritaikoma prie pakitusios aplinkos ir reikalavimų. Apima metrikas susijusias su galimybe skaidyti sistemą į modulius, plėsti analizę, atlikti pakeitimus ir testavimą.

Perkėlimo galimybė (angl. *portability*) atspindi, ar sistema gali būti perkeliama iš vienos techninės, programinės įrangos, naudojamos aplinkos į kitą. Susideda iš metrikų, susijusių su galimybe: pritaikyti, įdiegti ir pakeisti sistemos komponentą.

Išanalizavus sistemos kokybės sampratą, galima pereiti prie **kokybės užtikrinimo proceso**, kuris pagal IEEE [16] apibrėžiamas:
1. sistemingas ir suplanuotas veiksmų planas, padedantis užtikrinti, kad sistema atitinka iš anksto nustatytus techninius reikalavimus;
2. veiksmų planas, sukurtas įvertinti procesą, kuriuo remiantis buvo sukurta sistema.

Įvardyti apibrėžimai kokybės užtikrinimo procesą apriboja technine specifikacija ir programinės įrangos kūrimo procesu. Egzistuoja ir kitokio požiūrio apibrėžimas pagal D. Galin knygą [17]. Trečiasis apibrėžimas kokybės užtikrinimui suteikia didesnę prasmę programinės įrangos kūrimo procese. Šis apibrėžimas teigia, jog kokybės užtikinimo procesas (angl. *software quality assurance*) yra:
3. sistemingas, suplanuotas procesas, padedantis palaikyti tinkamą sistemos kūrimo ir priežiūros procesą, užtikrinantį sistemos atitikimą techninės specifikacijos reikalavimams, bei tvarkaraščio palaikymą ir sistemos plėtojimą pagal numatytą biudžetą.

Apibrėžus sistemos kokybės, jos užtikrinimo proceso sąvokas, galima pereiti prie testavimo proceso, kuris taip pat turi ne vieną apibrėžimą. Pagal IEEE [16] **testavimo procesas** yra:
1. sistemos ar komponento naudojimas pagal numatytas sąlygas stebint gaunamus rezultatus ir įvertinant tam tikrus sistemos ar komponento aspektus;
2. procesas, kurio metu analizuojama sistemos dalis, siekiant aptikti neatitikimus tarp sistemos ir sistemos reikalavimų dokumento, bei įvertinti sistemos funkcijas.

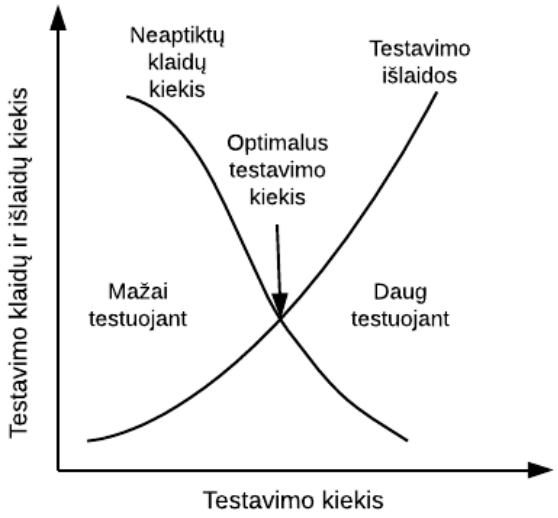
Įvardyti du apibrėžimai apima tik dalį testavimo proceso, kuris yra pakankamai didelis, kad būtų galima apibūdinti kelias sakiniais. Nepaisant to, IEEE apibrėžimai atspindi pagrindinę testavimo logiką. Antrasis apibrėžimas tinkamas tada, kai sistema turi detalią specifikaciją pagal kurią testuotojas gali testuoti sistemą, o pirmasis yra tinkamas tuo atveju, jeigu sistema nėra specifikuota ir testuotojas pagal savo nuožiūrą turi testuoti sistemos funkcionalumą.

Testavimo proceso tikslai

Programinės įrangos testavimas yra vykdomas dėl tam tikrų priežasčių, kiekvienas testavimo etapas turi tam tikrus tikslus ir priežastis. D. Galin knygoje („Software Quality Assurance“) įvardyti testavimo tikslai:

1. aptikti kuo daugiau klaidų sistemoje;
2. ištestuoti sistemą taip, kad po aptiktų klaidų ištaisymo ir pertestavimo sistema būtų tinkama klientui priimti;
3. numatytą testavimą atlikti efektyviai, pagal suplanuotą grafiką ir resursus;
4. sudaryti klaidų sąrašą, siekiant ateityje išvengti analogiškų klaidų.

Įvardyti tikslai atspindi, jog nėra įmanoma rasti ir ištaisyti visas sistemos klaidas. Dėl šios priežasties turi būti apibrėžtos testavimo ribos, kurios nurodytų, kada testavimas turi būti baigtas. Dažniausiai testavimo užbaigimo riba nustatoma iš kart po to, kai nustatoma, kad sukurta sistema tampa tinkama klientui priimti. Būtina pabrėžti, jog testavimo proceso apimtis, likusių sistemoje klaidų kiekis ir reikalingos testavimui išlaidos yra tarpusavyje susiję matai (8 pav.).



8 pav. Testavimo kiekio, klaidų ir išlaidų priklausomybė [7]

Testavimo apimtis pasirenkama pagal projekto rizikos lygį. Jeigu kuriama sistema yra kritinė (įvykus sistemos klaidai, gali nukentėti žmogaus gyvybė), tuomet testavimui yra skiriama daug dėmesio, tačiau toks testavimas yra itin brangus. Sistemoms, kurios nėra kritinės, dažniausiai yra pasirenkamas optimalus testavimo kiekis, kuris kiek įmanoma sumažina sistemoje likusių klaidų kiekį ir nėra labai brangus.

Testavimo tipai

Testavimas yra daugiau nei sistemos formų užpildymas ir patikrinimas, ar gauti teisingi rezultatai. Testuotojų veikla turėtų apimti ir projektinių dokumentų (reikalavimų specifikacija, naudotojo sąsajos prototipas, diegimo instrukcijos ir pan.) peržiūrą, pastabų šiems dokumentams pateikimą.

Testavimas gali būti:

- **Statinis** – tikrinamas sistemos programinis kodas (rankiniu būdu arba naudojant įrankius), bei visa reikiama projektinė dokumentacija (reikalavimų specifikacija, diegimo instrukcijos ir t. t.). Testavimo metu sistema nėra vykdoma. Šio tipo testavimui vykdyti naudojami formalūs ir neformalūs metodai. Nesvarbu, koks metodas pasirinktas, svarbu, kad šis testavimas būtų atliktas kaip galima greičiau, nes vėlyvose projekto stadijose aptiktų klaidų taisymas reikalauja daug resursų.
- **Dinaminis** – paleidžiama ir testuojama pati sistema.

Dinaminis testavimas gali būti :

- **Funkcinis** – tikrinama, ką sistema daro (t. y. sistemos funkcijos). Testuojant „ką“ sistema daro, reikia patikrinti ir tai, ar sistema neatlieka to, ko ji neturėtų daryti (ar nėra perteklinių funkcijų);
- **Nefunkcinis** – kai tikrinama, „kaip“ sistema veikia (greitaveika, saugumas ir pan.).

Testavimo lygiai

Testavimas yra vykdomas pagal projekto etapus. Kiekviename etape atliekamas tam tikro lygio testavimas. Galimi testavimo lygiai:

- **Vienetų testavimas** yra grindžiamas programinio kodo elementų testavimu. Įprastu atveju šį testavimą atlieka programuotojas darbinėje (DEV) aplinkoje. Vienetų testavimas taikomas smulkiems elementams, kurie įprastai nėra didesni nei programiniame kode aprašyta klasė.
- **Sąsajų testavimas** tikrina, kaip programiniai komponentai ištestuoti vienetų testavimo metu komunikuoja ir dera tarpusavyje. Šį testavimą taip pat dažniausiai atlieka programuotojas.
- **Sistemos testavimas** – testavimas, kurio metu testuotojų komanda pilnai ištestuoja sistemą. Atlikus šį testavimą, sistema turi būti tinkama perduoti kliento priėmimui.
- **Priėmimo testavimas** vykdomas kliento įmonėje, siekiant įvertinti, ar sistema pilnai tenkina lūkesčius ir iškeltus reikalavimus.
- **Regresinis testavimas** vykdomas rangovų, kai reikalingas pakartotinis pagrindinio sistemos funkcionalumo patikrinimas. Pagrindinis tikslas yra patikrinti, ar naujas funkcionalumas nesugadino iki pakeitimo egzistavusio funkcionalumo. Šio tipo testavimas atliekamas po egzistuojančios sistemos pakeitimo diegimo.
- **Alfa testavimas** – testavimas, kurio metu sistemos kūrėjai išbando sistemą turimoje realioje ar dirbtinai sukurtoje aplinkoje. Po šių bandymų pataisomos visos rastos klaidos.
- **Beta testavimas** – testavimas, kurio metu klientas išbando sistemą produkcinėje aplinkoje. Klientas gali duoti išbandyti sistemą tiesioginiams sistemos naudotojams. Po šio testavimo surinkti atsiliepimai ir pasiūlymai apie galimus patobulinimus siunčiami sistemos kūrėjų komandai.
- **Testavimas po diegimo** – testavimas, kurio metu sistemų administratorius, sudiegus sistemą arba pakeitimą, patikrina, ar visi pakeitimai susidiegė ir, ar veikia visos sistemos formos.

Visos įvardintos testavimo fazės yra svarbios, tačiau daugiausiai dėmesio yra skiriama vidiniam ir priėmimo testavimui.

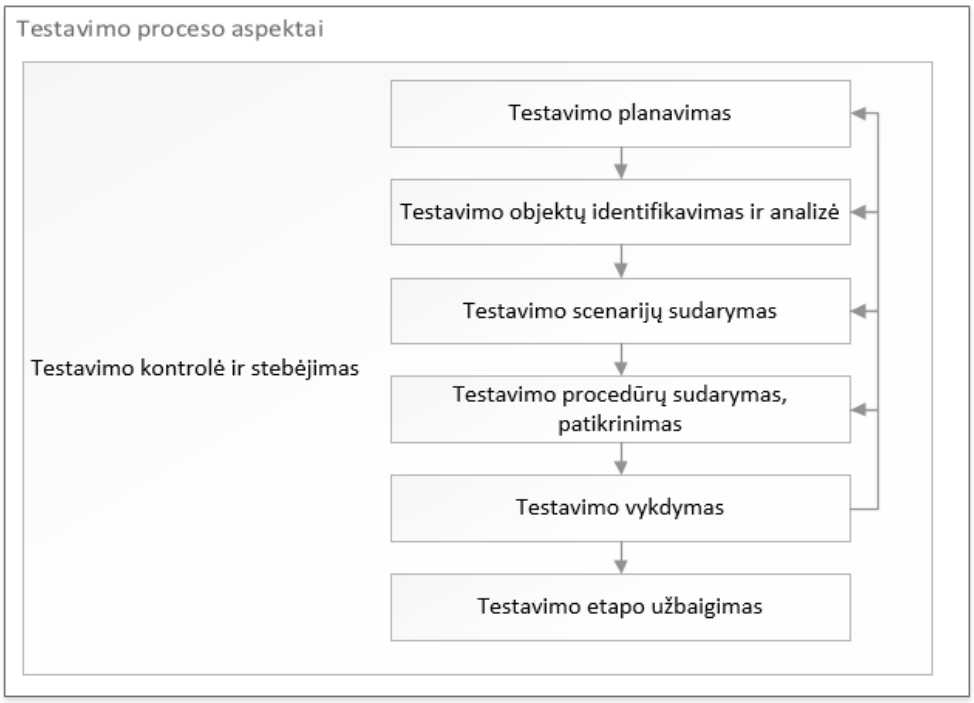
Testavimo proceso etapai

Testavimo proceso specifika priklauso nuo daugybės veiksnių: pasirinkto PĮ kūrimo metodo, projekto sudėtingumo, rizikos lygio ir apimties, išorinių ir vidinių įmonės standartų [19]. Nepaisant to, didžiausią įtaką testavimo proceso veikloms daro pasirinkta PĮ kūrimo metodika. Praktikoje testavimo proceso veiklas stengiamasi kuo anksčiau įtraukti į programinės įrangos kūrimo procesą. Dėl šios priežasties PĮ testavimas persidengia su kitomis PĮ kūrimo fazėmis ir tiesiogiai nuo jų priklauso. Žvelgiant į testavimą kaip į atskirą ir nepriklausomą procesą, galima išskirti septynis testavimo proceso etapus (9 pav.):

1. testavimo planavimas;
2. testavimo kontrolė ir stebėjimas;
3. testavimo objektų identifikavimas ir analizė;

4. testavimo scenarijų sudarymas;
5. testavimo procedūrų sudarymas, patikrinimas;
6. testavimo vykdymas;
7. testavimo etapo užbaigimas.

Testavimo proceso veiklos yra vykdomos pagal numatytą eiliškumą (9 pav.). Veiklos yra vykdomos nuosekliai, tačiau pagal poreikį iš vienos veiklos galima sugrįžti į jau vykdytą veiklą ir patikslinti reikiamus duomenis.



9 pav. Testavimo proceso pagrindiniai aspektai

Testavimo planavimo etapas, kurio metu atliekamos veiklos:
1. apibrėžiama apimtis, rizikos ir pagrindiniai objektai, kuriuos reikės ištestuoti;
2. pasirenkami testavimo metodai, kurie bus naudojami testavimo metu;
3. įgyvendinama testavimo politika ir (arba) testavimo strategiją (testavimo strategija – planas, apibūdinantis programinės įrangos kūrimo testavimo dalį. Jis sukurtas informuoti testuotojus, programuotojus apie kai kuriuos svarbiausius testavimo proceso aspektus. Testavimo strategija apima testavimo tikslus, testavimo metodą, bendrą laiką, išteklius, reikalingus projektui ir testavimo aplinkai);
4. apibrėžiami testavimui reikalingi resursai: žmonės, testavimo aplinkos, kompiuteriai ir pan.
5. paruošiamas testavimo analizės ir projektavimo užduočių, testavimo vykdymo ir vertinimo planas;
6. apibrėžiami įvykdymo kriterijai, kurie nurodo, kas turi būti atlikta testavimo metu. Tai padeda sekti progresą, norint nustatyti, ar testavimas vykdoma korektiškai. Šie kriterijai apibrėžia, kokios veiklos turi būti užbaigtos, tam kad sistemos testavimas būtų baigtas.

Testavimo objektų identifikavimo ir analizės etapas, kurio metu atliekamos veiklos:
1. peržiūrėti testavimo medžiagą (testavimo medžiaga – informacija, kuri reikalinga pradėti testavimo analizę ir paruošti testavimo scenarijus. Iš esmės tai yra dokumentas, kuriuo remiasi

testavimo scenarijai, pavyzdžiui, reikalavimų, projektavimo specifikacijos, rizikos analizė, architektūra ir sąsajos. Siekiant suprasti, ką realizuota sistema turės daryti galima naudoti įvardintus testavimo bandymams reikalingus dokumentus);
2. nustatyti testavimo sąlygas;
3. paruošti testus;
4. įvertinti reikalavimų ir sistemos patikimumą;
5. paruošti testavimo aplinką ir identifikuoti reikalingas infrastruktūros priemones.

Testavimo scenarijų sudarymo etapas, kurio metu atliekamos veiklos:
1. kurti scenarijus ir suteikti jiems prioritetus pagal technikas, bei pasiruošti duomenis, kurie bus naudingi testuojant sistemą (testavimo metu reikia suvesti tam tikrus duomenis, kurie bus naudojami ir gamybinėje aplinkoje naudojant naują funkcionalumą. Tokio tipo duomenys vadinami testavimo duomenimis);
2. paruošiamos testavimo instrukcijos dar kitaip vadinamos testavimo procedūromis;
3. pagal poreikį automatizuoti kai kuriuos testavimo scenarijus naudojant vykdomuosius programinio kodo fragmentus (angl. *scripts*);
4. paruošti testavimo rinkinius iš sukurtų testavimo scenarijų, siekiant atlikti efektyvų testavimą.

Testavimo procedūrų sudarymo, patikrinimo etapas, kurio metu atliekamos veiklos:
1. patikrinti testavimo žurnalus pagal laukiamus rezultatus, nurodytus testavimo plane;
2. įvertinti, ar reikia tęsti testavimą;
3. paruošti apibendrinančią testavimo ataskaitą, skirtą klientams.

Testavimo vykdymo etapas, kurio metu atliekamos veiklos:
1. paleisti ir patikrinti testavimo aplinką;
2. pagal numatytas testavimo procedūras vykdyti testavimo scenarijus ir atvejus;
3. atlikti pakartotinį testavimą;
4. fiksuoti testavimo rezultatus, nurodant sistemos versiją, testuotą elementą;
5. palyginti gautus rezultatus su laukiamais rezultatais;
6. radus neatitikimus tarp laukiamo ir gauto rezultato, užregistruoti klaidas, klaidų kontrolės sistemoje.

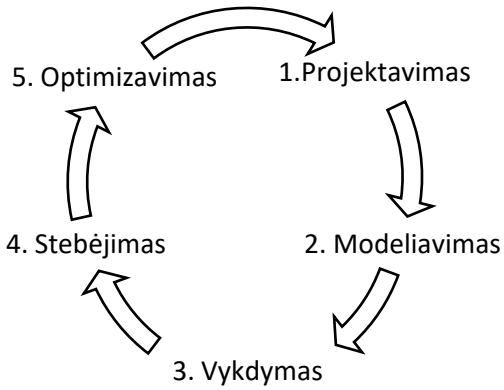
Testavimo etapo užbaigimas, kurio metu atliekamos veiklos:
1. patikrinti, ar testavimo atvejų laukiami rezultatai visi teigiami, ar tikrai visos klaidos išspręstos;
2. užbaigti ir suarchyvuoti visus surinktus rezultatus, testavimo aplinką, įrankius, tolimesniam pakartotiniam panaudojimui;
3. perduoti sistemą techninei priežiūrai;
4. įvertinti testavimo procesą, rezultatus, identifikuoti padarytas klaidas ir iš jų pasimokyti.

Testavimo kontrolės ir stebėjimo etapas, kurio metu atliekamos veiklos:
1. įvertinti ir analizuoti testavimo rezultatus;
2. stebėti, dokumentuoti progresą, testavimo rezultatus;
3. teikti informaciją apie testavimą;
4. inicijuoti reikiamas veiklas;
5. priimti sprendimus.

Testavimo procesas yra iteracinis, išvardintos projekto fazės vykdomos ne nuosekliai, o pagal poreikį iš vienos fazės galima grįžti į kitą. Kiekviena proceso fazė yra svarbi ir turi būti vykdoma nepriklausomai nuo projekto dydžio ar sudėtingumo.

1.3.3. Testavimo proceso ir vykdomaisiais modeliais grindžiamų sistemų suderinamumas

Pastaruoju metu dauguma organizacijų siekia optimizuoti, supaprastinti, kompiuterizuoti vykdomus veiklos procesus. Pagal BPM discipliną veiklos procesų gyvavimo ciklas turėtų susidėti iš šių veiklų: veiklos procesų projektavimo ir modeliavimo, veiklos procesų vykdymo, stebėjimo, tobulinimo. Visos išvardintos veiklos turi būti ciklišk ai kartojamos (10 pav.) kol procesai tampa pilnai optimizuoti [20].



10 pav. Bendras BPM gyvavimo ciklas [20]

Turint sumodeliuotus ir optimizuotus veiklos procesus organizacija gali pereiti prie procesų kompiuterizavimo etapo. Deja, tradicinė programinė įranga dažniausiai nėra pritaikytos dinamiškiems organizacijų veiklos procesams ir tokio tipo sistemų modifikavimas reikalauja daug resursų. Be to, tradicinės IS yra daugiau orientuotos į dalykinės srities kompiuterizavimą, bet ne į vykdomų procesų stebėjimą, siekiant identifikuoti silpnąsias vietas, kad prireikus būtų galima priimti reikiamus sprendimus. Dėl šios priežasties auga veiklos procesais grindžiamų informacinių sistemų poreikis ir paklausa. Informacinė sistema, realizuota pagal sumodeliuotą veiklos procesą, gali būti nesudėtingai modifikuojama be programuotojo pagalbos, sistema suteikia galimybę stebėti veiklos procesus realiu laiku. Be to, vykdomaisiais modeliais grindžiamos sistemos yra pradedamos kurti nuo veiklos procesų analizės ir modeliavimo, o sukurti veiklos procesų modeliai taip pat gali būti itin naudingi organizacijai. Sukurtus BPM modelius galima naudoti ne tik PĮ kūrimui, bet ir naujų darbuotojų mokymui, veiklos procesų pakeitimų demonstracijai ir pristatymui darbuotojams. Vizualus veiklos aprašas supaprastina ir sutrumpina darbuotojų mokymo laiką, sudėtingi veiklos procesai tampa paprasčiau suprantami, aiškiai išskiria esamas pareigybes ir užduotis už kurias yra atsakingi darbuotojai [21].

Vykdomuoju modeliu grindžiama sistema galėtų būti naudinga ir testavimo proceso kompiuterizavimui. Testavimo procese pagrindinė problema – nesusikalbėjimas tarp komandos narių, dažnai darbai testuotojams perduodami ne pagal grafiką, egzistuoja uždelsimas tarp užduočių vykdymo. Jeigu testavimo procesas būtų kompiuterizuojamas panaudojant BPM technologijomis grindžiamą sistemą, pats testavimo procesas, jo planavimas ir vykdymas taptų struktūrizuotas, testavimo veiklos vyktų pagal numatytą eiliškumą, grafikus ir prioritetus. Tokio tipo sistema

užtikrintų efektyvų testuotojų darbą, taip pat suteiktų galimybę testuotojų vadovui konkrečiu laiko momentu stebėti testuotojų darbus ir jų būsenas.

1.4. Programinės įrangos testavimo metodai

Dažnai sistemos testuojamos remiantis trijų tipų testavimo metodais: baltosios dėžės, juodosios dėžės ir praktika grindžiamais testavimo metodais. Visi šie PĮ testavimo metodų tipai turi savų privalumų ir trūkumų.

1.4.1. Baltosios dėžės testavimo metodai

Pagrindiniai principai

Baltosios dėžės testavimas yra taikomas siekiant aptikti logines klaidas programiniame kode. Tokio tipo testavimas atliekamas nuosekliai vykdant programinį kodą, stebint iškylančias klaidas. Tokiu būdu identifikuojami neteisingi sprendimai priimti programavimo metu. Baltosios dėžes testavimas gali būti naudojamas ir kitiems tikslams: reikalavimų analizei, projektavimui, testavimo scenarijų kūrimui. Šio tipo testavimo metodai skaidomi į dvi grupes [22]:

- Statinis baltosios dėžės testavimas:
  - „Atlikto darbo tikrinimas“ (angl. *desk checking*);
  - Nuoseklus kodo tikrinimas;
  - Formalūs patikrinimai.
- Struktūrinis baltosios dėžės testavimas:
  - Ciklų testavimas;
  - Duomenų srautų testavimas;
  - Padengimo (angl. *coverage*) testavimas;
  - Pagrindinio kelio testavimas.

Statinis baltosios dėžės testavimas apima tik sistemos programinį kodą, šio tipo testavimas vykdomas, kai kodas dar nėra pabaigtas ir dar nei karto nebuvo įvykdytas. Šio tipo testavimą vykdo konkretūs paskirti žmonės, kurie turi atrasti defektus programiniame kode. Pagrindinis tikslas – patikrinti, ar kodas atitinka funkcinius reikalavimus, sistemos projektą, programavimo taisykles ir standartus.

Struktūrinis baltosios dėžės testavimas atsižvelgia į programinio kodo struktūrą, vidinį dizainą ir kodavimo principą.

Pagrindiniai baltosios dėžės metodai: teiginių padengimo testavimas (angl. s*tatement coverage*), sprendimų testavimas (angl. *decision (branch) testing*), galimų kelių testavimas (angl. *path testing*).

Privalumai ir trūkumai

Pagrindiniai šių metodų privalumai: išsamiai testuojamas naujas kuriamas funkcionalumas, aprašomi visi naujo funkcionalumo testavimo atvejai, nesudėtinga automatizuoti, nėra imlus laikui, pagreitina testavimo procesą.

Pagrindiniai šių metodų trūkumai: kaina, reikalauja daug pakeitimų, jeigu yra automatizuotas, o kuriama PĮ nuolat keičiasi. Taip pat lieka nemažai neištestuotų atvejų, kurie susiję ne tik su nauju funkcionalumu.

1.4.2. Juodosios dėžės testavimo metodai

Pagrindiniai principai

Testuojant sistemą remiantis juodosios dėžės metodais, sistema yra traktuojama kaip juoda dėžė, kurios vidaus testuotojas nemato ir netestuoja. Pagrindinis šio testavimo metodo tikslas – patikrinti sistemos funkcijų grąžinamą rezultatą pateikus iš anksto numatytus duomenis. Šio tipo testavimas remiasi ne sistemos vidine struktūra, o jos specifikacija.

Tokio tipo testavimo metu testuotojai patikrina sistemą tik iš naudotojo perspektyvos, identifikuoja reikalavimų neišbaigtumą, sistemos atitikimą kliento poreikiams. Tam, kad testuotojai pilnai suprastų sistemai keliamus reikalavimus, testavimas pradedamas ankstyvoje analizės fazėje, testuotojai pilnai įtraukiami į kliento reikalavimų surinkimą ir analizę. Projektavimo metu, testuotojai pradeda rašytis testavimo scenarijus.

Pagrindiniai juodosios dėžės metodai: padalijimas į ekvivalenčias klases (angl. *equivalence partitioning*), ribinių reikšmių analizė (angl. *boundary value analysis*), SP lentelių testavimas (angl. *decision table testing*), būsenų perėjimų testavimas (angl. *state transition diagrams*), panaudojimo atvejų testavimas (angl. *use case testing*).

Privalumai ir trūkumai

Pagrindiniai šių metodų privalumai: procese testuotojas ir programuotojas yra vienas nuo kito nepriklausomi, testavimą galima pradėti vykdyti ankstyvoje fazėje, šio tipo testavimas išgrynina specifikacijoje esančias klaidas, testuotojams nereikalingos programavimo žinios, tokio tipo testavimą gali vykdyti ir asmuo, neturintis IT išsilavinimo.

Pagrindiniai šių metodų trūkumai: ištestuojama maža dalis scenarijų, neištestuojami sudėtingi ir komplikuoti kodo fragmentai.

1.4.3. Praktika grindžiami testavimo metodai

Pagrindiniai principai

Tokio tipo testavimas yra atliekamas pasikliaujant testuotojo sukaupta patirtimi testuojant analogiškas sistemas. Šio tipo testavimas ir jo sėkmė tiesiogiai priklauso nuo testuotojo kompetencijos ir turimų žinių, bei patirties. Praktika grindžiamas testavimas apima:

- Testavimas, kuris remiasi bandymu nuspėti klaidas. Testuotojas remiantis savo patirtimi susidaro tikėtinų klaidų sąrašą ir atlieka testavimą;
- Tiriamasis testavimas. Neformalia forma sukuriami, vykdomi ir pertestuojami testai, fiksuojami rezultatai;
- Kontroliniu sąrašu besiremiantis testavimas. Testavimo scenarijai sukuriami ir įvykdomi pagal sąrašą, kuris sukuriamas testuotojo ir papildomas kiekvieno pokyčio metu. Sąraše nurodytos gairės, ką ir kaip reikėtų ištestuoti.

Pagrindiniai praktika grindžiami testavimo metodai: tyrimu grindžiamas testavimas (angl. *Exploratory Testing*), Klaidų spėjimo metodas (angl. *Error Guessing*), testavimas remiantis sąrašu (angl. *Checklist-Based Testing*).

Privalumai ir trūkumai

Pagrindinis šių metodų privalumas: patikrinami specifiniai atvejai, kurių standartiniai testavimo metodai neapima.

Pagrindiniai šių metodų trūkumai: turi būti derinami su kitais testavimo metodais, norint pilnai ištestuoti sistemą, reikalingas daug patirties turintis testuotojas, sunku kontroliuoti testavimo apimtį ir efektyvumą.

1.4.4. Testavimo metodų palyginimas

Apibendrinantis testavimo metodų palyginimas pateikiamas lentelėje: 1 lentelė. Nurodytoje lentelėje testavimo technikos lyginamos pagal kriterijus: testavimo proceso struktūros aiškumas, automatizavimo galimybė, kiekybinis resursų poreikis, ištestuojamų scenarijų kiekis, galimybė vykdyti testavimą ankstyvoje PĮ kūrimo fazėje, žinių apie vidinę sistemos struktūrą poreikis, dalykinės srities specialisto poreikis.

1 lentelė. Esamų testavimo metodų grupių palyginimas

Palyginimo kriterijus	**Baltosios dėžės testavimas**	**Juodosios dėžes testavimas**	**Praktika grindžiamas testavimas**
Aiški testavimo proceso struktūra	+	+	-
Galimybė automatizuoti	+	-	+
Nereikalauja daug laiko/pinigų resursų	+-	+	+
Ištestuojamas maksimalus kiekis scenarijų	+-	-	+-
Galima vykdyti ankstyvoje PĮ kūrimo fazėje	-	+	-
Būtinos žinios apie vidinę sistemos struktūrą	+	-	-
Testavimui reikalingas srities specialistas	+	-	+-

Atlikus trijų technikų analizę, galima teigti, kad visos turi savų privalumų ir savų trūkumų. Nėra vienos technikos, kuri tiktų visiems projektams ir visiems nenumatytiems atvejams. Baltosios dėžės metodai tinkami, kai projektas nėra rizikingas. Jeigu kūrėjų komandoje yra patyręs testuotojas, tokiu atveju testavimą galima vykdyti remiantis praktika grindžiamais testavimo metodais. Jeigu sistemos reikalavimai turi būti ištestuoti pilnai, yra poreikis pradėti testavimą kuo anksčiau, tokiu atveju būtų tinkamas – juodosios dėžės testavimas. Esant poreikiui detaliai ištestuoti ne tik sistemos reikalavimus, bet ir patį sistemos kodą, jo struktūrą, reikėtų rinktis baltosios dėžės testavimą. Jeigu visi įvardyti projekto požymiai tarpusavyje persipina, idealu būtų naudoti skirtingoms technikoms priskiriamų metodų kombinacijas.

Bet kokiu atveju testuojant PĮ reikėtų testavimo metodą arba matodų kombinacijas pasirinkti atsižvelgiant į projekto aplinkybes, turimus resursus, dalykinę sritį ir kitus projekto parametrus.

1.5. Programinės įrangos testavimo proceso dalyvių analizė

Šiame skyriuje pateikiami su tyrimo objektu (testavimo procesu) susijusių naudotojų grupių aprašymai, remiantis asmenine patirtimi dirbant IT įmonėje ir [23] šaltinyje pateikta informacija.

Testavimo komandos dydis priklauso nuo projekto specifikos. Standartiškai į PĮ testavimo procesą bent dalinai tenka įsitraukti visai sistemos kūrimo komandai: analitikams, programuotojams, projektų vadovams, testuotojams, testuotojų vadovams. Į testavimo procesą įsitraukia praktiškai visa

komanda, nes stengiamasi testavimo procesą vykdyti lygiagrečiai su kitomis PĮ kūrimo veiklomis, siekiant identifikuoti klaidas kuo ankstesnėse PĮ kūrimo stadijose. Visa PĮ kūrimo komanda pasižymi bendrais bruožais:

- aukštas IT raštingumo lygis;
- kvalifikuoti IT srityje;
- dirba skirtinguose miestuose;
- įsitraukimas į kuriamos metodikos procesą yra būtinas.

Nepaisant to, kiekvienas komandos narys turi ir unikalių savybių, bei atsakomybių. Toliau tekste aprašomos visos komandos narių grupės, kurios bent dalinai yra susijusios su testavimo proceso vykdymu ir stebėjimu.

Analitikai

Pagrindinės atsakomybės: ruošia ir perduoda specifikaciją sistemos testuotojams ir programuotojams, teikia papildomą informaciją dėl būsimos sistemos funkcionalumo, komunikuoja su testuotojais ir programuotojais, atsižvelgia į testavimo ir programavimo metu pateiktas pastabas jau paruoštiems sistemos specifikacijos dokumentams.

Svarbios savybės: nėra testavimo srities ekspertai, tačiau gali pateikti svarbių ir profesionalių įžvalgų apie įmonėje vykdomus procesus. Tai vidutinio svarbumo naudotojai.

Šios grupės atstovams kuriama metodika ir jos realizacija aktuali, nes jie turi teikti informaciją testuotojams ir programuotojams apie baigtus darbus, o sukurtos metodikos realizacija automatiškai inicijuos testuotojų ir programuotojų darbus po specifikacijos paruošimo.

Programuotojai

Pagrindinės atsakomybės: programuoja sistemas, taiso testuotojų klaidas, registruoja klaidų būsenas, teikia pastabas specifikacijai, konsultuoja testuotojus.

Svarbios savybės: nėra testavimo srities ekspertai, bet pakankamai glaudžiai susiję su testavimo procesu. Tai vidutinio svarbumo naudotojai.

Šios grupės atstovams kuriama metodika ir jos realizacija aktuali, nes jie nuolatos laukia informacijos iš analitikų, kada bus galima pradėti darbus, bei turi informuoti testuotojus apie pataisytas klaidas. Šiuo atveju kuriamos metodikos realizacija automatiškai inicijuotų darbus.

Projektų vadovai

Pagrindinės atsakomybės: stebi testavimo procesą, fiksuoja proceso eigos rezultatus, tvirtina testavimo scenarijus ir ataskaitas.

Svarbios savybės: nėra testavimo srities ekspertai, tačiau itin svarbūs naudotojai, nes yra suinteresuoti gerinti testavimo procesą ir palengvinti proceso stebėjimą. Tai aukščiausio svarbumo naudotojai.

Šios grupės atstovams kuriama metodika ir jos realizacija aktuali, nes jiems reikia teikti progreso ataskaitas užsakovams, matyti vykdomo proceso silpnąsias vietas, priimti atitinkamus sprendimus.

Testuotojų poskyrio vadovai

Pagrindinės atsakomybės: ruošia testuotojams užduotis, koordinuoja testuotojus, padeda iškilus klausimams, stebi testuotojų darbus ir rezultatus. Tvirtina testavimo scenarijus ir ataskaitas.

Svarbios savybės: testavimo srities ekspertai, puikiai išmano testavimo procesą ir metodikas. Tai aukščiausio svarbumo naudotojai.

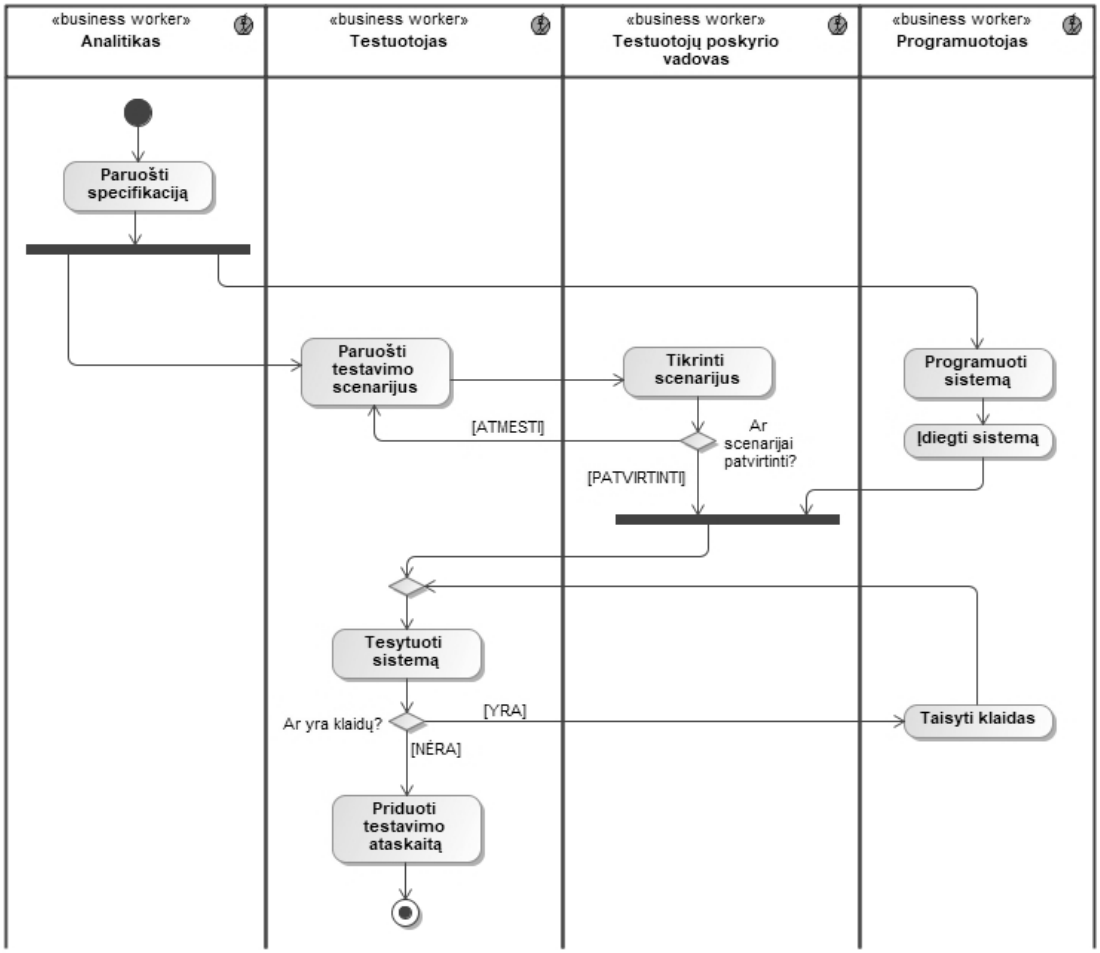
Testuotojai

Pagrindinės atsakomybės: atlieka sistemos ir dokumentacijos testavimą, pateikia analitikams pastabas, kuria testavimo planą, testavimo scenarijus, registruoja testavimo metu rastas klaidas, pildo ataskaitas. Pataisytas klaidas dar kartą ištestuoja, ruošia naudotojo vadovus, veda mokymus.

Svarbios savybės: užima pagrindinę rolę testavimo procese, testavimo sritį išmano gerai, geba taikyti testavimo metodikas.

Šios grupės atstovams kuriama metodika ir jos realizacija aktuali, nes jiems reikia patiems tarpusavyje derinti įvairias testavimo metodikas. Šie naudotojai taip pat nuolatos laukia informacijos iš analitikų, kada bus galima pradėti darbus.

Abstraktu procesas, atspindintis sąveikas tarp pagrindinių tyrimo objektų naudotojų, pateikiamas: 11 pav.



11 pav. Abstrakti sąveika tarp tyrimo objekto naudotojų

Identifikavus pagrindinius naudotojus ir jų veiklas, galima teigti, kad pagrindiniai tyrimo objekto naudotojai yra testuotojai, testuotojų vadovai ir projektų vadovai.

Visi identifikuoti naudotojai susiduria su tam tikromis problemomis iškylančiomis testavimo proceso metu. Eksperimentiniame tyrime dalyvaujančios įmonės testuotojų skyriuje buvo atlikta egzistuojančių problemų analizė ir aprašyti gauti rezultatai žemiau pateiktoje lentelėje.

2 lentelė. Naudotojų problemų aprašymas

Problema	Kaip viskas vyksta dabar?
1. Klientai priėmimo testavimo metu randa daug likusių klaidų.	Testuotojai testavimo scenarijus rašo pagal savo nuožiūrą, nesilaiko iš anksto numatyto proceso, nenaudoja rekomenduojamų testavimo metodų.
2. Projektų vadovai nežino testavimo būsenos konkrečiu laiko momentu.	Projektų vadovai skambina arba rašo el. laiškus komandos nariams, kai nori sužinoti testavimo proceso būseną.
3. Naujiems testuotojams sunku integruotis į komandą.	Prie komandos prisijungę nauji testuotojai pradeda darbus bandydami analizuoti ankstesniems pokyčiams paruoštus testavimo scenarijus.
4. Testuotojams nepraneša apie pasibaigusius analitikų darbus, testavimo darbai dažnai prasideda ne laiku.	Testuotojai sužino apie naują projektą tik tuomet, kai analitikas praneša projekto vadovui, kad reikia ištestuoti naują projektą.
5. PĮ kūrimo komandos nariai dažnai pamiršta kai kurias savo pareigas.	Šiuo metu komandos nariai bando atsikratyti kai kurių darbų, nes nėra aiškiai apibrėžto proceso, kuris atspindėtų, kas ir už ką yra atsakingas.
6. Testuotojams sudėtinga tarpusavyje derinti įvairius testavimo metodus.	Testuotojai pagal savo kompetencijas bando kombinuoti įvairius testavimo metodus tam tikrose PĮ kūrimo fazėse.
7. Komandos nariams sunku suvokti projekto būklę ir statusą.	Egzistuoja kelios komunikacinės priemonės: „Skype for business“, „Outlook“, „Open Project“, „Bugzilla“, „Mantis“. Informacija apie projektą yra įkelta į visas minėtas sistemas.

Tyrimo metu planuojama sukurti testavimo metodiką, kuri išnaudotų vykdomaisiais modeliais grindžiamų technologijų privalumus. Tokio tipo metodikos realizacija leistų projektų vadovams realiu laiku stebėti vykdomą testavimo procesą, testavimo komanda dirbtų pagal iš anksto numatytą procesą, kurį būtų galima pateikti ir naujiems komandos nariams, siekiant sumažinti naujokų mokymų periodo laikotarpį.

1.6. Programinės įrangos testavimo proceso valdymo sistemos

Testavimo procesui vykdyti ir stebėti įmonėse dažniausiai yra naudojami universalūs įrankiai, kuriuos sudėtinga pritaikyti prie konkrečioje įmonėje vykdomų veiklos procesų. Šiame skyriuje buvo pasirinkta aptarti tris populiariausius testavimo proceso vykdymui ir stebėjimui naudojamus įrankius: „Mantis“, „Bugzilla“, „Visual Studio Team System“.

Pirmoji pasirinkta IS, skirta testavimo proceso valdymui – „**Mantis**“ [24]. Tai atviro kodo sistema, skirta registruoti testavimo metu aptiktas klaidas, stebėti klaidų gyvavimo ciklą. Tai internete veikianti sistema. Šioje sistemoje galima registruoti klaidas, priskirti jas vykdytojams, filtruoti klaidų sąrašą pagal įvairius filtravimo parametrus. Sistema veikia pagal suprogramuotą algoritmą, jeigu norima pakeisti / pritaikyti sistemą konkrečiai įmonei, tam reikalingas programuotojas, kuris pakeistų iš esmės visą ar dalį sistemos algoritmo. Tokio tipo pakeitimai reikalauja nemažai resursų.

Antroji pasirinkta sistema – „**Bugzilla**“ [25]. Šis įrankis taip pat nemokamas, jame galima registruoti klaidas, jas priskirti vykdytojams, prisegti failus, rašyti komentarus, tačiau negalima valdyti pačio testavimo proceso, jo stebėti ar generuoti progreso ataskaitų. Ši sistema kaip ir „Mantis“ yra skirta tiesiog klaidų administravimui.

Trečioji pasirinkta programa – „**Visual Studio Team System**“ [26]. Ši programa yra mokama, apima ne tik testavimo procesą, bet ir visą projekto gyvavimo ciklą. Programa gali būti integruota su Visual Studio projektais, bet gali būti sudėtingai integruota su kitokio tipo sistemomis.

3 lentelė. Esamų sistemų palyginimas

Palyginimo kriterijus	Mantis	Bugzilla	Visual Studio Team System
Veikia pagal veiklos procesus	Ne	Ne	Taip
Atviro kodo	Taip	Taip	Ne
Procesų stebėjimas realiu laiku	Ne	Ne	Ne
Realizavimo technologija	Php	JavaScript	Java
El. laiškų siuntimas	Taip	Taip	Taip
Integracija su kitomis programavimo kalbomis	Ne	Ne	Ne
Galima kurti scenarijus	Ne	Ne	Taip
Galimybė generuoti ataskaitas, kurios atspindėtų testavimo proceso kokybę	Taip	Ne	Taip
Ar mokama?	Ne	Ne	Taip
Grafinė sąsaja geba prisitaikyti prie įrenginio ekrano dydžio (angl. *responsive design*)	Ne	Taip	Taip

Atlikus egzistuojančių įrankių analizę, galima daryti išvadą, jog norint nors vieną sistemą pritaikyti prie įmonėje vykdomo testavimo proceso, reikia atlikti programavimo darbus, o tai reikalauja daug resursų. Taip pat iš lyginamosios analizės galima išgryninti esmines funkcijas, kurias turi visos sistemos: el. laiškų siuntimas, klaidų registravimas. Taip pat galima išgryninti visų sistemų minusą – nei viena sistema nesuteikia galimybės stebėti testavimo proceso realiu laiku.

Lyginamoji egzistuojančių sistemų analizė padėjo išgryninti esmines testavimo proceso vykdymui reikalingas funkcijas, kurias privalės turėti ir kuriama sistema: klaidų registravimas, informacinių el. laiškų siuntimas. Taip pat išgrynintos funkcijos, kurios naujai sistemai sukurtų pranašumą lyginant su egzistuojančiomis: galimybė realiu laiku stebėti testavimo procesą, galimybė sistemą modifikuoti pagal organizacijos procesus.

1.7. Veiklos procesais grindžiamos platformos

Kuriant informacinių technologijų projektų valdymo sistemą, siekiama, kad sukurta sistema būtų pritaikoma unikalioms konkrečioje įmonėje vykdomoms veikloms. Tai įmanoma įgyvendinti, realizavus IS, veikiančią pagal įmonės veiklos procesą. Vienas iš galimų būdų realizuoti tokio tipo sistemą – BPM platformos naudojimas. Dauguma veiklos procesų modeliavimu grindžiamų platformų suteikia galimybę sukurti sistemą, kuri įgalina ne tik BPM modelius, bet ir sprendimo priėmimo lenteles (DMN), dėl šios priežastis nuspręsta kuriamą sistemą realizuoti naudojant BPM platformą.

Šiuo metu egzistuoja gausus pasirinkimas platformų, kurios suteikia galimybę realizuoti informacines sistemas, įgalinant konkrečius veiklos modelius. Tam, kad būtų lengviau išsirinkti platformą, būtina apžvelgti ir palyginti populiariausias tokio tipo technologijas. Populiariausių BPM platformų palyginimas (4 lentelė) pateikiamas pagal kuriamai sistemai aktualius ir lentelėje apibrėžtus kriterijus.

Pirmoji palyginimui pasirinkta platforma – „Activiti“ [27]. Ši platforma išsiskiria itin primityvia ir lengvai valdoma grafine sąsaja, detalia specifikacija, kurią paprasta suprasti neturint gilių programavimo žinių. Pagrindinis platformos minusas – nėra galimybės realiu laiku stebėti vykdomo proceso. Taip pat platformoje nėra DMN integracijos.

Antroji palyginimui pasirinkta platforma – „Bonitasoft“ [28]. Tai moderni, pritaikyta išmaniems telefonams platforma, kurioje galima modeliuoti procesus ir juos stebėti realiu laiku. Šios platformos išskirtinė funkcija – galimybė modifikuoti ir kurti prisitaikantį (angl. *responsive*) sistemos dizainą. Taip pat platforma suteikia ataskaitų generavimo funkcionalumą nemokamai ir be to, ataskaitas galima modifikuoti pagal savo poreikius naudojant specializuotus įrankius (pvz., „JasperReports iReport Designer“). Platformos minusas – nėra galimybės modeliuoti sprendimo priėmimo lentelių.

Trečioji palyginimui pasirinkta platforma – „Camunda BPM“ [29]. Pagrindinis platformos išskirtinumas – itin platus funkcionalumas. Naudojant šią platformą realizuojama sistema, kurioje galima realiu laiku stebėti vykdomus procesus, sumodeliuotus naudojant veiklos procesų modeliavimo kalbą ir sprendimo priėmimo lenteles. Be to, šios platformos funkcionalumą galima praplėsti įdiegiant įskiepius ir tai nereikalaus papildomų programavimo žinių. Šią platformą gali naudoti „java”, „php”, „c#” programavimo kalbas išmanantys programuotojai, nes platformoje veikiantis BPM variklis gali būti naudojamas per REST ryšio kanalą. Tai suteikia galimybę variklį naudoti sistemoms, kurios nėra suprogramuotos Java programavimo kalba.

4 lentelė. Galimų realizavimo technologijų palyginimas

Palyginimo kriterijus	**Activiti**	**Bonitasoft**	**Camunda**
Integracija su kitomis programavimo kalbomis	Ne	Ne	Taip
Platforma atviro kodo	Taip	Taip	Taip
Ar mokama?	Ne	Ne	Ne
Ataskaitų generavimo funkcionalumas	Ne	Taip	Taip, bet mokamas
Proceso stebėjimas realiu laiku	Ne	Taip	Taip
DMN (sprendimo priėmimo lentelės)	Ne	Ne	Taip
Realizavimo technologija	Java	Java	Java

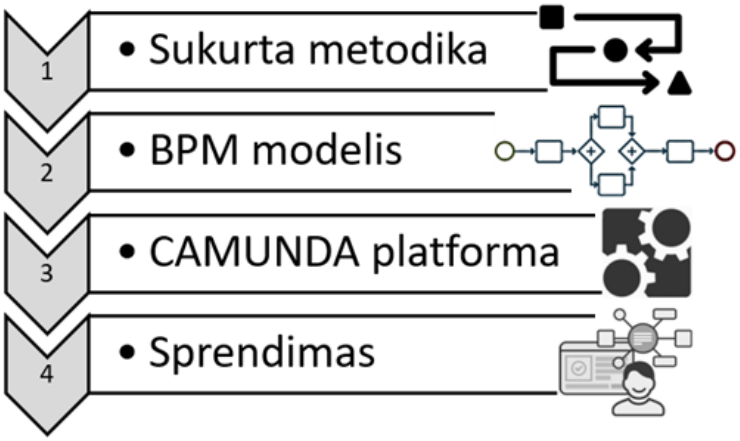
Atlikus galimų realizavimo technologijų lyginamąją analizę, galima daryti išvadą, kad dauguma populiarių BPM platformų yra atviro kodo. Pagrindinis skirtumas – tik viena iš analizuotų platformų gali būti naudojama sistemų, kurios suprogramuotos ne Java kalba ir turi DMN lentelių integraciją – tai „Camunda BPM“ platforma. Būtent ši platforma bus naudojama kuriamos sistemos realizacijoje.

1.8. Tyrime kuriamos programinės įrangos testavimo metodikos koncepcija

Numatytas problemos sprendimas – programinės įrangos testavimo metodika, išnaudojanti procesų valdymu grindžiamų sistemų teikiamus privalumus. Šis sprendimas bus įgyvendintas naudojant „Camunda BPM“ [29] platformą, kurioje galima sukurti sistemą, veikiančią pagal vykdomąjį veiklos

proceso modelį. Darbo apimtyje planuojama sukurti testavimo metodiką ir sumodeliuoti vykdomąjį BPM modelį testavimo procesui, kuris būtų grindžiamas sukurta testavimo metodika. Sukurtas vykdomasis modelis bus įkeltas į „Camunda BPM“ platformą, kurioje bus galima vizualiai pademonstruoti sukurtą programinės įrangos testavimo metodiką ir pagal ją veikiantį testavimo vykdymo ir valdymo procesą.

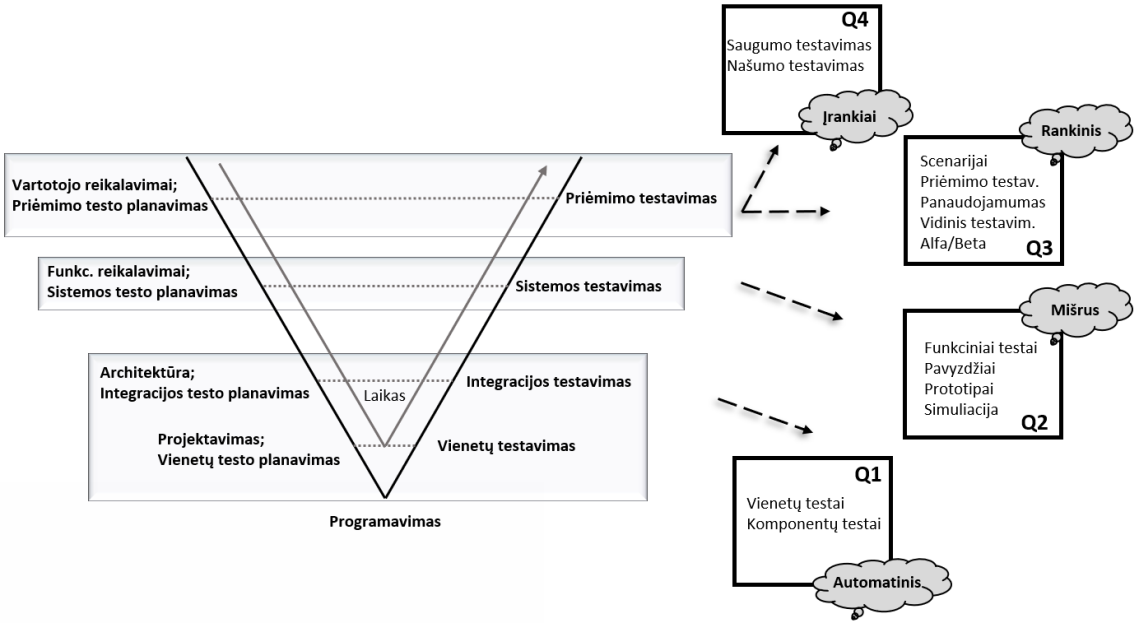
Abstrakti sprendimo schema (12 pav.) atspindi pagrindines veiklas, kurios bus vykdomos tiriamojo darbo apimtyje.



12 pav. Abstrakti sprendimo schema

Tyrimo problemos sprendimo schema (12 pav.) atspindi pagrindinius darbo etapus:
1. metodikos sukūrimas;
2. sukurta testavimo metodika grindžiamo BPM modelio sukūrimas;
3. BPM modelio įkėlimas į „Camunda BPM“ platformą ir sistemos konfigūravimas;
4. galutinio sprendimo pateikimas naudotojui.

Tyrimo darbas pradedamas nuo metodikos sukūrimo. Išanalizavus tris programinės įrangos kūrimo metodus, nuspręsta daugiausiai dėmesio skirti nuosekliai vykdomiems, „V“ modeliu grindžiamiems projektams ir iteracijomis grindžiamiems *Agile* tipo projektams. *Agile* tipo projektuose testavimo etape dažnai yra naudojami „Agile kvadratai“, todėl suderinamumas analizuojamas būtent tarp „V“ modelio ir *Agile* kvadratų metodo (13 pav.).



13 pav. „V“ modelio ir *Agile* kvadratų testavimo veiklų suderinamumas

Atlikus testavimo veiklų palyginimą *Agile* kvadratuose ir „V“ modelyje (13 pav.), matoma, kad žvelgiant į skirtingas metodikas iš testavimo perspektyvos, metodikos tarpusavyje yra suderinamos. Dėl šios priežasties tiriamajame darbe nuspęsta kurti metodiką, kuri būtų tinkama tiek tradiciniams metodams, tiek moderniems, *Agile* tipo. Kuriama testavimo metodika bus orientuota į šešis pagrindinius programinės įrangos kūrimo etapus: inicijavimą, analizę, projektavimą, kūrimą, testavimą ir priėmimą. Kiekvienam iš etapų išskiriamos konkrečios testavimo veiklos.

1. Inicijavimo etapas. Šio etapo metu yra peržiūrimas projekto vadovo inicijuotas projektas, sudaroma abstrakti testavimo strategija, pasirenkami testavimo metodai, kurie bus naudojami testuojant programinę įrangą. Taip pat šiame etape identifikuojama būsima testavimo komanda.
2. Analizės etapas. Šio etapo metu kuriami testavimo atvejai, skirti klientui priimti, todėl jie rašomi atitinkame abstrakcijos lygyje, kurį supranta tiek kūrėjų komanda, tiek klientas. Kuriama metodika apims testavimo veiklas:
   2.1. sudaromas testavimo planas;
   2.2. įvertinamos testavimo veiklos;
   2.3. atliekamas statinis testavimas, peržiūrimi ir įvertinami reikalavimai, išsiunčiamos pastabos;
   2.4. sudaromi pradiniai priėmimo testavimo scenarijai.
3. Projektavimo etapas. Šio etapo metu kuriama metodika apims testavimo veiklas (gali būti vykdomos ne visos veiklos, bus atsižvelgiama į projekto specifiką):
   3.1. paruošiami funkciniai testavimo scenarijai;
   3.2. paruošiami testavimo scenarijai, skirti veiklos proceso testavimui;
   3.3. pasiruošiama nefunkcinių reikalavimų testavimui:
      3.3.1. tinkamumo testavimas;
      3.3.2. greitaveikos testavimas;
      3.3.3. našumo testavimas;
      3.3.4. saugumo testavimas.
   3.4. pasiruošiama „Alfa“ ir „Beta“ testavimui.
4. Kūrimo etapas. Šio etapo metu kuriami testavimo atvejai skirti vidiniam ir pradiniam programuotojų testavimui. Jie rašomi kūrėjų komandai suprantama kalba, kuri nėra skirta klientui.

Šiame etape testavimas vykdomas automatiniu būdu. Etapo metu kuriama metodika apims testavimo veiklas:
   4.1. statinės kodo analizės vykdymas;
   4.2. vienetų testavimo atvejų sudarymas;
   4.3. komponentų testavimo atvejų sudarymas;
   4.4. testavimui reikalingų duomenų paruošimas;
   4.5. vienetų testavimo vykdymas;
5. Testavimo etapas. Vykdomi testavimo atvejai aprašyti ankstesnėse fazėse:
   5.1. sistemos funkcionalumo testavimas;
   5.2. veiklos proceso testavimas;
   5.3. „Alfa“, „Beta“ testavimas;
   5.4. klaidų fiksavimas;
   5.5. galutinės testavimo ataskaitos paruošimas.
6. Priėmimo etapas.
   6.1. nefunkcinių reikalavimų testavimas;
   6.2. priėmimo testavimas;
   6.3. klaidų fiksavimas;
   6.4. priėmimo rezultatų fiksavimas;
   6.5. įvykdyto testavimo proceso įvertinimas.

Apibrėžus pagrindinius kuriamos metodikos etapus ir vykdomas veiklas bus suprojektuota detali testavimo metodika: 2 Programinės įrangos testavimo metodikos sprendimas.

1.9. Analizės išvados

Atlikus tyrimo objekto analizę, buvo surinkta reikalinga informacija ir priimtos atitinkamos išvados:
1. Identifikavus tyrimo objektą, sritį ir problemą, buvo apibrėžtos aiškios tyrimo ribos ir kryptis. Pagal surinktus duomenis nuspręstą atlikti testavimo proceso vykdymo ir valdymo tyrimą ir paruošti testavimo metodiką, kuri prisidėtų prie testavimo proceso gerinimo.
2. Išanalizavus tyrimo objektą, buvo identifikuoti pagrindiniai testavimo proceso etapai, apibrėžta tyrimo sritis ir problema, kurią bus bandoma išspręsti tiriamojo darbo apimtyje. Pagal surinktus duomenis, priimta išvada, kad tyrimo objektas – testavimo metodika – turi būti glaudžiai susijusi su programinės įrangos kūrimo metodais, bei jų etapais, testavimo veiklos turi apimti visus PĮ gyvavimo ciklo etapus.
3. Identifikavus pagrindinius tyrimo objekto naudotojus, išsiaiškinta, kad į testavimo procesą įsitraukia praktiškai visa PĮ kūrimo komanda, dėl šios priežasties bus atsižvelgta į visų naudotojų poreikius. Tiriamojo darbo apimtyje bus siekiama, kad kuriama testavimo metodika išspręstų pagrindines problemas su kuriomis susiduria tiek testuotojai, tiek projektų vadovai testavimo metu.
4. Atlikus esamų testavimo metodų analizę ir juos palyginus, nuspręsta perimti gerąsias savybes iš egzistuojančių metodų ir pritaikyti jas kuriamoje metodikoje. Taip pat priimtas sprendimas kuriamos metodikos apimtyje maksimaliai išnaudoti procesų valdymu grindžiamų sistemų teikiamus privalumus. Tokio tipo sistemos pritaikymas, suteiktų galimybę realiu laiku stebėti testavimo proceso vykdymą.
5. Atlikus galimų realizavimo technologijų analizę ir palyginus egzistuojančias analogiškas sistemas, buvo priimti sistemos realizavimo sprendimai: kuriama sistema turi naudoti „Camunda

BPM“ platformą, kurioje bus naudojami ne tik BPM vykdomieji modeliai, bet ir sprendimų priėmimo lenteles.

6. Analizės etape apibrėžtas viso tyrimo tikslas – pagerinti programinės įrangos testavimo vykdymo ir valdymo procesus. Tikslui įgyvendinti buvo suformuoti uždaviniai, kurie turės būti įvykdyti tyrimo metu.
7. Tyrimo analizės pabaigoje atsižvelgiant į analizės metu surinktus duomenis, buvo apibrėžtas siekiamas problemos sprendimas, kuriuo remiantis bus atliekami tolimesni tyrimo darbai.

2. Programinės įrangos testavimo metodikos sprendimas

2.1. Reikalavimai keliami programinės įrangos testavimo metodikai

Atlikus detalią programinės įrangos testavimo proceso analizę, buvo surinkta naudinga informacija, bei rekomendacijos testavimo proceso kokybei gerinti. Surinkus visą šią informaciją suformuoti reikalavimai kuriamai metodikai:

1. kiekviena projekto realizavimo veikla turi turėti atitinkamą testavimo veiklą;
2. testavimo veiklos turi būti vykdomos lygiagrečiai su kitomis projekto vykdymo veiklomis;
3. kiekvienas testavimo etapas turi turėti apibrėžtą tikslą;
4. testavimo metodikoje turi atsispindėti, jog testuotojas yra įtraukiamas ne tik į testavimo, bet ir į reikalavimų išgryninimo, projekto apimties vertinimo veiklas;
5. testavimo metodika turi būti grindžiama standartinėmis programinės įrangos gyvavimo ciklo fazėmis: analizė, projektavimas, programavimas, testavimas, priėmimas;
6. testavimo metodikoje turi būti įtraukta bent viena testavimo proceso kokybės vertinimo veikla;
7. testavimo metodika turi išnaudoti BPM technologijos teikiamus privalumus: proceso stebėjimas realiu laiku, darbas pagal iš anksto numatytą apibrėžtą procesą, nesudėtingas proceso redagavimas pagal pasikeitusias aplinkybes.

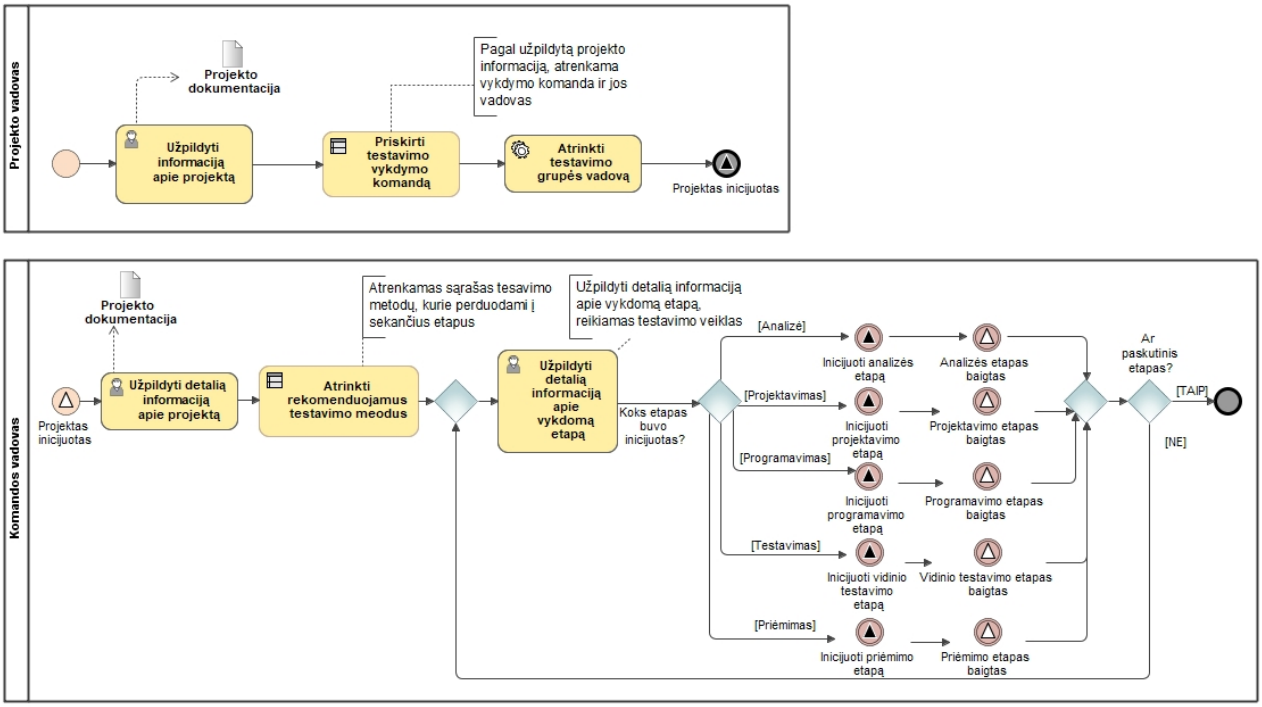
Sukūrus metodiką būtina vizualiai ją pademonstruoti naudojant eksperimentinę sistemą ir įvertinti, ar metodika tenkina visus išvardintus reikalavimus.

2.2. Programinės įrangos testavimo metodikos aprašas ir siekiamas proceso modelis

Tiriamojo darbo analizės metu buvo nustatyta, jog tradiciniai, „V“ modeliu grindžiami, projektai ir modernūs, *Agile* tipo projektai gali būti tarpusavyje suderinami, jeigu testavimo procesą projektuose išskaidome pagal standartinio programinės įrangos gyvavimo ciklo fazes (analizę, projektavimą, programavimą, testavimą ir priėmimą). Analizės metu surinkus reikiamą informaciją buvo apibrėžtas kuriamos metodikos koncepcinis modelis (1.8), kuris identifikuoja pagrindines testavimo veiklas ir rekomenduojamų naudoti testavimo metodų kombinacijas. Šiame etape detalizuojamas analizės metu sudarytas programinės įrangos testavimo metodikos prototipas.

Toliau skyriuje pateikiama detali kuriamos programinės įrangos testavimo metodikos informacija. Informacija grupuojama pagal projekto etapus, pateikiant suprojektuotą etapo procesą, bei detalų proceso aprašymą. Skyriuje aprašyti procesų modeliai transformuojami į vykdomuosius BPM modelius, kurie bus diegiami į „Camunda BPM“ platformą (žr. 3.4 Vykdomi programinės įrangos testavimo metodikos eksperimentinės sistemos procesų modeliai).

Testavimo procesas prasideda nuo projekto inicijavimo testuotojų skyriuje (14 pav.).

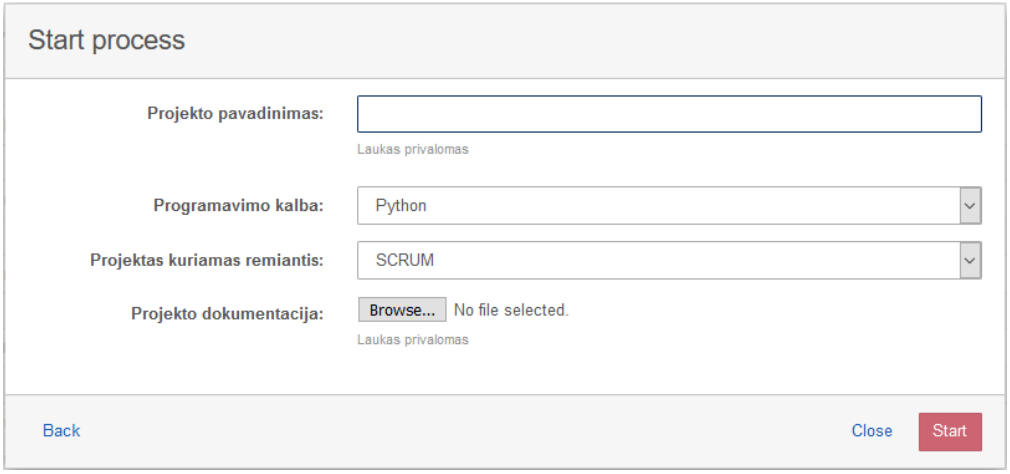


14 pav. Projekto etapo inicijavimo testuotojų skyriuje procesai

Projekto inicijavimo etapas

Šio etapo metu atliekamos pradinės veiklos, kurios yra būtinos projekto inicijavimui testuotojų skyriuje. Etapo tikslas – peržiūrėti projektų vadovo inicijuoto projekto reikalavimus, sudaryti projekto testavimo strategiją, nustatyti projekto etapus, identifikuoti testuotojų komandą, įvertinti projekto aplinkybes, kuriomis remiantis būtų galima nuspręsti, kokius testavimo metodus reikėtų naudoti. Etapas apima veiklas:

1.1. projektų vadovas inicijuoja projektą testavimo skyriuje, užpildo pagrindinę informaciją apie projektą: projekto pavadinimą, kūrimo metodą, programavimo kalbą, pateikia projekto dokumentaciją (15 pav.);



15 pav. Proceso inicijavimo testuotojų skyriuje realizacijos langas

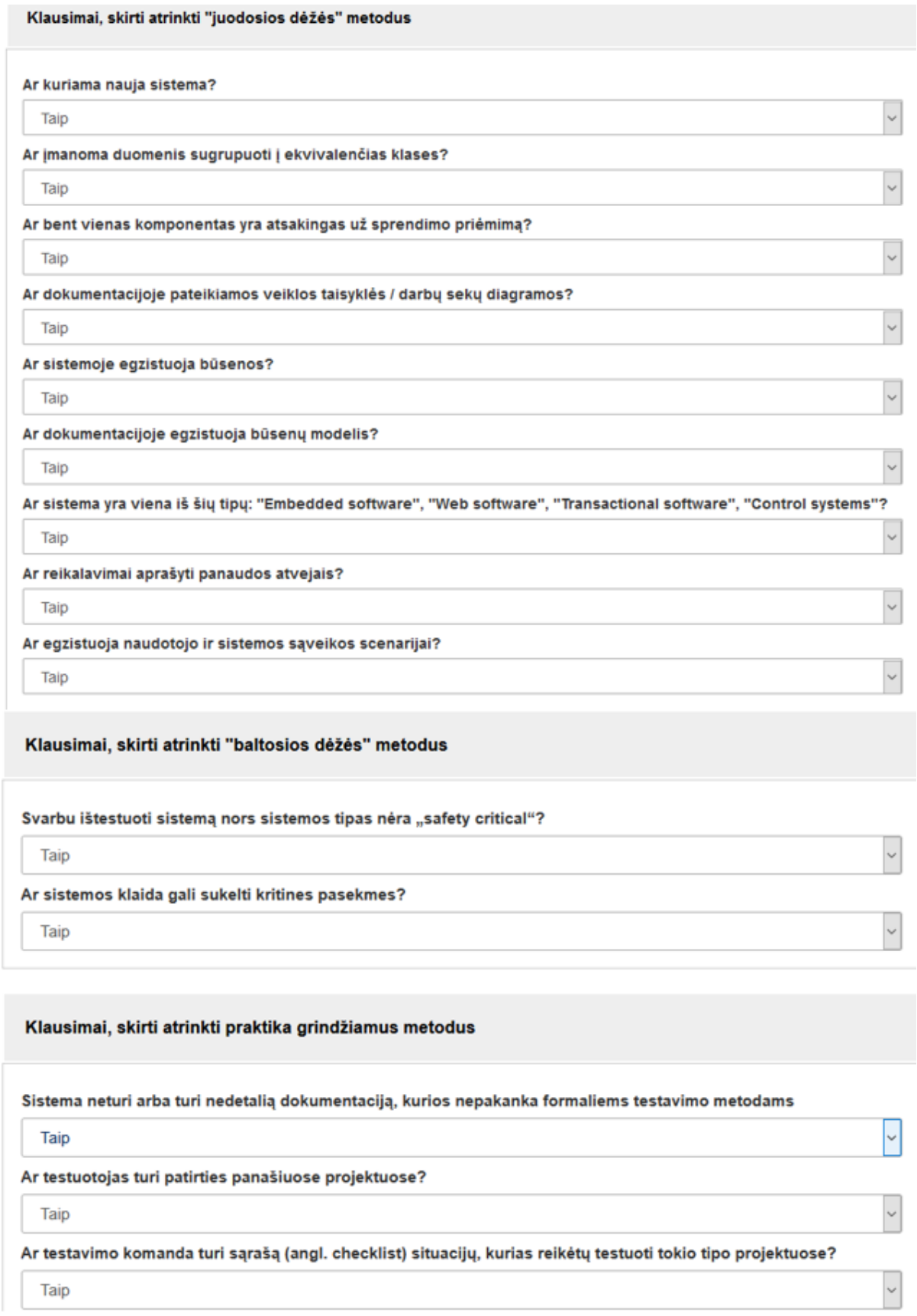
1.2. atrenkama testuotojų komanda, kuri gali vykdyti projektą. Testuotojų komanda atrenkama atsižvelgiant į projektų vadovo suvestus duomenis: kūrimo metodą, programavimo kalbą. Atrinkimas vykdomas naudojant sprendimo priėmimo lentelę (16 pav.), kurioje yra aprašytos

taisyklės, nurodančios prie kokių sąlygų, kurią testuotojų komandą reikia priskirti. Atrenkant komandą, taip pat turi būti surandamas ir komandos vadovas, kuriam priskiriama sekanti 1.3 užduotis;

Įeinantys duomenys		**Išvedami duomenys**	
Programavimo kalba	**Projekto vykdymo metodas**	**Komanda**	**Vadovas**
PHP	RUP	R16 komanda	Janina Janulytė
PHP	SRUM	A14 komanda	Janina Janulytė
PHP	„V“ modelis	V13 komanda	Jonas Jonaitis
Java	RUP	R10 komanda	Tadas Tadaitis
Java	SRUM	A11 komanda	Lina Linaitė
Java	„V“ modelis	V12 komanda	Petras Petraitis
...	...	...	...

16 pav. Testuotojų komandos priskyrimo sprendimo priėmimo lentelė

1.3. įvertinamas inicijuotas projektas. Testuotojų grupės vadovas perskaito dokumentaciją ir atsako į klausimus (17 pav.), kurie skirti surinkti visą reikiamą informaciją, naudojamą sudarant rekomenduojamų testavimo metodų (baltosios dėžės, juodosios dėžės, praktika grindžiamų) sąrašą;

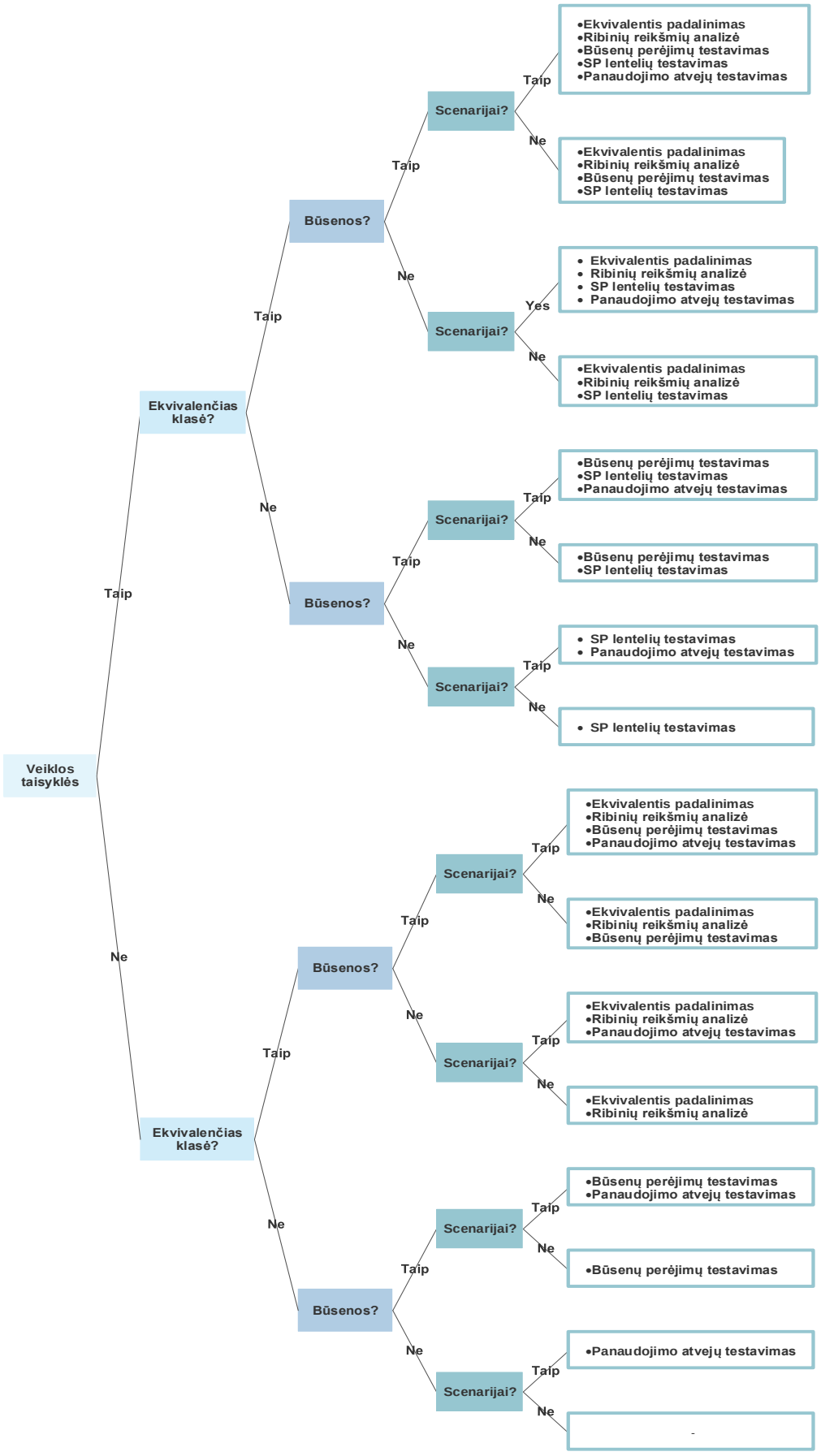


17 pav. Klausimai, skirti atrinkti rekomenduojamus testavimo metodus

1.4. sudaromas rekomenduojamų naudoti metodų sąrašas. Sistema naudojant sprendimo priėmimo lenteles sudaro rekomenduojamų testavimo metodų sąrašą. Testavimo metodai

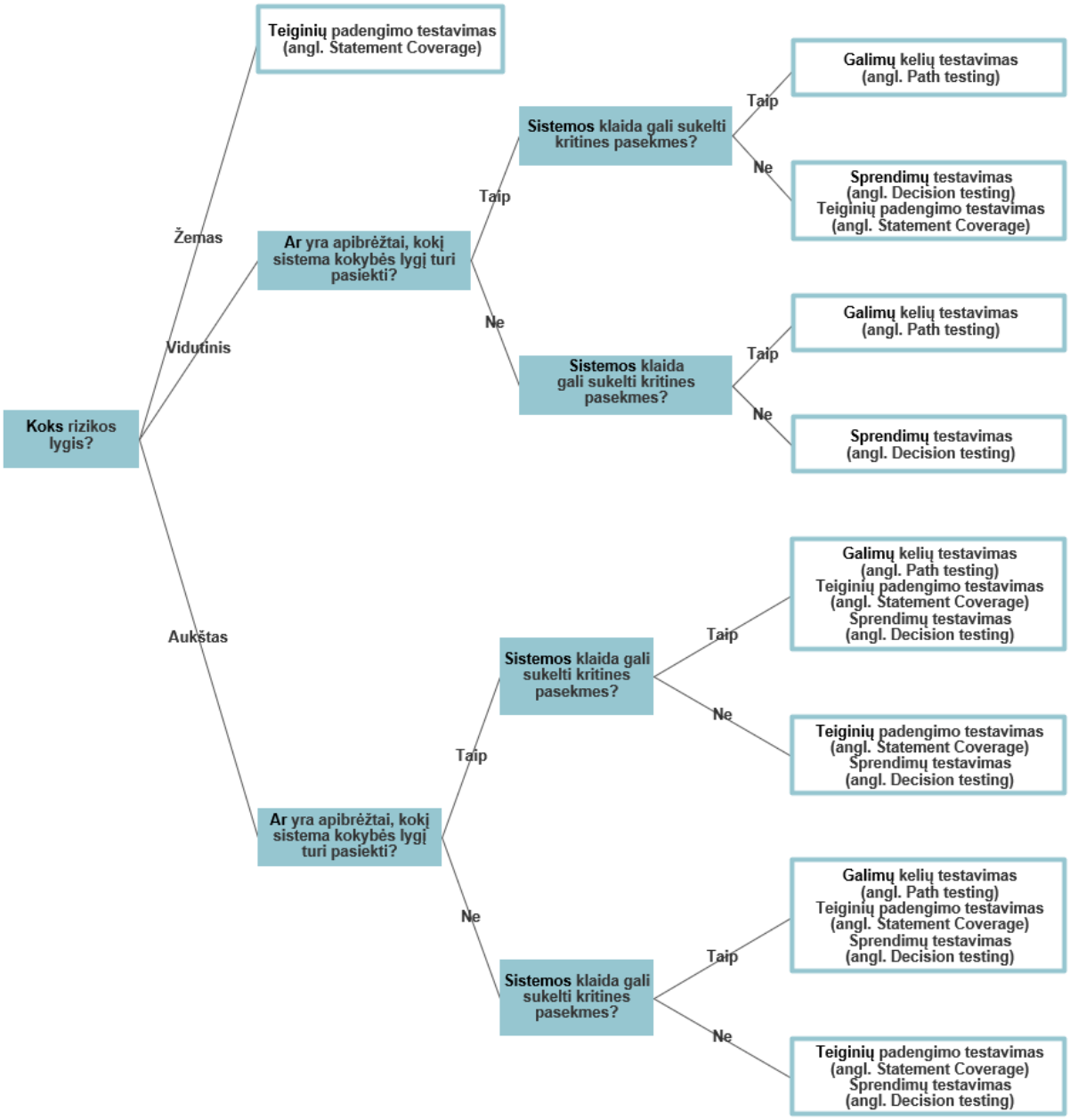
atrenkami pagal sprendimo priėmimo lentelėse aprašytas taisykles ir 1.3 žingsnyje užpildytą informaciją. Realizacijoje sprendimo priėmimo lentelės kuriamos pagal suprojektuotus sprendimų priėmimo medžius:

1.4.1. medis, skirtas atrinkti juodosios dėžės testavimo metodus (18 pav.). Juodosios dėžės metodų atrinkimo medis įvertina testuotojų grupės vadovo suvestus duomenis ir sudaro rekomenduojamų naudoti juodosios dėžės testavimo metodų sąrašą. Šis sprendimo priėmimo medis realizacijoje turi būti paverčiamas į DMN lentelę, kurios visi galimi įvesties ir išvesties duomenys pateikiami prieduose (1 priedas). Sprendimo priėmimo medis buvo sudaromas remiantis šaltiniais: [7], [30];



18 pav. Juodosios dėžės metodų atrinkimo medis

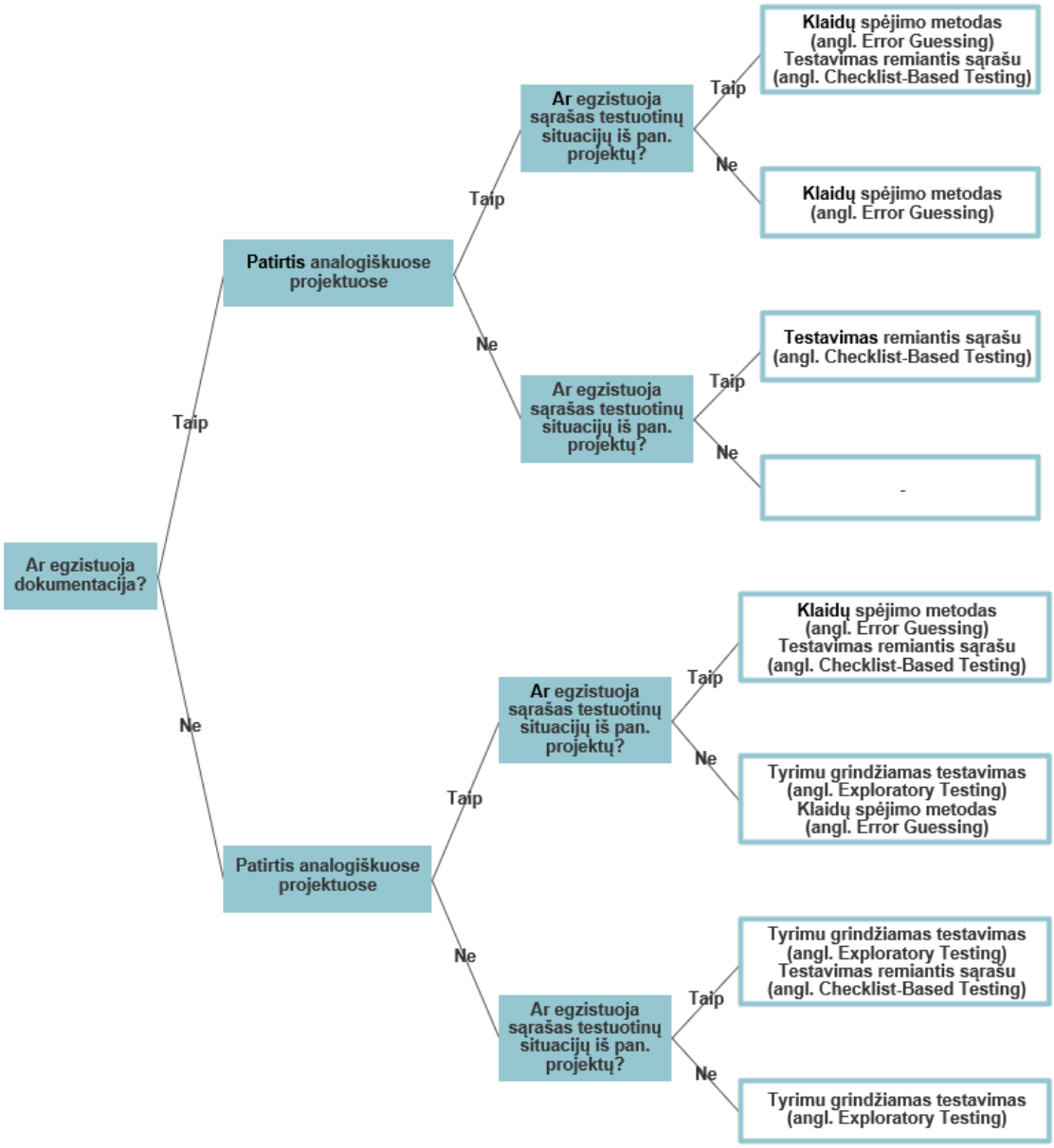
1.4.2. medis, skirtas atrinkti baltosios dėžės testavimo metodus (19 pav.). Baltosios dėžės metodų atrinkimo medis įvertina testuotojų grupės vadovo suvestus duomenis ir sudaro rekomenduojamų naudoti baltosios dėžės testavimo metodų sąrašą. Kuriama metodika neatsižvelgia į turimus įrankius. testuotojo kompetenciją ir visais atvejais rekomenduoja naudoti bent vieną baltosios dėžės testavimo metodą. Testuotojo kompetencijos, turimų resursų (įrankių, laiko) vertinimas paliekamas testuotojų grupės vadovui. Šis sprendimo priėmimo medis realizacijoje turi būti paverčiamas į DMN lentelę, kurios visi galimi įvesties ir išvesties duomenys pateikiami prieduose (1 priedas). Sprendimo priėmimo medis buvo sudaromas remiantis šaltiniais: [7], [30];



19 pav. Baltosios dėžės metodų atrinkimo medis

1.4.3. medis, skirtas atrinkti praktika grindžiamus testavimo metodus (20 pav.). Praktika grindžiamų metodų atrinkimo medis įvertina testuotojų grupės vadovo suvestus duomenis ir sudaro rekomenduojamų naudoti praktika grindžiamų testavimo metodų

sąrašą. Šis sprendimo priėmimo medis realizacijoje turi būti paverčiamas į DMN lentelę, kurios visi galimi įvesties ir išvesties duomenys pateikiami prieduose (1 priedas). Sprendimo priėmimo medis buvo sudaromas remiantis šaltiniais: [7], [30].



20 pav. Praktika grindžiamų metodų atrinkimo medis

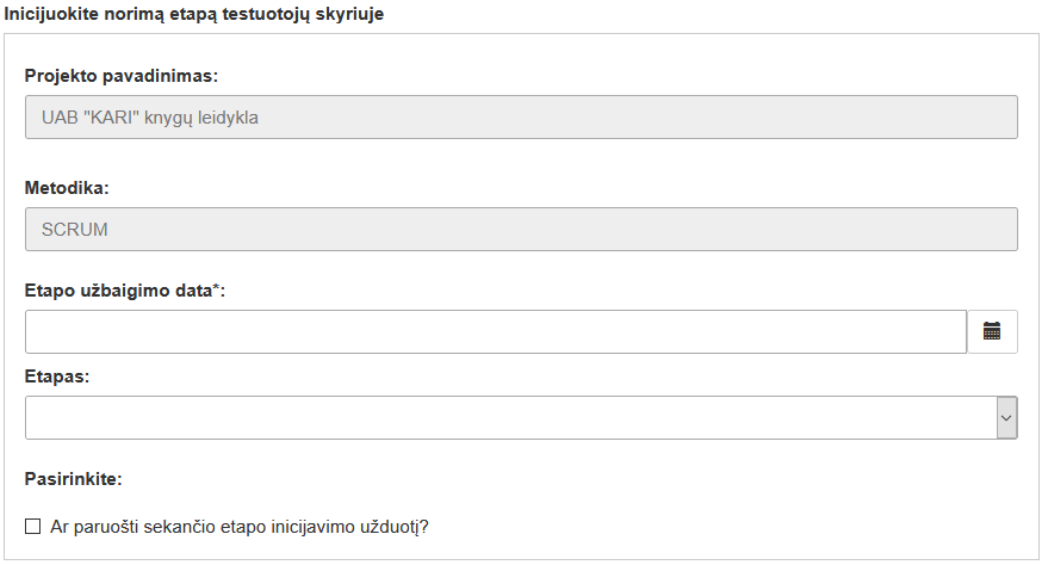
1.5. užpildoma projekto etapo inicijavimo informacija. Testuotojų grupės vadovas užpildo informaciją, reikalingą testavimo etapo inicijavimui: pasirenkamas norimas etapas, nurodomas etapo vykdymo terminas (21 pav.). Testavimo etapas turi būti pateikiamas atsižvelgiant į projekto vykdymo metodiką:

1.5.1. „Scrum“ – galimų etapų sąrašas: „Nulinis sprintas, pirmojo sprinto planavimas“, „Pasiruošimas pirmojo sprinto vykdymui (reikalavimų lygyje)“, „Pasiruošimas pirmojo sprinto vykdymui (kodo lygyje)“, „Sprinto vykdymas“, „Užbaigimas, priėmimas (angl. *end game*)“. Etapai išskirti remiantis [14] šaltiniu ir atliekant suderinamumo analizę su kitais tyrime naudojamais metodais („RUP“, „V“ modelis);

1.5.2. „V“ modelis – galimų etapų sąrašas: „Analizė“, „Projektavimas“, „Programavimas“, „Testavimas“, „Užbaigimas, priėmimas“. Etapai išskirti remiantis [7] šaltiniu ir atliekant

metodo suderinamumo analizę su kitais tyrime naudojamais metodais („Scrum“, „V“ modelis);

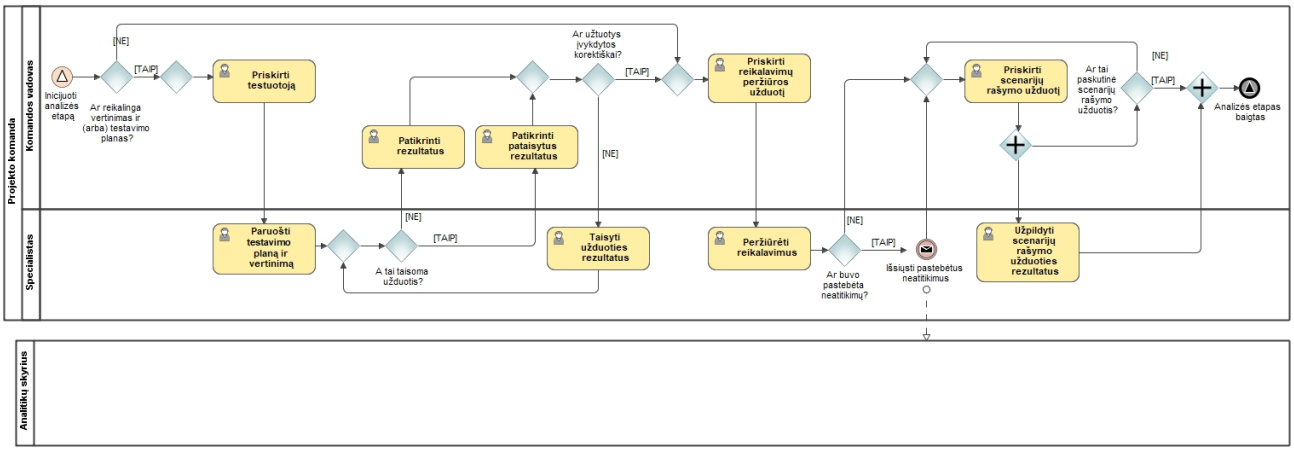
1.5.3. „RUP“ – galimų etapų sąrašas: „Inicijavimas“, „Vystymas“, „Kūrimas (programavimo lygyje)“, „Kūrimas (testavimo lygyje)“, „Perėjimas“. Etapai išskirti remiantis [31] šaltiniu ir atliekant metodo suderinamumo analizę su kitais tyrime naudojamais metodais („Scrum“, „V“ modelis).



21 pav. Testavimo etapo inicijavimo prototipo forma

1.6. Inicijuojamas testavimo etapas. Atsižvelgiant į testuotojų vadovo suvestus duomenis 1.5 žingsnyje, sistema automatiškai inicijuoja reikiamą etapą.

Sekantis pagal procesų inicijavimo eiliškumą gali būti inicijuotas analizės etapo procesas (22 pav.).



22 pav. Testavimo veiklų procesas vykdant analizės etapą

Analizės etapas

Šiame etape itin svarbu kruopščiai peržiūrėti sistemai keliamus reikalavimus, nustatyti, ar reikalavimai vienas kitam neprieštarauja, ar yra pakankamai detaliai aprašyti. Taip pat šio etapo metu yra kuriami pradiniai testavimo atvejai skirti klientui priimti, todėl jie rašomi atitinkamame

abstrakcijos lygyje, kurį supranta tiek kūrėjų komanda, tiek klientas. Kuriama metodika apims analizės etapo veiklas:

1.1. identifikuojamas testuotojas, atsakingas už testavimo scenarijų rašymo, reikalavimų peržiūros, vertinimo ir (arba) testavimo plano paruošimo užduotis (rekomenduojama, jog planą ir vertinimą ruoštų tas pats testuotojas, testavimo scenarijus gali ruošti ir kitas testuotojas, tam turi būti realizuota atskira užduoties priskyrimo veikla);

1.2. detaliai peržiūrimi ir patikrinami reikalavimai (vykdoma statinė reikalavimų analizė). Jeigu nustatoma, kad reikalavimai ne pakankamai detaliai aprašyti, vienas kitam prieštarauja arba turi kitų trūkumų, tuomet testuotojas pateikia pastabas projekto analitikui (išsiunčiamas el. laiškas);

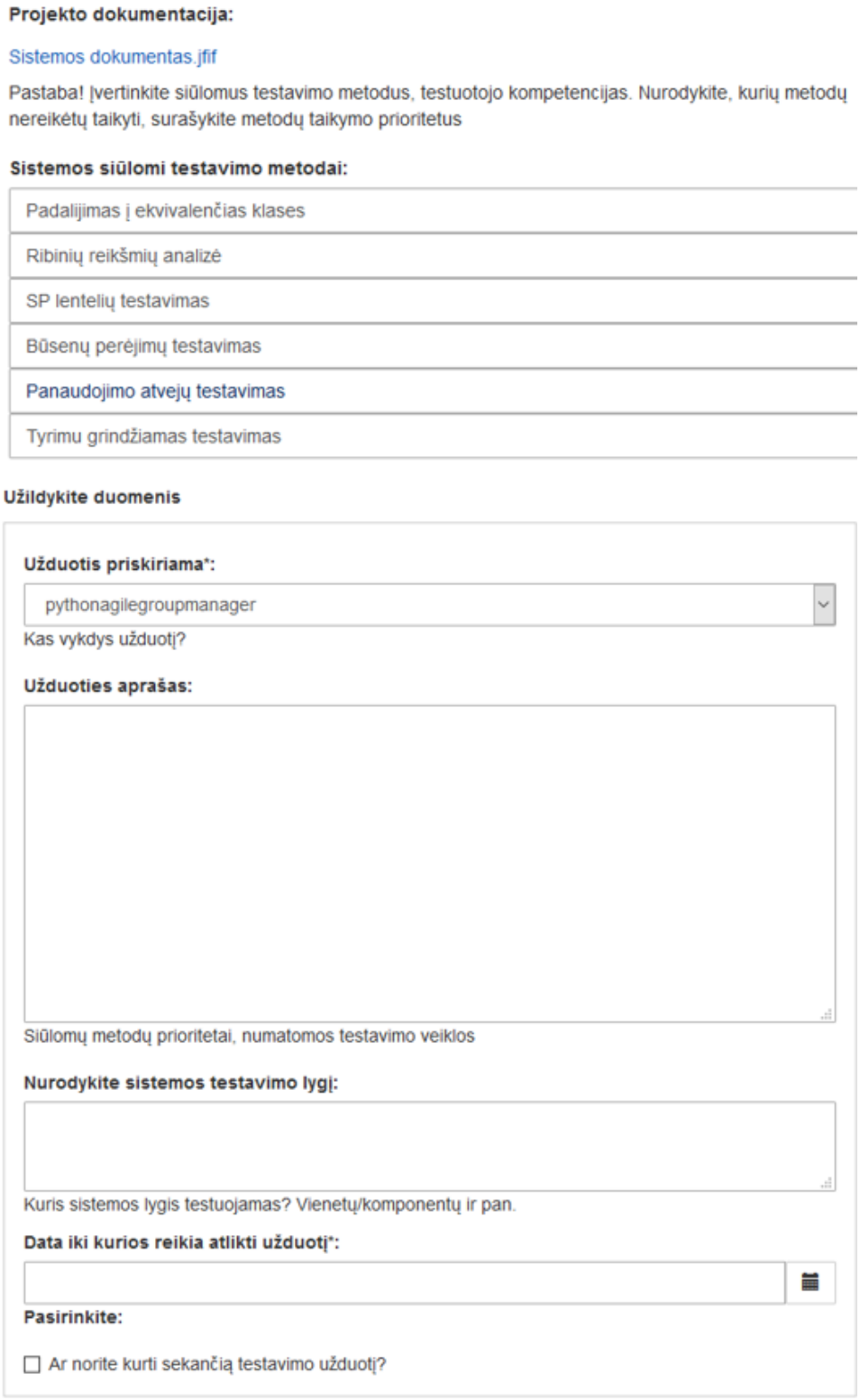
1.3. įvertinama darbų apimtis, paruošiamas vertinimas ir planas. Dokumentai derinami su komandos vadovu. Jeigu komandos vadovas turi pastabų, tuomet dokumentai grąžinami taisymui;

1.4. komandos vadovas įvertina projektą, peržiūri sistemos rekomenduojamus testavimo metodus, užpildo informaciją, kuri yra svarbi testavimo užduoties priskyrimui (23 pav.):

1.4.1. nurodomas testavimo lygis, kurį reikia testuoti. Detaliai aprašoma testavimo užduotis, nurodoma, kokio tipo testavimui užduotis yra skirta;

1.4.2. įvertinami turimi resursai, atsakingo testuotojo kompetencijos, surašomi prioritetai sistemos siūlomiems testavimo metodams (sistemos pateikiamas rekomenduojamų metodų sąrašas redaguojamas, jame turi būti galima prie kiekvieno metodo nurodyti reikiamą informaciją);

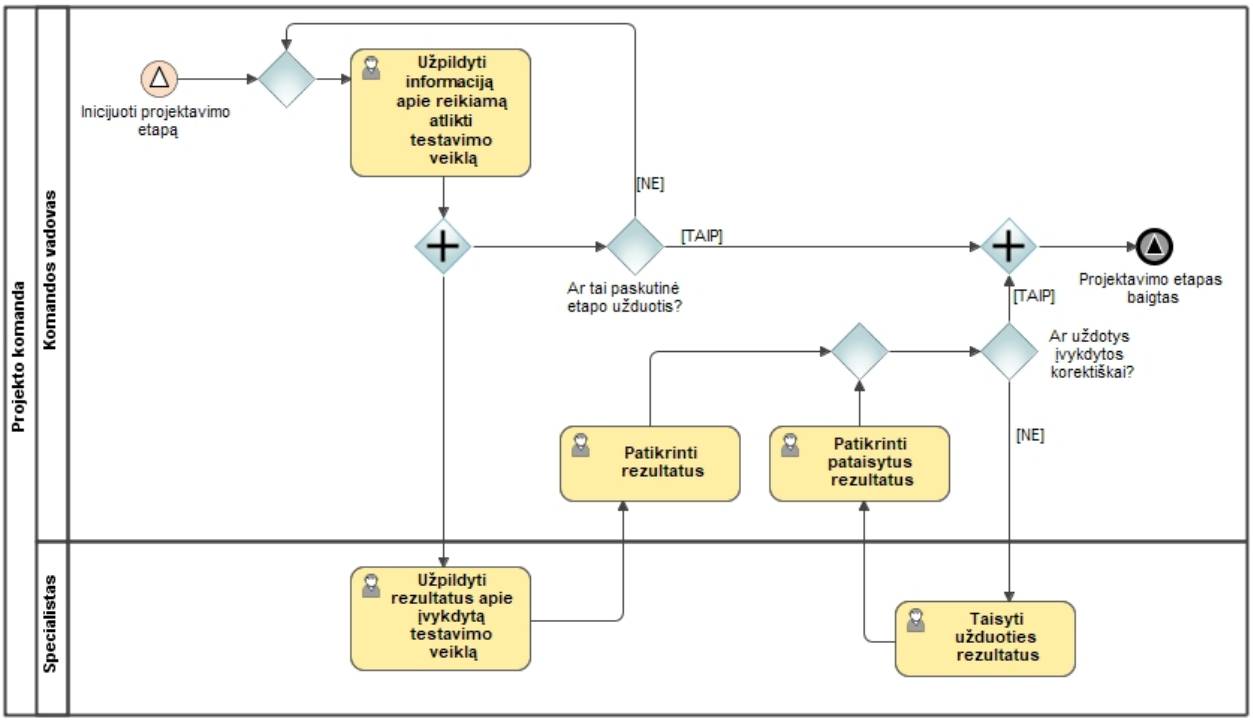
1.4.3. nurodomas darbų atlikimo terminas.



23 pav. Testavimo scenarijų rašymo užduoties priskyrimo lango prototipas

1.5. sistema įvertina suvestą komandos vadovo informaciją, sukuria testavimo scenarijų rašymo užduotį ir priskiria atsakingam testuotojui. Užduotyje pateikiami priėmimo testavimo nurodymai atsižvelgiant į komandos vadovo suvestą informaciją;
1.6. specialistas užpildo įvykdytos testavimo scenarijų rašymo užduoties rezultatų informaciją;
1.7. jeigu buvo vykdoma paskutinė užduotis, procesas baigiasi, jeigu nepaskutinė, kartojami žingsniai 2.4 – 2.7.

Sekantis pagal procesų inicijavimo eiliškumą gali būti inicijuotas projektavimo etapo procesas (24 pav.).



24 pav. Testavimo veiklų procesas vykdant projektavimo etapą

Projektavimo etapas

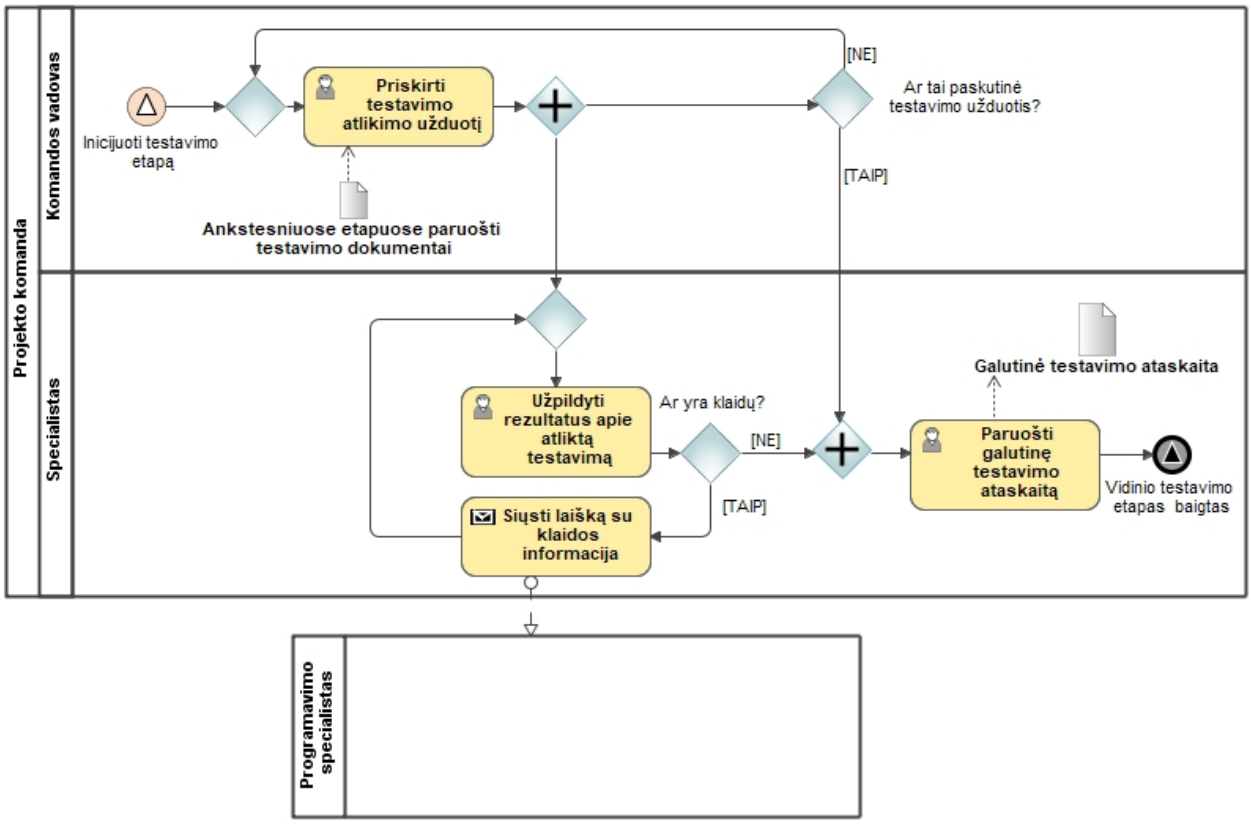
Šiame etape kuriami testavimo scenarijai, skirti sistemos testavimui. Etapo vykdymui pateikiami rekomenduojami testavimo metodai, kuriuos peržiūri testuotojų vadovas. Siekiant užtikrinti, kad metodika ir eksperimentinė sistema būtų pritaikoma kuo įvairesniems projektams, šio etapo realizaciniame modelyje neturi būti įvardintos konkrečios testavimo veiklos. Realizacijos vykdomasis modelis turi suteikti galimybę kurti neapibrėžtą įvairių tipų testavimo užduočių skaičių. Kokią konkrečiai testavimo užduotį reikia vykdyti, nusprendžia testuotojų komandos vadovas atsižvelgiant į realų poreikį ir projekto aplinkybes. Jeigu reikia sukurti užduotį regresinio testavimo scenarijų kūrimui, tuomet testuotojų vadovas privalo turėti tokią galimybę eksperimentinėje sistemoje. Vykdomasis kuriamos testavimo metodikos modelis turi apimti šias veiklas:

1.1. testavimo scenarijų rašymo užduoties priskyrimas. Užduoties rezultatuose testuotojų komandos vadovas pasirenka komandos testuotoją, nurodo konkrečiai, kokiam testavimo tipui, kuriame lygyje turi būti rašomi testavimo scenarijai. Šiame etape būtina sukurti testavimo scenarijus sistemos lygiui. Taip pat įvertinami turimi resursai, atsakingo testuotojo kompetencijos, surašomi prioritetai sistemos siūlomiems testavimo metodams (sistemos pateikiamas rekomenduojamų metodų sąrašas redaguojamas, jame turi būti galima prie kiekvieno metodo nurodyti reikiamą informaciją). Galimas užduoties rezultatų suvedimo langas pateikimas: 23 pav.;

1.2. testavimo scenarijų užduoties rezultatų pildymas. Testuotojas atlikęs testavimo scenarijų rašymo užduotį, įkelia rezultatų failą ir pateikia vadovo peržiūrai;

1.3. testavimo scenarijų užduoties tikrinimas. Testuotojų grupės vadovas privalo gauti testavimo scenarijų užduoties rezultatų failą ir jį peržiūrėti. Jeigu vadovas turi pastabų – jas pateikia ir grąžina failą taisymui, jeigu patvirtina – vykdomas 3.5 žingsnis;
1.4. jeigu reikia, atliekamas testavimo scenarijų rašymo užduoties rezultatų taisymas. Jeigu testuotojų vadovas pateikė pastabas testavimo scenarijų užduoties rezultatams, tuomet testuotojas privalo pataisyti scenarijus pagal vadovo pastabas ir pateikti pakartotinai peržiūrai, inicijuojamas 3.3 žingsnis;
1.5. jeigu yra poreikis, inicijuojama ir vykdoma kita testavimo užduotis kartojant 3.1 – 3.4 žingsnius. Eksperimentinėje sistemoje turi būti realizuota galimybė vykdyti neapibrėžtą testavimo veiklų skaičių šiame etape.

Sekantis pagal procesų inicijavimo eiliškumą gali būti inicijuotas programavimo etapo procesas (25 pav.).



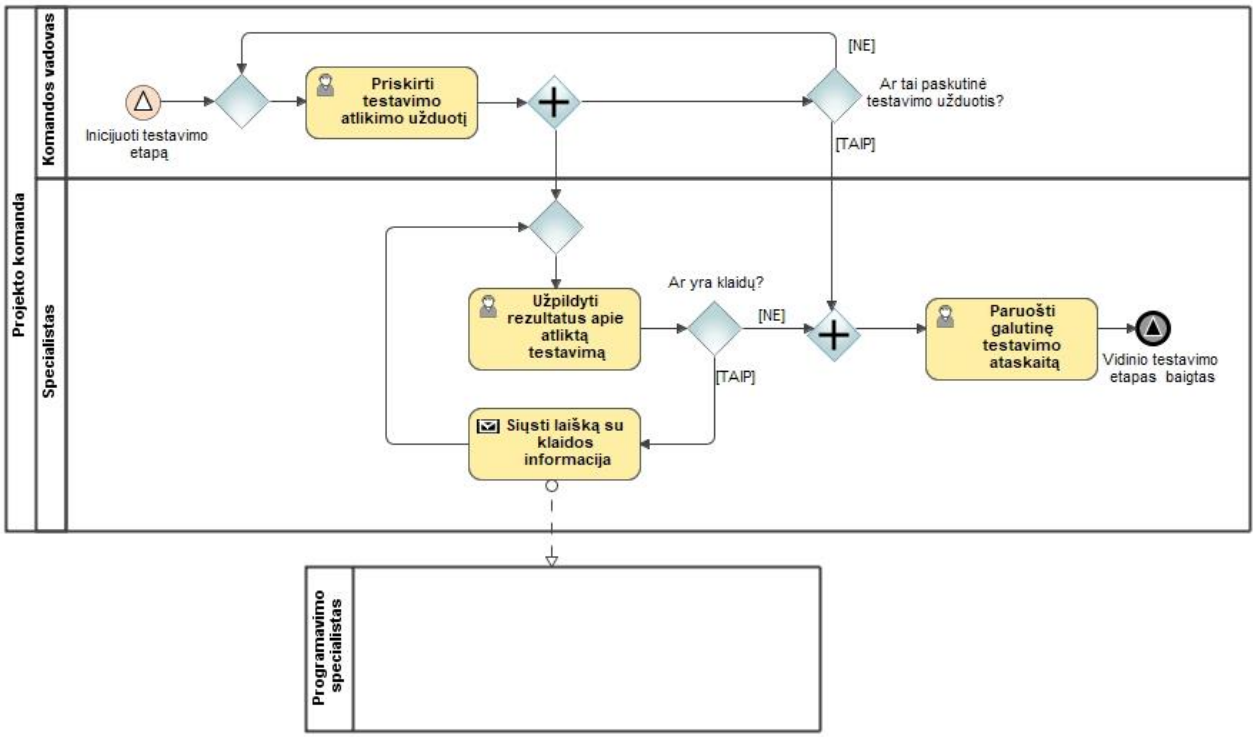
25 pav. Testavimo veiklų procesas vykdant programavimo etapą

Programavimo etapas

Šio etapo metu kuriami testavimo atvejai, skirti vidiniam testavimui, kurį atlieka pats programuotojas arba perduoda testuotojui. Testavimo scenarijai rašomi kūrėjų komandai suprantama kalba, kuri nėra skirta klientui. Šiame etape atsižvelgiant į gerąsias praktikas rekomenduojama atlikti statinę kodo analizę naudojant „Gendarme“ įrankį [32], paruošti testavimo scenarijus sistemos vienetams ir komponentams, paruošti testavimui reikiamus duomenis, bei atlikti vienetų testavimą. Etapo metu kuriama metodika apims testavimo veiklas:

1.1. Statinės kodo analizės užduoties priskyrimas. Testuotojų grupės vadovas peržiūri projekto informaciją, įvertina projektą ir priskiria statinės kodo analizės užduotį. Nurodoma informacija: testuotojas, terminas, detalus užduoties aprašas;
1.2. statinės kodo analizės rezultatų pateikimas. Atsakingam testuotojui priskiriama statinės kodo analizės užduotis, kurią rekomenduojama vykdyti naudojant „Gendarme“ įrankį [32];
1.3. komandos vadovas įvertina etapą, užpildo informaciją, kuri yra aktuali tolimesnei testavimo veiklai:
1.3.1. parenkamas atsakingas etapo testuotojas, kuris vykdys testavimo užduotis (testuotojai filtruojami pagal atrinktą etapo komandą);
1.3.2. testavimo lygis, kurį reikia testuoti;
1.3.3. įvertinami turimi resursai, atsakingo testuotojo kompetencijos, surašomi prioritetai sistemos siūlomiems testavimo metodams (sistemos pateikiamas rekomenduojamų metodų sąrašas redaguojamas, jame turi būti galima prie kiekvieno metodo nurodyti reikiamą informaciją);
1.4. sistema įvertina suvestą komandos vadovo informaciją, sukuria užduotį atsakingam testuotojui. Užduotyje pateikiami siūlomi testavimo metodai, bei komandos vadovo suvestos pastabos ir papildoma informacija;
1.5. specialistas užpildo įvykdytos testavimo scenarijų rašymo užduoties rezultatų informaciją (pateikiamas rezultatų failas);
1.6. jeigu tai paskutinė užduotis vykdomas 4.6 žingsnis, jeigu nepaskutinė, kartojami žingsniai 4.3 – 4.5;
1.7. testuotojas paruošia testavimo duomenis. Atsakingam testuotojui priskiriama testavimui reikalingų duomenų paruošimo užduotis;
1.8. testavimo rezultatų pateikimas. Atsakingas testuotojas pateikia atlikto testavimo rezultatus.

Sekantis pagal procesų inicijavimo eiliškumą gali būti inicijuotas testavimo etapo procesas (26 pav.).



26 pav. Testavimo vykdymo procesas

Testavimo etapas

Šio etapo metu turi būti įvykdyti visi testavimo atvejai, kurie buvo paruošti ankstesniuose etapuose. Užduotis priskiria testuotojų grupės vadovas, kuriam turi būti pateikiami visi paruošti testavimo dokumentai ankstesniuose etapuose. Taip pat etapo metu registruojamos ir el. paštu išsiunčiamos rastos klaidos programuotojams. Veiklų sąrašas:

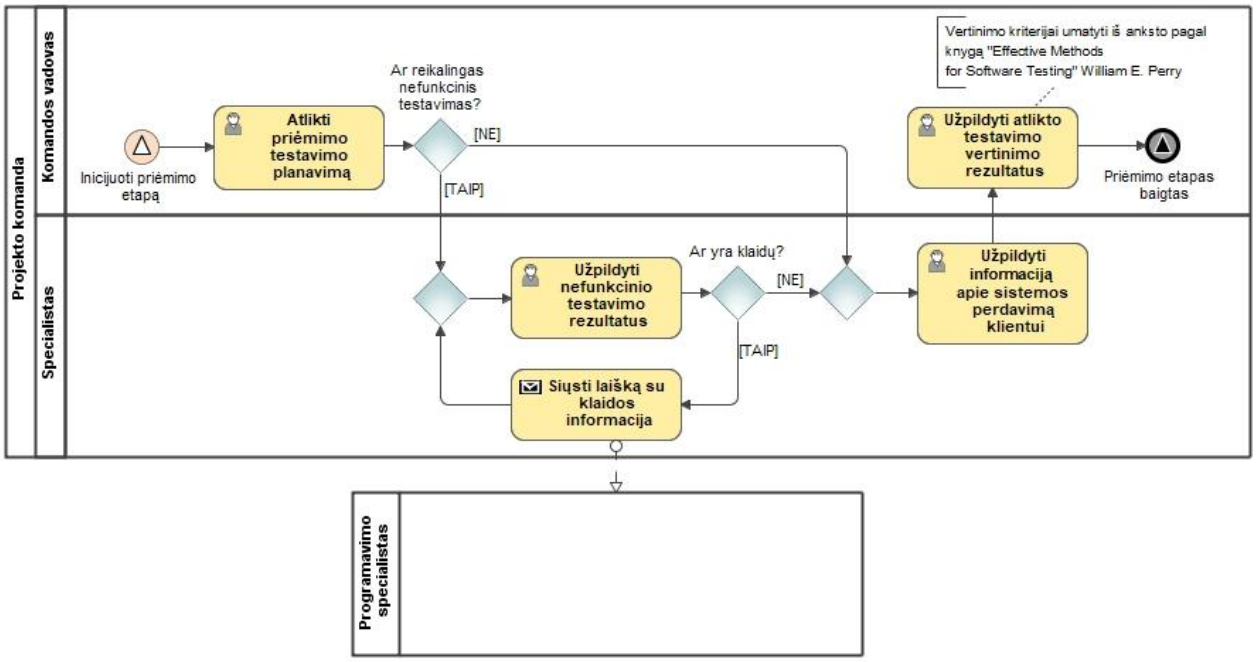
1.1. testavimo užduoties priskyrimas. Testuotojų grupės vadovas įvertina ankstesniuose etapuose paruoštus testavimo dokumentus ir pagal juos aprašo atitinkamas užduotis, kurias priskiria atsakingam testuotojui. Užpildoma detali užduoties informacija, pasirenkamas testuotojas, nurodomas užduoties atlikimo terminas;

1.2. testavimo rezultatų pildymas. Atsakingas testuotojas atlieka priskirtą užduotį, įkelia užduoties rezultatų failą, jeigu testuotojas randa klaidų, jas užregistruoja ir išsiunčia el. paštu programuotojui. Eksperimentinėje sistemoje turi būti galimybė tiesiogiai iš rezultatų pildymo formos pildyti klaidas, nurodyti el. paštą ir juo išsiųsti rastas klaidas. Veikla kartojama, kol suregistruojamos visos klaidos ir galutinai užpildomi testavimo rezultatai;

1.3. jeigu tai paskutinė užduotis, pereinama į 5.4 žingsnį, jeigu ne – kartojami 5.1-5.2 žingsniai;

1.4. sudaroma apibendrinta testavimo ataskaita. Atsakingas testuotojas, užpildo galutinę testavimo ataskaitą ir ją pateikia eksperimentinėje sistemoje.

Sekantis pagal procesų inicijavimo eiliškumą gali būti inicijuotas priėmimo etapo procesas (27 pav.).



27 pav. Sistemos priėmimo etapo procesas testuotojų skyriuje

Priėmimo etapas

Šio etapo metu pagal poreikį vykdomi nefunkciniai sistemos testai, sistemos perdavimas kliento testavimui, užfiksuojamas sistemos priėmimo faktas, kokybiškai įvertinamas vykdytas testavimo procesas. Etapo veiklos:

1.1. priėmimo etapo planavimas. Testuotojų grupės vadovas sistemoje nurodo, ar bus reikalingas nefunkcinis sistemos testavimas, jeigu taip, nurodomas, koks konkrečiai. Taip pat pasirenkamas atsakingas etapo testuotojas ir pateikiamas užduoties atlikimo terminas;

1.2. nefunkcinių testų rezultatų pateikimas. Atsakingas testuotojas atlieka priskirtą užduotį, įkelia užduoties rezultatų failą, jeigu testuotojas randa klaidų, jas užregistruoja ir išsiunčia el. paštu programuotojui. Eksperimentinėje sistemoje turi būti galimybė tiesiogiai iš rezultatų pildymo formos pildyti klaidas, nurodyti el. paštą ir juo išsiųsti rastas klaidas. Veikla kartojama, kol suregistruojamos visos klaidos ir galutinai užpildomi testavimo rezultatai;

1.3. sistemos perdavimas klientui. Atsakingas testuotojas eksperimentinėje sistemoje užfiksuoja faktą, kad sistema buvo sėkmingai perduota priėmimo testavimui;

1.4. testavimo proceso vertinimas. Sukurta metodika ir eksperimentinė sistema rekomenduoja kokybiškai įvertinti atliktą testavimo procesą pagal [23] šaltinyje aprašytus kriterijus. Eksperimentinėje sistemoje testuotojų grupės vadovui priskiriama testavimo proceso vertinimo užduotis, kurioje pateikiami visi vertinimo kriterijai ir vadovas turi pažymėti, ar buvo procesas kokybiškai įvertintas. Kokybės vertinimo klausimai:

- Ar vadovai vis dar palaiko testavimo proceso tobulinimo veiklas?
- Ar buvo priskiriami resursai testavimo proceso tobulinimui?
- Ar buvo priskirtas asmuo atsakingas už testavimo proceso tobulinimo priežiūrą?
- Ar buvo kaupiami testavimo rezultatai?
- Ar buvo naudojami rekomenduojami testavimo metodai?
- Ar testuotojams užteko darbo įrankių, skirtų testavimo rezultatų apibendrinimui, fiksavimui, klaidų pranešimui?

- Ar užfiksuotos testavimo proceso silpnosios vietos buvo atrinkto argumentuotai ir pagrįstai?
- Ar testavimo proceso peržiūra buvo atliekama reguliariai?
- Ar pastebėjus tobulinamas testavimo proceso vietas, jos buvo užfiksuotos ir įtrauktos į tobulinimo planą?
- Ar testavimo proceso kokybės rezultatai yra saugomi? Siekiant įvertinti, ar įdiegtas įmonėje patobulinimas tikrai pagerino testavimo procesas.

Detaliai aprašius kuriamą programinės įrangos testavimo metodiką ir sudarius siekiamus procesų modelius, pradedamas eksperimentinės sistemos projektavimas ir kūrimas (žr. 3 Programinės įrangos testavimo metodikos demonstracinės sistemos sprendimas ir realizacija).

3. Programinės įrangos testavimo metodikos demonstracinės sistemos sprendimas ir realizacija

3.1. Programinės įrangos testavimo metodikos demonstracinės sistemos panaudojimo atvejų modelis ir reikalavimų specifikacija

Sistemos reikalavimų specifikacijos skyriuje apibrėžiami funkciniai ir nefunkciniai reikalavimai, keliami testavimo metodikos eksperimentinei sistemai.

3.1.1. Funkciniai sistemos reikalavimai

Sistemai keliami funkciniai reikalavimai aprašomi: **5 lentelė.** PĮ testavimo valdymo sistemai keliami funkciniai reikalavimai.

5 lentelė. PĮ testavimo valdymo sistemai keliami funkciniai reikalavimai

Reikalavimo Nr.	Reikalavimo formuluotė
FR-1.3.1.	Turi būti galimybė atsijungti nuo sistemos.
FR-1.3.2.	Turi būti galimybė prisijungti prie sistemos.
FR-1.3.3.	Sistemoje turi būti galimybė redaguoti profilį.
FR-1.4.9.	Sistemoje turi būti galimybė inicijuoti testavimo etapą.
FR-1.4.1.	Sistemoje turi būti galimybė sukurti užduotį
FR-1.5.1.	Sistemoje turi būti galimybė stebėti testavimo procesą
FR-1.5.2.	Sistemoje turi būti galimybė nutraukti testavimo procesą
FR-1.4.7.	Sistemoje turi būti galimybė užpildyti užduoties rezultatų formą
FR-1.4.2.	Sistemoje turi būti galimybė peržiūrėti užduotį
FR-1.4.4.	Sistemoje turi būti galimybė peržiūrėti užduočių sąrašą.
FR-1.4.6.	Sistemoje turi būti galimybė pakomentuoti užduotį.
FR-1.4.5.	Sistemoje turi būti galimybė pataisyti užduoties rezultatus.
FR-1.4.8.	Sistemoje užduotys turi būti vykdomos pagal vykdomąjį veiklos modelį.

Sistemos reikalavimus galima sugrupuoti į tris grupes:
1. Reikalavimai naudotojų valdymo moduliui: turi būti galimybė prisijungti, atsijungti nuo sistemos, bei redaguoti profilį.
2. Reikalavimai užduočių valdymo moduliui: turi būti galimybė sukurti, peržiūrėti testavimo užduotis, jas inicijuoti, užpildyti užduoties įvykdymo rezultatus, patikrinti rezultatus, peržiūrėti užduočių sąrašą, parašyti komentarą, bei inicijuoti projekto etapą.
3. Reikalavimai proceso stebėjimo moduliui: turi būti galimybė stebėti ir nutraukti vykdomą testavimo procesą.

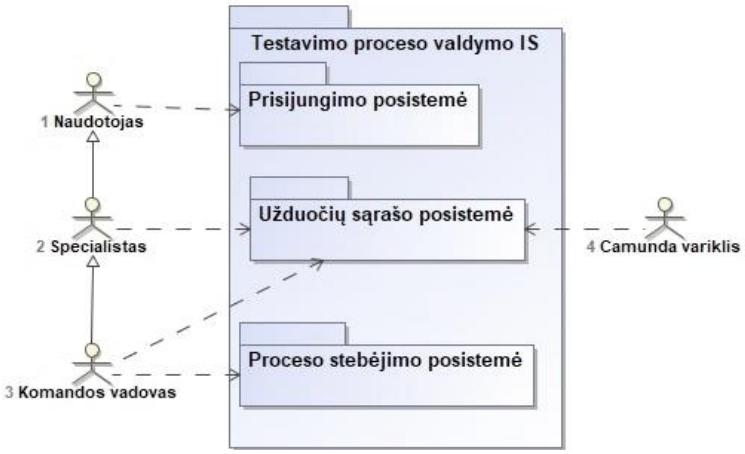
Atsižvelgiant į funkcinių reikalavimų aprašą, kuriamas demonstracinės sistemos panaudojimo atvejų modelis (3.1.2).

3.1.2. Panaudojimo atvejų modelis

Kompiuterizuojamų panaudojimo atvejų diagrama atvaizduoja sistemos naudotojo (aktoriaus) ir kuriamos sistemos sąveiką per sistemos funkcijas (panaudojimo atvejus). Ši diagrama apibrėžia

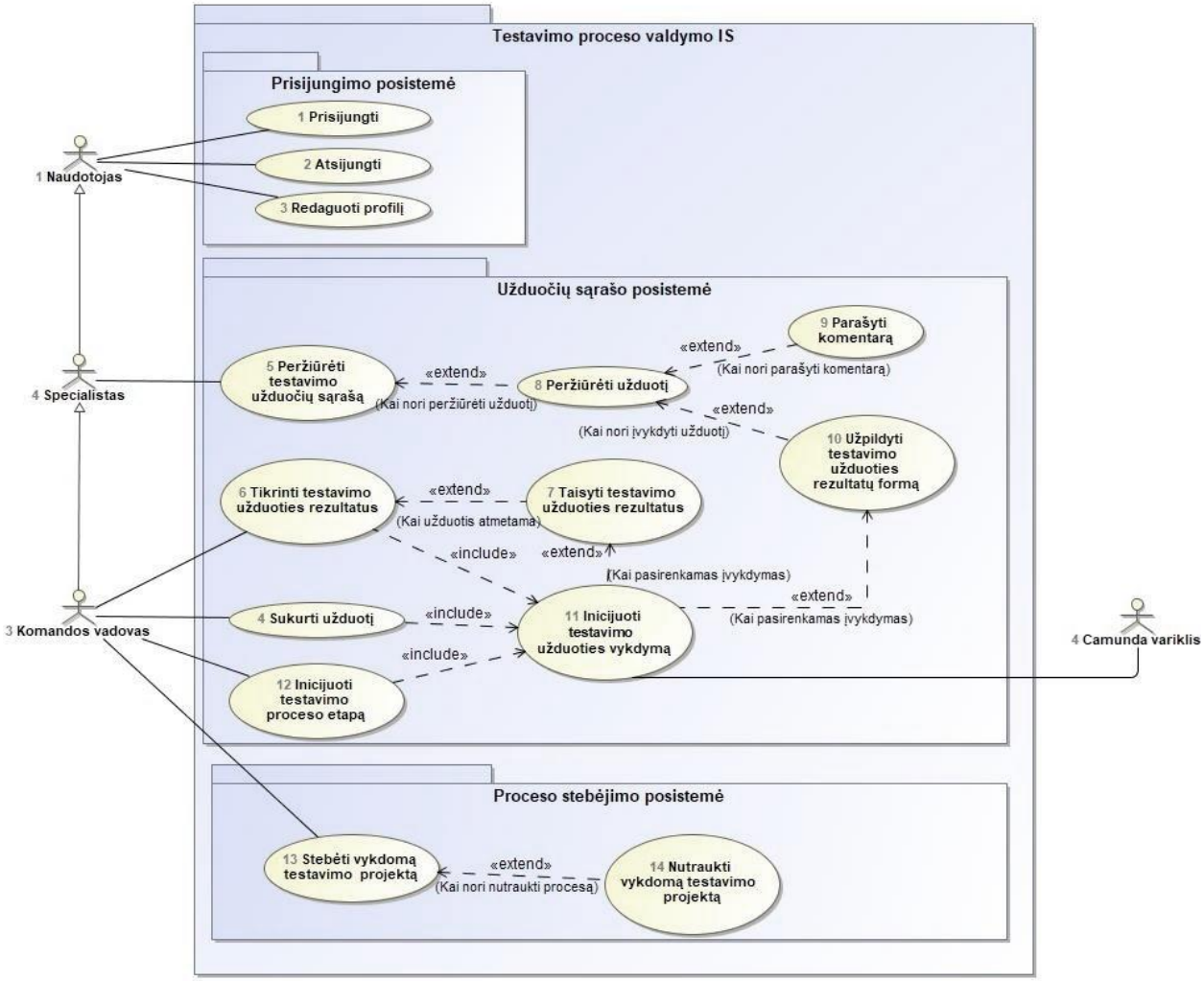
sistemai keliamus funkcinius reikalavimus. PĮ testavimo proceso valdymo sistemos kompiuterizuojamų panaudojimo atvejų diagrama pateikiama dviem abstrakcijos lygiais:

1. Pirmame abstrakcijos lygyje (28 pav.) vaizduojama: PĮ testavimo valdymo sistemos suskirstymas į posistemes, tarpusavio sąveika tarp aktorių ir sistemos posistemių. Šiame abstrakcijos lygyje konkretūs panaudojimo atvejai nepateikiami.



28 pav. PĮ testavimo proceso valdymo informacinės sistemos apibendrinta panaudojimo atvejų diagrama

2. Antrame abstrakcijos lygyje (29 pav.) detalizuojama kiekviena sistemos posistemė: pateikiami panaudojimo atvejai ir sąveikas atspindintys ryšiai. Šiame abstrakcijos lygyje vizualizuojama, kaip konkretus aktorius sąveikauja su konkrečiais panaudojimo atvejais ir kitais aktoriais. Taip pat diagramoje pateikiami panaudojimo atvejų tarpusavio sąveikas atspindintys ryšiai.



29 pav. PĮ testavimo proceso valdymo informacinės sistemos panaudojimo atvejų diagrama

PĮ testavimo proceso valdymo informacinė sistema išskaidoma į 3 posistemes, kurios sugrupuoja kompiuterizuojamus panaudojimo atvejus pagal realizuojamo funkcionalumo logiką:

- **Prisijungimo posistemė** – panaudojimo atvejų, realizuojančių prisijungimo, atsijungimo, profilio valdymo funkcionalumą, posistemė;
- **Užduočių sąrašo posistemė** – apjungia visus panaudojimo atvejus, kurie susiję su projekto užduočių valdymu: užduočių kūrimas, priskyrimas, vykdymas, inicijavimas;
- **Analizės posistemė** – saugomi panaudojimo atvejai, realizuojantys vykdomų projektų stebėjimą.

Detaliame kiekvienos posistemės apraše pateikiama: panaudojimo atvejų aprašas (2 priedas), kiekvieno panaudojimo atvejo realizacijos projekto klasėmis aprašas (3 priedas).

3.1.3. Nefunkciniai sistemos reikalavimai

PĮ testavimo proceso valdymo sistemai keliami nefunkciniai reikalavimai (6 lentelė) grupuojami pagal atitinkamas kategorijas ir pateikiami siūlomi patikrinimo būdai, kaip reikėtų įvertinti, ar realizuota sistema tenkina iš anksto apibrėžtus nefunkcinius reikalavimus.

6 lentelė. Nefunkciniai reikalavimai keliami PĮ testavimo proceso valdymo sistemai

Kategorija	Nr.	Reikalavimas	Patikrinimas
Tinkamumo reikalavimai	NFR-1.	Sistema apriboja galimybę naudotojui suklysti įvedant duomenis – kur įmanoma, laukuose pateikiamas „pasirinkimų sąrašas“, nurodomos pradinės reikšmės.	Peržiūrėti, ar naudojami pasirinkimų sąrašai.
	NFR-2.	Klaidos ir sėkmingos operacijos atlikimo atvejais, sistema turi pateikti vartotojui atsako pranešimus. Klaidos atveju – pateikiamas raudonos spalvos klaidos pranešimas, sėkmės atveju – pateikiamas žalios spalvos informacinis pranešimas.	Ištestuoti visas situacijas sistemoje.
	NFR-3.	Atliekant šalinimo, redagavimo ar kūrimo funkcijas yra galimybė atšaukti funkcijos vykdymą.	Peržiūrėti, ar visose formose yra mygtukas [Atšaukti].
	NFR-4	Sistema pateikia informaciją vartotojui apie vykdomų projektų būklę tiek tekstiniu, tiek grafiniu būdu.	Patikrinti, ar projekto analizės posistemėje informacija apie projekto būseną pateikiama ne tik tekstu, bet ir grafinėmis diagramomis.
Programinės įrangos palaikymas	NFR-5.	Esant poreikiui, turi būti galimybė modifikuoti procesą, pagal kurį veikia sistema.	Sistemos proceso diagramoje pridėti naują veiklą.
Reikalavimai programinei įrangai	NFR-6.	Sistema palaikoma šiose naršyklėse: - Google Chrome 72.0.3626.121 versija; - Mozilla Firefox 65.0.2 versija; - Internet Explorer 10 / 11.	Ištestuoti su visomis paminėtomis naršyklėmis.
	NFR-7.	Sistemai reikalinga ši programinė įranga: - MySQL 5.7.22; - Java 8; - Apache Tomcat 8.5.	Peržiūrėti.

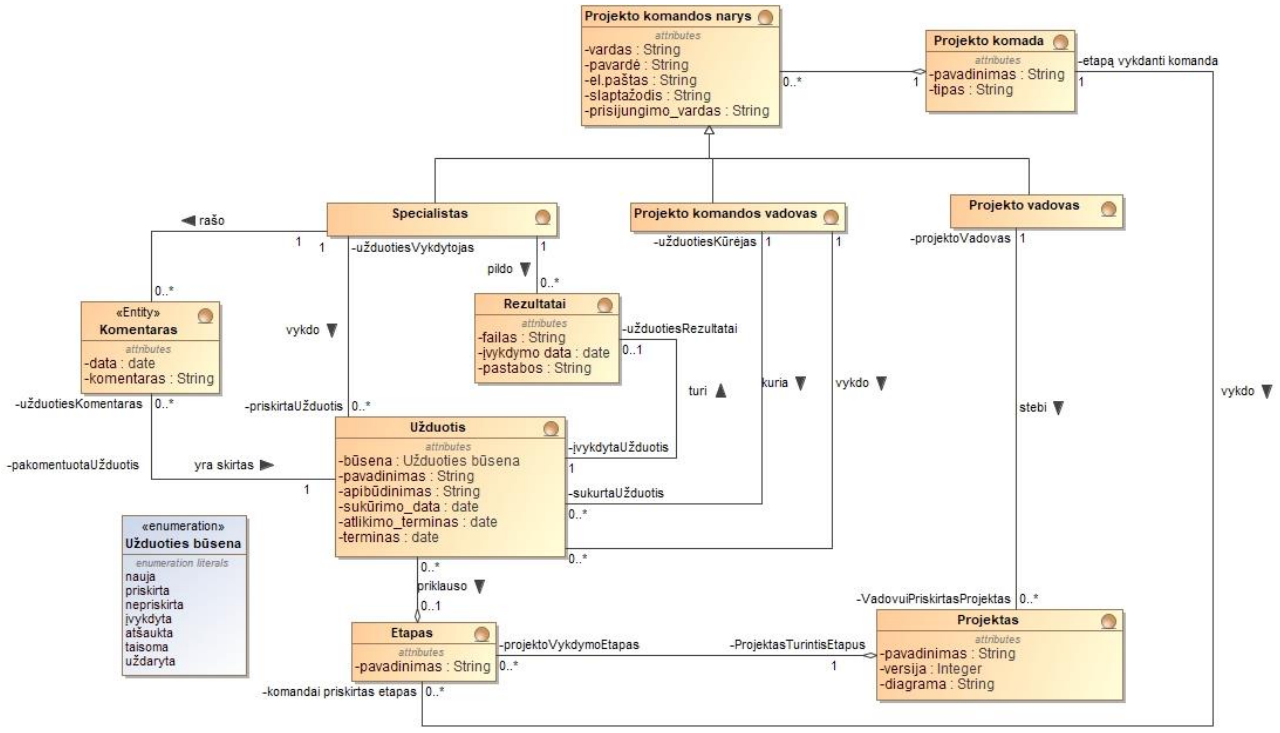
Pagrindiniai sistemai keliami nefunkciniai reikalavimai priskiriami sistemos tinkamumo reikalavimų grupei, nes realizuojant PĮ testavimo proceso valdymo sistemą, bus siekiama atskleisti pagrindinius pasirinktos BPM platformos „Camunda BPM“ privalumus. Pasirinktoje realizavimo platformoje galima nesudėtingai ir nesunaudojant daug laiko resursų:

- pateikti informaciją lengvai suprantamu grafiniu būdu;
- modifikuoti sistemos funkcionavimo procesą;
- keisti sistemos formas, pateikti laukuose pavyzdines reikšmes, sukurti funkcijos atšaukimo galimybę;
- pateikti grįžtamojo ryšio klaidos/informacinius pranešimus.

Visus šiuos platformos privalumus, bus siekiama atskleisti realizuojant PĮ testavimo proceso valdymo sistemą, dėl to nefunkciniai reikalavimai orientuoti į sistemos tinkamumą ir palaikymo paprastumą.

3.2. Programinės įrangos testavimo metodikos demonstracinės sistemos duomenų modelis

Probleminės srities analizės etape identifikavus esminius veiklos objektus ir jų priklausomybes sudaroma dalykinės srities esybių klasių diagrama (30 pav.), kurioje pateikiamos esybės struktūrizuota ir artimesne būsimai PĮ testavimo proceso valdymo sistemos realizacijai forma.

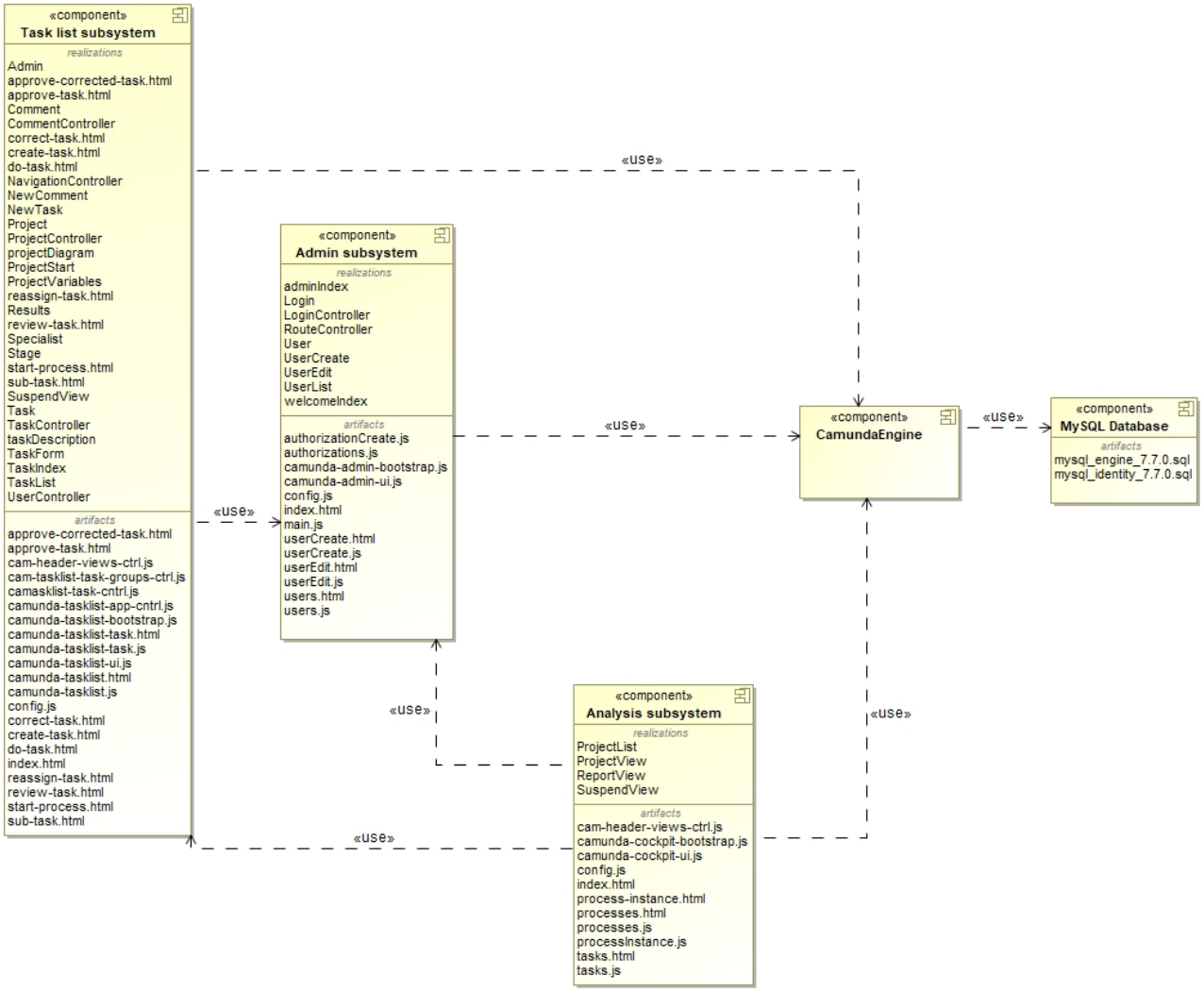


30 pav. PĮ testavimo proceso dalykinės srities diagrama

Dalykinės srities esybių diagramoje pateikiami aktualių esybių tarpusavio sąveikas atspindintys ryšiai. Diagramoje vaizduojama esybė „Užduotis“ gali įgyti vieną iš būsenų („Nauja“, „Priskirta“, „Nepriskirta“, „Įvykdyta“, „Atšaukta“, „Taisoma“, „Uždaryta“). Visos būsenos įvardintos „enumeration″ tipo elemente. Užduotims būsenos gali būti priskiriamos nebūtinai nuosekliai.

3.3. Programinės įrangos testavimo metodikos demonstracinės sistemos realizacijos modelis

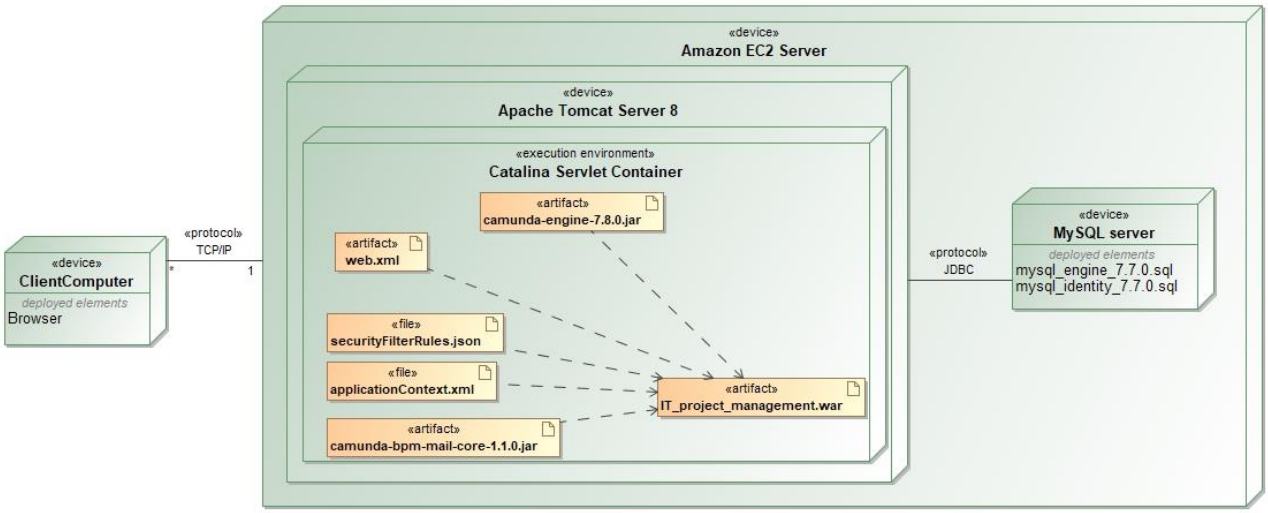
Sistemą realizuojančius elementus ir jų tarpusavio sąveikas atspinti IS realizuojančių komponentų diagrama (31 pav.).



31 pav. Sukurtos testavimo metodikos demonstracinės sistemos komponentų diagrama

Sukurtos testavimo metodikos demonstracinė sistema sudaryta iš trijų posistemių. Kiekviena posistemė su duomenų baze komunikuoja per „Camunda BPM“ platformos BPM variklį. Kiekviena posistemė vaizduojama kaip atskiras komponentas, kurio viduje pateikiami visi tą posistemę realizuojantys elementai.

Planuojamą sistemos diegimo struktūrą atspindi diegimo diagrama (32 pav.).



32 pav. Sukurtos testavimo metodikos demonstracinės sistemos diegimo diagrama

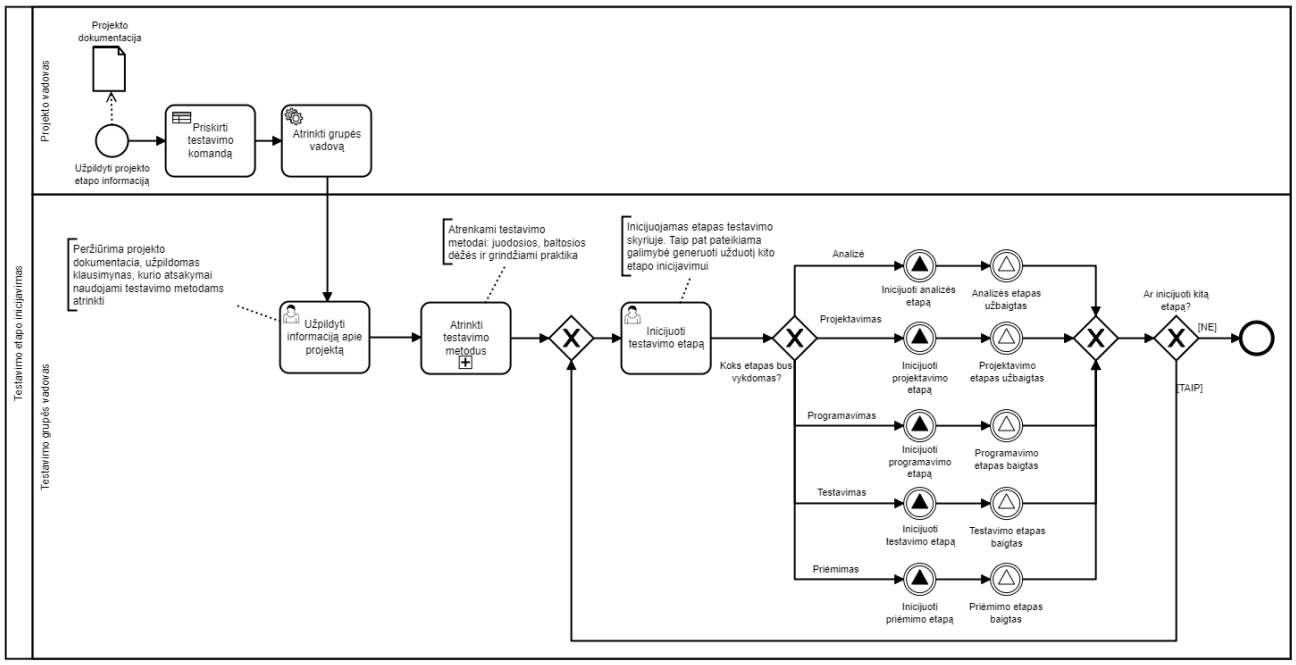
Realizuota sistema bus talpinama į Amazon EC2 serverį, kuriame įdiegtas „Apache Tomcat 8.5“ serveris ir „Catalina Servlet“ konteineris. Sukurtos testavimo metodikos demonstracinei sistemai sugeneruojamas vykdomasis .war tipo failas ir įkeliamas į serverį. Vykdomojo failo viduje bibliotekos pavidalu yra įterptas BPM vykdomasis valiklis, kuris automatiškai paleidžiamas, pradėjus darbą su sistema.

Siekiant neapkrauti diegimo diagramos, joje pateikiami tik aktualūs artefaktai, kurie susiję su rašto darbe naudojamais komponentais.

3.4. Vykdomi programinės įrangos testavimo metodikos eksperimentinės sistemos procesų modeliai

Programinės įrangos testavimo metodika ir jos demonstracinė sistema realizuota veiklos procesais grindžiamoje platformoje „Camunda BPM“. Dėl šios priežasties suprojektuoti modeliai (2.2 Programinės įrangos testavimo metodikos aprašas ir siekiamas proceso modelis) transformuojami į vykdomuosius BPM modelius, kuriuos galima sudiegti į „Camunda BPM“ platformą.

Pradinis procesas yra „Naujo projekto inicijavimas“ (33 pav.). Nuo šio proceso paleidimo pradedamas darbas su sistema.



33 pav. Naujo projekto inicijavimo vykdomasis modelis

Naujo projekto inicijavimo metu atliekamos veiklos, skirtos testavimo proceso vykdymo pasiruošimui. Procesas pradedamas tuomet, kai projektų vadovas suveda pagrindinę informaciją apie projektą, įkelia turimą projekto dokumentaciją, o sistema pagal supildytus projektų vadovo duomenis (programavimo kalbą, projekto vykdymo metodiką) atrenka testuotojų komandą, kuri gali vykdyti projektą. Komandos atrinkimas vykdomas automatiškai pagal DMN lentelėje esančius duomenis (34 pav.).

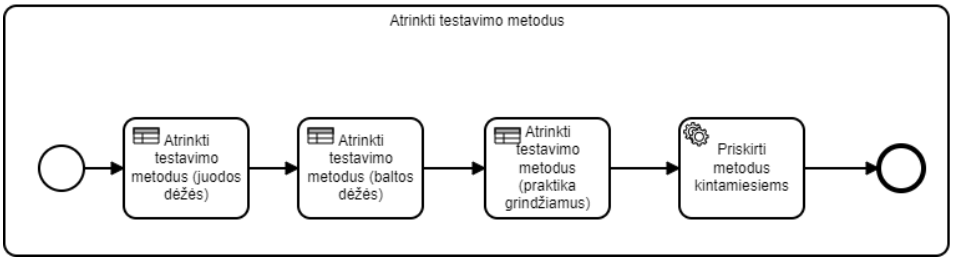
Assign team

`assign-team`

U	Input +		Output +
	language	model	result
	string	string	string
1	"Python"	"SCRUM"	"pythonagilegroup"
2	"Python"	"V modelis"	"pythongroup"
3	"Python"	"RUP"	"pythongroup"
4	"Java"	"SCRUM"	"javaagilegroup"
5	"Java"	"V modelis"	"javagroup"
6	"Java"	"RUP"	"javagroup"
7	"PHP"	"SCRUM"	"phpagilegroup"
8	"PHP"	"V modelis"	"phpgroup"
9	"PHP"	"RUP"	"phpgroup"
10	"C#"	"SCRUM"	"cagilegroup"
11	"C#"	"V modelis"	"cgroup"
12	"C#"	"RUP"	"cgroup"
+	-	-	

34 pav. Testavimo komandos atrinkimo sprendimo priėmimo lentelė

Atrinkus testuotojų komandą, identifikuojamas komandos vadovas ir sistema jam automatiškai priskiria užduotį „Užpildyti informaciją apie projektą“. Testuotojų grupės vadovas peržiūri projekto dokumentaciją, atsako į reikiamus klausimus, supildo užduoties rezultatus. Pagal supildytus testuotojų komandos vadovo duomenis sistema atrenka rekomenduojamus testavimo metodus (35 pav.).



35 pav. Išskleistas vykdomasis sub-procesas „Atrinkti testavimo metodus“

Naujo projekto inicijavimo metu taip pat vykdomas automatinis rekomenduojamų naudoti juodosios dėžės metodų atrinkimas naudojant DMN lentelę (36 pav.).

Assign method

`assign-method`

U	Ar įmanoma PĮ duomenis sugrupuoti į ekvivalenčias klases?	Ar bent vienas komponentas yra atsakingas už SP?	[...]	Output + pirmas label	antras	trecias	ketvirtas	penktas
	string	string	string	string	string	string	string	string
1	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	"Use case testing"
2	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	-
3	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	"Use case testing"
4	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	"Use case testing"
5	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	"Use case testing"
6	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	-
7	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	"Use case testing"
8	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	"Use case testing"
9	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	"Use case testing"
10	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	-
11	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	-	"Use case testing"
12	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	-	"Use case testing"
13	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	-	"Use case testing"
14	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	-	-
15	"Taip"	"Taip"	"Taip"	"Equivalence partitioning"	"Boundary value analysis"	"Decision table testing"	"State transition diagrams"	"Use case testing"

36 pav. Juodosios dėžės metodų atrinkimo sprendimo priėmimo lentelės fragmentas

Juodosios dėžės metodai atrenkami pagal testuotojų grupės suvestą informaciją (visi klausimai į kuriuos turi atsakyti testuotojų grupės vadovas pateikiami 1-ame priede).

Naujo projekto inicijavimo metu taip pat vykdomas automatinis rekomenduojamų naudoti baltosios dėžės metodų atrinkimas naudojant DMN lentelę (37 pav.).

Assign method

`assign-method1`

U	Input + Sistemos rizikos lygis?	Svarbu ištestuoti sistemos kodą, bet sistema nėra kritiškai svarbi („safety critical“)?	Sistemos („safety critical“) klaida gali sukelti kritines pasekmes?	Output + sestas	septintas	astuntas
	string	string	string	string	string	string
1	"Žemas"	"Taip"	"Ne"	"Statement Coverage"	-	-
2	"Vidutinis"	"Taip"	"Ne"	-	"Decision (branch) testing"	-
3	"Vidutinis"	"Ne"	"Taip"	-	-	"Path Testing"
4	"Aukštas"	"Ne"	"Taip"	-	-	"Path Testing"
5	"Aukštas"	"Taip"	"Ne"	-	"Decision (branch) testing"	-
+	-	-	-			

37 pav. Baltosios dėžės metodų atrinkimo sprendimo priėmimo lentelė

Baltosios dėžės metodai atrenkami pagal testuotojų grupės suvestą informaciją. Vertinama, ar sistemos klaidos gali sukelti kritines pasekmes, bei atsižvelgiama į komandos vadovo nurodytą sistemos rizikos lygį.

Naujo projekto inicijavimo metu taip pat vykdomas automatinis rekomenduojamų naudoti praktika grindžiamų metodų atrinkimas naudojant DMN lentelę (38 pav.).

Assign method

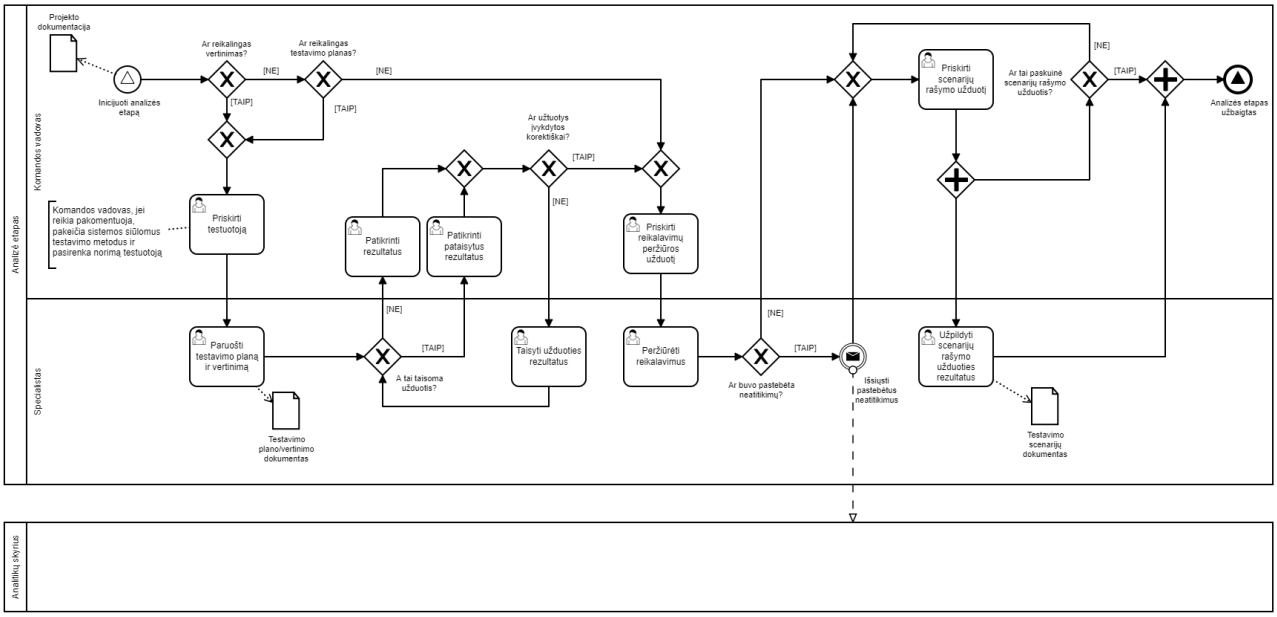
`assign-method2`

U	Input +			Output +		
	Sistema neturi/turi nedetalią dokumentaciją?	Ar testuotojas turi patirties pan. projektuose?	Ar testavimo komanda turi susidarę sąrašą atvejų, kuriuos reikėtų testuoti tokio tipo projektuose?	devintas	desimtas	vienuoliktas
	string	string	string	string	string	string
1	"Taip"	"Ne"	"Ne"	"Exploratory Testing"	-	-
2	"Taip"	"Ne"	"Taip"	"Exploratory Testing"	-	"Checklist-Based Testing"
3	"Taip"	"Taip"	"Ne"	"Exploratory Testing"	"Error Guessing"	-
4	"Taip"	"Taip"	"Taip"	-	"Error Guessing"	"Checklist-Based Testing"
5	"Ne"	"Ne"	"Ne"	-	-	-
6	"Ne"	"Taip"	"Ne"	-	"Error Guessing"	-
7	"Ne"	"Taip"	"Taip"	-	"Error Guessing"	"Checklist-Based Testing"
8	"Ne"	"Ne"	"Taip"	-	-	"Checklist-Based Testing"
+	-	-	-		-	-

38 pav. Praktika grindžiamų metodų atrinkimo sprendimo priėmimo lentelė

Praktika grindžiami metodai atrenkami pagal testuotojų grupės suvestą informaciją. Vertinama, ar prie projekto dirbantys testuotojai turi patirties analogiškuose projektuose, ar egzistuoja atvejų, kuriuos reikėtų testuoti, sąrašas. Taip pat atsižvelgiama į sistemos dokumentacijos detalumą.

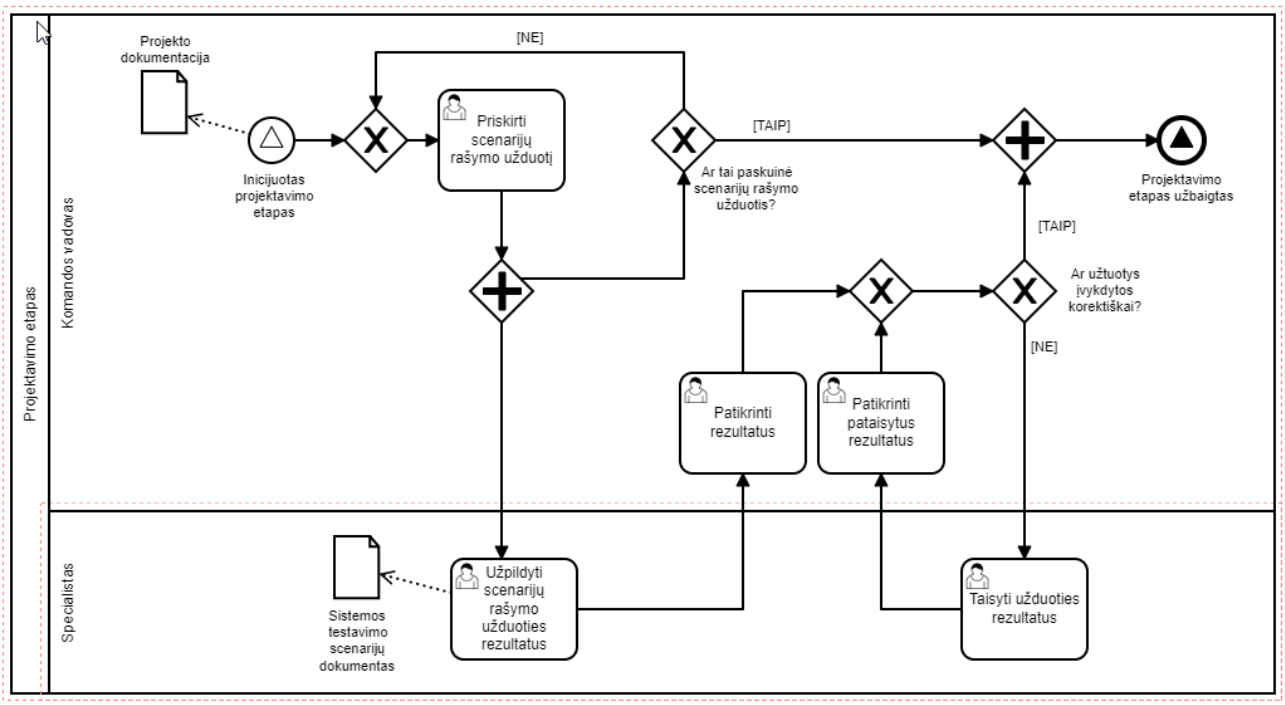
Sekantis testuotojų skyriuje gali būti inicijuojamas analizės etapo vykdymo procesas (39 pav.).



39 pav. Analizės etapo vykdomasis modelis

Analizės etapo vykdymo metu testuotojai ruošiasi sistemos testavimui. Testuotojų komandos vadovas priskiria užduotis testuotojams, testuotojai paruošia testavimo planą ir įvertina planuojamus testavimo darbus, užduoties rezultatus perduoda grupės vadovui. Testuotojų vadovas patikrina užpildytus rezultatus, jeigu reikia, pateikia pastabas pagal kurias testuotojai pataiso dokumentus. Atlikus vertinimo ir planavimo užduotis, testuotojų vadovas priskiria reikalavimų peržiūros užduotį. Atsakingas testuotojas atlikęs užduotį, jeigu reikia supildo pastebėtus neatitikimus ir nurodo, kokiu el. pašto adresu išsiųsti pastabas. Pastabos iš eksperimentinės sistemos nusiunčiamos analitikui nurodytu el. paštu. Atlikus reikalavimų peržiūrą, testuotojų vadovas priskiria testavimo scenarijų rašymo užduotį, nurodydamas konkretų testuojamą sistemos lygį. Jeigu reikia, vadovas priskiria testavimo scenarijų rašymo užduotį kelis kartus.

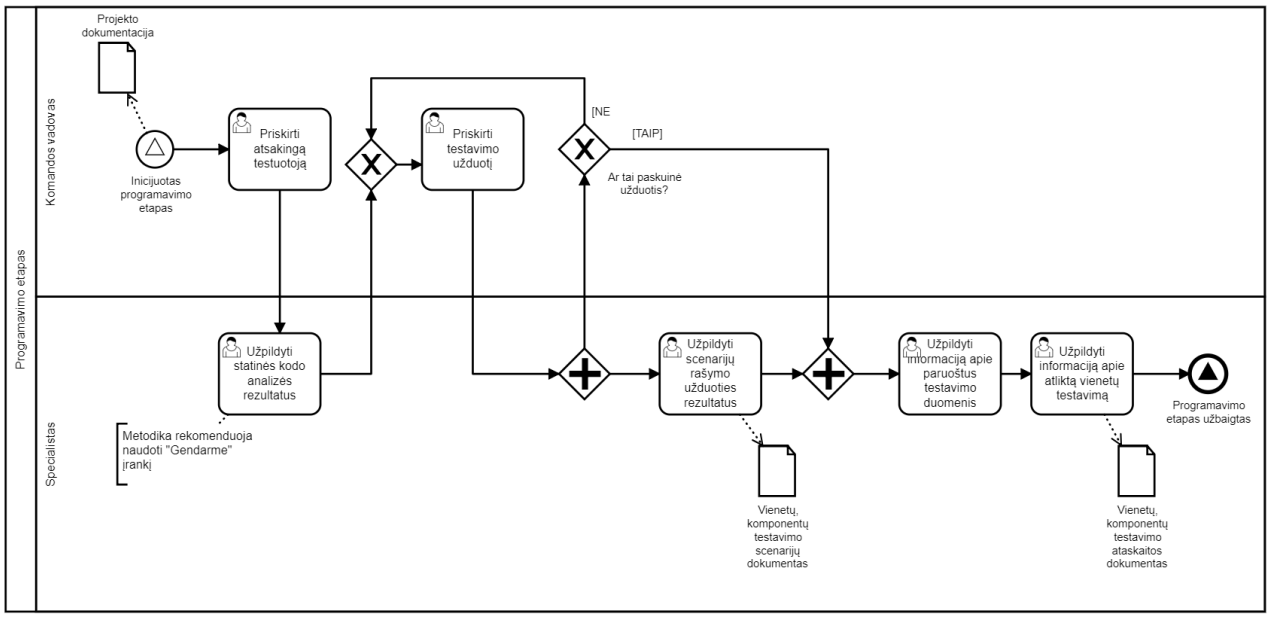
Sekantis testuotojų skyriuje gali būti inicijuojamas projektavimo etapo vykdymo procesas (40 pav.).



40 pav. Projektavimo etapo vykdomasis modelis

Projektavimo etapas pradedamas nuo testavimo scenarijų rašymo užduoties priskyrimo. Šiame etape rekomenduojama sukurti testavimo scenarijus sistemos lygiui, tačiau prie konkrečių užduočių procesas nėra pririštas. Testuotojų grupės vadovas gali pagal savo nuožiūrą priskirti neapribotą skaičių užduočių, skirtų skirtingiems testavimo lygiams ir tipams. Kai testuotojas atlieka priskirtą užduotį, rezultatus pateikia vadovui, kuris, jeigu reikia, pateikia pastabas, o testuotojas atsižvelgdamas į pastabas pataiso rezultatų dokumentą.

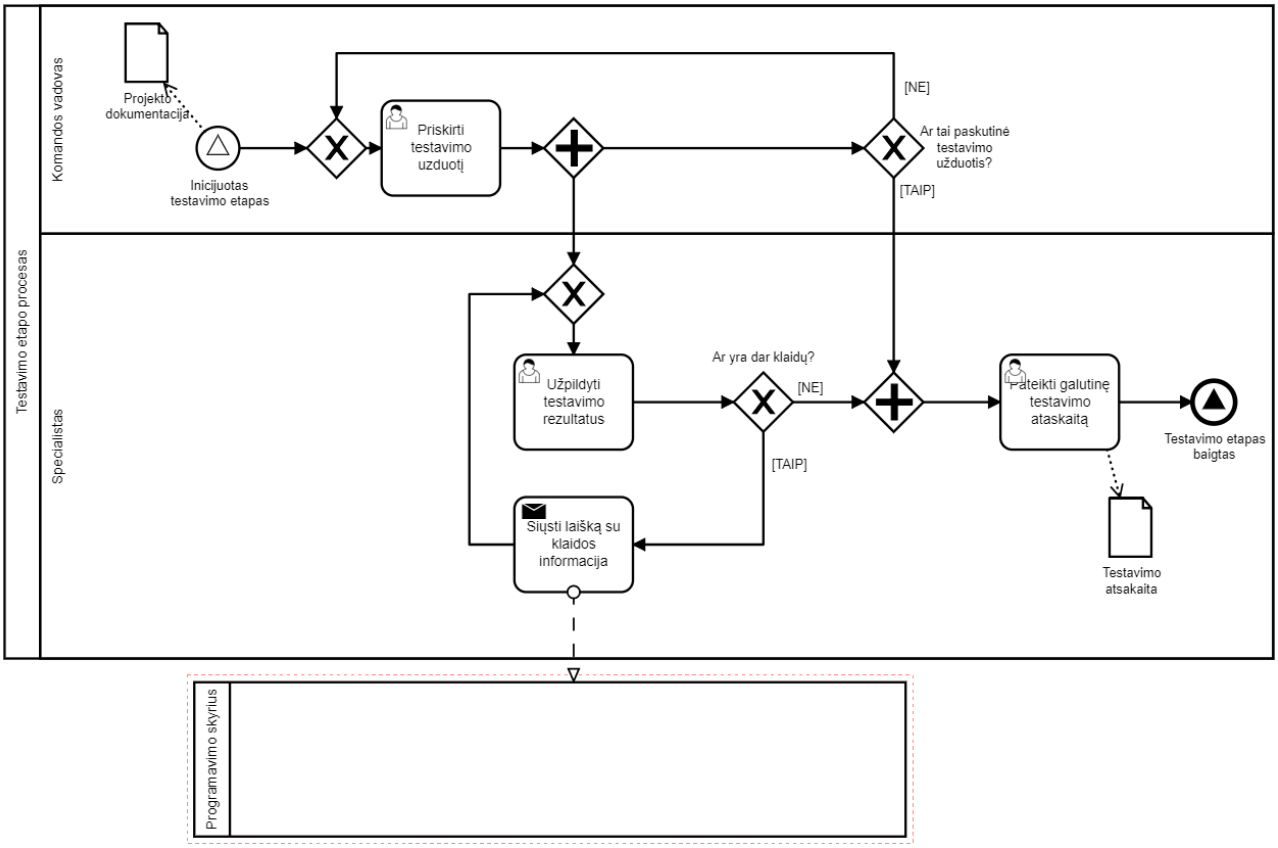
Testuotojų skyriuje taip pat gali būti inicijuojamas programavimo etapo vykdymo procesas (41 pav.).



41 pav. Programavimo etapo vykdomasis modelis

Programavimo etapo vykdymo metu testuotojai taip pat gali vykdyti testavimo scenarijų rašymo užduotis, skirtas, pavyzdžiui, vienetų lygiui. Proceso metu testuotojų grupės vadovas testuotojams priskiria užduotis, o testuotojai jas vykdo. Jeigu reikia, procesas kartojamas. Taip pat šio etapo metu yra vykdoma statinė kodo analizė, kurią vykdyti rekomenduojama „Gendarme“ įrankyje. Rekomendacija pateikiama remiantis [32] šaltinyje pateikiamais įrankio privalumais. Be to, programavimo etape yra surenkami visi testavimui būtini duomenys ir atliekamas vienetų, bei komponentų testavimas, užfiksuojami testavimo rezultatai.

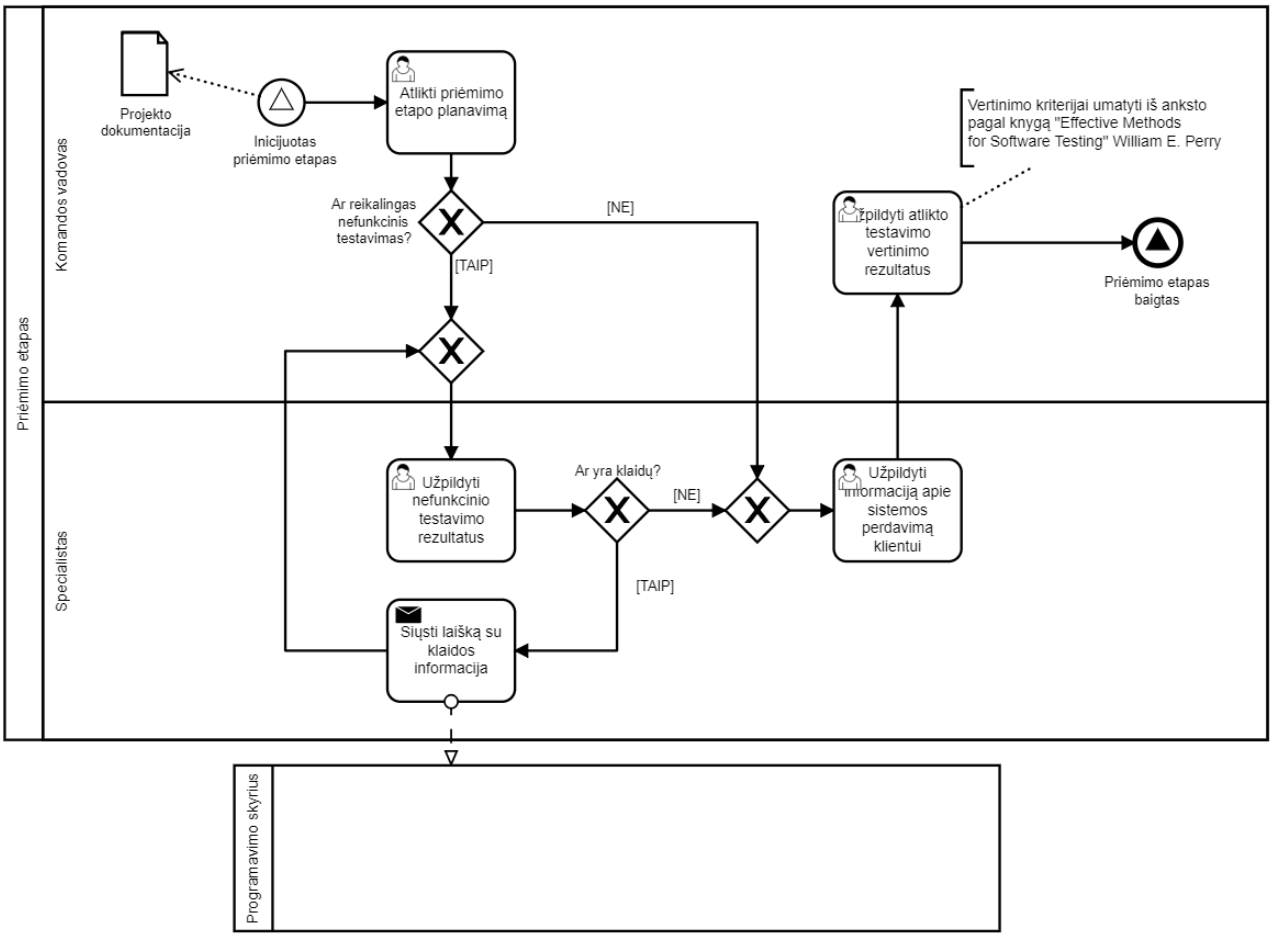
Testuotojų skyriuje vienas iš pagrindinių procesų yra programinės įrangos testavimo etapo vykdymo procesas (42 pav.).



42 pav. Testavimo etapo vykdomasis modelis

Testavimo etapo vykdymo metu testuotojai atlieka sistemos testavimą, registruoja rastas klaidas, bei pateikia testavimo rezultatų ataskaitą. Proceso metu testuotojų grupės vadovas testuotojams priskiria testavimo vykdymo užduotis, o testuotojai jas vykdo. Jeigu reikia, procesas kartojamas. Taip pat testuotojai registruoja klaidas, bei pildo galutinę testavimo ataskaitą.

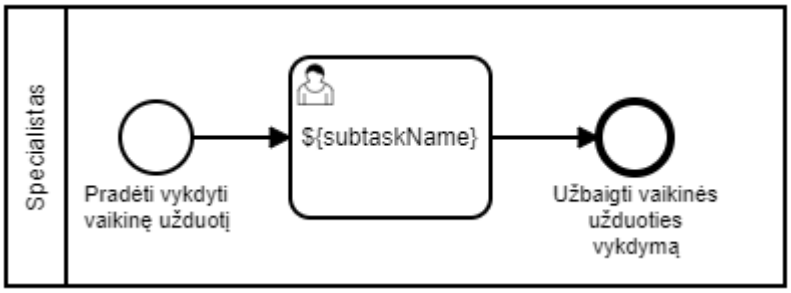
Atlikus vidinį sistemos testavimą, inicijuojamas sistemos priėmimo etapas (43 pav.).



43 pav. Priėmimo etapo vykdomasis modelis

Sistemos priėmimo etapo vykdymas pradedamas nuo planavimo užduoties, kurią atlieka testuotojų grupės vadovas. Užpildoma informacija apie nefunkcinį testavimą, nurodoma, ar jis reikalingas. Jeigu reikalingas, vykdomas nefunkcinis testavimas ir fiksuojamos klaidos išsiunčiamos programuotojams el. paštu. Jeigu nefunkcinis testavimas nereikalingas, pereinama iškart prie sistemos atidavimo klientui užduoties, kurioje užfiksuojamas faktas, jog sistema buvo perduota klientui. Galiausiai remiantis [23] šaltiniu yra atliekamas testavimo proceso vertinimas.

Atliekant testavimo procesų veiklas taip pat gali būti inicijuojamas vaikinių užduočių vykdymo procesas (44 pav.).



44 pav. Vaikinių užduočių inicijavimo vykdomasis modelis

Vaikinių užduočių vykdymas inicijuojamas automatiniu būdu iš tėvinės užduoties tiek kartų, kiek tėvinė užduotis turi vaikinių užduočių. Tėvinė užduotis tampa įvykdoma tuo metu, kai visos jos

vaikinės užduotys įvykdomos. Vaikinės užduoties pavadinimas gaunamas iš tėvinės užduoties, kurioje yra nurodytas vaikinių užduočių masyvas su vaikinių užduočių pavadinimais ir aprašymas.

4. Programinės įrangos testavimo metodikos eksperimentinės sistemos testavimas

Realizuota sistema testuojama rankiniu būdu, trimis etapais: pirmiausia, ištestuojama, ar sistema atitinka funkcinius reikalavimus (4.1), toliau testuojami nefunkciniai reikalavimai (4.2) ir galiausiai atliekamas integracinis testavimas (4.3).

Taip pat planuojama testuoti formas, kuriose galimas duomenų įvedimas, naudojantis principais:

- testuojamos ribinės, bei vidurinės įėjimo reikšmės;
- įvedami korektiški duomenys;
- įvedami klaidingi duomenys;
- neužpildomi privalomi laukai.

Peržiūros formose bus testuojama, ar atvaizduojama teisinga informacija, taip pat planuojama testuoti ištisą procesą. Detalūs atlikti testavimo atvejai ir jų rezultatai pateikiami prieduose.

4.1. Funkciniai testai

Šiame poskyryje pateikiami funkcinių testų rezultatai. Detaliai ištestuojami visi analizės etape apibrėžti funkciniai reikalavimai ir fiksuojami gauti rezultatai (4.2). Gautų rezultatų lentelėje pateikiama tik po vieną ištestuotą formą, daugiau testavimo atvejų pateikiama prieduose (4 priedas).

7 lentelė. Funkcinių reikalavimų testavimo rezultatai

Testuojami reikalavimai	Atliekami veiksmai	Nuoroda į TC	Laukiamas rezultatas	Gautas rezultatas
FR-1.3.1 Turi būti galimybė atsijungti nuo sistemos.	Naudotojas bando atsijungti.	TC-1	Sėkmingai atsijungta.	Sėkmingas
FR-1.3.2 Turi būti galimybė prisijungti prie sistemos.	Naudotojas bando prisijungti.	TC-1	Sėkmingai prisijungta.	Sėkmingas
FR-1.3.3 Sistemoje turi būti galimybė redaguoti profilį.	Naudotojas bando pakeisti profilio duomenis.	TC-1	Sėkmingai duomenys pakeičiami.	Sėkmingas
FR-1.4.9 Sistemoje turi būti galimybė inicijuoti testavimo etapą.	Bandoma inicijuoti testavimo etapą.	TC-2	Etapas inicijuojamas sėkmingai.	Sėkmingas
FR-1.4.1. Sistemoje turi būti galimybė sukurti užduotį.	Bandoma sukurti ir priskirti užduotį.	TC-3	Užduotis sukuriama ir priskiriama sėkmingai.	Sėkmingas
FR-1.5.1. Sistemoje turi būti galimybė stebėti testavimo procesą	Peržiūrima stebėjimo aplikacija.	TC-4	Stebėjimo aplikacijos duomenys atnaujinami realiu laiku.	Sėkmingas
FR-1.5.2. Sistemoje turi būti galimybė nutraukti testavimo procesą	Pabandoma terminuoti vieną iš procesų.	TC-4	Proceso vykdymas sustabdomas.	Sėkmingas
FR-1.4.7. Sistemoje turi būti galimybė užpildyti užduoties rezultatų formą	Pabandoma užpildyti užduoties rezultatus.	TC-5	Užduoties rezultatai išsaugomi sėkmingai.	Sėkmingas
FR-1.4.2. Sistemoje turi būti galimybė peržiūrėti užduotį	Bandoma peržiūrėti užduotį.	TC-5	Užduoties duomenys atvaizduojami korektiškai.	Sėkmingas

Testuojami reikalavimai	Atliekami veiksmai	Nuoroda į TC	Laukiamas rezultatas	Gautas rezultatas
FR-1.4.4. Sistemoje turi būti galimybė peržiūrėti užduočių sąrašą.	Bandoma peržiūrėti užduočių sąrašą.	TC-5	Užduočių sąrašas atvaizduojamas korektiškai.	Sėkmingas
FR-1.4.6. Sistemoje turi būti galimybė pakomentuoti užduotį.	Bandoma sukurti komentarą prie užduoties.	TC-5	Komentaras išsaugomas.	Sėkmingas
FR-1.4.3. Sistemoje turi būti galimybė patikrinti užduoties rezultatus	Naudotojas vykdo rezultatų tikrinimo užduotį.	TC-6	Rezultatų tikrinimo užduotis egzistuoja, jos duomenys išsaugomi.	Sėkmingas
FR-1.4.5. Sistemoje turi būti galimybė pataisyti užduoties rezultatus.	Naudotojas vykdo rezultatų taisymo užduotį.	TC-7	Rezultatų taisymo užduotis egzistuoja, jos duomenys išsaugomi.	Sėkmingas
FR-1.4.8. Sistemoje užduotys turi būti vykdomos pagal vykdomąjį veiklos modelį.	Naudotojas peržiūri, ar sistemoje yra sudiegti vykdomieji modeliai.	-	Sistemoje sudiegti vykdomieji modeliai.	Sėkmingas

Atlikus funkcinių reikalavimų testavimą, rastos 5 klaidos, kurios buvo ištaisytos testavimo metu. Ištaisius klaidas, galima teigti, jog eksperimentinė sistema atitinka analizės etape iškeltus funkcinius reikalavimus.

4.2. Nefunkciniai testai

Šiame poskyryje pateikiami nefunkcinių testų rezultatai. Detaliai ištestuojami visi analizės etape apibrėžti nefunkciniai reikalavimai ir fiksuojami gauti rezultatai (8 lentelė). Gautų rezultatų lentelėje pateikiama tik po vieną ištestuotą formą, daugiau testavimo atvejų pateikiama prieduose (5).

8 lentelė. Nefunkcinių reikalavimų testavimo rezultatai

Nefunkcinis reikalavimas	Atliekami veiksmai	Nuoroda į TC	Laukiamas rezultatas	Gautas rezultatas
NFR-1. Sistema apriboja galimybę naudotojui suklysti įvedant duomenis – kur įmanoma, laukuose pateikiamas „pasirinkimų sąrašas“, nurodomos pradinės reikšmės.	Peržiūrėti, ar naudojami pasirinkimų sąrašai.	NFR-1	Prie įvedimo laukų pateikiama pagalbinė informacija.	Sėkmingas
NFR-2. Klaidos ir sėkmingos operacijos atlikimo atvejais, sistema turi pateikti vartotojui atsako pranešimus. Klaidos atveju – pateikiamas raudonos spalvos klaidos pranešimas, sėkmės atveju – pateikiamas žalios spalvos informacinis pranešimas.	Ištestuoti visas situacijas sistemoje.	NFR-2	Klaidos, sėkmingos operacijos atlikimo atvejais vartotojui pateikiami atsako pranešimai.	Sėkmingas
NFR-3. Atliekant šalinimo, redagavimo ar kūrimo funkcijas yra galimybė atšaukti funkcijos vykdymą.	Peržiūrėti, ar visose formose yra mygtukas [Atšaukti].	NFR-3	Sistemos formose yra galimybė užbaigti funkciją neišsaugojus duomenų.	Sėkmingas
NFR-4. Sistema pateikia informaciją vartotojui apie vykdomų projektų būklę tiek tekstiniu, tiek grafiniu būdu.	Patikrinti, ar projekto analizės posistemėje informacija apie projekto būseną pateikiama ne tik tekstu, bet ir grafinėmis diagramomis.	NFR-4	Analizės posistemėje grafiniu būdu pateikiama projekto informacija.	Sėkmingas
NFR-5. Esant poreikiui, turi būti galimybė modifikuoti procesą, pagal kurį veikia sistema.	Sistemos proceso diagramoje pridėti naują veiklą.	NFR-5	Sistema tinkamai funkcionuoja su nauja veikla.	Sėkmingas
NFR-6. Sistema palaikoma šiose naršyklėse: – Google Chrome 72.0.3626.121 versija; – Mozilla Firefox 65.0.2 versija; – Internet Explorer 10 / 11.	Peržiūrėti, ar naudojami pasirinkimų sąrašai.	NFR-6	Sistema veikia korektiškai su visomis naršyklėmis.	Sėkmingas
NFR-7. Sistemai reikalinga ši programinė įranga: – Java 8; – MySQL 5.7.22; – Apache Tomcat 8.5.	Peržiūrėti.	NFR-7	Peržiūrėti, ar serveryje yra tokia PĮ.	Sėkmingas

Atlikus nefunkcinių reikalavimų testavimą, rastos 3 smulkios klaidos, kurios buvo ištaisytos testavimo metu. Ištaisius klaidas, galima teigti, jog eksperimentinė sistema atitinka analizės etape iškeltus nefunkcinius reikalavimus.

4.3. Integraciniai testai

Sistemoje realizuotas elektroninių laiškų siuntimas, todėl būtina atlikti integracinį testavimą. Elektroniniai laiškai iš testavimo proceso valdymo sistemos siunčiami naudojant „JavaMail API“ metodą, siunčiant laiškus per „Gmail SMTP“ serverį, naudojant 587 portą. Integracinio testavimo rezultatai pateikiami 9 lentelė. Integracinių testų rezultatai.

9 lentelė. Integracinių testų rezultatai

Eil. Nr.	Testavimo scenarijus	Laukiamas rezultatas	Gautas rezultatas
IT-1.	Pradinė sąlyga: projekto komandos vadovas priskiria užduotį komandos nariui. 1. Komandos narys peržiūri užduočių sąrašą. 2. Komandos narys peržiūri naujai gautą užduotį. 3. Komandos narys tikrina savo elektroninio pašto dėžutę.	Komandos narys elektroniniu paštu gauna informacinį pranešimą apie priskirtą užduotį. El. laiške nurodyta užduotis ir nuoroda į pačią užduotį.	Sėkmingas
IT-2.	Pradinė sąlyga: projekto komandos vadovas priskiria užduotį komandos nariui, kuris sistemoje nenurodė savo el. pašto. 1. Komandos narys peržiūri užduočių sąrašą. 2. Komandos narys peržiūri naujai gautą užduotį.	Sistema nesustoja veikti, neįvyksta klaida, nors komandos narys ir yra nenurodęs savo elektroninio pašto.	Sėkmingas

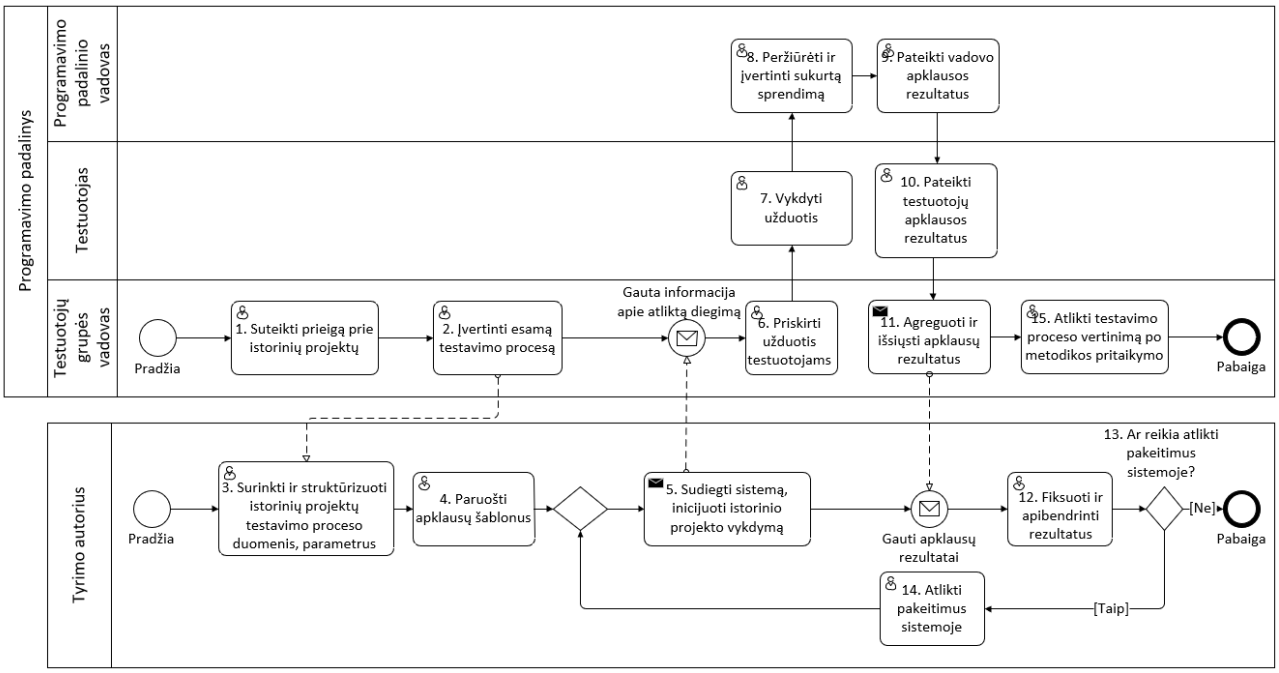
Atlikus testavimą, rastos klaidos buvo pataisytos ir atnaujinti testavimo duomenys. Ištaisius rastas klaidas, galima teikti, kad sistema atitinka jai keliamus funkcinius ir nefunkcinius reikalavimus.

5. Programinės įrangos testavimo metodikos pritaikymo IT įmonės testavimo proceso gerinime tyrimas

Norint įvertinti sukurtos programinės įrangos testavimo metodikos pritaikymo galimybes IT įmonėje, sudaromas tyrimo planas, atliekamas vertinimas, detaliai aprašomi tyrimo rezultatai, pateikiamas testavimo metodikos kokybės kriterijų įvertinimas, bei taikymo rekomendacijos.

5.1. Programinės įrangos testavimo metodikos pritaikymo IT įmonės testavimo proceso gerinime tyrimo planas

Sukurtos programinės įrangos testavimo metodikos ir eksperimentinės sistemos pritaikymo tyrimą planuojama atlikti pagal iš anksto apibrėžtą procesą (45 pav.).



45 pav. Programinės įrangos testavimo metodikos pritaikymo IT įmonės testavimo proceso gerinime planuojamo tyrimo procesas

Tyrimą planuojama vykdyti tiesiogiai bendraujant su IT įmonės darbuotojais, konsultuojant susidūrus su problemomis, realiu laiku fiksuojant projekto parametrus, bei iškilusias problemas. Siekiant ištobulinti sukurtą metodiką, bei jos demonstracinę sistemą tyrimą planuojama vykdyti keliomis iteracijomis, jeigu tik sistemos naudotojai ar kiti suinteresuoti asmenys pateiks įžvalgas apklausoje ir nurodys vietas, kurias reikėtų tobulinti. Planuojamas tyrimo procesas susideda iš šių veiklų:

1. suteikti prieigą prie istorinių projektų. Prieš pradedant vykdyti sukurtos testavimo metodikos tyrimą, tyrimo autoriui turi būti suteikiama prieiga prie istorinių projektų. Teises prie projektų suteikia testuotojų grupės vadovas;
2. įvertinti esamą testavimo procesą įmonėje. Vertinimą atlieka testuotojų grupės vadovas (apklausa pateikiama 8-tame priede);
3. surinkti ir struktūrizuoti istorinių projektų testavimo proceso duomenis, parametrus. Tyrimo autorius išanalizuoja gauto istorinio projekto duomenis, identifikuoja esmines projekto charakteristikas: atliekamų testuotojų ir vadovo veiksmų kiekis esamoje testavimo proceso valdymo sistemoje, darbų koordinavimui skirtas laikas, naudoti testavimo metodai;

4. paruošti apklausų šablonus. Tyrimo autorius paruošia apklausų šablonus, kurie būtų tinkami visiems suinteresuotiems asmenims apklausti;
5. sudiegti sistemą, inicijuoti istorinių projektų vykdymą. Tyrimo autorius sudiegia sistemą, suteikia prieigą suinteresuotiems asmenims;
6. priskirti užduotis testuotojams. Testuotojų grupės vadovas gavęs informaciją apie sudiegtą sistemą, inicijuoja istorinio projekto vykdymą, priskiria užduotis testuotojams, susipažįsta su visu sistemos funkcionalumu;
7. vykdyti užduotis. Testuotojai užpildo rezultatus gautoms užduotims;
8. peržiūrėti ir įvertinti sukurtą sprendimą. Programavimo padalinio vadovas peržiūri, įvertina sukurtą sprendimą, jeigu reikia, identifikuoja vietas, kurias būtų galima patobulinti;
9. pateikti vadovo apklausos rezultatus. Programavimo padalinio vadovas pateikia apibendrintus savo vertinimo rezultatus, užpildo apklausą;
10. pateikti testuotojų apklausos rezultatus. Testuotojai užpildo apklausą, pateikia savo įžvalgas ir atsiliepimus;
11. agreguoti ir pateikti apklausų rezultatus. Testuotojų grupės vadovas užpildo savo apklausą ir surinktus viso skyriaus apklausų duomenis perduoda tyrimo autoriui;
12. fiksuoti ir apibendrinti rezultatus. Tyrimo autorius išanalizuoja surinktus rezultatus, atlieka įžvalgas ir kokybinį vertinimą;
13. atlikti tikrinimą, ar reikia atlikti pakeitimus sistemoje? Tyrimo autorius nusprendžia, ar reikia atlikti pakeitimus sukurtoje metodikoje ir vykdyti dar vieną tyrimo iteraciją.
14. atlikti pakeitimus sistemoje. Ši veikla vykdoma, jeigu 12-ame žingsnyje tyrimo autorius nusprendžia, jog reikia atlikti pakeitimus sistemoje/ kuriamoje sistemoje. Jeigu reikia, tuomet pakeitimai atliekami ir tyrimas vėl tęsiamas sugrįžtant į ketvirtą veiklą.
15. atlikti testavimo proceso vertinimą po metodikos pritaikymo. Vertinimą atlieka testuotojų grupės vadovas (apklausa pateikiama 8-tame priede).

Siekiant įvertinti sukurtos programinės įrangos testavimo metodikos pritaikymo galimybes IT įmonėje, tyrimo metu surinkta istorinių projektų informacija, konkretūs parametrai bus lyginami su tų pačių projektų vykdymo sukurtoje eksperimentinėje sistemoje parametrais. Surinkta informacija bus vertinama lyginant projektų vykdymo charakteristikas (5.1.1) ir vertinant surinktus atsiliepimus iš suinteresuotų asmenų (5.1.2).

5.1.1. Programinės įrangos testavimo proceso charakteristikų palyginimas

IT įmonėje, kuri dalyvauja tyrime, visų projektų testavimo procesas fiksuojamas „Open Project“ sistemoje, kuri nėra grindžiama veiklos procesais, užduotys kiekvienam projektui individualiai kuriamos testuotojų grupės vadovo rankiniu būdu, testavimo metodai parenkami prie projekto dirbančio atsakingo testuotojo. Taip testuotojai kiekvieną dieną „Open Project“ sistemoje žymi prie užduoties praleistą laiką, o užsibaigus projektui, testuotojų grupės vadovai generuoja ataskaitas, kuriose pateikiami neagreguoti duomenys apie atliktas veiklas ir praleistą laiką. Be to, užduoties rezultatų pateikimui papildomai naudojama „Outlook“ programinė įranga, kurią naudojant grupės vadovas informuoja testuotojus apie naują projektą, o testuotojai užbaigę užduotį siunčia laišką grupės vadovui, pateikia užduoties rezultatus, bei nurodo, jog atliko užduotį. Sekančią užduotį rankiniu būdu vėl inicijuoja testuotojų grupės vadovas. Dėl įvardintų priežasčių buvo išskirtos palyginimo charakteristikos: užduočių kiekis, kurį turi įvykdyti testuotojų grupės vadovas ir testuotojai, laikas skirtas koordinuojantiems darbams (projekto inicijavimas, užduočių sukūrimas, priskyrimas, praleisto laiko prie užduoties fiksavimas, užduoties uždarymas, rezultatų pateikimas),

testavimo metodų parinkimo kokybiškumas. Planuojama paimti 2019 metais vykdytus projektus, išanalizuoti jo duomenis (suskaičiuoti vykdytų užduočių kiekį, apskaičiuoti koordinavimui reikalingą laiką, identifikuoti naudotus testavimo metodus) ir tą patį projektą inicijuoti sukurtos programinės įrangos testavimo metodikos demonstracinėje sistemoje. Užbaigus vykdyti projektą demonstracinėje sistemoje planuojama palyginti gautus parametrų rezultatus su 2019 metais vykdyto projekto parametrų rezultatais. Tokiu būdu siekiama identifikuoti ir įvertinti, kokią įtaką daro sukurta testavimo metodika ir jos realizacinė sistema realiam testavimo procesui.

5.1.2. Programinės įrangos testavimo metodikos eksperimentinės sistemos vertinimas iš suinteresuotų asmenų perspektyvos

Sukurta programinės įrangos testavimo metodika ir jos eksperimentinė sistema daro įtaką įvairiems suinteresuotiems asmenims. Tiriant sukurtos metodikos pritaikymo galimybes IT įmonės testavimo proceso gerinime bus analizuojama šių suinteresuotų asmenų nuomonė, bei įžvalgos: testuotojai, testuotojų grupės vadovė (tyrimo autorė), projektų vadovas, bei programavimo padalinio vadovas.

Testuotojai yra pagrindiniai testavimo proceso dalyviai, todėl jų darbą sukurta testavimo metodika, bei demonstracinė sistema įtakos labiausiai. Pirmiausia, testuotojai gali pateikti itin naudingų įžvalgų iš technologinės pusės – palyginti standartinę iki šiol naudotą informacinę sistemą su veiklos procesais grindžiama sukurta sistema. Taip pat testuotojai galės įvertinti sukurtos sistemos pateikiamus rekomenduojamus testavimo metodus su tais, kuriuos patys pasirinko ir naudojo 2019 metais vykdydami projektą. Be to, testuotojų darbą turėtų palengvinti sukurta sistema, nes veiklos procesais grindžiamoje sistemoje nereikės patiems kiekvieną dieną žymėti prie užduoties praleisto laiko, užduoties rezultatų nereikės siųsti atskiru laišku vadovui, rezultatai bus pateikiami sistemoje. Taip pat nereikės klausti dėl tolimesnių darbų, nes testuotojas automatiškai pagal veiklos procesą gaus sekančią užduotį. Visa ši iš testuotojų surinkta informacija bus itin naudinga lyginant tarpusavyje procesų charakteristikas. Testuotojų atsiliepimai bus renkami atliekant apklausą.

Testavimo grupės vadovė taip pat yra testavimo proceso dalyvė ir sukurtos sistemos naudotoja. Testavimo grupės vadovė stebi testuotojų progresą, mokina naujus testuotojus, priskiria ir kuria užduotis, bei tikrina atliktų užduočių rezultatus. Testuotojų vadovė pateiks įžvalgas iš proceso stebėjimo, koordinavimo, bei naujų testuotojų mokymo perspektyvos. Testuotojų grupės vadovė bus apklausiama atliekant apklausą.

Projektų vadovas inicijuoja projekto vykdymą, jam itin aktuali testavimo proceso kokybė, projekto terminai, užduočių atlikimo trukmė. Projektų vadovas kaip suinteresuotas asmuo pateiks nuomonę dėl sistemos siūlomų testavimo metodų, laiko reikalingo testavimo proceso koordinavimui. Projektų vadovas bus apklausiamas interviu principu.

Programavimo padalinio vadovas sukurta testavimo metodika, bei demonstracine sistema nesinaudos, bet šis asmuo yra atsakingas už finansų paskirstymą, vidinių sistemų plėtrą, bei palaikymą. Dėl šios priežasties padalinio vadovo nuomonė itin įdomi vertinant sukurtos sistemos palaikymo ir plėtros kaštus atsižvelgiant į pasirinktą realizacinę technologiją. Programavimo padalinio vadovas bus apklausiamas interviu principu.

Apklausus visus įvardintus suinteresuotus asmenis atliekamas apklausų apibendrinimas ir išskiriamos esminės įžvalgos, kurios pateikiamos kaip rekomendacijos kitoms įmonėms siekiant pritaikyti ir naudoti tyrimo darbe sukurtą programinės įrangos testavimo metodiką, bei demonstracinę sistemą.

5.2. Programinės įrangos testavimo metodikos pritaikymo IT įmonės testavimo proceso gerinime tyrimo rezultatai

5.2.1. Programinės įrangos testavimo metodikos pritaikymo eksperimentas ir testavimo proceso charakteristikų palyginimo rezultatai

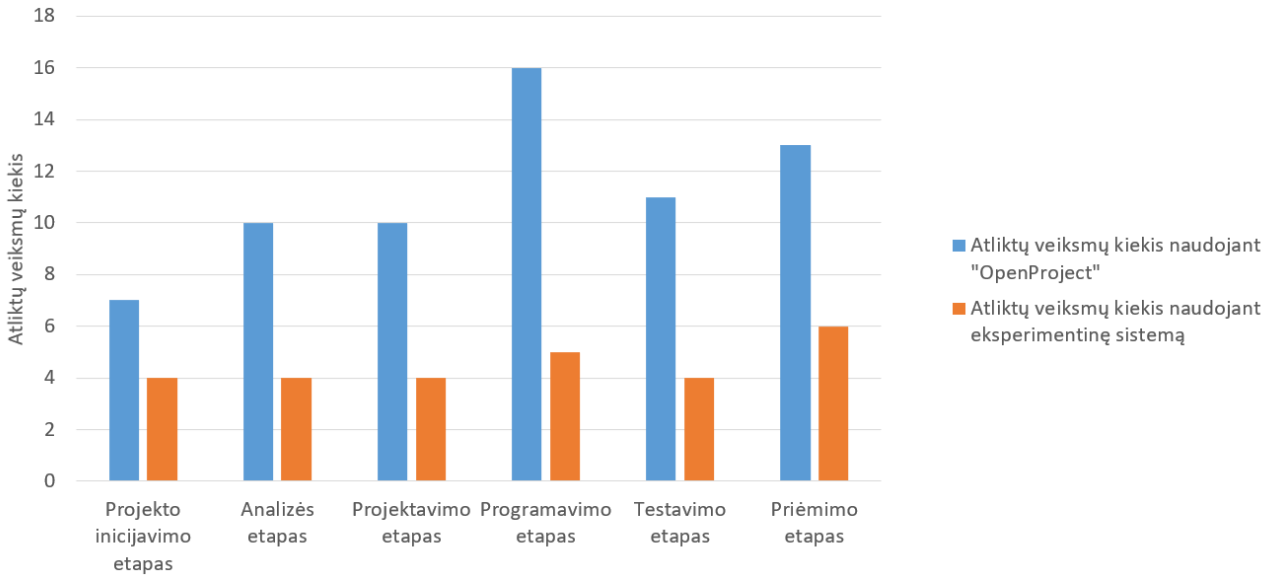
Prieš pradedant programinės įrangos testavimo metodikos, bei jos eksperimentinės sistemos pritaikymo IT įmonėje galimybių tyrimą, buvo nuspręsta įvertinti šiuo metu įmonėje naudojamo testavimo proceso kokybę. Vertinimas buvo atliekamas pagal knygoje [23] aprašytą vertinimo metodą, kuris apima aštuonias vertinamas sritis: testavimo aplinkos planavimas, testavimo proceso valdymas, testavimo proceso naudojimas, testavimo įrankių naudojimas, testavimo mokymų vykdymas, naudotojų (klientų) pasitenkinimo užtikrinimas, testavimo įverčių naudojimas, testavimo kokybės valdymas. Atlikus tyrimą dar kartą buvo vykdomas vertinimas pagal tą patį metodą ir gauti rezultatai buvo lyginami tarpusavyje. Detali gautų rezultatų kokybinė analizė pateikiama 5.3 skyriuje.

Tyrimą nuspręsta vykdyti dviem etapais. Pirmame etape tiriamas vienas istorinis projektas ir skaičiuojama, kiek naudotojas atlieka veiksmų vykdant pagrindines testavimo proceso veiklas. Antrame etape siekiama ištirti ir palygini metodus, kuriuos siūlo sistema naudoti. Antrojo etapo vykdymo metu naudojami trys istoriniai projektai ir vienas reikalavimas iš einamu laiko momentu vykdomo projekto. Istoriniai projektai buvo atrinkti atsižvelgiant į tuo metu prie projekto dirbančių testuotojų patirtį. Prie pirmojo projekto dirbo 3 mėn. patirties turėjęs testuotojas, prie antro projekto 1,5 metų patirties turėjęs testuotojas ir prie trečio 2,5 metų dirbantis testuotojas. Visiems trims projektams iš istorinio archyvo buvo atrinkti duomenys apie projektų metu naudotus testavimo metodus. Turint reikiamą istorinių projektų informaciją, projektai inicijuojami eksperimentinėje aplinkoje, užfiksuojami sistemos siūlomi testavimo metodai ir gauti duomenys palyginami su istoriniais duomenimis. Taip pat antrame etape naudojamas vienas iš einamu laiko momentu vykdomų projektų prie kurio dirba 4 mėn. patirties turintis testuotojas. Siekiama ištirti, ar sistemos siūlomi testavimo metodai padėtų nedaug patirties turinčiam testuotojui atlikti testavimą kokybiškiau. Testuotojui priskiriamas reikalavimas, jis pagal savo nuomonę parašo testavimo scenarijus, ištestuoja reikalavimą ir baigus darbus, pradedama nauja reikalavimo testavimo iteracija, kurios metu testuotojas parašo papildomus testavimo scenarijus ir ištestuoja pakartotinai reikalavimą pagal sistemos siūlomus metodus.

Pirmas tyrimo etapas (atliekamų veiksmų kiekio tyrimas)

Pirmas tyrimo etapas atliekamas su vienu istoriniu projektu. Pasirinktas projektas inicijuojamas eksperimentinėje sistemoje ir fiksuojami naudotojų atliekami veiksmai. Projekto vykdymas eksperimentinėje sistemoje imituojamas lyg būtų vykdomas iš tikrųjų: veiklas vykdo pagal procesą numatyti darbuotojai. Projektą inicijuoja projekto vadovas, testuotojų grupės vadovas įvertina projektą, paskirsto testuotojams užduotis. Projektas įvykdomas nuo inicijavimo etapo iki priėmimo. Vykdant tyrimą fiksuojami atliekamų veiklų kiekiai kiekviename etape ir supildoma eksperimentui skirtų duomenų lentelė (9 Priedas).

Atlikus pirmąją tyrimo dalį, supildomi istoriniai projekto duomenys (2019m., kai projektas buvo vykdomas iš tikrųjų), skirti eksperimento tyrimui (9 Priedas) ir gauti rezultatai palyginami su eksperimentinėje sistemoje vykdyto projekto rezultatais (46 pav.).



46 pav. Atliekamų veiksmų kiekio palyginimas įmonės naudojamoje ir eksperimentinėje sistemoje

Atliekami veiksmai buvo apskaičiuojami kiekvienam projekto etapui individualiai. Atlikta duomenų analizė atspindi, kad visuose projekto etapuose reikiamų atlikti veiksmų kiekis sumažėjo. Pagrindinė sumažėjusio veiksmų kiekio priežastis – veiklos procesų valdymu grindžiamoje eksperimentinėje sistemoje nereikia kiekvienam projekto etapui kurti užduočių rankiniu būdu, žymėti praleistų valandų skaičiaus, priskirti testuotojų komandos. Eksperimentinėje sistemoje užduotys yra kuriamos automatiškai pagal iš anksto numatytą procesą, testuotojų komanda atrenkama pagal sprendimo priėmimo lentelėje nurodytas taisykles, o užduoties pradžios ir pabaigos laikas fiksuojamas automatiškai. Vertinant visą procesą, neišskaidant į atskirus etapas, remiantis surinktais duomenimis (9 Priedas), galima teigti, kad vykdant projektą eksperimentinėje sistemoje atliekamų veiksmų skaičius sumažėja 59,7%.

Antras tyrimo etapas (naudojamų testavimo metodų tyrimas)

Antro tyrimo etapo metu buvo naudojami trys 2019 m. vykdyti istoriniai projektai, siekiant palyginti projektų vykdymo metu naudotus metodus su eksperimentinės sistemos siūlomais metodais. Istoriniai projektai pasirinkti atsižvelgiant į tuo metu prie projektų dirbusių testuotojų patirtį. Toks sprendimas priimtas, nes testuotojų patirtis gali tiesiogiai daryti įtaką metodų pasirinkimo kokybei. Mažiau patyrę testuotojai gali neturėti pakankamai teorinių žinių apie egzistuojančius testavimo metodus, dėl to buvo pasirinkti skirtingas kompetencijas turinčių testuotojų projektai, siekiant užtikrinti, kad tyrimas apimtų kuo daugiau galimų atvejų.

Inicijavus visus tris istorinius projektus eksperimentinėje sistemoje buvo užfiksuoti sistemos siūlomi metodai ir sudaryta palyginimo lentelė (10 lentelė) su iš tikrųjų projektuose naudotais testavimo metodais.

10 lentelė. Testavimo metodų naudotų istoriniuose projektuose ir vykdant projektus eksperimentinėje sistemoje palyginimas

Dirbusio testuotojo patirtis	Naudoti testavimo metodai vykdant projektą 2019m.	Atrinkti testavimo metodai eksperimentinėje sistemoje
3 mėn.	1. Panaudojimo atvejų testavimas.	1. Panaudojimo atvejų testavimas. 2. Ekvivalentus padalinimas. 3. Ribinių reikšmių analizė. 4. Teiginių padengimo testavimas.
1,5 m.	1. Panaudojimo atvejų testavimas 2. Ekvivalentus padalinimas. 3. Ribinių reikšmių analizė.	1. Panaudojimo atvejų testavimas. 2. Ekvivalentus padalinimas. 3. Ribinių reikšmių analizė. 4. Sprendimų testavimas. 5. Klaidų spėjimas.
2,5 m.	1. Panaudojimo atvejų testavimas. 2. Ekvivalentus padalinimas. 3. Ribinių reikšmių analizė. 4. Klaidų spėjimas	1. Panaudojimo atvejų testavimas. 2. Ekvivalentus padalinimas. 3. Ribinių reikšmių analizė. 4. SP lentelių testavimas. 5. Sprendimų testavimas. 6. Klaidų spėjimas.

Atlikus įmonėje naudotų testavimo metodų analizę, nustatyta, kad testuotojai nenaudoja baltosios dėžės metodų ir retai naudoja praktika grindžiamą testavimą. Baltosios dėžės testavimas itin naudingas programavimo etape testuojant sistemos vienetus. Baltosios dėžės testavimo metodai padeda aptikti ankstyvoje programos fazėje logines, spausdinimo, bei nekorektiškas funkcijas programiniame kode [33]. Dėl šių priežasčių testavimo metodikoje baltosios dėžės metodai yra įtraukti ir eksperimentinė sistema juos rekomenduoja naudoti. Nepaisant to, baltosios dėžės testavimo metodai reikalauja techninių žinių ir įrankių, todėl įmonė turėtų įvertinti turimus resursus, testuotojų kompetenciją ir nuspręsti individualiai, ar baltosios dėžės testavimo metodus reikėtų pradėti taikyti. Priešingai nei baltosios dėžės metodai – praktika grindžiami metodai yra mažiau reiklūs resursams. Jeigu įmonėje patyrę testuotojai pagal savo sukauptą patirtį sudarytų standartinių, dažnai pasitaikančių klaidų sąrašą ir juo pasidalintų su pradedančiaisiais testuotojais – įmonės visi testuotojai galėtų taikyti praktika grindžiamą testavimo metodą, grindžiamą turimu sąrašu. Kiti praktika grindžiami metodai reikalauja patirties analogiškuose projektuose, tokio tipo metodus galėtų plačiau taikyti patirties turintys testuotojai. Pagrindiniai praktika grindžiamų metodų privalumai: apima dažnai kitų testavimo metodų netestuotas vietas, greičiau surandamos klaidos, nereikalauja didelių testavimų ataskaitų, metodai yra neformalūs [34]. Šių metodų taikymas įmonėje galėtų prisidėti prie testavimo rezultatų ir kokybės gerinimo.

Antrame tyrimo etape taip pat buvo vykdomas papildomas eksperimentas su einamu laiko momentu vykdomu projektu prie kurio dirbo 5 mėn. patirties turintis testuotojas. Buvo nuspręsta palyginti testuotojo darbą nenaudojant sukurtos eksperimentinės sistemos ir naudojant. Eksperimentui buvo pasirinktas vienas reikalavimas, kuris buvo susietas su vienu panaudos atveju. Pasirinktas reikalavimas turėjo būti realizuotas dviejose analogiškose darbalaukio aplikacijos posistemėse (reikalavimą abiejose posistemėse realizavo tas pats programuotojas). Pirmai posistemei testuotojas rašė testavimo scenarijus ir testavo reikalavimą pats pasirinkęs panaudojimo atvejų testavimo metodą. Testuotojas sukūrė du testavimo atvejus pagrindiniam ir alternatyviam panaudos atvejo scenarijui, ištestavo reikalavimą ir rado vieną klaidą. Testuojant antrąją posistemę testuotojas naudojo

eksperimentinę sistemą, kuri pasiūlė naudoti šiuos testavimo metodus: panaudojimo atvejų testavimas, ekvivalentus padalinimas, ribinių reikšmių analizė, teiginių padengimo testavimas. Prieš inicijuojant testavimo užduotis visus metodus įvertino testuotojų grupės vadovas ir iš sistemos siūlyto sąrašo atsisakė „teiginių padengimo testavimas“ metodo, nes dirbantis testuotojas neturėjo reikiamų techninių žinių. Atliekant testavimą pagal eksperimentinės sistemos metodus testuotojas pasirašė 5 testavimo scenarijus (pagrindiniam panaudos atvejo scenarijui parašyti 4 testavimo scenarijai orientuoti į ekvivalenčias klases ir jų ribines reikšmes ir vienas scenarijus parašytas alternatyviam panaudos atvejo scenarijui) ir pagal juos atliko testavimą. Testavimo metu antroje posistemėje buvo rastos trys klaidos. Nusprendus grįžti ir patikrinti, ar pirmoje posistemėje egzistuoja tos pačios klaidos – spėjimas pasitvirtino ir paaiškėjo, kad pirmoje posistemėje viena klaida sutampa ir yra užregistruota, o likusios dvi liko nepastebėtos. Natūralu, kad vykdant daugiau testavimo scenarijų išauga testavimo apimtis ir padidėja rastų klaidų skaičius, tačiau interviu principu apklausus dirbusį testuotoją, jis įvardino eksperimentinės sistemos privalumus. Kaip teigė eksperimente dalyvavęs testuotojas antrąją posistemę buvo lengviau testuoti, nes sistema pateikė galimus testavimo metodus, kurių iki šiol testuotojas nežinojo. Pasidomėjus apie siūlomus metodus, mažai patirties turėjusiam testuotojui tapo lengviau sugalvoti daugiau testavimo scenarijų, kuriuos vykdant pavyko aptikti daugiau klaidų nei testuojant tik pagal dokumentacijoje aprašytus panaudos atvejus.

Atlikus antrąjį tyrimo etapą buvo nustatyta, jog IT įmonėje yra nenaudojami baltosios dėžės ir retai naudojami praktika grindžiami testavimo metodai. Šių tipų metodus rekomenduoja naudoti eksperimentinė sistema, nes metodai suteikia galimybę ankstyvoje fazėje aptikti logines klaidas programiniame kode ir gali sutrumpinti klaidų aptikimo laiką, jeigu testuotojas turi pakankamai patirties. Atsižvelgiant į tyrimo rezultatus įmonei buvo rekomenduota įvertinti visas aplinkybes ir apsvarstyti galimybes pradėti naudoti baltosios dėžės testavimo metodus ir plačiau taikyti patirtimi grindžiamus metodus. Taip pat antrame etape buvo atliktas papildomas eksperimentas, kurio metu nustatyta, jog eksperimentinė sistema gali padėti mažai patirties turintiems testuotojams lengviau generuoti testavimo atvejus pagal siūlomų testavimo metodų aibę. Be to, papildomo tyrimo metu buvo nustatyta, jog eksperimentinė sistema visais atvejais siūlo naudoti bent vieną baltosios dėžės metodą, neatsižvelgiant, ar dirbantis testuotojas turi pakankamai patirties.

Be to, dar liko keletas sudėtingai ištiriamų eksperimentinės sistemos vietų, kurios gali daryti teigiamą įtaką IT įmonės testavimo procesui. Pavyzdžiui, dirbant įmonėje pastebėta, kad vykdant užduotis dažnai atsiranda uždelsimas: pabaigus užduotį testuotojai rašo laišką vadovui, laukia atsakymo, testuotojų vadovas kreipiasi į projekto vadovą, klausia dėl galimų užduočių, tuomet testuotojų vadovas sukuria užduotis ir priskiria testuotojams. Šio sudėtingo proceso eksperimentinėje sistemoje nelieka, nes visos užduotys vykdomos pagal numatytą procesą, kai tik projekto vadovas inicijuoja projektą sistemoje. Dėl šios priežasties teoriškai būtų galima teigti, jog eksperimentinė sistema gali prisidėti ir prie užduočių vykdymo uždelsimo mažinimo. Taip pat kitas svarbus aspektas – sistemos palaikymo kaštai. Standartinėje sistemoje pridedant vieną lauką reikia atlikti pakeitimus duomenų bazėje, formoje ir procesuose, kuriuose šis laukas bus naudojamas. Eksperimentinėje sistemoje pridėti vieną lauką reikalautų mažiau resursų, nes nereikėtų atlikti pakeitimų duomenų bazėje. „Camunda BPM“ platforma yra grindžiama dinamine duomenų baze, kurioje automatiškai kuriami duomenys pagal formose pridėtus naujus parametrus. Taip pat veiklos procesais grindžiamos sistemos diegimas yra kur kas paprastesnis nei standartinės sistemos. Atliekant veiklos procesais grindžiamos sistemos diegimą, galima tiesiog įkelti naujus vykdomuosius modelius paliekant senuosius ir dvi procesų versijos sistemoje galės veikti nepriklausomai viena nuo kitos. Kai tik

pasibaigia neaktualios procesų versijos darbai, senuosius procesus galima archyvuoti ir dirbti tik su naujai sudiegtomis procesų versijomis.

5.2.2. Programinės įrangos testavimo metodikos eksperimentinės sistemos suinteresuotų asmenų vertinimo rezultatai

Atliekant tyrimą ir vykdant istorinius projektus eksperimentinėje sistemoje tiesiogiai su sistema tiesiogiai dirbo du testuotojai, testuotojų grupės vadovė ir projekto vadovė. Likusiems skyriaus testuotojams, bei programavimo padalinio vadovui sistema buvo pademonstruota ir pristatytas pagrindinis funkcionalumas. Siekiant įvertinti tyrimo rezultatus iš skirtingų darbuotojų perspektyvos, testuotojams ir testuotojų grupės vadovei buvo sudaryti apklausų šablonai (7 Priedas), o programavimo padalinio vadovas ir projektų vadovė buvo apklausti interviu principu.

Atlikus IT įmonės testuotojų skyriuje apklausą apie sukurtą metodiką ir eksperimentinę sistemą, bei agregavus testuotojų, testuotojų grupės vadovės, bei projektų vadovės apklausų rezultatus, galima išskirti pagrindinius įvardintus sistemos privalumus:

- Automatinis užduočių kūrimas ir inicijavimas. Sukurtoje eksperimentinėje sistemoje testavimo proceso vykdymui reikalingos užduotys generuojamos ir inicijuojamos automatiškai pagal sukurtus vykdomuosius veiklos procesus. Tai palengvina testuotojų vadovo darbą, nes reikia mažiau laiko skirti koordinavimo darbams ir galima daugiau dėmesio skirti pačio testavimo vykdymui.
- Procesų stebėjimo posistemėje galima peržiūrėti ir eksportuoti vykdomų, bei jau įvykdytų procesų duomenis įvairiais pjūviais. Projektų vadovas ir testuotojų grupės vadovas eksperimentinėje sistemoje gali pasirinkti norimus duomenų filtravimo kriterijus ir peržiūrėti, bei eksportuoti duomenis tiek procesų, tiek veiklų lygyje. Lyginant su įmonėje naudojama sistema, pačioje eksperimentinėje sistemoje galima atlikti daugiau paieškos ir duomenų agregavimo veiksmų, todėl nereikia papildomai redaguoti sugeneruotos ataskaitos duomenų, priešingai nei naudojant „OpenProject“ sistemą.
- Trumpesnis naujokų mokymo apie testavimo etapus, užduočių vykdymo eiliškumą periodas. Siekiant sutrumpinti naujokų mokymo laiką, testuotojų grupės vadovė identifikavo vieną iš galimų mokymo alternatyvų – vykdomo veiklos modelio ir jo aprašo pateikimą naujokams. Vykdomieji modeliai yra pakankamai aiškūs ir atspinti testavimo etapus, vykdomas veiklas, atsakomybes, veiklų vykdymo eiliškumą, todėl modelius galima naudoti mokinant naujus darbuotojus.
- Vieningas testavimo procesas testavimo skyriuje. Palengvina perėjimą iš vieno projekto į kitą, lengviau aprašyti testavimo procesą įvairiuose dokumentuose, skyriaus darbuotojams paprasčiau suprasti kas ir už ką yra atsakingas.
- Automatizuotas sprendimų priėmimas (testuotojų grupės priskyrimas, testavimo metodų rekomendavimas). Eksperimentinėje sistemoje automatiškai yra priskiriama testavimą galinti vykdyti testuotojų grupė, taip pat automatiškai pagal iš anksto numatytas taisykles priskiriami testavimo metodai. Tai palengvina testuotojų grupės vadovo, bei projektų vadovo darbą, nes nereikia minėtų veiklų atlikti rankiniu būdu.

Dirbant su eksperimentine sistema taip pat iškilo keletas nesklandumų ir buvo įvardinti šie sistemos trūkumai:

- Nepakankamas žinių kiekis reikalingas rekomenduojamų metodų taikymui. Eksperimentinė sistema gali pateikti baltosios dėžės testavimo metodus, kuriems reikia tam tikrų techninių

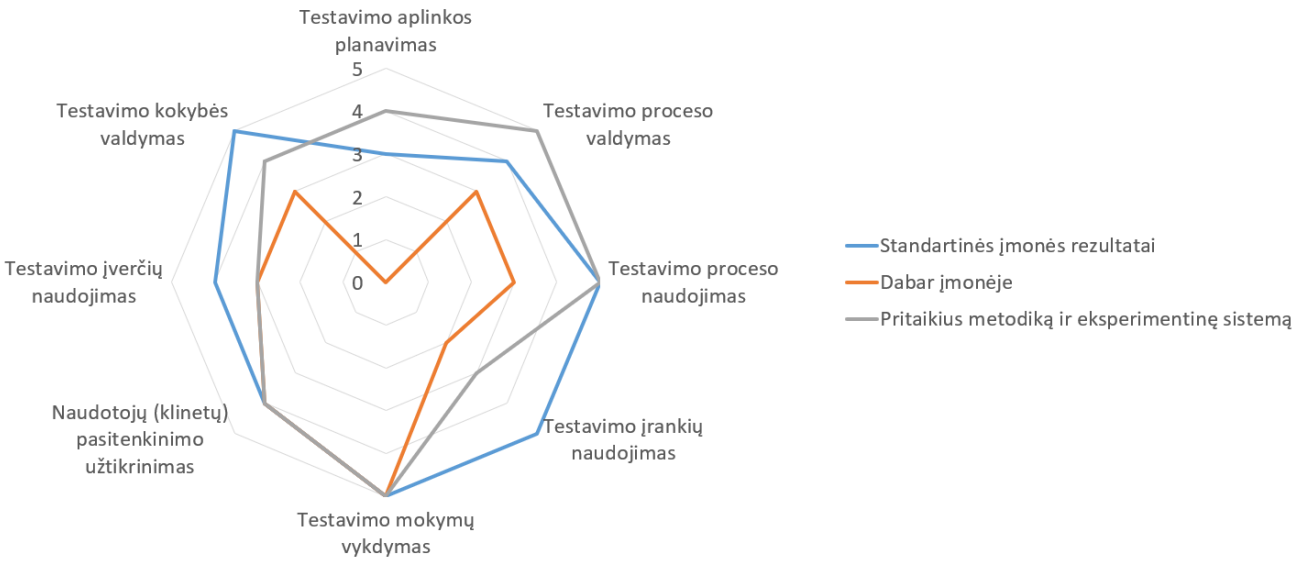
žinių. Dėl šios priežasties testuotojų grupės vadovas turi peržiūrėti, įvertinti situaciją, testuotojo kompetenciją ir tam tikrais atvejais atsisakyti kai kurių sistemos siūlomų metodų, o tai reikalauja papildomo laiko.

– Nepakankamai tikslus užduoties vykdymo laikotarpis. Eksperimentinėje sistemoje pateikiamas užduoties vykdymo laikotarpis skaičiuojamas nuo užduoties priskyrimo momento iki užduoties užbaigimo. Jeigu darbuotojas užduoties priskyrimo metu dirba prie kitos užduoties, tuomet užduoties vykdymas bus pradedamas ne iškart, o po kurio laiko. Dėl šios priežasties sistemoje pateikiamas užduoties vykdymo laikotarpis negali būti naudojamas kaip metrika siekiant palyginti suplanuotus laiko resursus su realiai praleistomis valandomis.

Ne mažiau svarbi yra programavimo padalinio vadovo nuomonė apie sukurtą metodiką, eksperimentinę sistemą ir jos taikymo galimybes, nes padalinio vadovas skirsto resursus, sprendžia dėl naujų įrankių naudojimo galimybių, bei turi ilgametę patirtį IT srityje. Padalinio vadovas remiantis sukaupta patirtimi, patvirtino, jog veiklos procesais grindžiamų sistemų palaikymas reikalauja mažiau resursų, tačiau dažnai vis tiek nepavyksta atlikti pakeitimo be programuotojo įsikišimo ir negalime drąsiai teigti, jog tokio tipo sistemos yra plečiamos vos tik pakeitus vykdomąjį modelį. Dažnai realūs įmonių procesai yra sudėtingi ir tarpusavyje persipynę, didžiuliai duomenų kiekiai perduodami tarp procesų, todėl pridėjus naują lauką tenka keisti ir programinį kodą, kuris skirtas duomenų perdavimui iš vieno proceso į kitą. Nepaisant to, laiko sutaupoma, nes nereikia keisti duomenų bazės struktūros, kuri yra dinaminė, paprastesnis sistemos diegimas leidžia sutaupyti laiko ir resursų. Taip pat padalinio vadovas įvardino, jog įmonei yra svarbu turėti vieningą testavimo procesą, kurį būtų galima stebėti realiu laiku – darbuotojams didesnė motyvacija stengtis, o vadovams paprasčiau stebėti procesą, vertinti testavimo kokybę ne tik projekto apimtyje, bet ir visos įmonės mastu. Taip pat vieningas testavimo procesas gerina įmonės įvaizdį, o tai gali būti naudinga dalyvaujant viešuose konkursuose, ruošiant techninį pasiūlymą būtų galima pateikti įmonėje apibrėžtą ir aprašytą testavimo procesą. Nepaisant identifikuotų privalumų, buvo įvardytas trūkumas – eksperimente naudota „Camunda BPM“ platformos versija yra mokama ir reikėtų įvertinti, ar naudojant eksperimentinę sistemą nemokamoje „Camunda BPM“ platformos versijoje būtų išlaikomi pagrindiniai privalumai. Įmonė šiuo metu neturi galimybės pereiti nuo nemokamo įrankio prie mokamo, todėl turėtų būti atliktas papildomas nemokamos platformos versijos vertinimas.

5.3. Programinės įrangos testavimo metodikos analizė, kokybės kriterijų įvertinimas

Atlikus eksperimentą IT įmonėje pakartotinai buvo vykdomas testavimo veiklų vertinimas. Vertinimą kaip ir prieš eksperimentą vykdė testuotojų grupės vadovė. Detalus atliktų vertinimų palyginimas pateikiamas: 8 Priedas. Remiantis surinktais testavimo proceso vertinimo duomenimis sudaryta radaro tipo diagrama (47 pav.), kuri atspindi esminių testavimo proceso kokybės vertinimo kriterijų rezultatų palyginimą: esamos situacijos įmonėje, pritaikius testavimo metodiką, bei eksperimentinę sistemą ir standartinės IT įmonės duomenis pateikiamus [23] šaltinyje.



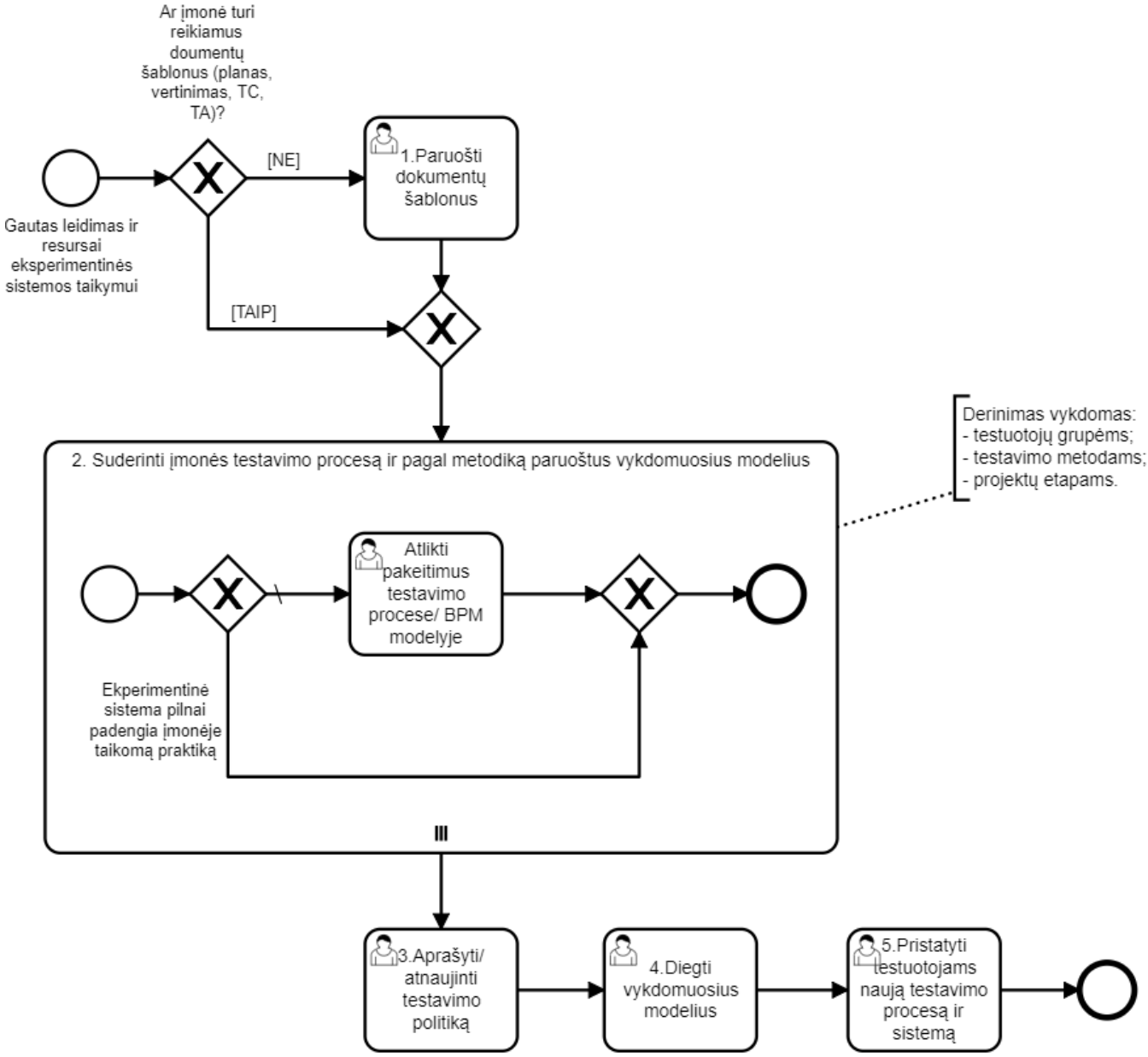
47 pav. Testavimo proceso vertinimo rezultatų palyginimas tarp standartinės IT įmonės, eksperimente dalyvaujančios įmonės istorinių projektų ir projektų vykdomų eksperimentinėje sistemoje

Atlikus testavimo proceso vertinimą ir palyginimą tarp esamos situacijos įmonėje, pritaikius testavimo metodiką, bei eksperimentinę sistemą įmonėje ir standartinės IT įmonės rodiklių, matome, jog dviejų kokybės rodiklių („Testavimo mokymų vykdymas“, „Naudotojų (klientų) pasitenkinimo užtikrinimas“) rezultatai sutampa. Šių kokybės vertinimo kriterijų eksperimentinė sistema nepadengia, todėl buvo palikti tokie patys įmonės rezultatai, nes traktuojama, kad įmonė kaip ir anksčiau pagal seną tvarką vykdytų testuotojų mokymus, bei rūpintųsi, kad klientai būtų patenkinti galutiniu produktu. Taip pat sutampa vertinimo kriterijaus „Testavimo įverčių naudojimas“ esamos situacijos įmonėje ir pritaikius eksperimentinę sistemą rezultatai, o standartinės IT įmonės rodiklis yra didesnis. Iš to būtų galima daryti išvadą, jog įmonei reikėtų perplanuoti testavimo proceso įverčių naudojimo procedūras. Visų kitų įverčių atžvilgiu lyginant esamą situaciją įmonėje ir pritaikius eksperimentinę sistemą, galima teigti, jog eksperimentinė sistema įmonės testavimo proceso kokybei suteikia ženkliai geresnius rezultatus. Itin geri rezultatai gauti vertinant kriterijus „Testavimo kokybės valdymas“, „Testavimo aplinkos planavimas“, „Testavimo proceso valdymas“, „Testavimo proceso naudojimas“. Pagrindinė gerų eksperimentinės sistemos rezultatų priežastis – testavimas, grindžiamas vieningu, realiu laiku stebimu ir valdomu vykdomuoju modeliu. Taip pat metodikos, bei eksperimentinės sistemos kūrimo metu buvo remiamasi gerosiomis praktikomis, kurios teigia, kad IT įmonė privalo turėti testavimo planą, strategiją, apibrėžtą procesą, testuotojai turi būti įtraukiami į projektą nuo pat inicijavimo etapo. Remiantis gerosiomis praktikomis buvo kuriama metodika ir sistema ir tai suteikė gerus testavimo proceso kokybės vertinimo rezultatus.

5.4. Programinės įrangos testavimo metodikos taikymo rekomendacijos

Atlikus tyrimą, buvo nustatyta, jog sukurta programinės įrangos testavimo metodika, bei jos eksperimentinė sistema gali būti pritaikyta IT įmonėje siekiant pagerinti įmonės testavimo procesą. Išanalizavus gautus tyrimo rezultatus, galima teigti, jog sukurtas sprendimas didžiausią naudą teiktų įmonėms, kurios testavimo skyriuje neturi apibrėžto vieningo testavimo proceso, naudoja universalius nemokamus įrankius, kurie pilnai nepadengia įmonėje vykdomo testavimo proceso.

Siekiant palengvinti sukurto sprendimo taikymą IT įmonės testavimo skyriuje buvo sudarytas rekomendacinių veiklų procesas (48 pav.), kuriuo remiantis būtų lengviau pritaikyti testavimo metodiką, bei eksperimentinę sistemą savo įmonėje.



48 pav. Sukurtos testavimo metodikos ir eksperimentinės sistemos rekomendacinis taikymo procesas

Rekomenduojamas sprendimo procesas apima visas reikalingas esamos situacijos įmonėje įvertinimo veiklas, veiklų suderinimo tarp eksperimentinės sistemos ir įmonėje taikomo testavimo proceso sub-procesą, bei sistemos diegimo, bei pristatymo skyriaus darbuotojams veiklas. Kiekvienai rekomenduojamai veiklai buvo sukurtas detalus aprašas (11 lentelė).

11 lentelė. Sukurtos testavimo metodikos ir eksperimentinės sistemos taikymo rekomenduojamų veiklų aprašas

Proceso žingsnis	Detalus aprašymas
1. Paruošti dokumentų šablonus	Įmonė privalo peržiūrėti visus dokumentų šablonus, kurių reikalauja eksperimentinė sistema (testavimo plano, testavimo vertinimo, testavimo scenarijų įvairiems testavimo lygiams, testavimo ataskaitų

Proceso žingsnis	Detalus aprašymas
	įvairiems testavimo lygiams ir kt.). Jeigu bent vieno šablono įmonė neturi, tuomet reikėtų paruošti.
2. Suderinti įmonės testavimo procesą ir pagal metodiką paruoštus vykdomuosius modelius	Prieš pradedant taikyti PĮ testavimo metodiką, bei jos eksperimentinę sistemą, reikėtų įvertinti, ar įmonėje taikoma testavimo politika yra suderinama su eksperimentinės sistemos. Lyginimas turi būti atliktas šiems elementams: testuotojų komandoms, projektų etapams ir testavimo metodams. Atliekant testuotojų komandų suderinamumo analizę, reikia peržiūrėti, ar įmonėje egzistuojančios testavimo komandos sutampa su sprendimo priėmimo „assign-team“ lentelėje aprašytomis komandomis. Jeigu nesutampa, tuomet reikėtų papildyti sprendimo priėmimo lentelę reikiamomis reikšmėmis arba pakeisti esamas. Atliekant projektų etapų suderinimą, reikėtų įmonėje apibrėžti procedūrą, kaip suskaidyti įmonėje vykdomus projektus į penkis etapus pagal eksperimentinę sistemą. Atliekant testavimo metodų suderinamumo analizę, įmonės atstovai turėtų peržiūrėti visus eksperimentinėje sistemoje siūlomus testavimo metodus ir įvertinti, ar visi metodai įmonėje gali būti taikomi (ar pakanka žinių, įrankių). Jeigu nusprendžiama, kad kažkurie metodai bus nenaudojami, tuomet įmonė turi apibrėžti kokie tai metodai ir visad jų atsisakyti.
3. Aprašyti arba atnaujinti turimą testavimo politiką	Pagal gautus suderinamumo rezultatus įmonė turi atsinaujinti turimą arba susikurti naują įmonėje taikomo testavimo proceso politiką, kurioje atsispindėtų eksperimentinės sistemos ir PĮ testavimo metodikos taikymas.
4. Diegti vykdomuosius modelius	Įmonė turi sudiegti vykdomuosius modelius į savo serverius ir padaryti sistemą prieinamą iš vidinio tinklo.
5. Pristatyti testuotojams naują testavimo procesą ir sistemą	Sudiegus sistemą, visi testavimo skyriaus darbuotojai, projektų vadovai, bei kiti suinteresuoti asmenys turi būti supažindinti su sistemos funkcionalumu ir vykdomais procesais.

Sukurtas detalus rekomenduojamų veiklų aprašas turėtų padėti lengviau taikyti tiriamajame darbe sukurtą testavimo metodiką, bei eksperimentinę sistemą.

Išvados

1. Atlikus probleminės srities analizę, nustatyta, jog dažnai testavimo procesas yra nestruktūrizuotas, įmonės neturi aiškiai apibrėžto proceso, kurį būtų galima pritaikyti įvairiais metodais kuriamiems projektams. Dėl šios priežasties buvo nuspręsta kurti apibendrintą testavimo metodiką, kuri atsižvelgtų į vykdomo projekto parametrus, pateiktų rekomenduojamus testavimo metodus ir nebūtų tiesiogiai susieta su konkrečiu programinės įrangos kūrimo metodu. Siekiant užtikrinti, kad kuriamą metodiką būtų galima pritaikyti ne tik klasikiniuose nuosekliuose PĮ kūrimo metoduose (pvz., „Krioklys“), bet ir moderniuose (pvz., „Scrum“), priimtas sprendimas kurti metodiką, orientuotą į bendrines standartinių IT projektų fazes: analizę, projektavimą, programavimą, testavimą ir diegimą.
2. Atlikus galimų realizacijos technologijų analizę, bei jas palyginus, buvo nuspręsta eksperimentinę sistemą kurti „Camunda BPM“ platformoje. Pasirinkta technologija ne tik suteiks galimybę vizualiai pademonstruoti sukurtą metodiką, bet ir suteiks papildomų privalumų. Remiantis sukurta metodika bus sudaromi vykdomieji BPM modeliai pagal kuriuos veiks eksperimentinė sistema. Be to, veiklos procesais grindžiama platforma suteiks galimybę realiu laiku stebėti vykdomus testavimo procesus, naudojant sprendimų priėmimo lenteles sistema automatiškai priims sprendimus pagal iš anksto testavimo metodikoje numatytas taisykles, inicijuos testavimo užduotis automatiniu būdu.
3. Atsižvelgiant į analizės etape surinktą informaciją, kuriamai metodikai ir eksperimentinei sistemai iškeltus reikalavimus, bei pasirinktą realizavimo technologiją, projektavimo metu buvo sudaryti detalūs testavimo metodikos modeliai su aprašu, bei eksperimentinės sistemos architektūriniai klasių, duomenų bazės modeliai, kurie atspindi būsimos sistemos struktūrą.
4. Pagal projektavimo etape sudarytus modelius ir detaliai aprašytą programinės įrangos testavimo metodiką buvo sukurti vykdomieji modeliai ir sudiegti į „Camunda BPM“ platformą. Sukurta eksperimentinė sistema buvo atidžiai ištestuota ir perduota bandomajai eksploatacijai. Eksperimentinės sistemos bandomosios eksploatacijos metu buvo pastebėtos ne tik smulkios sisteminės klaidos, bet ir loginės, susijusios su sukurta programinės įrangos testavimo metodika – nenumatyta regresinio testavimo veikla, nerealizuota galimybė žymėti užduoties atlikimo progreso. Atsižvelgiant į aptiktus neatitikimus buvo pataisyti ir pakartotinai sudiegti vykdomieji modeliai.
5. Sukurta metodika ir sistema buvo eksperimentiškai ištirtos, identifikuoti sukurtos metodikos privalumai ir trūkumai. Nustatyta, jog sukurta metodika, bei eksperimentinė sistema palengvina testuotojų, bei testuotojų grupės vadovo darbą. Testuotojams nebereikia kiekvienam projektui individualiai parinkti testavimo metodų, atlikus užduotį ar radus klaidą nereikia naudotis keliomis sistemomis. Testuotojų grupės vadovas naudojant sukurtą metodiką, bei eksperimentinę sistemą gali greičiau išmokyti naujus darbuotojus pateikdami sukurtos metodikos modelius, bei aprašą. Taip pat gali realiu laiku stebėti testuotojų progresą, vykdomas užduotis, vadovams nebereikia kiekvienam projektui individualiai, rankiniu būdu kurti, bei inicijuoti užduočių. Be to, buvo sudarytas programinės įrangos testavimo metodikos pritaikymo rekomendacijų sąrašas, kuris tikėtina palengvins testavimo metodikos taikymo standartinėse IT įmonėse procesą.

Literatūros sąrašas

[1] „ISO/IEC/IEEE International Standard - Systems and software engineering - Software life cycle processes,“ New York, USA:IEEE, 2017.

[2] ISO/IEC/IEEE International Standart - Systems and software engineering - Life cycle management - Part 1: Guidelines for life cycle management (24748-1-2018), New York, USA, 2018.

[3] A. Alshamrani ir A. Bahattab, „A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model,“ *IJCSI International Journal of Computer Science Issues,* t. 12, nr. 1, pp. 106-111, 2015.

[4] I. Sommerville, Software engineering, 8th ed., Harlow, England, 2007.

[5] C. &. B. V. Larman, „Iterative and incremental developments. a brief history,“ *Computer, 36(6),* pp. 47-56, 2003.

[6] J. Sheffield, „Systemic knowledge and the V-model,“ *Int. J. Business Information Systems,* t. 1, p. 83–101, 2005.

[7] A. Spilner, T. Rossner, M. Winter ir T. Linz, „Software Testing Practice: Test Management,“ 2007, pp. 28-30.

[8] Lisa Crispin, Janet Gregory, „Agile Testing,“ 2009.

[9] T. Tan, Q. Li, B. Boehm, Y. Yang, M. He ir R. Moazeni, „Productivity trends in incremental and iterative software development,“ *International Symposium on Empirical Software Engineering and Measurement,* pp. 1-3, 2009.

[10] „IBM,“ Rational Software, 1998. [Tinkle]. Available: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf.

[11] P. Abrahamsson, O. Salo, J. Ronkainen ir J. Warsta, „Agile software development methods: Review and analysis,“ *VTT publication 478,* 2002.

[12] I. Mosaic, „Mosaic’s Testing Process And the Rational Unified Process. A Powerful Union for Quality Systems,“ Mosaic, Inc, 2001.

[13] S. W. Ambler, Going Beyond Scrum. Disciplined Agile Delivery, 2013.

[14] van Den Broek, R, B. M. M. C. M. ir . v. M. H. , „Integrating testing into Agile software development processes,“ *2014 2nd International Conference on Model-Driven Engineering and Software Development,* pp. 561-574, 2014.

[15] J. Gregory ir L. Crispin, More Agile Testing: Learning Journeys for the Whole Team by Lisa, 2015.

[16] „IEEE,“ 18 Liepa 2008. [Tinkle]. Available: https://cow.ceng.metu.edu.tr/Courses/download_courseFile.php?id=4046.

[17] D. Galin, Software Quality, 2004.

[18] „ISO 25000,“ [Tinkle]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&limitstart=0.

[19] „ISTQB,“ International Software Testing Qualifications Board , 2018. [Tinkle]. Available: https://www.istqb.org/downloads/send/51-ctfl2018/208-ctfl-2018-syllabus.html.

[20] W. M. P. Van Der Aalst, Barros, F, He, X, Holgado-Terriza, J. A ir Rolland, C, „Business Process Management: A Comprehensive Survey,“ *ISRN Software Engineering,* t. 2013, 2013.

[21] O. a. c. t. c. s. w. BPMN, „Business Process Management Journal,“ *Business Process Management Journal,* t. 16(1), pp. 181-201, 2010.

[22] J. D. Srinivas Nidhra, BLACK BOX AND WHITE BOX TESTING, June, 2012.

[23] W. E. Perry, Effective Methods for Software Testing, Indianapolis, Indiana, 2006.

[24] „Mantis Bug Tracer,“ [Tinkle]. Available: https://www.mantisbt.org/.

[25] „Bugzilla,“ [Tinkle]. Available: https://www.bugzilla.org/.

[26] „Microsoft,“ [Tinkle]. Available: https://docs.microsoft.com/en-us/previous-versions/fda2bad5(v=vs.80).

[27] „Activiti,“ [Tinkle]. Available: https://www.activiti.org/.

[28] „ Bonitasoft,“ Bonitasoft, 2018. [Tinkle]. Available: https://www.bonitasoft.com/.

[29] „Camunda,“ Camunda Services GmbH, 2018. [Tinkle]. Available: https://camunda.com/.

[30] ISTQB, „Advanced Level Syllabus Test Analyst,“ 2019.

[31] R. Software, „Rational Unified Process. Best Practices for Software,“ 2010.

[32] Novak, J, Krajnc, A ir Žontar, Rok, „Taxonomy of static code analysis tools,“ *International Convention MIPRO,* pp. 418-422, 2010.

[33] D. Jagruthi ir N. Srinivas, „Black box and white box testing techniques-a literature review,“ *International Journal of Embedded Systems and Applications,* t. 2, 2012.

[34] P. C. Jorgensen, Software Testing A Craftsman’s Approach, 2014.

[35] M. Harman, C. Henard, M. Papadakis, Y. Jia ir Y. L. Traon, „Comparing White-box and Black-box Test Prioritization,“ įtraukta *Proceedings of the 38th International Conference on Software Engineering,* New York, 2016.

Priedai

1 priedas. Programinės įrangos testavimo metodų atrinkimo lentelės

12 lentelė. Juodosios dėžės testavimo metodų atrinkimo kriterijai

Ar įmanoma PĮ duomenis sugrupuoti į ekvivalenčias klases?	**Ar bent vienas komponentas yra atsakingas už SP?**	**Ar spec. pateikiamos veiklos taisyklės / darbų sekų diagramos?**	**Ar sistemoje egzistuoja būsenos?**	**Ar dok. egzistuoja būsenų modelis?**	**Ar sistema yra viena iš šių tipų: „Embedded software“, „Web software“, „Transactional Software“, „Control systems“.**	**Ar reikalavimai aprašyti panaudos atvejais?**	**Ar egzistuoja naudotojo sistemos sąsajos scenarijai?**	**Juodosios dėžės metodai (grindžiami specifikacija)**				
								Padalijimas į ekvivalenčias klases (angl. Equivalence Partitioning)	Ribinių reikšmių analizė (angl. Boundary Value Analysis)	SP lentelių testavimas (angl. Decision Table Testing)	Būsenų perėjimų testavimas (angl. States Transition Diagrams)	Panaudojimo atvejų testavimas (angl. Use Case Testing)
TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	TAIP	TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	TAIP	TAIP	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	TAIP	TAIP	NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	TAIP	TAIP	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	TAIP	TAIP	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	TAIP	NE	NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP

Ar įmanoma PĮ duomenis sugrupuoti į ekvivalenčias klases?	**Ar bent vienas komponentas yra atsakingas už SP?**	**Ar spec. pateikiamos veiklos taisyklės / darbų sekų diagramos?**	**Ar sistemoje egzistuoja būsenos?**	**Ar dok. egzistuoja būsenų modelis?**	**Ar sistema yra viena iš šių tipų: „Embedded software“, „Web software“, „Transactional Software“, „Control systems“.**	**Ar reikalavimai aprašyti panaudos atvejais?**	**Ar egzistuoja naudotojo sistemos sąsajos scenarijai?**	**Juodosios dėžės metodai (grindžiami specifikacija)**				
								Padalijimas į ekvivalenčias klases (angl. *Equivalence Partitioning*)	Ribinių reikšmių analizė (angl. *Boundary Value Analysis*)	SP lentelių testavimas (angl. *Decision Table Testing*)	Būsenų perėjimų testavimas (angl. *State Transition Diagrams*)	Panaudojimo atvejų testavimas (angl. *Use Case Testing*)
TAIP	TAIP	TAIP	NE	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	TAIP	NE	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
TAIP	TAIP	NE	TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	NE	TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	TAIP	NE	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	NE	TAIP	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	NE	TAIP	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	NE	TAIP	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
TAIP	TAIP	NE	NE	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	NE	NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	NE	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	NE	NE	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	NE	NE	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	NE	NE	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
TAIP	NE	TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP

Ar įmanoma PĮ duomenis sugrupuoti į ekvivalenčias klases?	**Ar bent vienas komponentas yra atsakingas už SP?**	**Ar spec. pateikiamos veiklos taisyklės / darbų sekų diagramos?**	**Ar sistemoje egzistuoja būsenos?**	**Ar dok. egzistuoja būsenų modelis?**	**Ar sistema yra viena iš šių tipų: „Embedded software“, „Web software“, „Transactional Software“, „Control systems“.**	**Ar reikalavimai aprašyti panaudos atvejais?**	**Ar egzistuoja naudotojo sistemos sąsajos scenarijai?**	**Juodosios dėžės metodai (grindžiami specifikacija)**				
								Padalijimas į ekvivalenčias klases (angl. *Equivalence Partitioning*)	Ribinių reikšmių analizė (angl. *Boundary Value Analysis*)	SP lentelių testavimas (angl. *Decision Table Testing*)	Būsenų perėjimų testavimas (angl. *State Transition Diagrams*)	Panaudojimo atvejų testavimas (angl. *Use Case Testing*)
TAIP	TAIP	TAIP	NE	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	TAIP	NE	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
TAIP	TAIP	NE	TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	NE	TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	TAIP	NE	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	NE	TAIP	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	TAIP	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	NE	TAIP	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	NE	TAIP	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
TAIP	TAIP	NE	NE	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	NE	NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	TAIP	NE	NE	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	TAIP	NE	NE	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	NE	NE	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	TAIP	NE	NE	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
TAIP	NE	TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP

Ar įmanoma PĮ duomenis sugrupuoti į ekvivalenčias klases?	**Ar bent vienas komponentas yra atsakingas už SP?**	**Ar spec. pateikiamos veiklos taisyklės / darbų sekų diagramos?**	**Ar sistemoje egzistuoja būsenos?**	**Ar dok. egzistuoja būsenų modelis?**	**Ar sistema yra viena iš šių tipų: „Embedded software“, „Web software“, „Transactional Software“, „Control systems“.**	**Ar reikalavimai aprašyti panaudos atvejais?**	**Ar egzistuoja naudotojo sistemos sąsajos scenarijai?**	**Juodosios dėžės metodai (grindžiami specifikacija)**				
								Padalijimas į ekvivalenčias klases (angl. *Equivalence Partitioning*)	Ribinių reikšmių analizė (angl. *Boundary Value Analysis*)	SP lentelių testavimas (angl. *Decision Table Testing*)	Būsenų perėjimų testavimas (angl. *State Transition Diagrams*)	Panaudojimo atvejų testavimas (angl. *Use Case Testing*)
TAIP	NE	TAIP	TAIP	TAIP	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	NE	TAIP	TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	NE	TAIP	TAIP	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	TAIP	TAIP	NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	TAIP	TAIP	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	NE	TAIP	TAIP	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	NE	TAIP	TAIP	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
TAIP	NE	TAIP	NE	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	TAIP	NE	NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	NE	TAIP	NE	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	NE	TAIP	NE	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
TAIP	NE	NE	TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	NE	TAIP	TAIP	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	NE	TAIP	TAIP	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	NE	NE	TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP

Ar įmanoma PĮ duomenis sugrupuoti į ekvivalenčias klases?	**Ar bent vienas komponentas yra atsakingas už SP?**	**Ar spec. pateikiamos veiklos taisyklės / darbų sekų diagramos?**	**Ar sistemoje egzistuoja būsenos?**	**Ar dok. egzistuoja būsenų modelis?**	**Ar sistema yra viena iš šių tipų: „Embedded software“, „Web software“, „Transactional Software“, „Control systems“.**	**Ar reikalavimai aprašyti panaudos atvejais?**	**Ar egzistuoja naudotojo sistemos sąsajos scenarijai?**	**Juodosios dėžės metodai (grindžiami specifikacija)**				
								Padalijimas į ekvivalenčias klases (angl. *Equivalence Partitioning*)	Ribinių reikšmių analizė (angl. *Boundary Value Analysis*)	SP lentelių testavimas (angl. *Decision Table Testing*)	Būsenų perėjimų testavimas (angl. *States Transition Diagrams*)	Panaudojimo atvejų testavimas (angl. *Use Case Testing*)
TAIP	NE	NE	TAIP	TAIP	NE	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	NE	NE	TAIP	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	NE	TAIP	NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	NE	TAIP	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	NE	NE	TAIP	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	NE	NE	TAIP	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	NE	NE	TAIP	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
TAIP	NE	NE	NE	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	NE	NE	NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	TAIP
TAIP	NE	NE	NE	NE	TAIP	NE	NE	TAIP	NE	NE	TAIP	NE
TAIP	NE	NE	NE	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	NE	NE	NE	NE	NE	NE	TAIP	TAIP	NE	NE	NE	TAIP
TAIP	NE	NE	NE	NE	NE	NE	NE	TAIP	NE	NE	NE	NE
NE	TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	TAIP	TAIP	TAIP	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	TAIP	TAIP	TAIP	TAIP	NE	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	TAIP	TAIP	TAIP	NE	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	TAIP	TAIP	TAIP	NE	NE	NE	NE	NE	NE	TAIP	NE
NE	TAIP	TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP

Ar įmanoma PĮ duomenis sugrupuoti į ekvivalenčias klases?	**Ar bent vienas komponentas yra atsakingas už SP?**	**Ar spec. pateikiamos veiklos taisyklės / darbų sekų diagramos?**	**Ar sistemoje egzistuoja būsenos?**	**Ar dok. egzistuoja būsenų modelis?**	**Ar sistema yra viena iš šių tipų: „Embedded software“, „Web software“, „Transactional Software“, „Control systems“.**	**Ar reikalavimai aprašyti panaudos atvejais?**	**Ar egzistuoja naudotojo sistemos sąsajos scenarijai?**	**Juodosios dėžės metodai (grindžiami specifikacija)**				
								Padalijimas į ekvivalenčias klases (angl. *Equivalence Partitioning*)	Ribinių reikšmių analizė (angl. *Boundary Value Analysis*)	SP lentelių testavimas (angl. *Decision Table Testing*)	Būsenų perėjimų testavimas (angl. *States Transition Diagrams*)	Panaudojimo atvejų testavimas (angl. *Use Case Testing*)
NE	TAIP	TAIP	TAIP	NE	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	TAIP	TAIP	NE	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	TAIP	TAIP	TAIP	NE	NE	TAIP	TAIP	NE	NE	NE	NE	TAIP
NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	NE	NE	NE	NE	TAIP
NE	TAIP	TAIP	TAIP	NE	NE	NE	NE	NE	NE	NE	NE	NE
NE	TAIP	TAIP	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	TAIP	NE	NE	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	TAIP	NE	NE	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	TAIP	TAIP	NE	NE	NE	TAIP	TAIP	NE	NE	NE	NE	TAIP
NE	TAIP	TAIP	NE	NE	NE	NE	TAIP	NE	NE	NE	NE	TAIP
NE	TAIP	TAIP	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE
NE	TAIP	NE	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	NE	TAIP	TAIP	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	TAIP	NE	TAIP	TAIP	NE	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	NE	TAIP	TAIP	NE	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	NE	TAIP	TAIP	NE	NE	NE	NE	NE	NE	TAIP	NE
NE	TAIP	NE	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	NE	TAIP	NE	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	NE	TAIP	NE	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	TAIP	NE	TAIP	NE	NE	TAIP	TAIP	NE	NE	NE	NE	TAIP

Ar įmanoma PĮ duomenis sugrupuoti į ekvivalenčias klases?	**Ar bent vienas komponentas yra atsakingas už SP?**	**Ar spec. pateikiamos veiklos taisyklės / darbų sekų diagramos?**	**Ar sistemoje egzistuoja būsenos?**	**Ar dok. egzistuoja būsenų modelis?**	**Ar sistema yra viena iš šių tipų: „Embedded software“, „Web software“, „Transactional Software“, „Control systems“.**	**Ar reikalavimai aprašyti panaudos atvejais?**	**Ar egzistuoja naudotojo sistemos sąsajos scenarijai?**	**Juodosios dėžės metodai (grindžiami specifikacija)**				
								Padalijimas į ekvivalenčias klases (angl. *Equivalence Partitioning*)	Ribinių reikšmių analizė (angl. *Boundary Value Analysis*)	SP lentelių testavimas (angl. *Decision Table Testing*)	Būsenų perėjimų testavimas (angl. *States Transition Diagrams*)	Panaudojimo atvejų testavimas (angl. *Use Case Testing*)
NE	TAIP	NE	TAIP	NE	NE	NE	TAIP	NE	NE	NE	NE	TAIP
NE	TAIP	NE	TAIP	NE	NE	NE	NE	NE	NE	NE	NE	NE
NE	TAIP	NE	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	NE	NE	NE	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	TAIP	NE	NE	NE	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	TAIP	NE	NE	NE	NE	TAIP	TAIP	NE	NE	NE	NE	TAIP
NE	TAIP	NE	NE	NE	NE	NE	TAIP	NE	NE	NE	NE	TAIP
NE	TAIP	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE
NE	NE	TAIP	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	TAIP	TAIP	TAIP	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	TAIP	TAIP	TAIP	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	NE	TAIP	TAIP	TAIP	NE	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	TAIP	TAIP	TAIP	NE	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	NE	NE	NE	TAIP	NE
NE	NE	TAIP	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	TAIP	TAIP	NE	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	TAIP	TAIP	NE	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	NE	TAIP	TAIP	NE	NE	TAIP	TAIP	NE	NE	NE	NE	TAIP
NE	NE	TAIP	TAIP	NE	NE	NE	TAIP	NE	NE	NE	NE	TAIP
NE	NE	TAIP	TAIP	NE	NE	NE	NE	NE	NE	NE	NE	NE

Ar įmanoma PĮ duomenis sugrupuoti į ekvivalenčias klases?	**Ar bent vienas komponentas yra atsakingas už SP?**	**Ar spec. pateikiamos veiklos taisyklės / darbų sekų diagramos?**	**Ar sistemoje egzistuoja būsenos?**	**Ar dok. egzistuoja būsenų modelis?**	**Ar sistema yra viena iš šių tipų: „Embedded software“, „Web software“, „Transactional Software“, „Control systems“.**	**Ar reikalavimai aprašyti panaudos atvejais?**	**Ar egzistuoja naudotojo sistemos sąsajos scenarijai?**	**Juodosios dėžės metodai (grindžiami specifikacija)**				
								Padalijimas į ekvivalenčias klases (angl. *Equivalence Partitioning*)	Ribinių reikšmių analizė (angl. *Boundary Value Analysis*)	SP lentelių testavimas (angl. *Decision Table Testing*)	Būsenų perėjimų testavimas (angl. *States Transition Diagrams*)	Panaudojimo atvejų testavimas (angl. *Use Case Testing*)
NE	NE	TAIP	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	TAIP	NE	NE	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	TAIP	NE	NE	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	NE	TAIP	NE	NE	NE	TAIP	TAIP	NE	NE	NE	NE	TAIP
NE	NE	TAIP	NE	NE	NE	NE	TAIP	NE	NE	NE	NE	TAIP
NE	NE	TAIP	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE
NE	NE	NE	TAIP	TAIP	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	NE	TAIP	TAIP	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	NE	NE	TAIP	TAIP	NE	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	NE	TAIP	TAIP	NE	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	NE	TAIP	TAIP	NE	NE	NE	NE	NE	NE	TAIP	NE
NE	NE	NE	TAIP	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	NE	TAIP	NE	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	NE	TAIP	NE	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	NE	NE	TAIP	NE	NE	TAIP	TAIP	NE	NE	NE	NE	TAIP
NE	NE	NE	TAIP	NE	NE	NE	TAIP	NE	NE	NE	NE	TAIP
NE	NE	NE	TAIP	NE	NE	NE	NE	NE	NE	NE	NE	NE

Ar įmanoma PĮ duomenis sugrupuoti į ekvivalenčias klases?	**Ar bent vienas komponentas yra atsakingas už SP?**	**Ar spec. pateikiamos veiklos taisyklės / darbų sekų diagramos?**	**Ar sistemoje egzistuoja būsenos?**	**Ar dok. egzistuoja būsenų modelis?**	**Ar sistema yra viena iš šių tipų: „Embedded software“, „Web software“, „Transactional Software“, „Control systems“.**	**Ar reikalavimai aprašyti panaudos atvejais?**	**Ar egzistuoja naudotojo sistemos sąsajos scenarijai?**	**Juodosios dėžės metodai (grindžiami specifikacija)**				
								Padalijimas į ekvivalenčias klases (angl. *Equivalence Partitioning*)	Ribinių reikšmių analizė (angl. *Boundary Value Analysis*)	SP lentelių testavimas (angl. *Decision Table Testing*)	Būsenų perėjimų testavimas (angl. *State Transition Diagrams*)	Panaudojimo atvejų testavimas (angl. *Use Case Testing*)
NE	NE	NE	NE	NE	TAIP	TAIP	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	NE	NE	NE	TAIP	NE	TAIP	NE	NE	NE	TAIP	TAIP
NE	NE	NE	NE	NE	TAIP	NE	NE	NE	NE	NE	TAIP	NE
NE	NE	NE	NE	NE	NE	TAIP	TAIP	NE	NE	NE	NE	TAIP
NE	NE	NE	NE	NE	NE	NE	TAIP	NE	NE	NE	NE	TAIP
NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE

13 lentelė. Baltosios dėžės testavimo metodų atrinkimo kriterijai

Koks rizikos lygis?	**Sistemą svarbu gerai ištestuoti?**	**Sistemos („safety critical“) klaida gali sukelti kritines pasekmes?**	**Baltosios dėžės metodai (grindžiami sistemos struktūra)**		
			Teiginių padengimo testavimas (angl. *Statement Coverage*)	Sprendimų testavimas (angl. *Decision (branch) testing*)	Galimų kelių testavimas (angl. *Path Testing*)
Žemas	NE	NE	TAIP	-	-
Vidutinis	TAIP	NE	-	TAIP	-
Vidutinis	NE	TAIP	-	-	TAIP
Aukštas	NE	TAIP	-	-	TAIP
Aukštas	TAIP	NE	-	TAIP	-

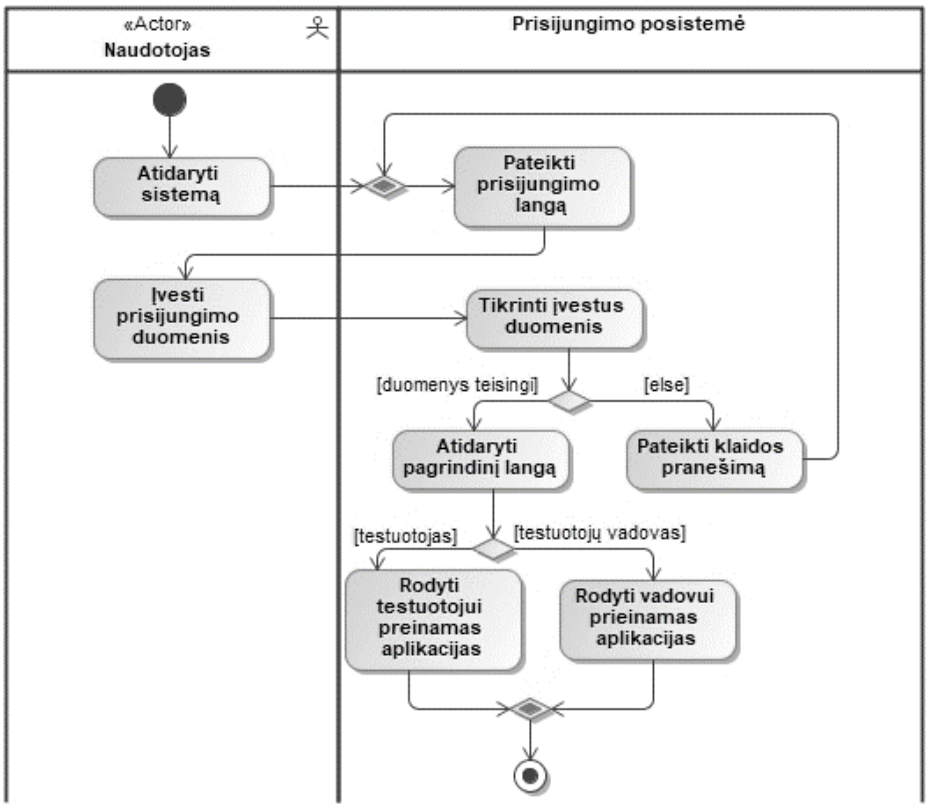
14 lentelė. Praktika grindžiamų testavimo metodų atrinkimo kriterijai

Ar egzistuoja dokumentacija?	**Ar testuotojas turi patirties pan. projektuose?**	**Ar testavimo komanda turi susidarę sąrašą (angl. *checklist*) situacijų, kurias reikėtų testuoti tokio tipo projektuose?**	**Praktika grindžiami metodai**		
			Tyrimu grindžiamas testavimas (angl. *Exploratory Testing*)	Klaidų spėjimo metodas (angl. *Error Guessing*)	Testavimas remiantis sąrašu (angl. *Checklist-Based Testing*)
NE	NE	NE	TAIP	-	-
NE	NE	TAIP	TAIP	-	TAIP
NE	TAIP	NE	TAIP	TAIP	-
NE	TAIP	TAIP	-	TAIP	TAIP
TAIP	NE	NE	-	-	-
TAIP	TAIP	NE	-	TAIP	-
TAIP	TAIP	TAIP	-	TAIP	TAIP
TAIP	NE	TAIP	-	-	TAIP

2 priedas. Programinės įrangos testavimo metodikos demonstracinės sistemos panaudojimo atvejų specifikacija

Prisijungti

Prisijungimo posistemei priskiriamo panaudojimo atvejo „Prisijungti“ veiklos diagrama (49 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



49 pav. Prisijungimo veiklos diagrama

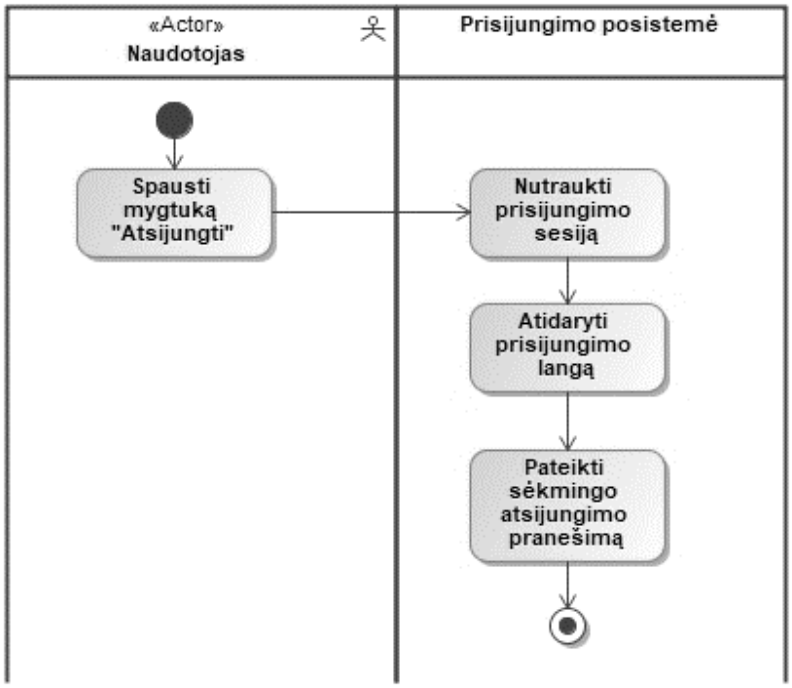
Panaudojimo atvejis „Prisijungti“ detalizuojamas specifikacijos lentele (15 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

15 lentelė. Panaudojimo atvejo „Prisijungi“ specifikacijos lentelė

PA „Prisijungti“		
Aprašymas. Vykdant panaudojimo atvejį, naudotojas nurodo prisijungimo duomenis ir yra prijungiamas prie sistemos. Neteisingai nurodžius duomenis, pateikiamas klaidos pranešimas.		
Prieš sąlyga:		Naudotojas yra neprisijungęs prie sistemos.
Aktorius:		Naudotojas.
Sužadinimo sąlyga:		Naudotojas atsidaro sistemos prisijungimo langą.
Susiję PA	Išplečiantys PA:	-
	Apimami PA:	-
	Specializuoja PA:	-
Po sąlyga:		Naudotojas yra prisijungęs prie sistemos.

Atsijungti

Prisijungimo posistemei priskiriamo panaudojimo atvejo „Atsijungti“ veiklos diagrama (50 pav.) atspindi pagrindinį panaudojimo atvejo vykdymo scenarijų.



50 pav. Atsijungimo veiklos diagrama

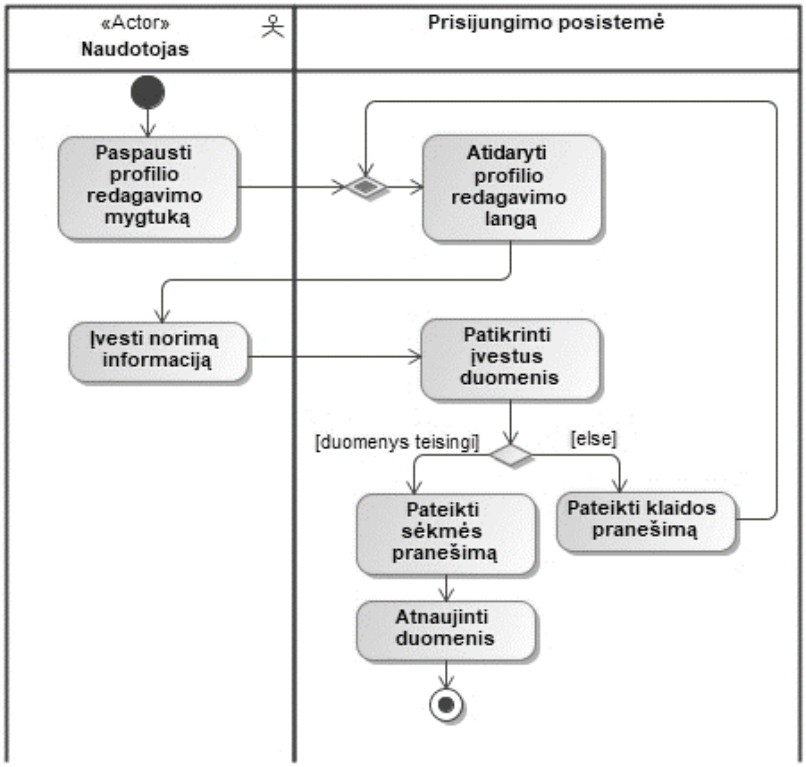
Panaudojimo atvejis „Atsijungti“ detalizuojamas specifikacijos lentele (16 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

16 lentelė. Panaudojimo atvejo „Atsijungti“ specifikacijos lentelė

PA „Atsijungti“		
Aprašymas. Vykdant panaudojimo atvejį, naudotojas paspaudžia atsijungimo mygtuką ir atsijungia nuo sistemos.		
Prieš sąlyga:		Naudotojas yra prisijungęs prie sistemos.
Aktorius:		Naudotojas.
Sužadinimo sąlyga:		Naudotojas paspaudžia atsijungimo mygtuką.
Susiję PA	Išplečiantys PA:	-
	Apimami PA:	-
	Specializuoja PA:	-
Po sąlyga:		Naudotojas yra atsijungęs nuo sistemos.

Redaguoti profilį

Prisijungimo posistemei priskiriamo panaudojimo atvejo „Redaguoti profilį“ veiklos diagrama (51 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



51 pav. Profilio redagavimo veiklos diagrama

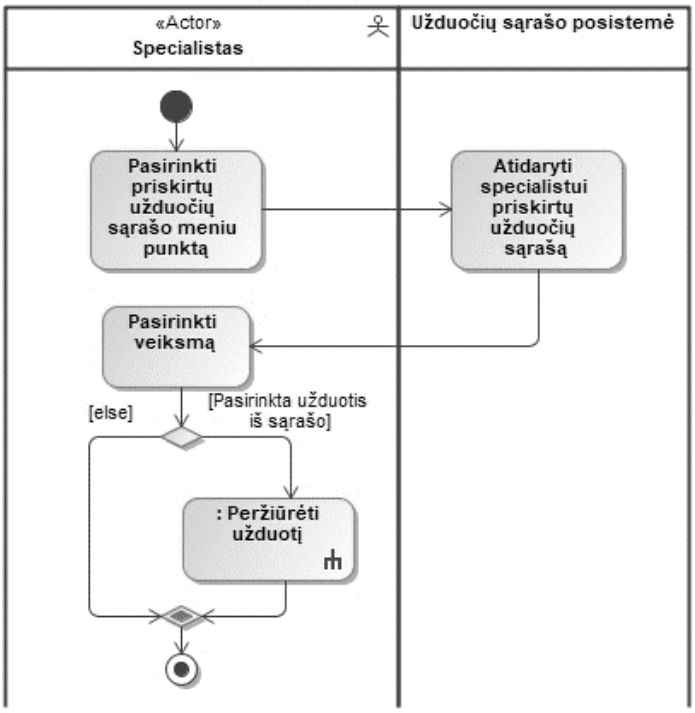
Panaudojimo atvejis „Redaguoti profilį“ detalizuojamas specifikacijos lentele (17 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

17 lentelė. Panaudojimo atvejo „Redaguoti profilį“ specifikacijos lentelė

PA „Redaguoti profilį“		
Aprašymas. Vykdant panaudojimo atvejį, naudotojas įveda ir išsaugo atnaujintus profilio duomenis. Neteisingai nurodžius duomenis, pateikiamas klaidos pranešimas.		
Prieš sąlyga:		Naudotojas yra prisijungęs prie sistemos ir atsidaręs pagrindinį langą.
Aktorius:		Naudotojas.
Sužadinimo sąlyga:		Naudotojas paspaudžia profilio redagavimo mygtuką.
Susiję PA	Išplečiantys PA:	-
	Apimami PA:	-
	Specializuoja PA:	-
Po sąlyga:		Naudotojas profilio informacija atnaujinta.

Peržiūrėti užduočių sąrašą

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Peržiūrėti užduočių sąrašą“ veiklos diagrama (52 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



52 pav. Užduočių sąrašo peržiūrėjimo veiklos diagrama

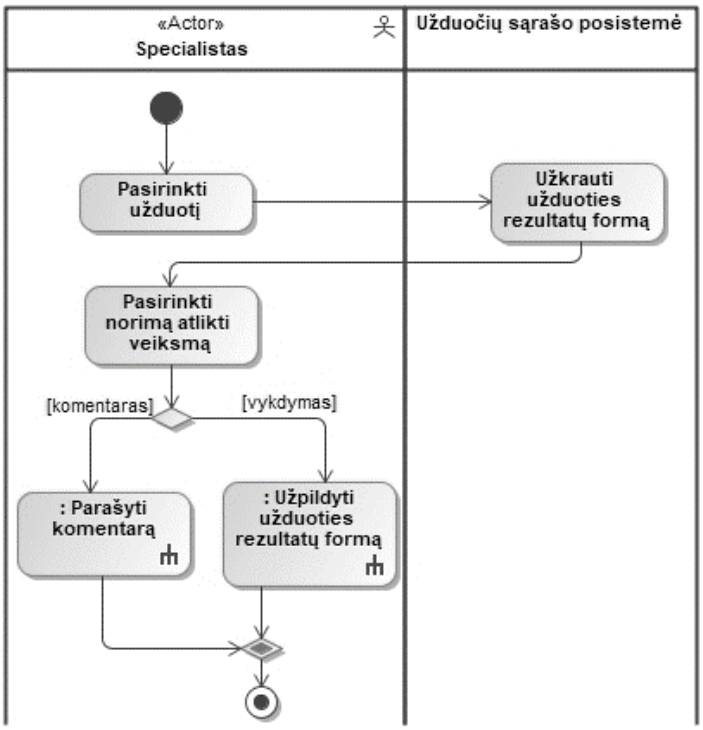
Panaudojimo atvejis „Peržiūrėti užduočių sąrašą“ detalizuojamas specifikacijos lentele (18 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

18 lentelė. Panaudojimo atvejo „Peržiūrėti užduočių sąrašą“ specifikacijos lentelė

PA „Peržiūrėti užduočių sąrašą“		
Aprašymas. Vykdant panaudojimo atvejį, specialistas pasirenka jam priskirtų užduočių sąrašo peržiūrėjimo meniu ir peržiūri priskirtų užduočių sąrašą.		
Prieš sąlyga:		Specialistas turi priskirtą bent vieną užduotį.
Aktorius:		Specialistas.
Sužadinimo sąlyga:		Specialistas pasirenka užduočių sąrašo meniu punktą.
Susiję PA	Išplečiantys PA:	Peržiūrėti užduotį.
	Apimami PA:	-
	Specializuoja PA:	-
Po sąlyga:		Užduočių sąrašas peržiūrėtas.

Peržiūrėti užduotį

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Peržiūrėti užduotį“ veiklos diagrama (53 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



53 pav. Užduoties peržiūrėjimo veiklos diagrama

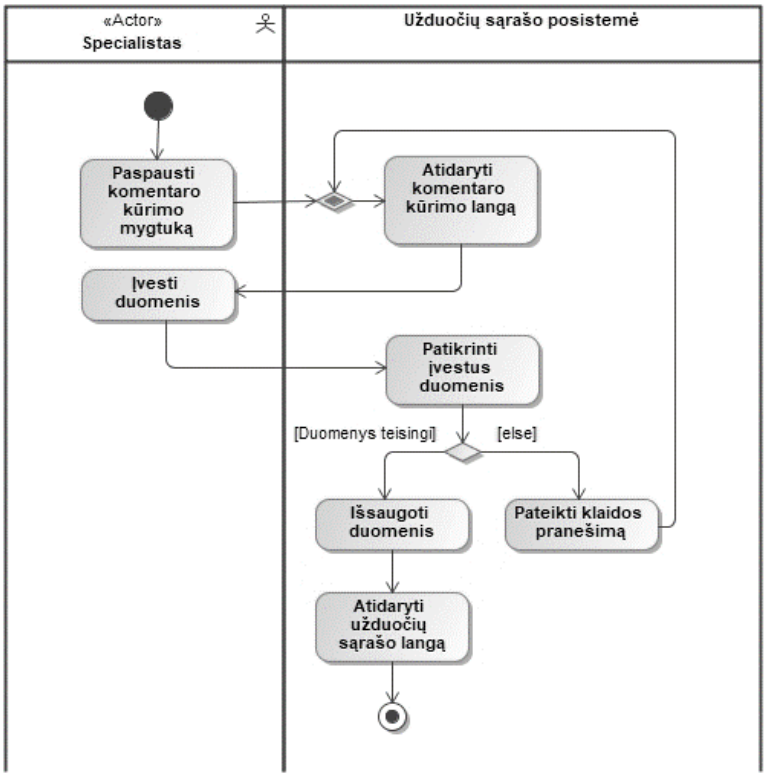
Panaudojimo atvejis „Peržiūrėti užduotį“ detalizuojamas specifikacijos lentele (19 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

19 lentelė. Panaudojimo atvejo „Peržiūrėti užduotį“ specifikacijos lentelė

PA „Peržiūrėti užduotį“		
Aprašymas. Vykdant panaudojimo atvejį, specialistas peržiūri užduotį ir gali pasirinkti tolimesnius norimus atlikti veiksmus su užduotimi.		
Prieš sąlyga:		Specialistas yra atsidaręs užduočių sąrašą.
Aktorius:		Specialistas.
Sužadinimo sąlyga:		Specialistas pasirenka norimą užduotį.
Susiję PA	Išplečiantys PA:	Parašyti komentarą, Užpildyti užduoties rezultatų formą.
	Apimami PA:	-
	Specializuoja PA:	-
Po sąlyga:		Užduoties duomenys peržiūrėti.

Parašyti komentarą

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Parašyti komentarą“ veiklos diagrama (54 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



54 pav. Komentaro rašymo veiklos diagrama

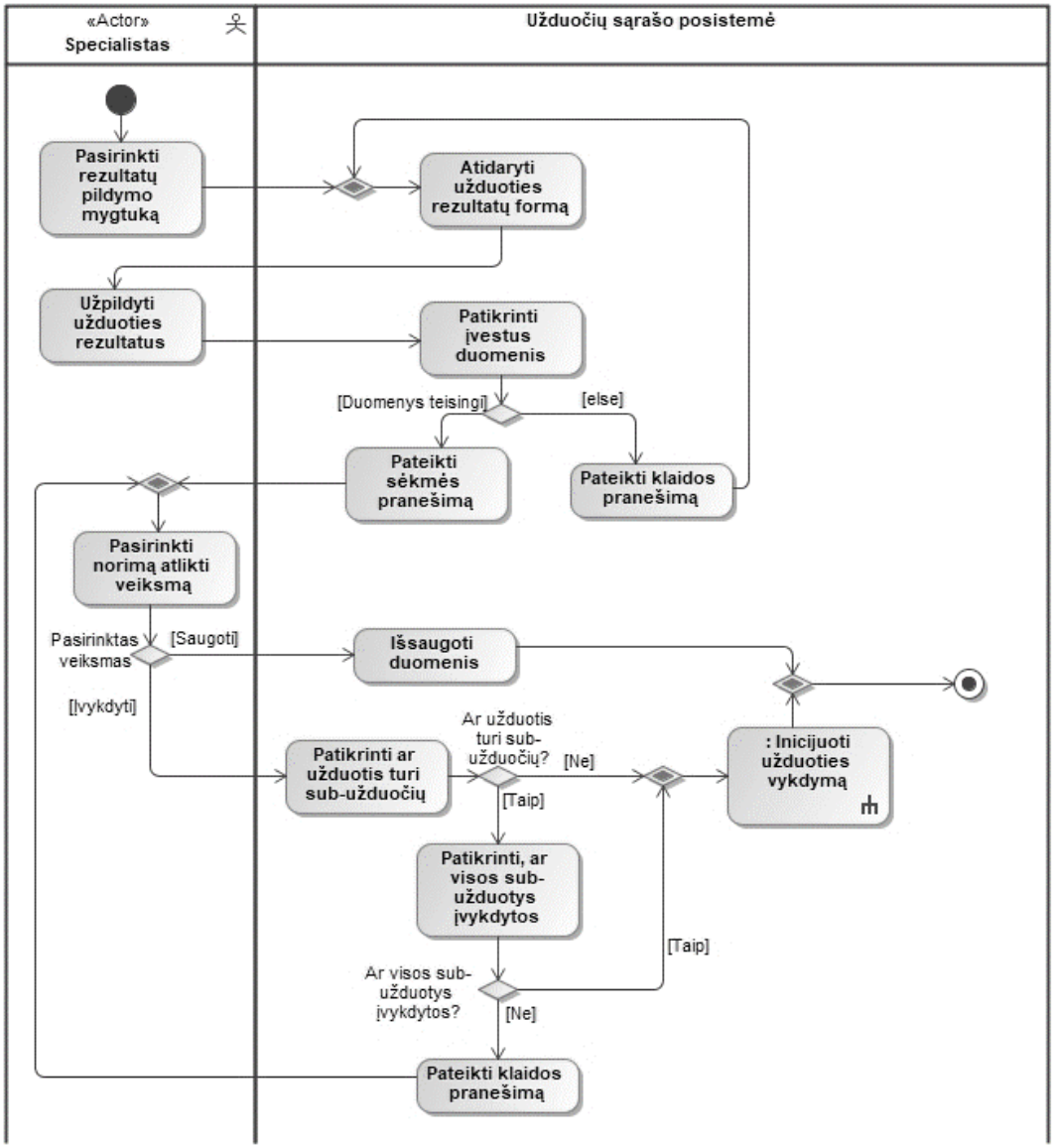
Panaudojimo atvejis „Parašyti komentarą“ detalizuojamas specifikacijos lentele (20 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

20 lentelė. Panaudojimo atvejo „Parašyti komentarą“ specifikacijos lentelė

PA „Parašyti komentarą“		
Aprašymas. Vykdant panaudojimo atvejį, specialistas sukuria prie savo užduoties komentarą.		
Prieš sąlyga:		Specialistas yra atsidaręs užduoties peržiūros langą.
Aktorius:		Specialistas.
Sužadinimo sąlyga:		Specialistas paspaudžia komentaro kūrimo mygtuką.
Susiję PA	Išplečiantys PA:	-
	Apimami PA:	-
	Specializuoja PA:	-
Po sąlyga:		Komentaras sukurtas.

Užpildyti užduoties rezultatų formą

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Užpildyti užduoties rezultatų formą“ veiklos diagrama (55 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



55 pav. Užduoties rezultatų pildymo veiklos diagrama

Panaudojimo atvejis „Užpildyti užduoties rezultatų formą“ detalizuojamas specifikacijos lentele (21 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

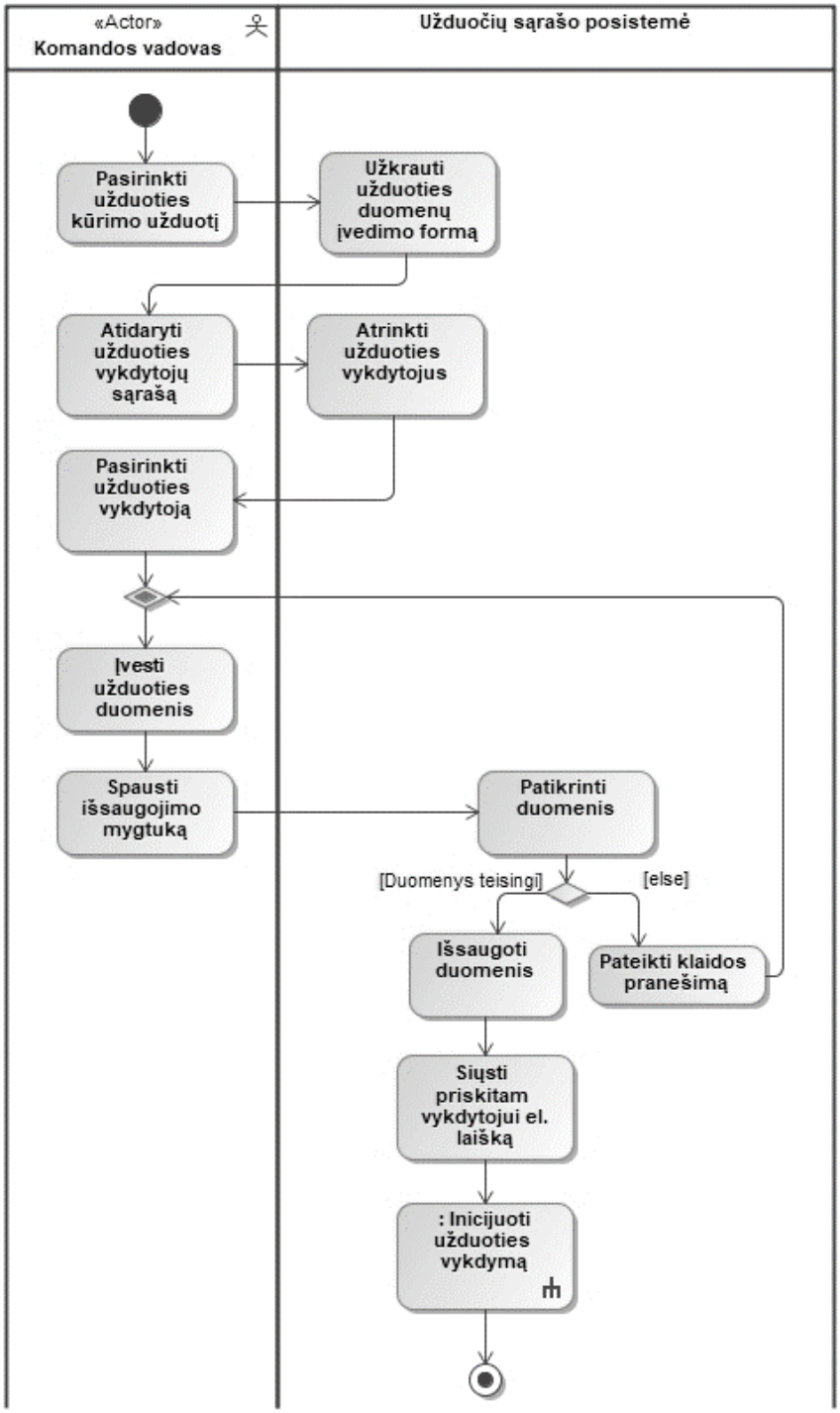
21 lentelė. Panaudojimo atvejo „Užpildyti užduoties rezultatų formą“ specifikacijos lentelė

PA „Užpildyti užduoties rezultatų formą“		
Aprašymas. Vykdant panaudojimo atvejį, užpildoma specialistui priskirtos užduoties rezultatai.		
Prieš sąlyga:		Specialistas yra atsidaręs užduoties peržiūros langą.
Aktorius:		Specialistas.
Sužadinimo sąlyga:		Specialistas pasirenka rezultatų pildymo formos mygtuką.
Susiję PA	Išplečiantys PA:	Inicijuoti užduoties vykdymą.
	Apimami PA:	-
	Specializuoja PA:	-

Po sąlyga:	Užduoties rezultatai išsaugoti.

Sukurti užduotį

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Sukurti užduotį“ veiklos diagrama (56 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



56 pav. Užduoties kūrimo veiklos diagrama

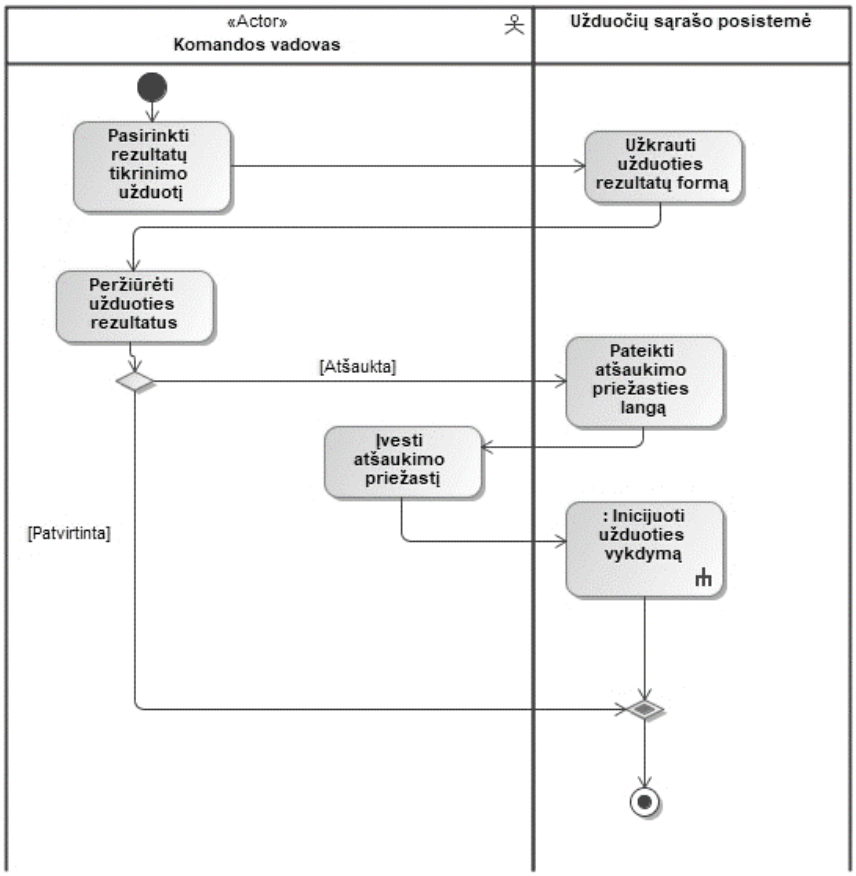
Panaudojimo atvejis „Sukurti užduotį“ detalizuojamas specifikacijos lentele (22 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

22 lentelė. Panaudojimo atvejo „Sukurti užduotį“ specifikacijos lentelė

PA „Sukurti užduotį“		
Aprašymas. Vykdant panaudojimo atvejį, sukuriama užduotis ir priskiriama vykdytojui. Neteisingai nurodžius duomenis, pateikiamas klaidos pranešimas.		
Prieš sąlyga:		Komandos vadovas yra gavęs užduotį „Sukurti užduotį“
Aktorius:		Komandos vadovas.
Sužadinimo sąlyga:		Komandos vadovas atsidaro užduoties kūrimo užduotį.
Susiję PA	Išplečiantys PA:	-
	Apimami PA:	Inicijuoti užduoties vykdymą.
	Specializuoja PA:	-
Po sąlyga:		Užduoties duomenys išsaugoti.

Tikrinti testavimo užduoties rezultatus

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Tikrinti užduoties rezultatus“ veiklos diagrama (57 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



57 pav. Užduoties rezultatų tikrinimo veiklos diagrama

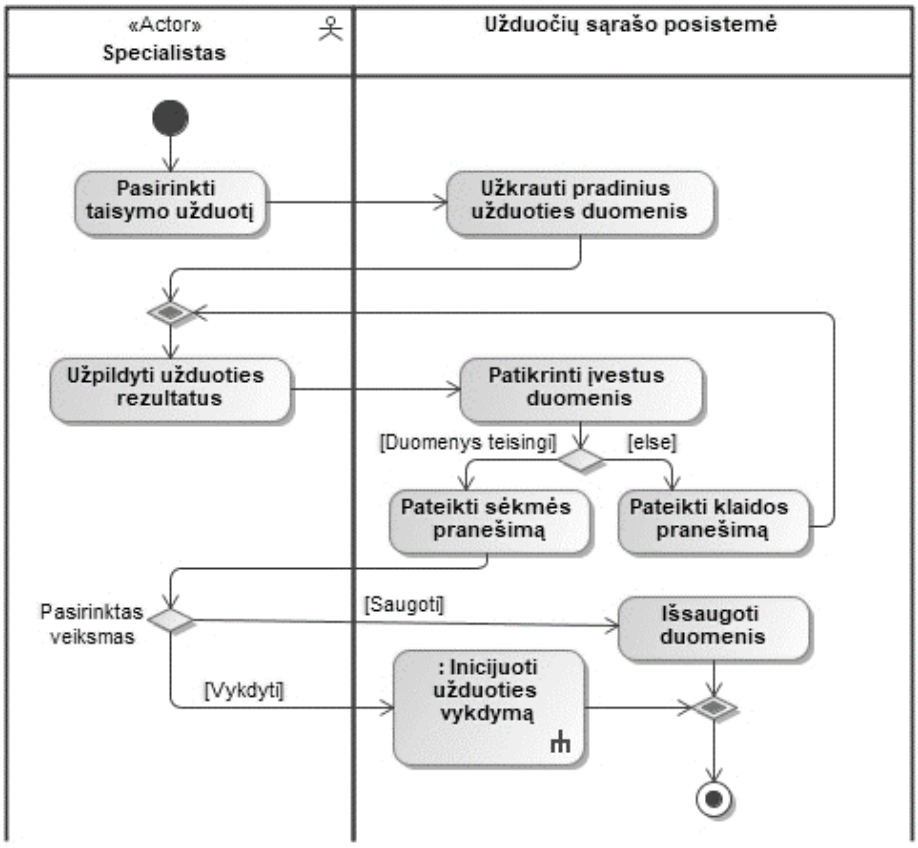
Panaudojimo atvejis „Tikrinti užduoties rezultatus“ detalizuojamas specifikacijos lentele (23 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

23 lentelė. Panaudojimo atvejo „Tikrinti testavimo užduoties rezultatus“ specifikacijos lentelė

PA „Tikrinti testavimo užduoties rezultatus“		
Aprašymas. Vykdant panaudojimo atvejį, patvirtinami užduoties rezultatai.		
Prieš sąlyga:		Komandos vadovas yra gavęs užduoties rezultatų tikrinimo užduotį.
Aktorius:		Komandos vadovas.
Sužadinimo sąlyga:		Komandos vadovas paspaudžia užduoties peržiūrėjimo mygtuką.
Susiję PA	Išplečiantys PA:	Taisyti užduoties rezultatus.
	Apimami PA:	Inicijuoti užduoties vykdymą.
	Specializuoja PA:	-
Po sąlyga:		Užduoties rezultatai patvirtinti.

Taisyti testavimo užduoties rezultatus

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Taisyti užduoties rezultatus“ veiklos diagrama (58 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



58 pav. Užduoties taisymo veiklos diagrama

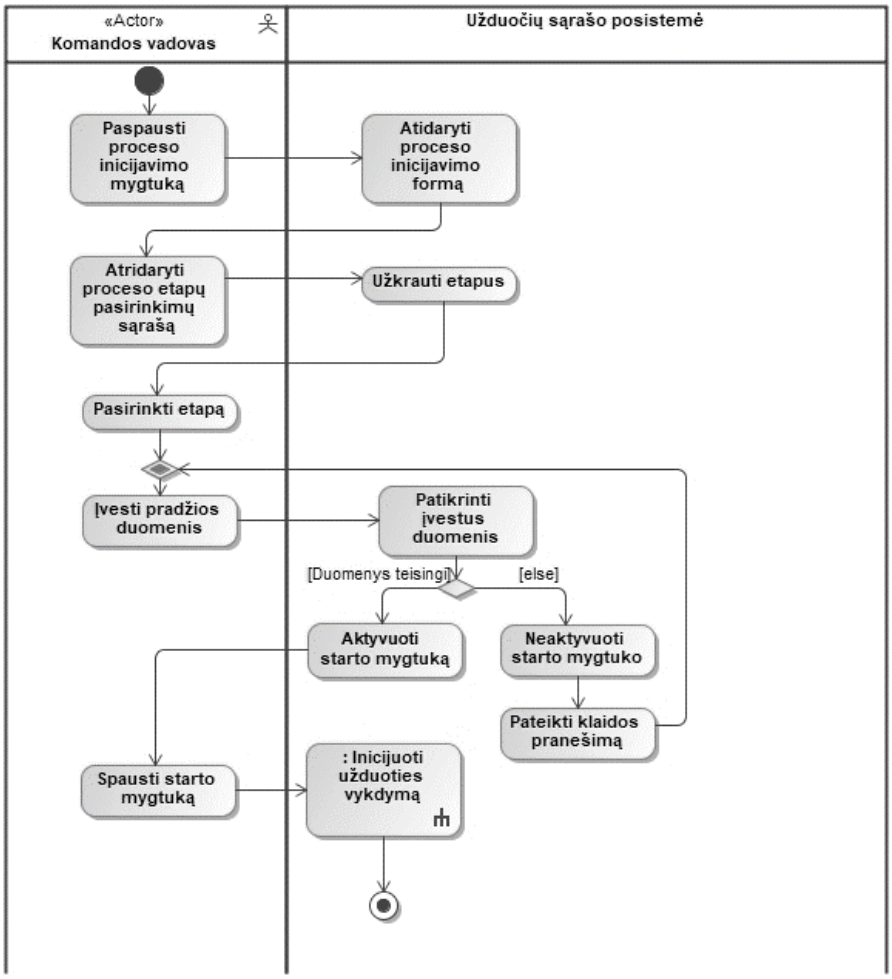
Panaudojimo atvejis „Taisyti užduoties rezultatus“ detalizuojamas specifikacijos lentele (24 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

24 lentelė. Panaudojimo atvejo „Taisyti testavimo užduoties rezultatus“ specifikacijos lentelė

PA „Taisyti testavimo užduoties rezultatus“		
Aprašymas. Vykdant panaudojimo atvejį, išsaugomi pataisyti užduoties rezultatai.		
Prieš sąlyga:		Specialistas yra gavęs užduoties rezultatų taisymo užduotį.
Aktorius:		Specialistas.
Sužadinimo sąlyga:		Specialistas paspaudžia užduoties peržiūrėjimo mygtuką.
Susiję PA	Išplečiantys PA:	Inicijuoti užduoties vykdymą.
	Apimami PA:	-
	Specializuoja PA:	-
Po sąlyga:		Išsaugomi ištaisyti užduoties rezultatai.

Inicijuoti testavimo proceso etapą

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Inicijuoti testavimo proceso etapą“ veiklos diagrama (59 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



59 pav. Testavimo proceso etapo inicijavimo veiklos diagrama

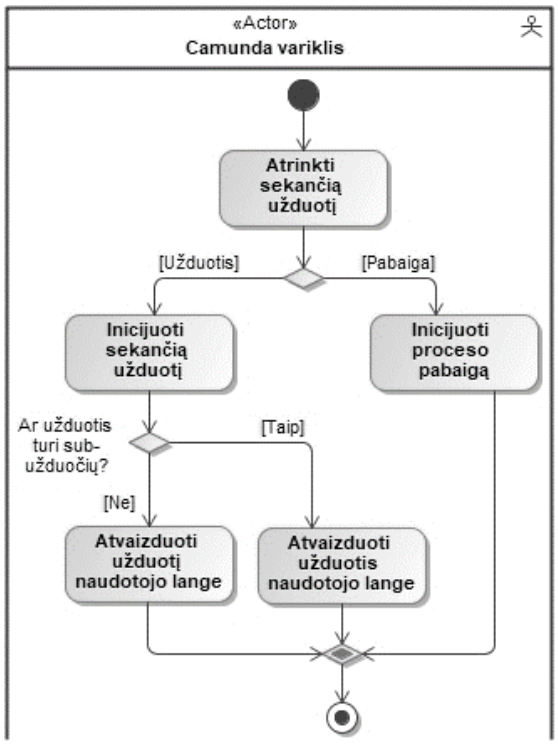
Panaudojimo atvejis „Inicijuoti projekto pradžią“ detalizuojamas specifikacijos lentele (25 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

25 lentelė. Panaudojimo atvejo „Inicijuoti testavimo proceso etapą“ specifikacijos lentelė

PA „Inicijuoti testavimo proceso etapą“		
Aprašymas. Vykdant panaudojimo atvejį, paspaudžiamas projekto inicijavimo mygtukas ir projekto vykdymas pradedamas.		
Prieš sąlyga:		Komandos vadovas yra atsidaręs savo užduočių sąrašą.
Aktorius:		Komandos vadovas.
Sužadinimo sąlyga:		Komandos vadovas paspaudžia projekto inicijavimo mygtuką.
Susiję PA	Išplečiantys PA:	-
	Apimami PA:	Inicijuoti užduoties vykdymą.
	Specializuoja PA:	-
Po sąlyga:		Projektas inicijuotas.

Inicijuoti testavimo užduoties vykdymą

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Inicijuoti užduoties vykdymą“ veiklos diagrama (60 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



60 pav. Užduoties vykdymo inicijavimo veiklos diagrama

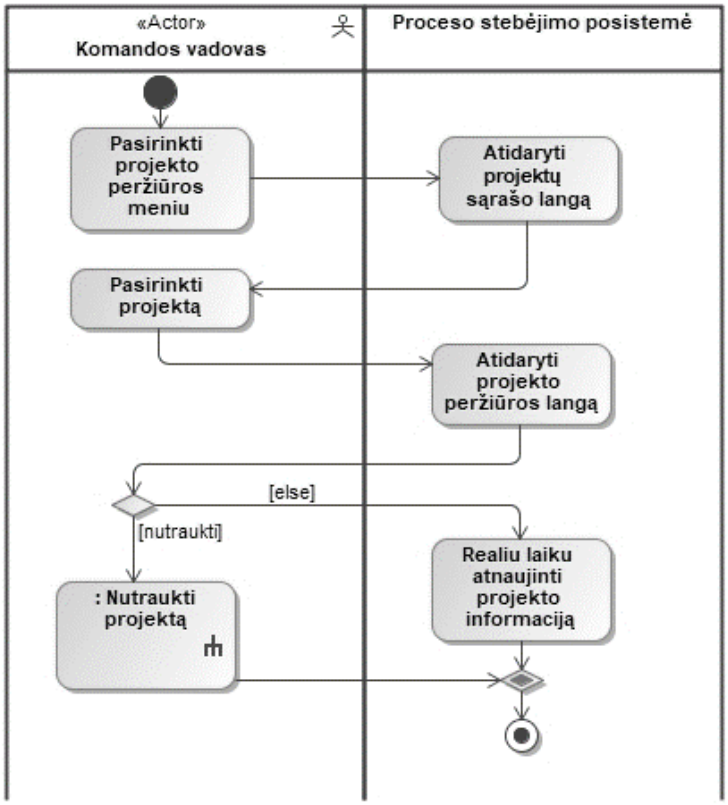
Panaudojimo atvejis „Inicijuoti užduoties vykdymą“ detalizuojamas specifikacijos lentele (26 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

26 lentelė. Panaudojimo atvejo „Inicijuoti testavimo užduoties vykdymą“ specifikacijos lentelė

PA „Inicijuoti testavimo užduoties vykdymą“		
Aprašymas. Vykdant panaudojimo atvejį, inicijuojama sekanti proceso užduotis.		
Prieš sąlyga:		Buvo įvykdyta ankstesnė užduotis.
Aktorius:		„Camunda BPM“ variklis.
Sužadinimo sąlyga		„Camunda BPM“ variklis gauna užduoties įvykdymo duomenis.
Susiję PA	Išplečiantys PA:	-
	Apimami PA:	-
	Specializuoja PA:	-
Po sąlyga:		Sekanti užduotis inicijuota – perduota priskirtam specialistui.

Stebėti vykdomą testavimo projektą

Analizės posistemei priskiriamo panaudojimo atvejo „Stebėti vykdomą projektą“ veiklos diagrama (61 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



61 pav. Vykdomo testavimo projekto stebėjimo veiklos diagrama

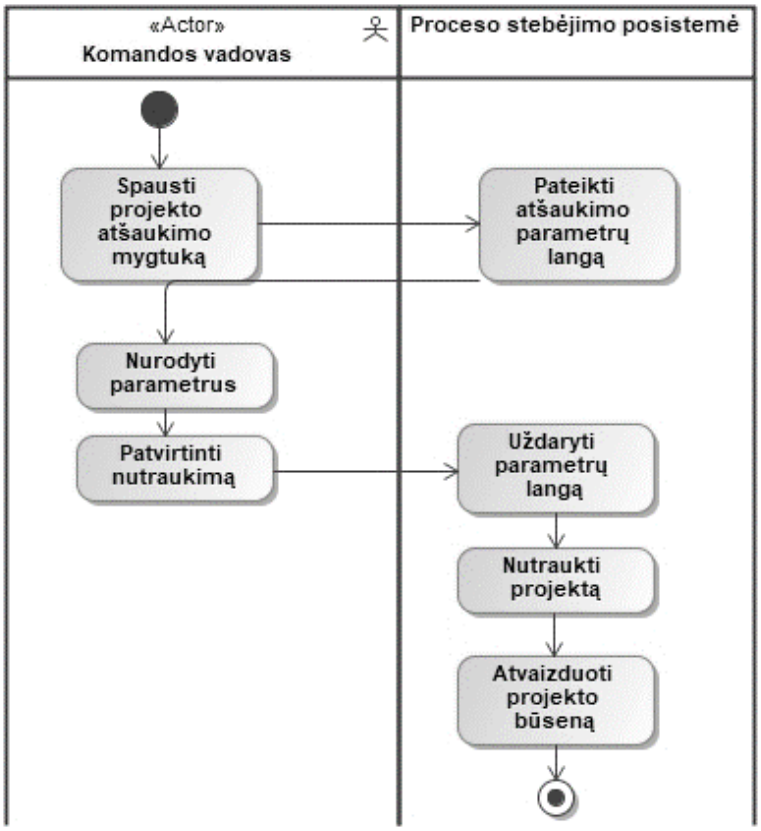
Panaudojimo atvejis „Stebėti vykdomą projektą“ detalizuojamas specifikacijos lentele (27 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

27 lentelė. Panaudojimo atvejo „Stebėti vykdomą testavimo projektą“ specifikacijos lentelė

PA „Stebėti vykdomą testavimo projektą“		
Aprašymas. Panaudos atvejo vykdymo metu, komandos vadovas atsidaro ir peržiūri vykdomus projektus.		
Prieš sąlyga:		Komandos vadovas yra prisijungęs prie sistemos.
Aktorius:		Komandos vadovas.
Sužadinimo sąlyga:		Komandos vadovas paspaudžia projektų meniu mygtuką.
Susiję PA	Išplečiantys PA:	Nutraukti projektą.
	Apimami PA:	-
	Specializuoja PA:	-
Po sąlyga:		Vykdomas projektas peržiūrėtas.

Nutraukti vykdomą testavimo projektą

Analizės posistemei priskiriamo panaudojimo atvejo „Nutraukti projektą“ veiklos diagrama (62 pav.) atspindi pagrindinį ir alternatyvius panaudojimo atvejo vykdymo scenarijus.



62 pav. Projekto nutraukimo testavimo veiklos diagrama

Panaudojimo atvejis „Nutraukti projektą“ detalizuojamas specifikacijos lentele (28 lentelė), kurioje nurodomas panaudojimo atvejo scenarijaus trumpas aprašas, prieš, po ir sužadinimo sąlygos, bei susiję panaudojimo atvejai.

28 lentelė. Panaudojimo atvejo „Nutraukti vykdomą testavimo projektą“ specifikacijos lentelė

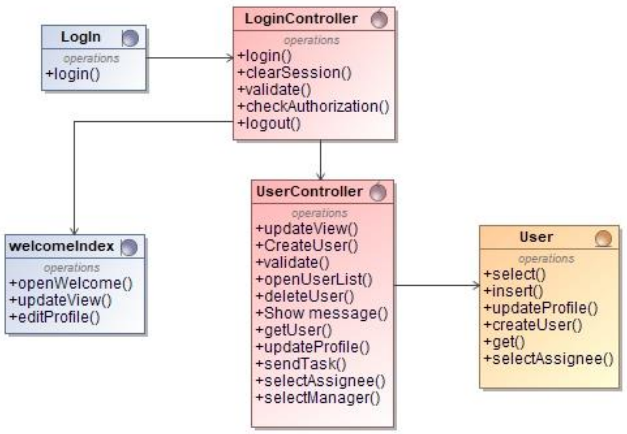
PA „Nutraukti vykdomą testavimo projektą“		
Aprašymas. Vykdant panaudojimo atvejį, komandos vadovas pasirenka ir nutraukia norimo projekto vykdymą.		
Prieš sąlyga:		Komandos vadovas yra atsidaręs vykdomų procesų langą.
Aktorius:		Komandos vadovas.
Sužadinimo sąlyga:		Komandos vadovas paspaudžia projekto atšaukimo mygtuką.
Susiję PA	Išplečiantys PA:	-
	Apimami PA:	-
	Specializuoja PA:	-
Po sąlyga:		Vykdomas projektas nutrauktas.

3 priedas. Kompiuterizuojamų panaudojimo atvejų realizacija projekto klasėmis

Šiame skyriuje pateikiamos visus kompiuterizuojamus panaudojimo atvejus specifikuojančios diagramos: realizavimo projekto klasėmis, sekų. Šios diagramos yra artimiausios projekto realizacijai ir turi su ja sinchronizuotis. Tolimesniuose poskyriuose specifikuojamas kiekvienas panaudos atvejis.

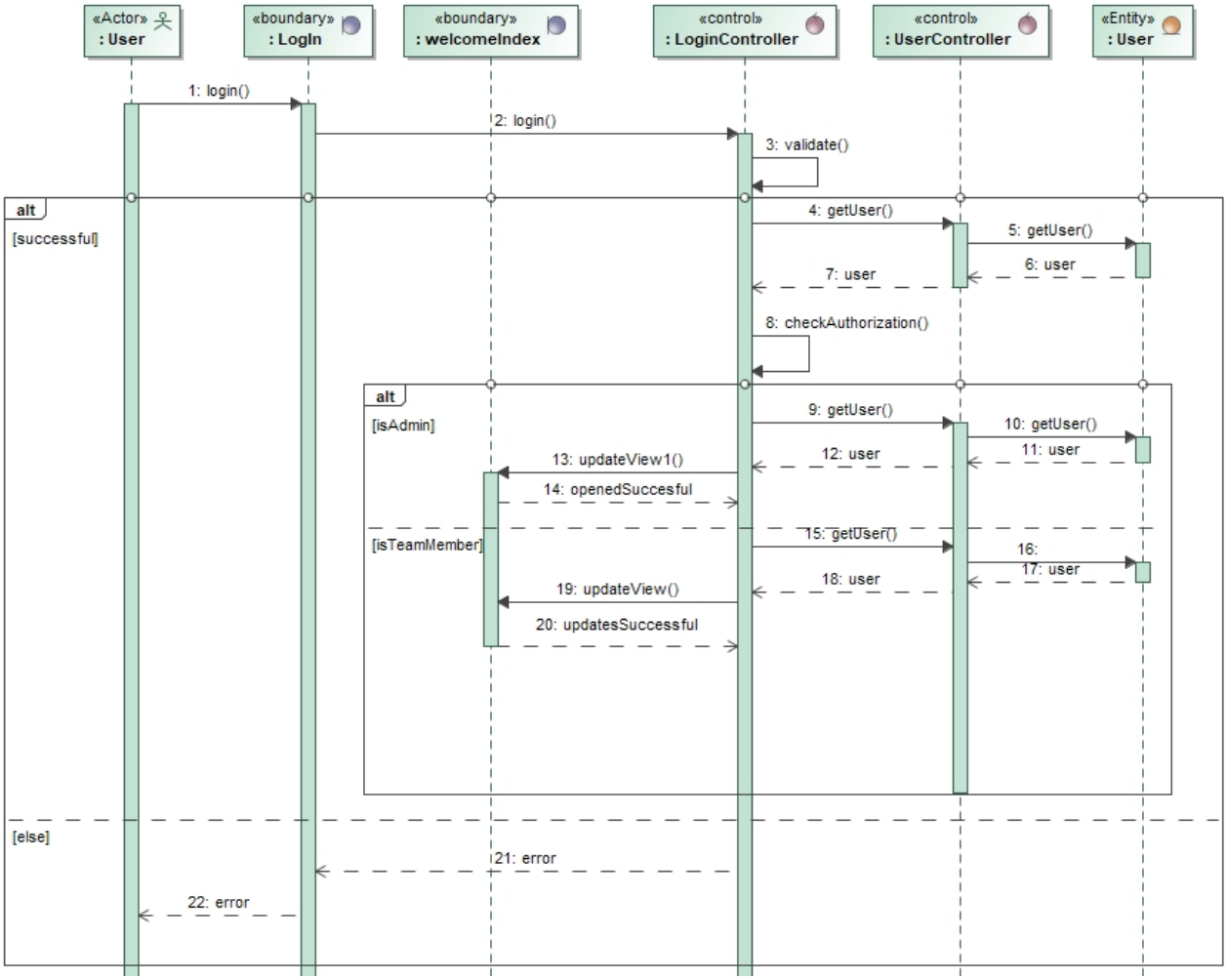
Prisijungti

Prisijungimo posistemei priskiriamo panaudojimo atvejo „Prisijungti“ realizavimo projekto klasėmis diagrama (63 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



63 pav. Prisijungimo realizavimo klasių diagrama

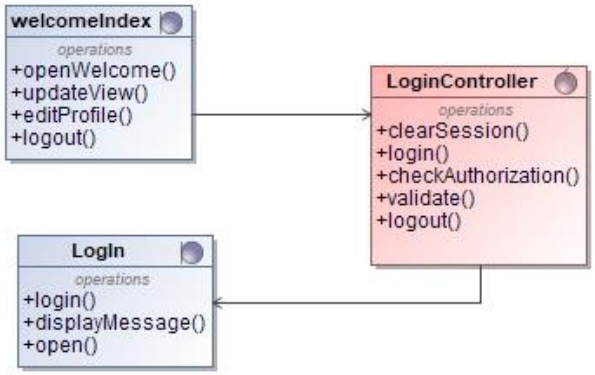
Prisijungimo posistemei priskiriamo panaudojimo atvejo „Prisijungti“ sekų diagrama (64 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



64 pav. Prisijungimo sekų diagrama

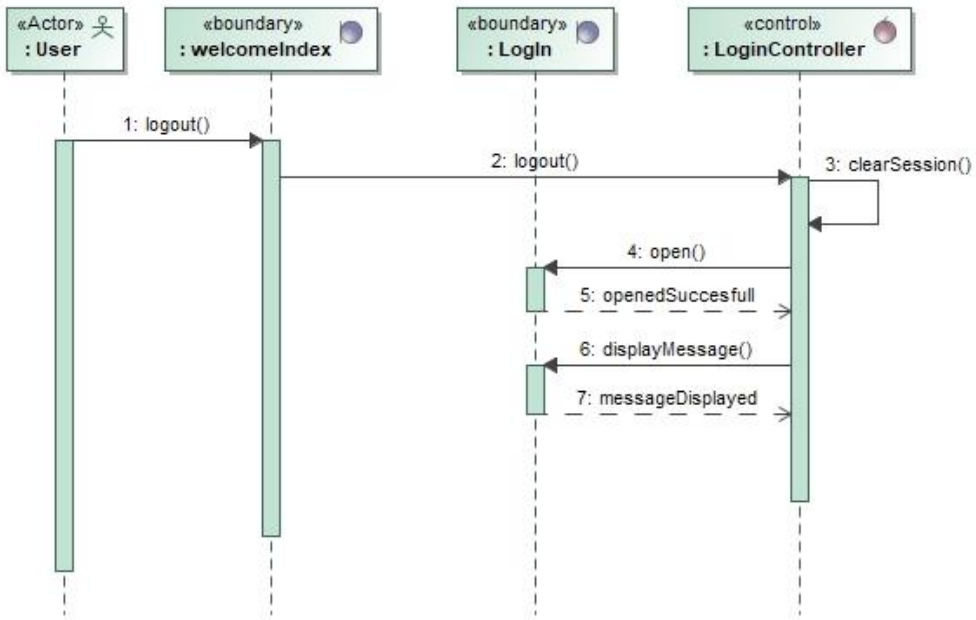
Atsijungti

Prisijungimo posistemei priskiriamo panaudojimo atvejo „Atsijungti“ realizavimo projekto klasėmis diagrama (65 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



65 pav. Atsijungimo realizavimo klasių diagrama

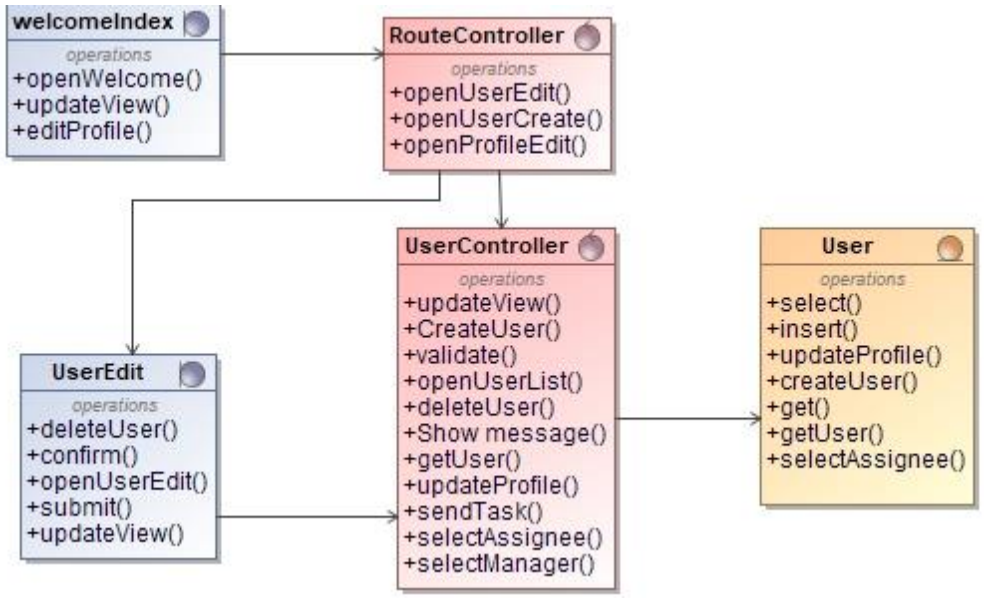
Prisijungimo posistemei priskiriamo panaudojimo atvejo „Atsijungti“ sekų diagrama (66 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



66 pav. Atsijungimo sekų diagrama

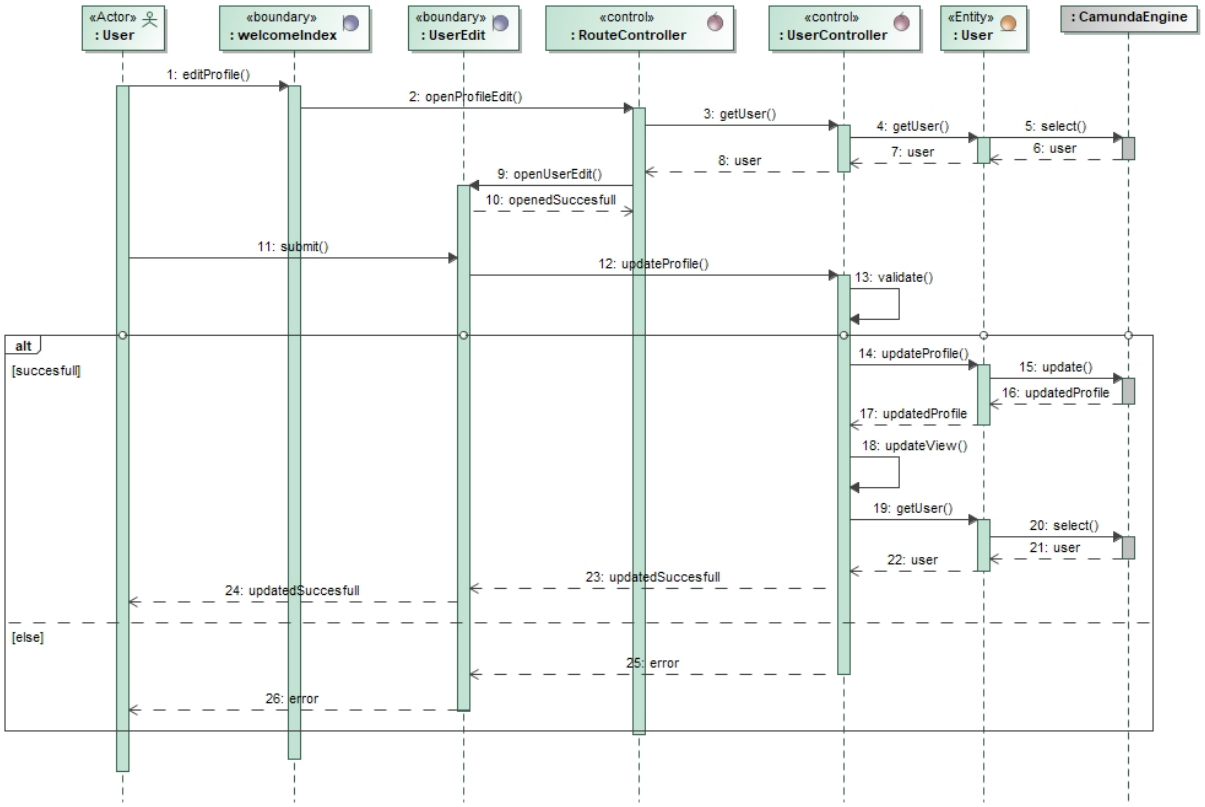
Redaguoti profilį

Prisijungimo posistemei priskiriamo panaudojimo atvejo „Redaguoti profilį“ realizavimo projekto klasėmis diagrama (67 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



67 pav. Profilio redagavimo realizavimo klasių diagrama

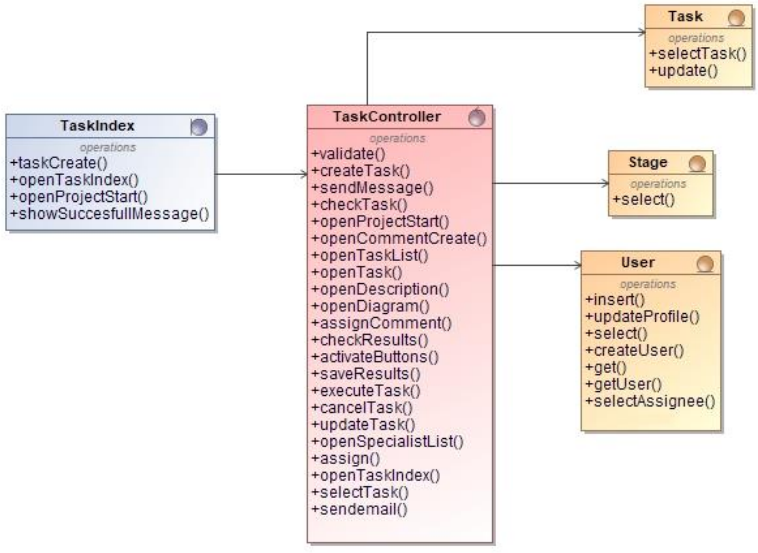
Prisijungimo posistemei priskiriamo panaudojimo atvejo „Redaguoti profilį“ sekų diagrama (68 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



68 pav. Profilio redagavimo sekų diagrama

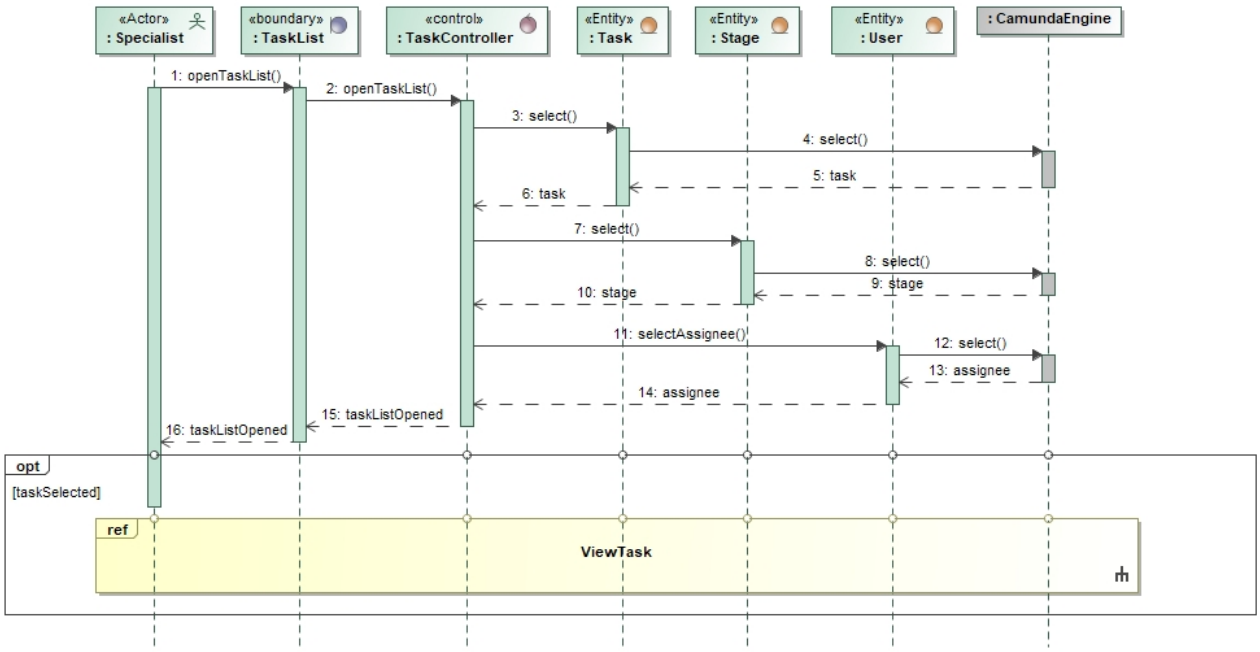
Peržiūrėti užduočių sąrašą

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Peržiūrėti užduočių sąrašą“ realizavimo projekto klasėmis diagrama (69 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



69 pav. Užduočių sąrašo peržiūrėjimo realizavimo klasių diagrama

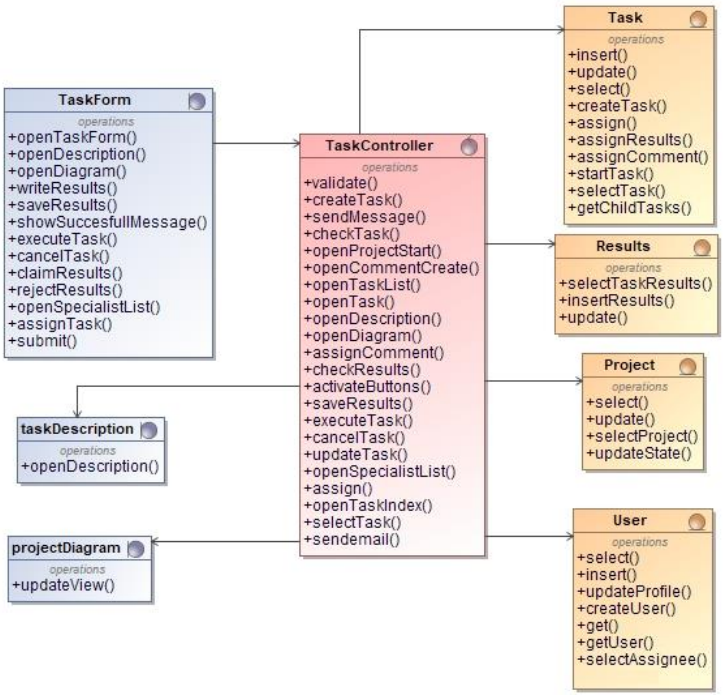
Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Peržiūrėti užduočių sąrašą“ sekų diagrama (70 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



70 pav. Užduočių sąrašo peržiūrėjimo sekų diagrama

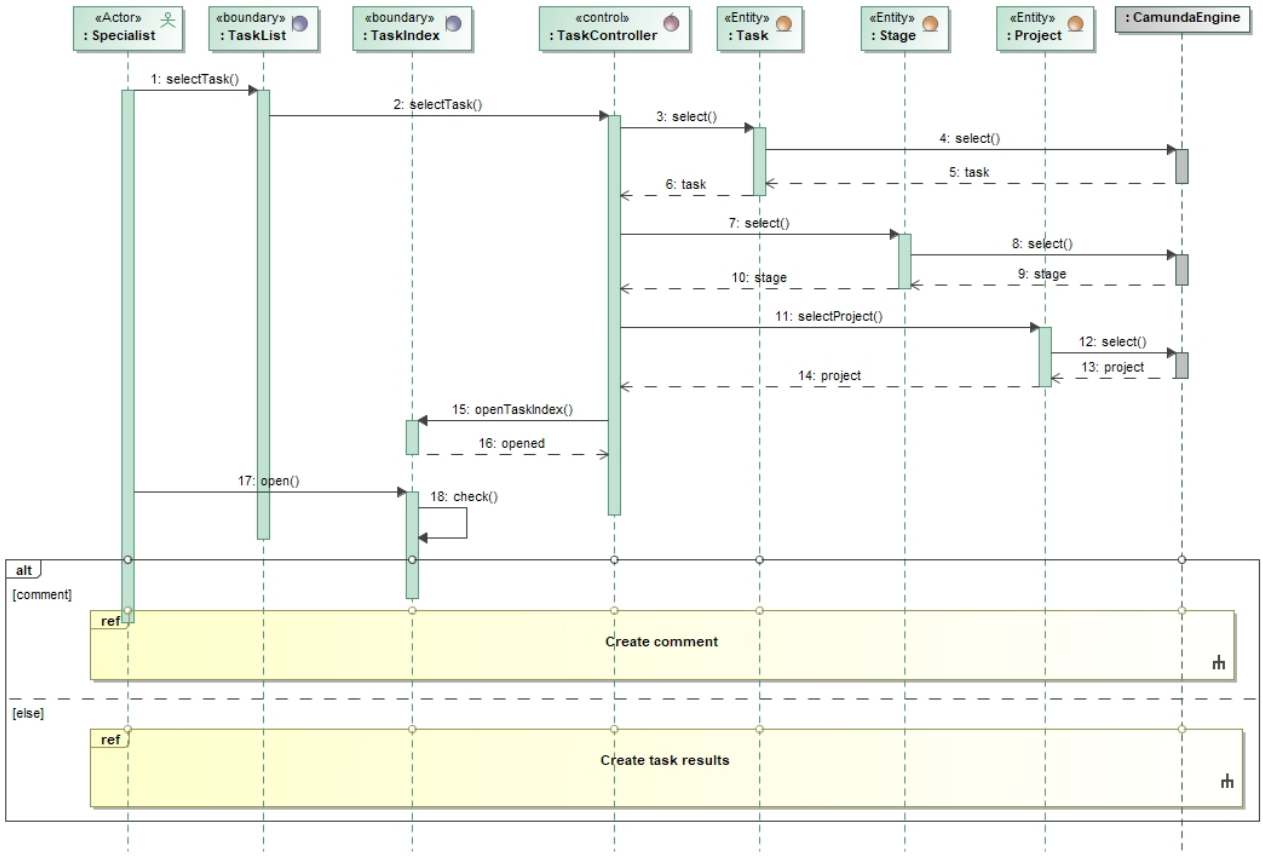
Peržiūrėti užduotį

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Peržiūrėti užduotį“ realizavimo projekto klasėmis diagrama (71 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



71 pav. Užduoties peržiūrėjimo realizavimo klasių diagrama

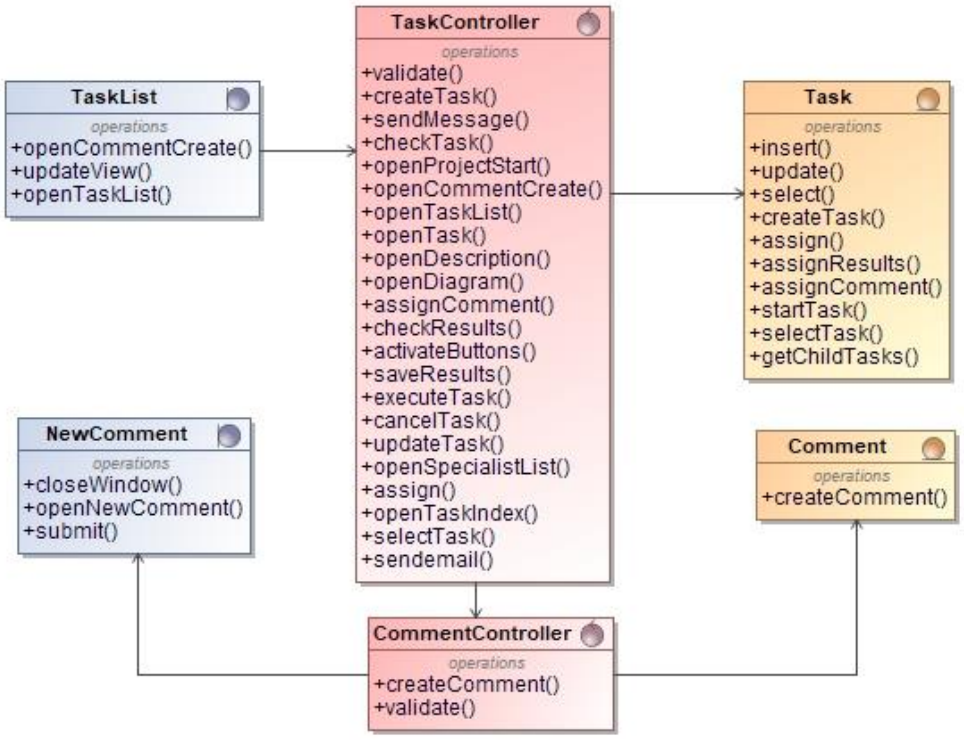
Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Peržiūrėti užduotį“ sekų diagrama (72 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



72 pav. Užduoties peržiūrėjimo sekų diagrama

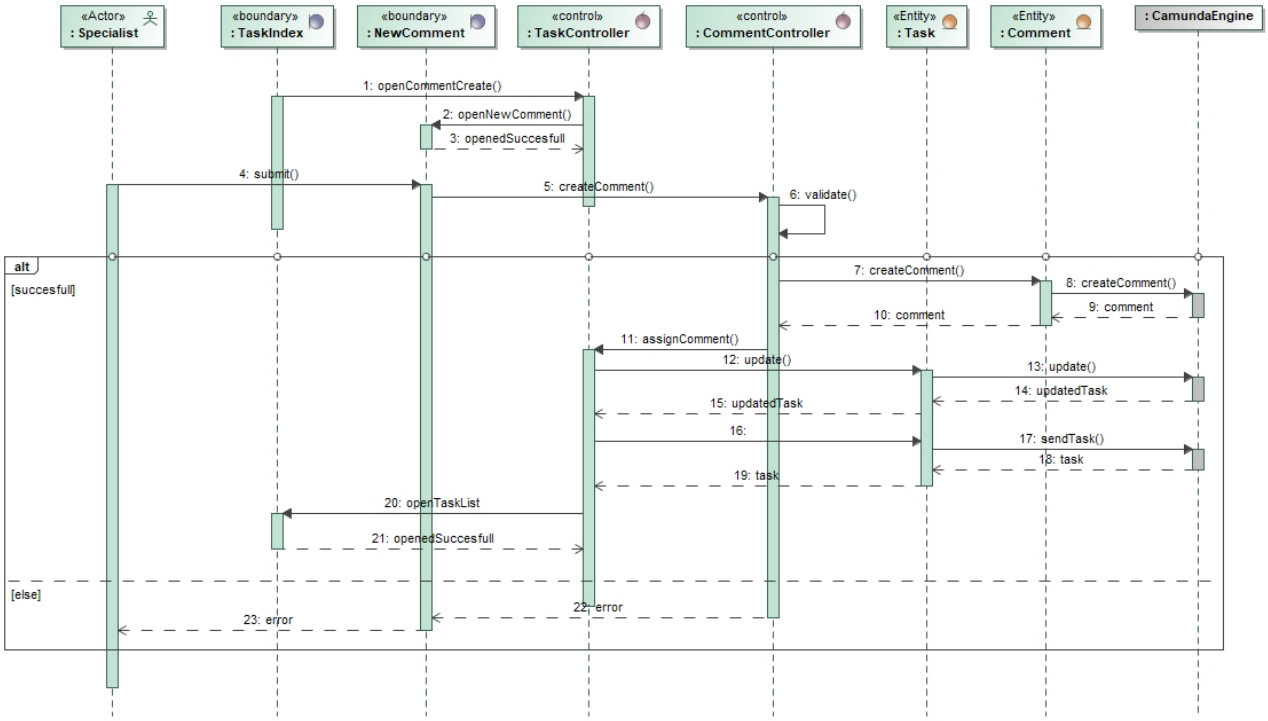
Parašyti komentarą

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Parašyti komentarą“ realizavimo projekto klasėmis diagrama (73 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



73 pav. Komentaro kūrimo realizavimo klasių diagrama

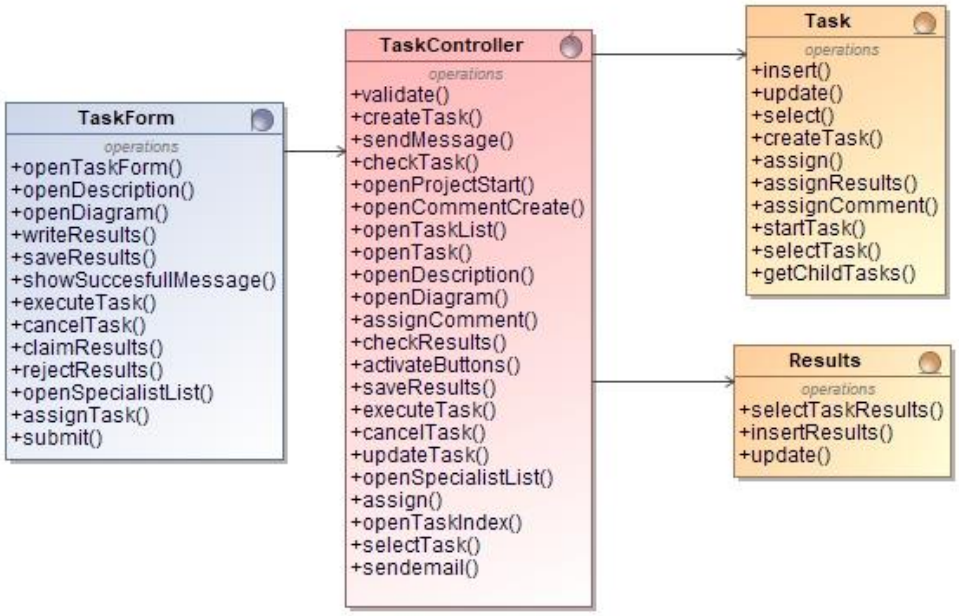
Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Parašyti komentarą“ sekų diagrama (74 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



74 pav. Komentaro kūrimo sekų diagrama

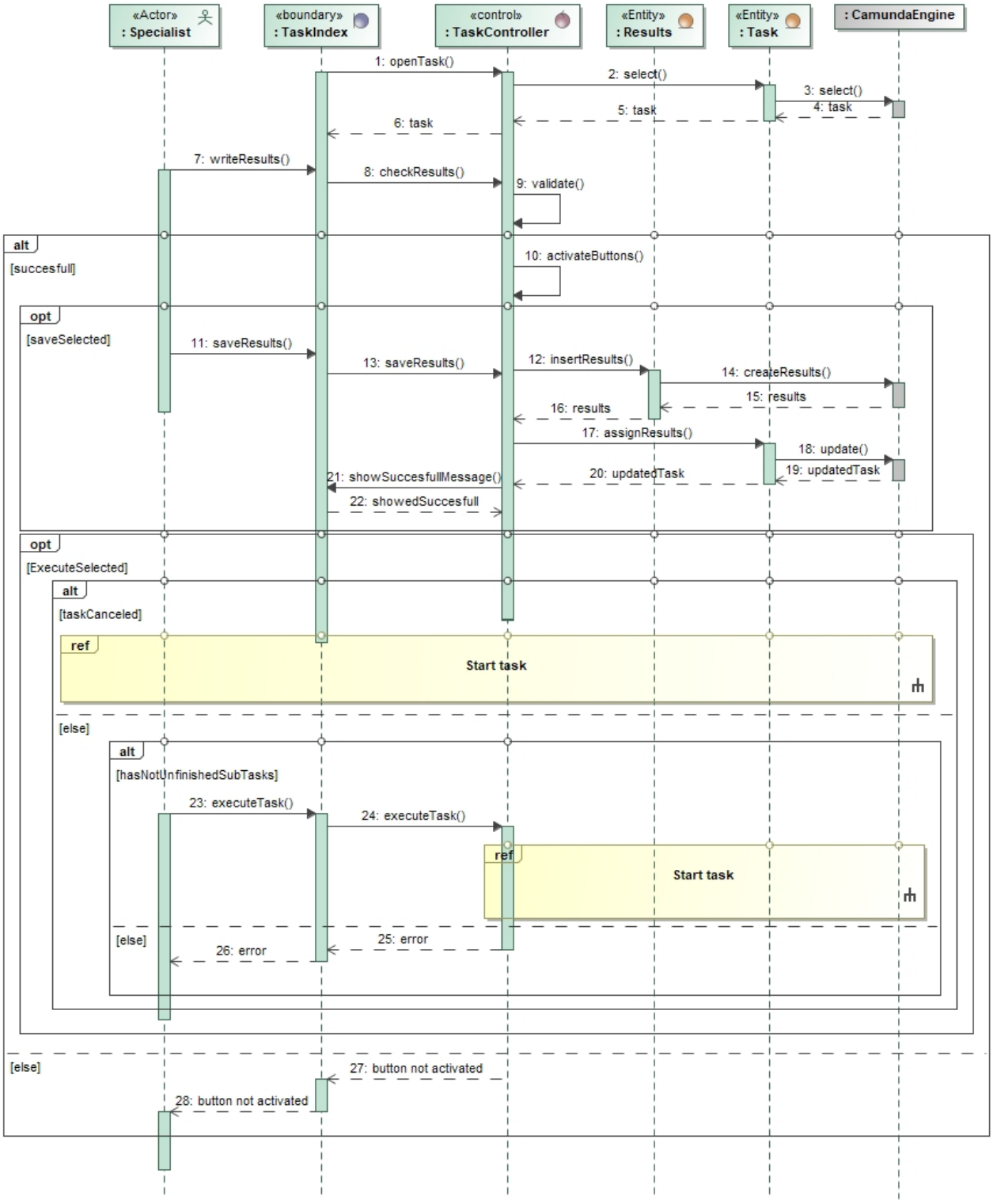
Užpildyti užduoties rezultatų formą

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Užpildyti užduoties rezultatų formą“ realizavimo projekto klasėmis diagrama (75 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



75 pav. Užduoties rezultatų pildymo realizavimo klasių diagrama

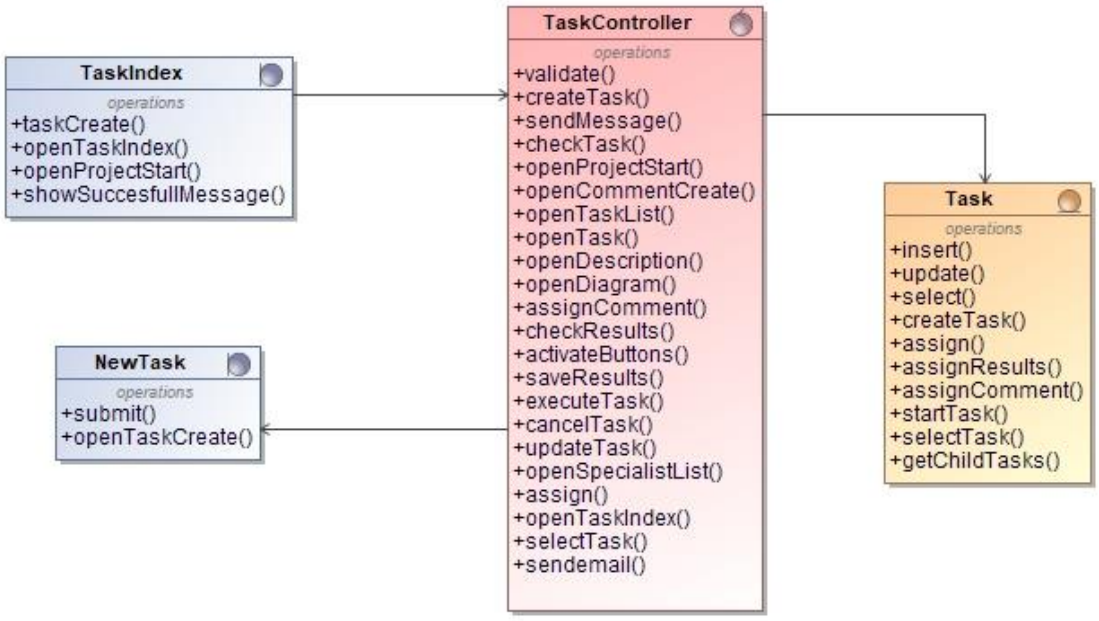
Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Užpildyti užduoties rezultatų formą“ sekų diagrama (76 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



76 pav. Užduoties rezultatų pildymo sekų diagrama

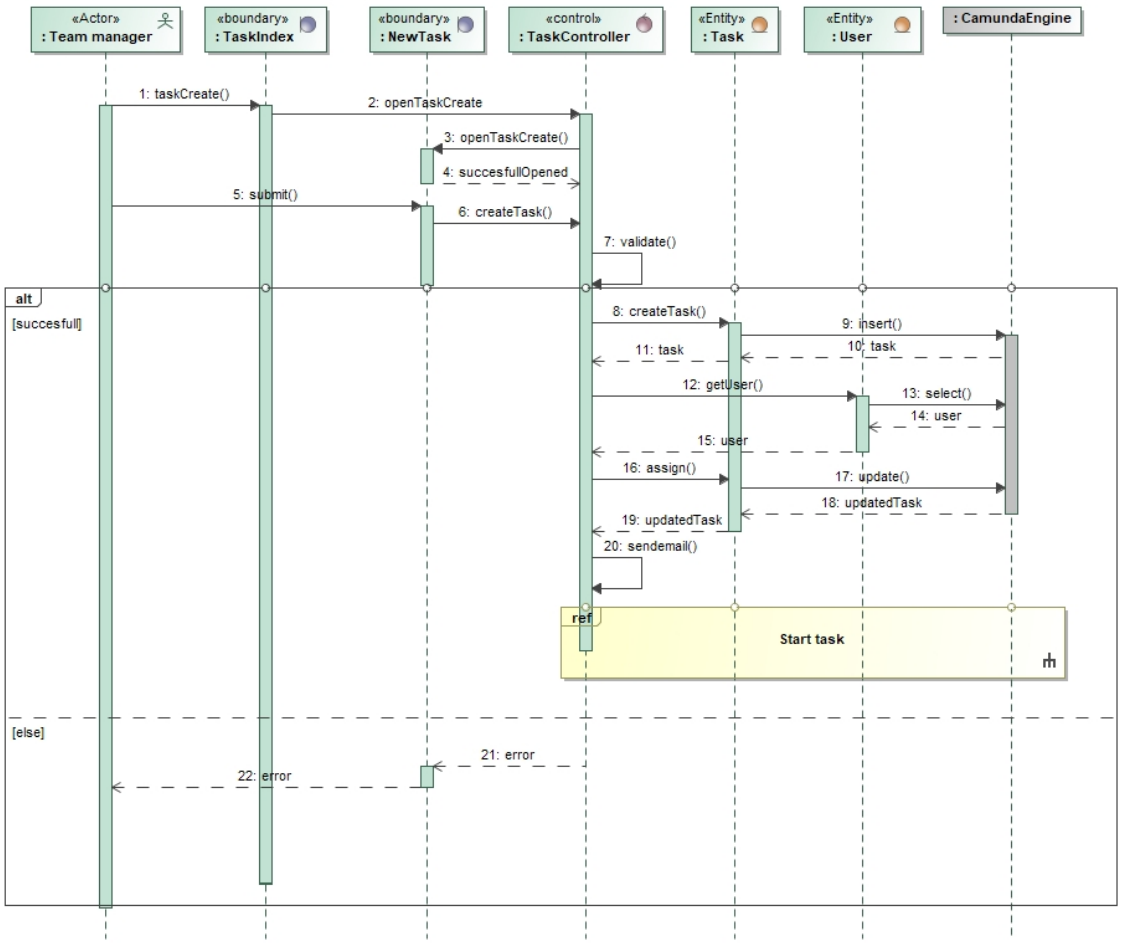
Sukurti užduotį

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Sukurti užduotį“ realizavimo projekto klasėmis diagrama (77 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



77 pav. Užduoties kūrimo realizavimo klasių diagrama

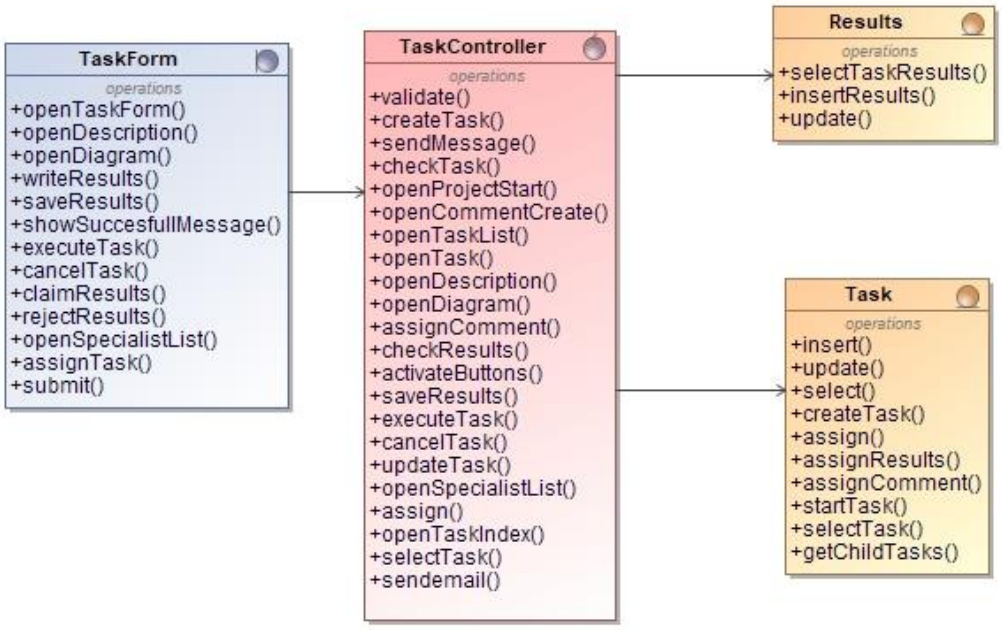
Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Sukurti užduotį“ sekų diagrama (78 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



78 pav. Užduoties kūrimo sekų diagrama

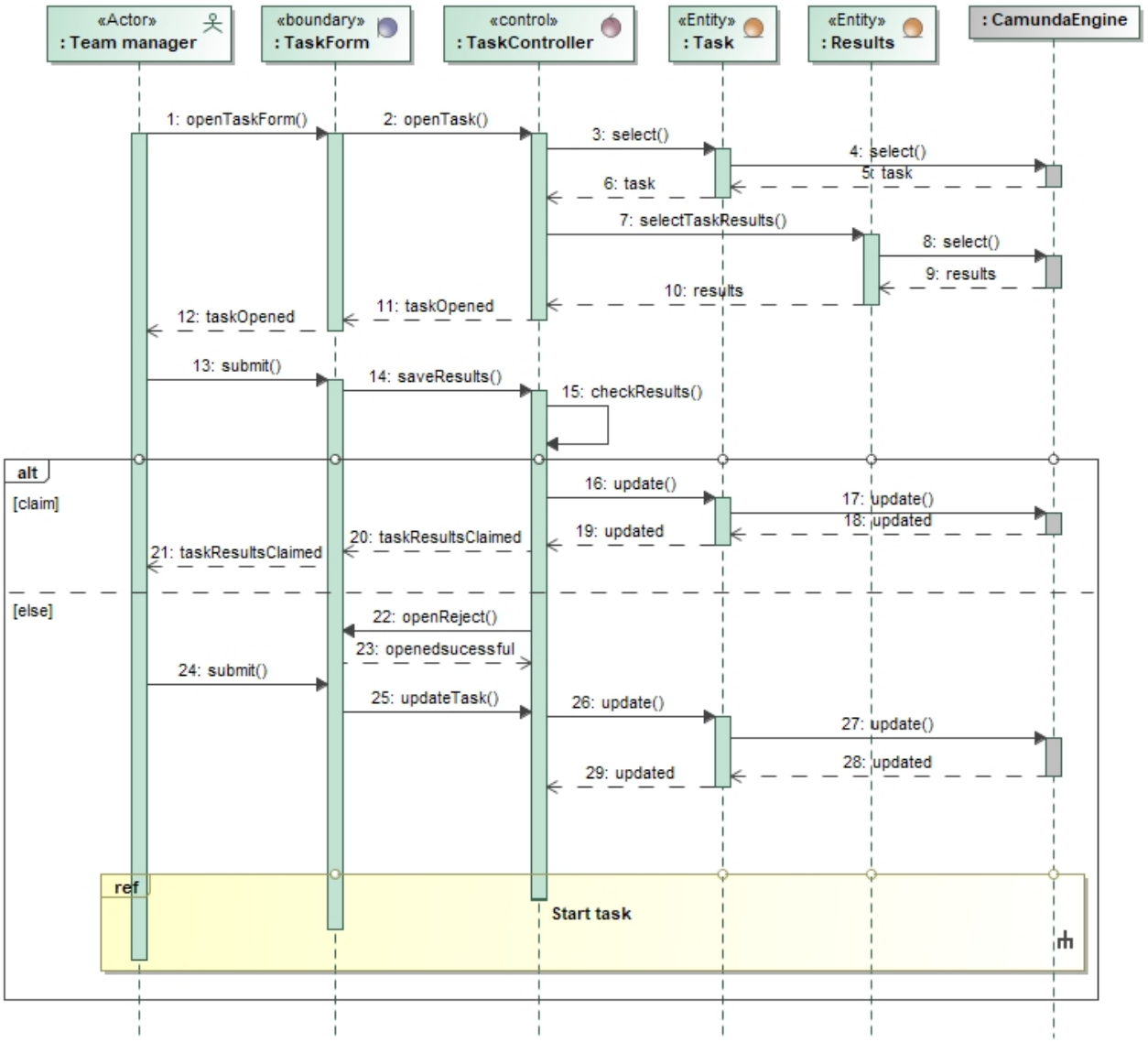
Tikrinti užduoties testavimo rezultatus

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Tikrinti užduoties rezultatus“ realizavimo projekto klasėmis diagrama (79 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



79 pav. Užduoties rezultatų tikrinimo realizavimo klasių diagrama

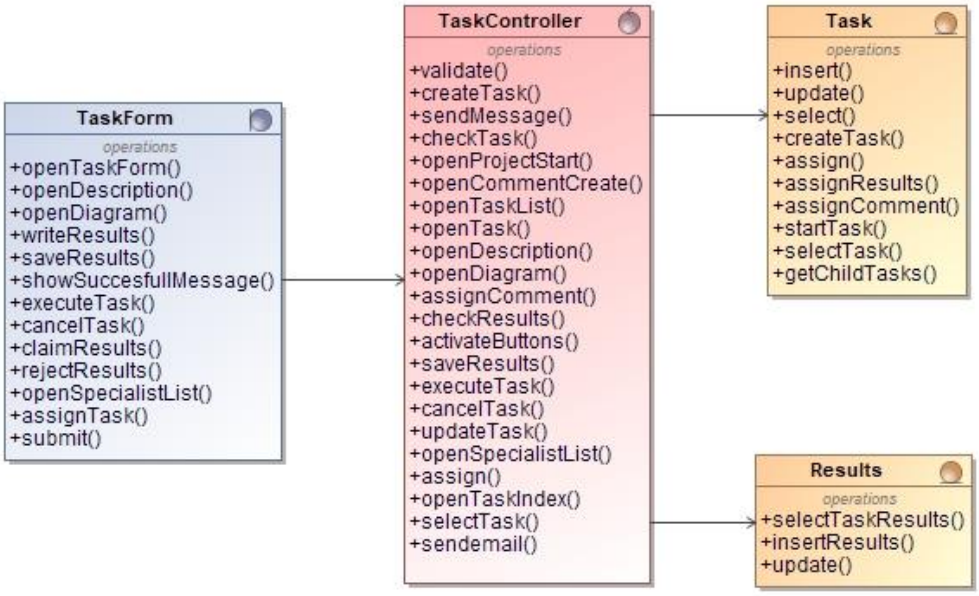
Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Tikrinti užduoties rezultatus“ sekų diagrama (80 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



80 pav. Užduoties rezultatų tikrinimo sekų diagrama

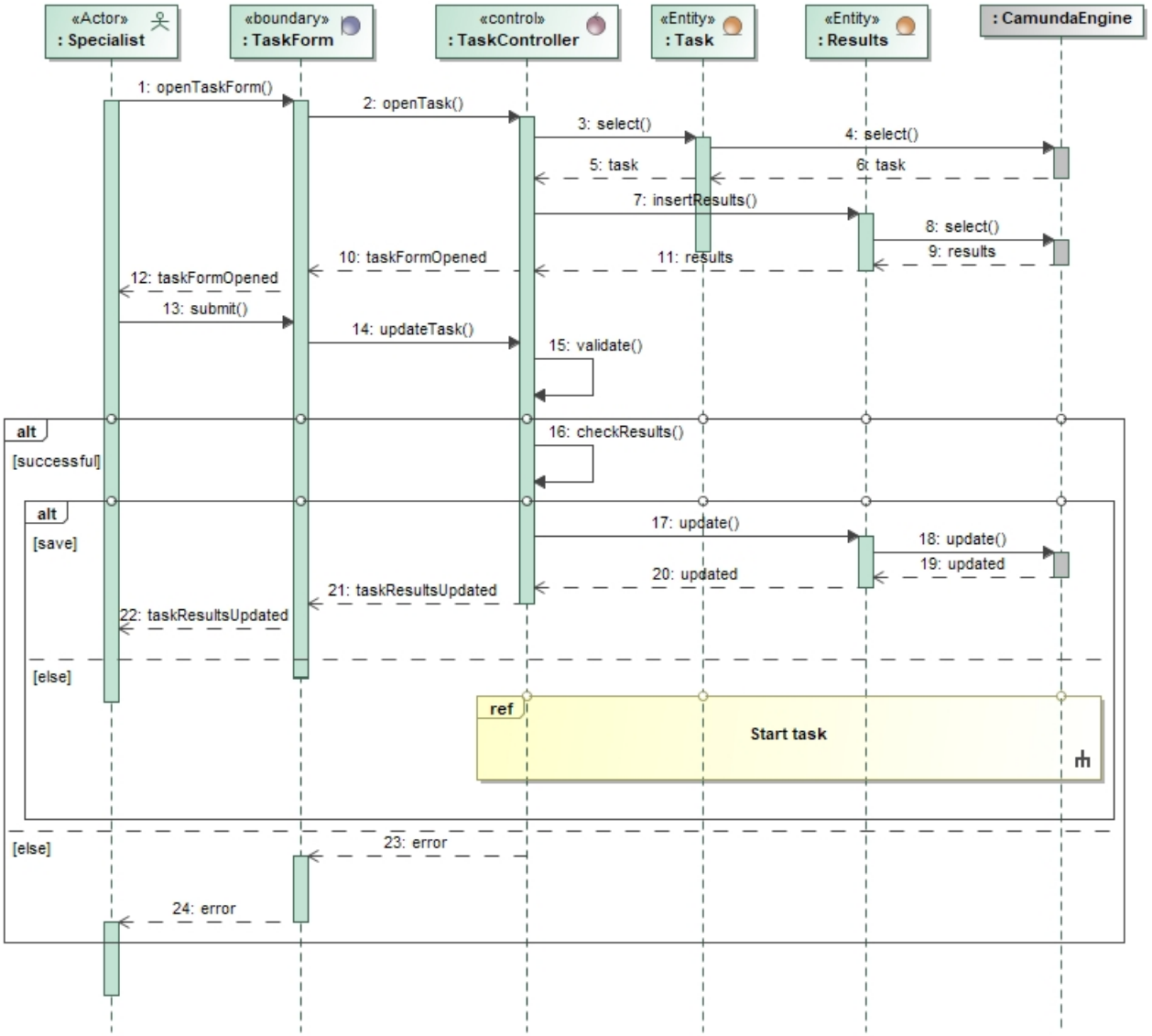
Taisyti testavimo užduoties rezultatus

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Taisyti užduoties rezultatus“ realizavimo projekto klasėmis diagrama (81 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



81 pav. Užduoties taisymo realizavimo klasių diagrama

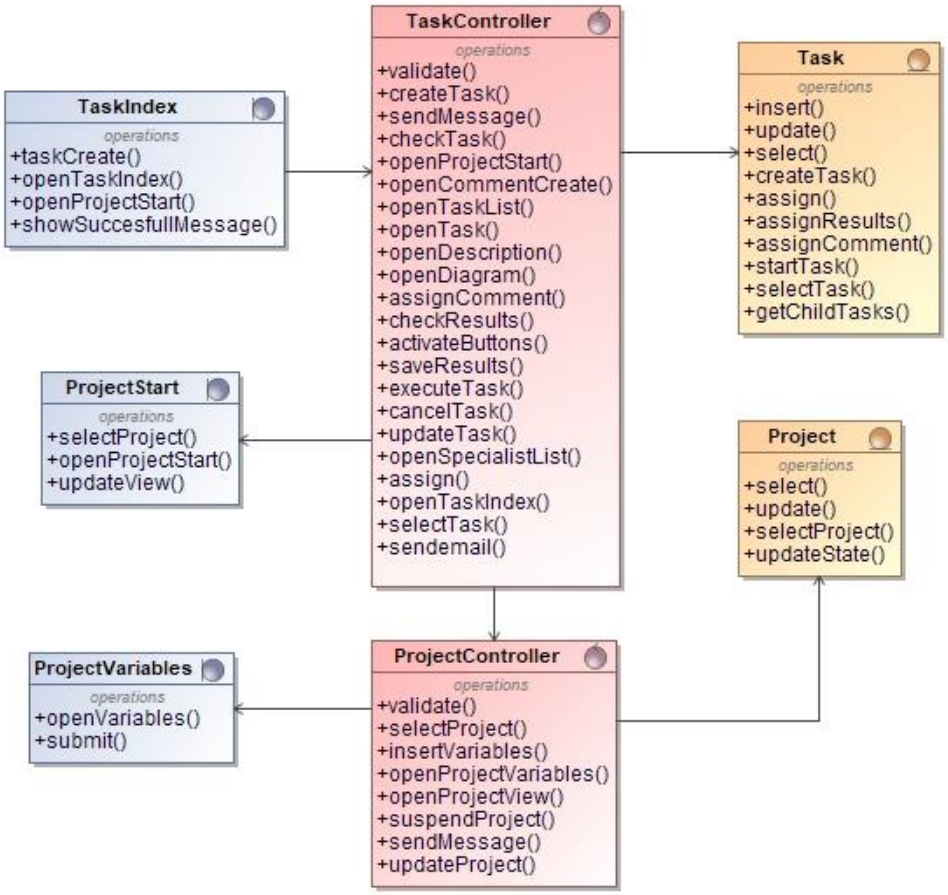
Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Taisyti užduoties rezultatus“ sekų diagrama (82 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



82 pav. Užduoties taisymo sekų diagrama

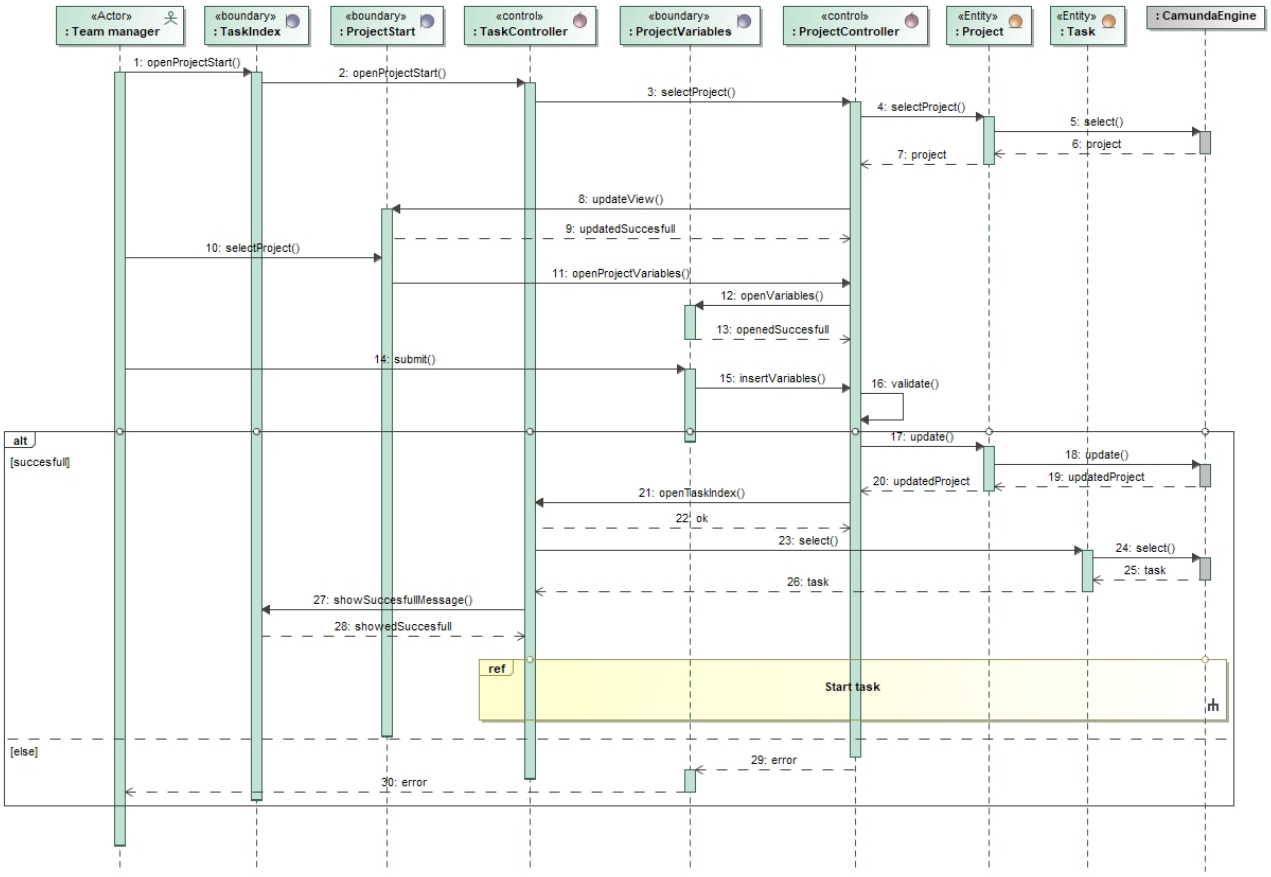
Inicijuoti testavimo proceso etapą

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Inicijuoti testavimo proceso etapą“ realizavimo projekto klasėmis diagrama (83 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



83 pav. Testavimo etapo inicijavimo realizavimo klasių diagrama

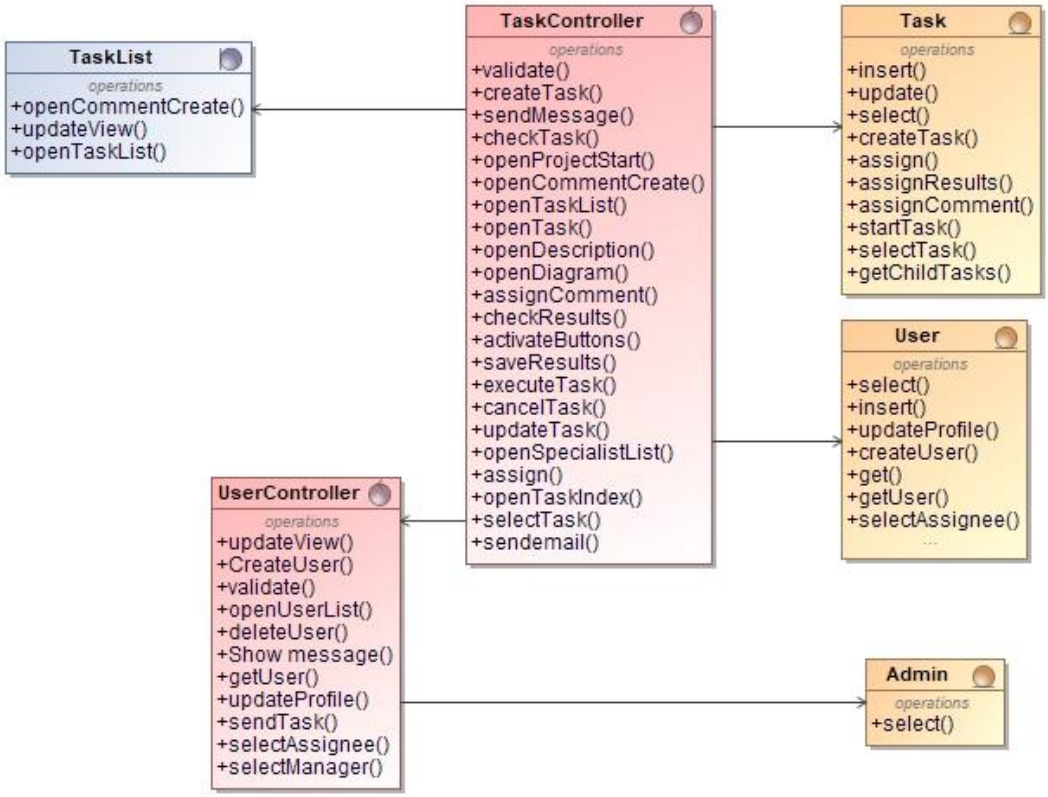
Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Inicijuoti testavimo proceso etapą“ sekų diagrama (84 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



84 pav. Testavimo etapo inicijavimo sekų diagrama

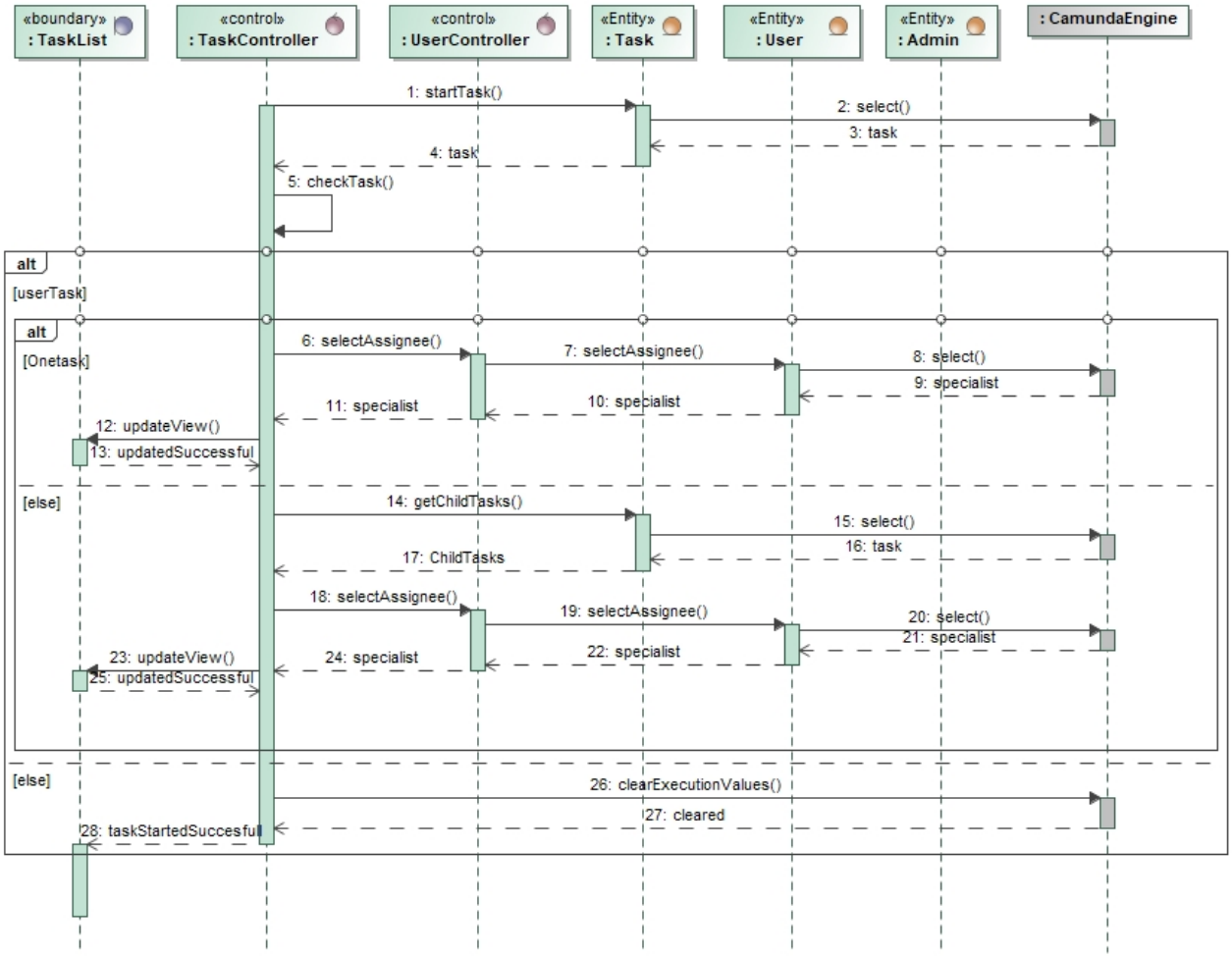
Inicijuoti testavimo užduoties vykdymą

Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Inicijuoti užduoties vykdymą“ realizavimo projekto klasėmis diagrama (85 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



85 pav. Užduoties inicijavimo realizavimo klasių diagrama

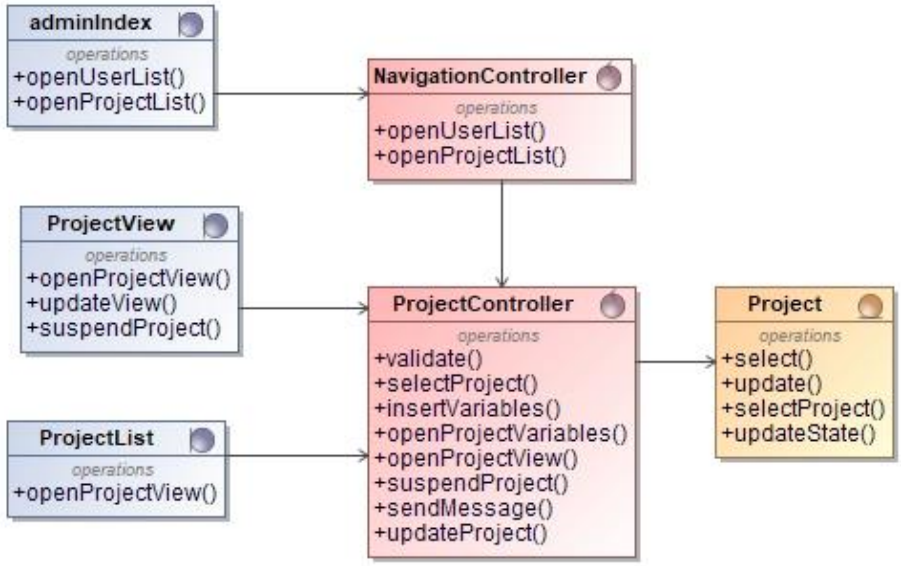
Užduočių sąrašo posistemei priskiriamo panaudojimo atvejo „Inicijuoti užduoties vykdymą“ sekų diagrama (86 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



86 pav. Užduoties inicijavimo sekų diagrama

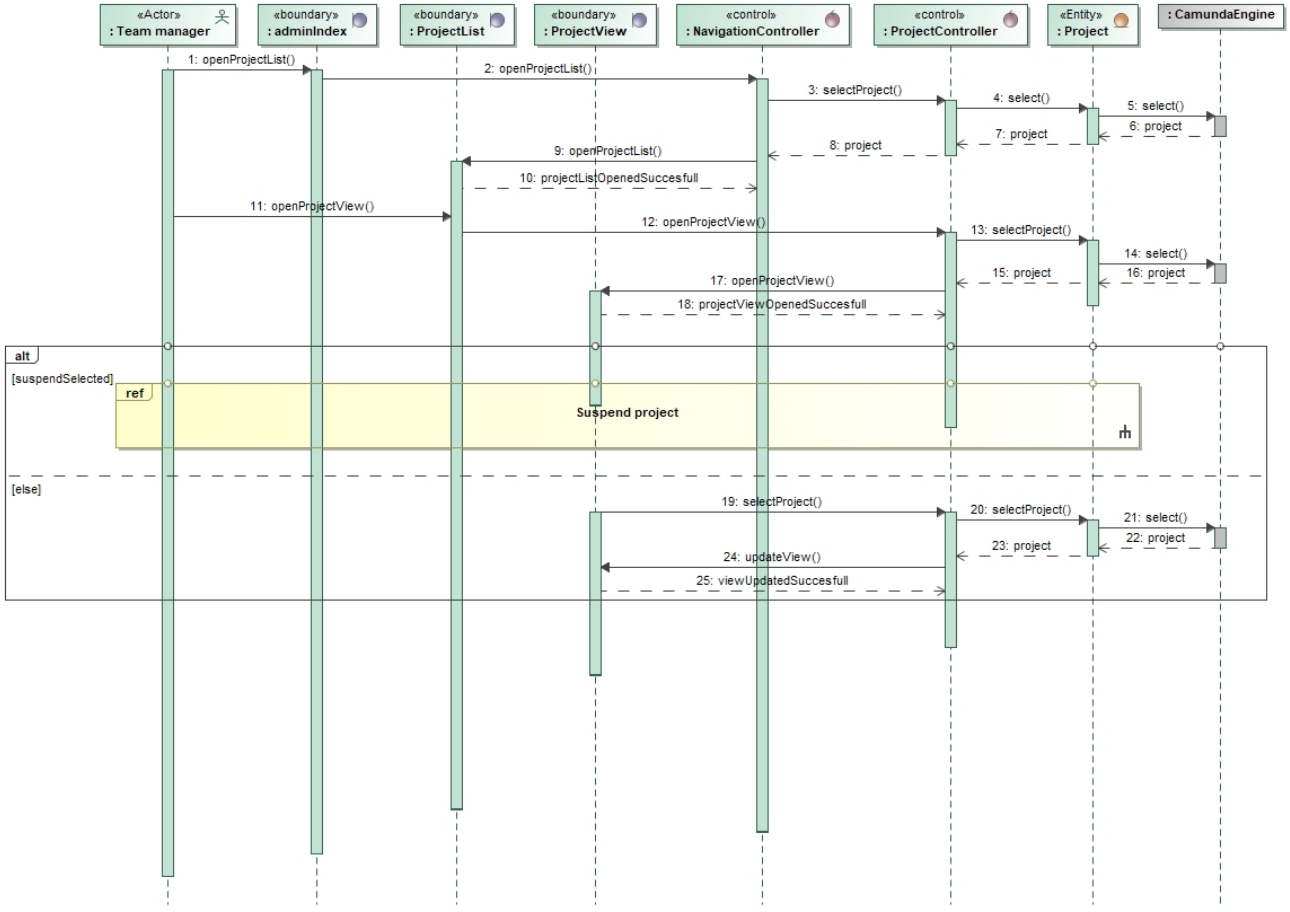
Stebėti vykdomą testavimo projektą

Analizės posistemei priskiriamo panaudojimo atvejo „Stebėti vykdomą projektą“ realizavimo projekto klasėmis diagrama (87 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



87 pav. Projekto valdymo realizavimo klasių diagrama

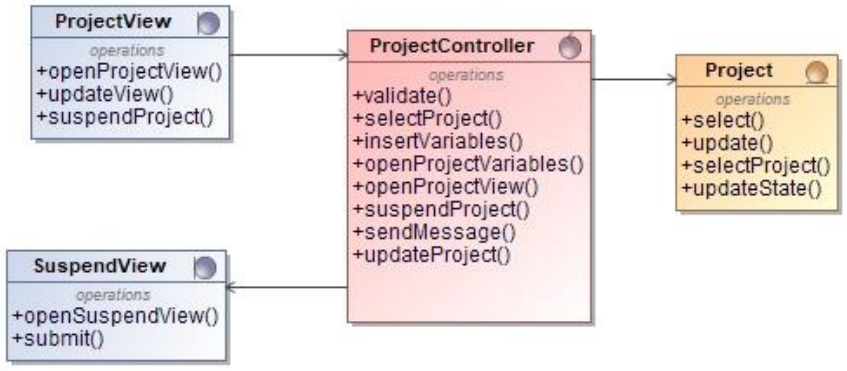
Analizės posistemei priskiriamo panaudojimo atvejo „Stebėti vykdomą projektą“ sekų diagrama (88 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



88 pav. Projekto valdymo sekų diagrama

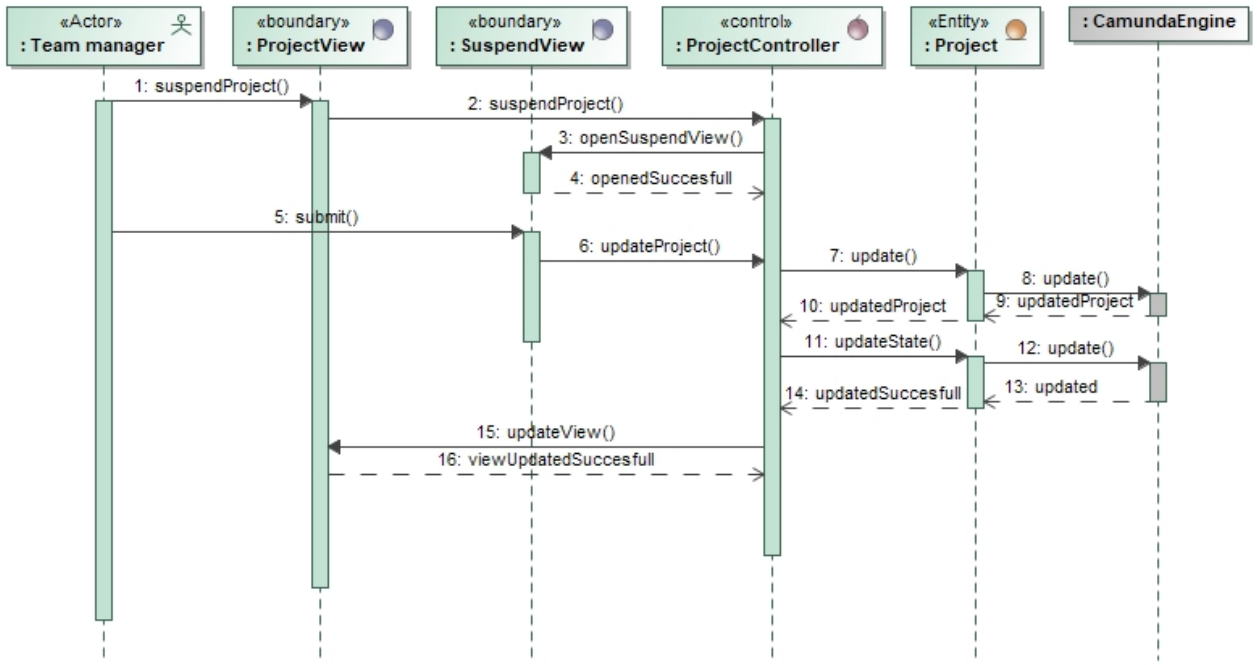
Nutraukti vykdomą testavimo projektą

Analizės posistemei priskiriamo panaudojimo atvejo „Nutraukti vykdomą projektą“ realizavimo projekto klasėmis diagrama (89 pav.) atspindi klases, kurios realizuoja panaudojimo atvejį ir tų klasių tarpusavio sąveikas.



89 pav. Projekto nutraukimo realizavimo klasių diagrama

Analizės posistemei priskiriamo panaudojimo atvejo „Nutraukti vykdomą projektą“ sekų diagrama (90 pav.) atspindi detalią panaudojimo atvejo veikimo logiką, bei operacijas, kuriomis bendrauja realizavimo klasės.



90 pav. Projekto nutraukimo sekų diagrama

4 priedas. Funkcinio testavimo duomenys

TC-1 testavimo rezultatai

Reikalavimai: FR-1.3.1 turi būti galimybė atsijungti nuo sistemos, FR-1.3.2 turi būti galimybė prisijungti prie sistemos, FR-1.3.3 sistemoje turi būti galimybė redaguoti profilį.

29 lentelė. TC-1 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: naudotojas bando prisijungti prie sistemos. **Duomenys**: nurodomi korektiški duomenys.	Sėkmingas	Sėkmingas
2.	**Veiksmas**: naudotojas bando prisijungti prie sistemos. **Duomenys**: nurodomas nekorektiškas prisijungimo vardas.	Klaida	Klaida
3.	**Veiksmas**: naudotojas bando prisijungti prie sistemos. **Duomenys**: nurodomas nekorektiškas slaptažodis.	Klaida	Klaida
4.	**Veiksmas**: naudotojas bando atsijungti iš užduočių sąrašo aplikacijos.	Sėkmingas	Sėkmingas
5.	**Veiksmas**: naudotojas bando atsijungti iš administratoriaus aplikacijos.	Sėkmingas	Sėkmingas
6.	**Veiksmas**: naudotojas bando atsijungti iš analizės aplikacijos.	Sėkmingas	Sėkmingas
7.	**Veiksmas**: naudotojas pakeičia profilio duomenis, bet paspaudžia mygtuką [Cancel].	Duomenys neatnaujinami	Duomenys neatnaujinami
8.	**Veiksmas**: naudotojas redaguoja profilio duomenis. **Duomenys**: neužpildomi privalomi laukai.	Mygtukas [Update] neaktyvus	Mygtukas [Update] neaktyvus
9.	**Veiksmas**: naudotojas redaguoja profilio duomenis ir spaudžia mygtuką [Update]. **Duomenys**: užpildomi privalomi laukai, duomenys korektiški.	Duomenys atnaujinami	Duomenys atnaujinami
10.	**Veiksmas**: naudotojas nieko nepakeičia iš profilio duomenų ir bando spausti mygtuką [Update].	Mygtukas [Update] neaktyvus	Mygtukas [Update] neaktyvus
11.	**Veiksmas**: naudotojas keičia elektroninio pašto adresą. **Duomenys**: lauke įvedamas ne elektroninis paštas.	Klaida	Klaida
12.	**Veiksmas**: naudotojas keičia slaptažodį. **Duomenys:** nurodomi korektiški duomenys.	Sėkmingas	Sėkmingas
13.	**Veiksmas**: naudotojas keičia slaptažodį. **Duomenys:** antrą kartą įvedamas nesutampantis naujas slaptažodis.	Klaida	Klaida

TC-2 testavimo rezultatai

Reikalavimas: FR-1.4.9. sistemoje turi būti galimybė inicijuoti testavimo etapą.

30 lentelė. TC-2 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: testuotojų grupės vadovas inicijuoja testavimo etapą.	Sėkmingas	Sėkmingas
2.	**Veiksmas**: komandos narys bando inicijuoti projekto etapą.	Klaida	Klaida

TC-3 testavimo rezultatai

Reikalavimas: FR-1.4.1. sistemoje turi būti galimybė sukurti užduotį.

31 lentelė. TC-3 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: naudotojas pildo užduoties „Priskirti atsakingą testuotoją“ rezultatų formą ir spaudžia mygtuką [Complete]. **Duomenys:** privalomi laukai užpildomi, įvedami korektiški duomenys.	Sėkmingas	Sėkmingas
2.	**Veiksmas**: naudotojas pildo užduoties „Priskirti atsakingą testuotoją“ rezultatų formą ir spaudžia mygtuką [Complete]. **Duomenys:** neužpildomi privalomi laukai.	Klaida	Klaida
3.	**Veiksmas**: naudotojas pildo užduoties „Priskirti atsakingą testuotoją“ rezultatų formą ir spaudžia mygtuką [Complete]. **Duomenys**: įvedamas klaidingas datos formatas.	Klaida	Klaida
4.	**Veiksmas**: naudotojas pildo užduoties „Priskirti atsakingą testuotoją“ rezultatų formą ir spaudžia mygtuką [Save]. **Duomenys:** privalomi laukai užpildomi, įvedami korektiški duomenys.	Sėkmingas	Sėkmingas
5.	**Veiksmas**: naudotojas pildo užduoties „Priskirti atsakingą testuotoją“ rezultatų formą ir spaudžia mygtuką [Save]. **Duomenys:** neužpildomi privalomi laukai.	Klaida	Klaida
6.	**Veiksmas**: naudotojas pildo užduoties „Priskirti atsakingą testuotoją“ rezultatų formą ir spaudžia mygtuką [Save]. **Duomenys**: įvedamas klaidingas datos formatas.	Klaida	Klaida

TC-4 testavimo rezultatai

Reikalavimai: FR-1.5.1. sistemoje turi būti galimybė stebėti testavimo procesą, FR-1.5.2. sistemoje turi būti galimybė nutraukti testavimo procesą.

32 lentelė. TC-4 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: komandos vadovas bando prisijungti prie stebėjimo aplikacijos ir peržiūrėti, ar duomenys atnaujinami realiu laiku.	Sėkmingas	Sėkmingas
2.	**Veiksmas**: projektų vadovas bando prisijungti prie stebėjimo aplikacijos.	Sėkmingas	Sėkmingas
3.	**Veiksmas**: komandos narys bando prisijungti prie stebėjimo aplikacijos.	Klaida	Klaida
4.	**Veiksmas:** komandos vadovas bando prisijungti prie stebėjimo aplikacijos ir nutraukti vykdomą procesą.	Sėkmingas	Sėkmingas

TC-5 testavimo rezultatai

Reikalavimai: FR-1.4.7. sistemoje turi būti galimybė užpildyti užduoties rezultatų formą, FR-1.4.2. sistemoje turi būti galimybė peržiūrėti užduotį, FR-1.4.4. sistemoje turi būti galimybė peržiūrėti užduočių sąrašą, FR-1.4.6. sistemoje turi būti galimybė pakomentuoti užduotį.

33 lentelė. TC-5 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: naudotojas peržiūri priskirtų užduočių sąrašą.	Sėkmingas	Sėkmingas
2.	**Veiksmas**: naudotojas peržiūri priskirtą užduotį.	Sėkmingas	Sėkmingas
3.	**Veiksmas**: naudotojas parašo komentarą ir spaudžia mygtuką [Close].	Komentaras neišsaugomas	Komentaras neišsaugomas
4.	**Veiksmas**: naudotojas parašo komentarą ir spaudžia mygtuką [Save].	Komentaras išsaugomas	Komentaras išsaugomas

TC-6 testavimo rezultatai

Reikalavimas: FR-1.4.3. sistemoje turi būti galimybė patikrinti užduoties rezultatus.

34 lentelė. TC-6 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: komandos vadovas atmeta užduotį. **Duomenys**: atmetimo priežastis nenurodoma.	Sėkmingas	Sėkmingas
2.	**Veiksmas**: komandos vadovas atmeta užduotį. **Duomenys**: atmetimo priežastis nurodoma.	Sėkmingas	Sėkmingas
3.	**Veiksmas**: komandos vadovas patvirtina užduotį.	Sėkmingas	Sėkmingas

TC-7 testavimo rezultatai

Reikalavimas: FR-1.4.5. Sistemoje turi būti galimybė pataisyti užduoties rezultatus.

35 lentelė. TC-7 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: testuotojas užpildo užduoties „Taisyti užduoties rezultatus“ rezultatus ir išsaugo.	Sėkmingas	Sėkmingas

5 priedas. Nefunkcinio testavimo duomenys

NFR-1 testavimo rezultatai

Reikalavimas: sistema apriboja galimybę suklysti naudotojui įvedinėjant duomenis – kur įmanoma, pateikiamas „pasirinkimų sąrašas“.

36 lentelė. NFR-1 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: naudotojas peržiūri užduočių sąrašo posistemės formas, patikrina įvedamus laukus.	Sėkmingas	Sėkmingas
2.	**Veiksmas**: naudotojas peržiūri administratoriaus posistemės formas, patikrina įvedamus laukus.	Sėkmingas	Sėkmingas
3.	**Veiksmas**: naudotojas peržiūri stebėjimo posistemės formas, patikrina įvedamus laukus.	Sėkmingas	Sėkmingas

NFR-2 testavimo rezultatai

Reikalavimas: klaidos, sėkmingos operacijos atlikimo atvejais vartotojui sistema pateikia atsako pranešimus. Klaidos atveju – pateikiamas raudonos spalvos klaidos pranešimas, sėkmės atveju – pateikiamas žalios spalvos informacinis pranešimas.

37 lentelė. NFR-2 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: naudotojas peržiūri užduočių sąrašo posistemės formas, atlieka bent vieną sėkmingą veiksmą ir peržiūri informacinį pranešimą.	Sėkmingas	Sėkmingas
2.	**Veiksmas**: naudotojas peržiūri administratoriaus posistemės formas, atlieka bent vieną sėkmingą veiksmą ir peržiūri informacinį pranešimą.	Sėkmingas	Sėkmingas
3.	**Veiksmas**: naudotojas peržiūri stebėjimo posistemės formas, atlieka bent vieną sėkmingą veiksmą ir peržiūri informacinį pranešimą.	Sėkmingas	Sėkmingas
4.	**Veiksmas**: naudotojas peržiūri užduočių sąrašo posistemės formas, patikrina, įveda klaidingus duomenis ir peržiūri klaidos pranešimą.	Sėkmingas	Sėkmingas
5.	**Veiksmas**: naudotojas peržiūri administratoriaus posistemės formas, įveda klaidingus duomenis ir peržiūri klaidos pranešimą.	Sėkmingas	Sėkmingas
6.	**Veiksmas**: naudotojas peržiūri stebėjimo posistemės formas, įveda klaidingus duomenis ir peržiūri klaidos pranešimą.	Sėkmingas	Sėkmingas

NFR-3 testavimo rezultatai

Reikalavimas: atliekant šalinimo, redagavimo ar kūrimo funkcijas yra galimybė atšaukti funkcijos vykdymą.

38 lentelė. NFR-3 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: naudotojas peržiūri užduočių sąrašo posistemės formas, patikrina, ar yra galimybė atšaukti vykdomą funkciją.	Sėkmingas	Sėkmingas
2.	**Veiksmas**: naudotojas peržiūri administratoriaus posistemės formas, patikrina, ar yra galimybė atšaukti vykdomą funkciją.	Sėkmingas	Sėkmingas
3.	**Veiksmas**: naudotojas peržiūri stebėjimo posistemės formas, patikrina, ar yra galimybė atšaukti vykdomą funkciją.	Sėkmingas	Sėkmingas

NFR-4 testavimo rezultatai

Reikalavimas: sistema informaciją apie vykdomų projektų būklę vartotojui pateikia tiek tekstiniu, tiek grafiniu būdu.

39 lentelė. NFR-4 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: naudotojas peržiūri stebėjimo aplikaciją, patikrina, ar yra galimybė realiu laiku stebėti vykdomą projektą.	Sėkmingas	Sėkmingas

NFR-5 testavimo rezultatai

Reikalavimas: esant poreikiui turi būti galimybė modifikuoti procesą, pagal kurį veikia sistema.

40 lentelė. NFR-5 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: naudotojas proceso diagramoje prideda naują veiklą.	Sėkmingas	Sėkmingas

NFR-6 testavimo rezultatai

Reikalavimas: sistemai reikalinga ši programinė įranga: Java 8, MySQL 5.7.22, Apache Tomcat 8.5.

41 lentelė. NFR-6 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: naudotojas tikrina, ar serveryje yra ši programinė įranga: Java 8, MySQL 5.7.22, Apache Tomcat 8.5.	Sėkmingas	Sėkmingas

NFR-7 testavimo rezultatai

Reikalavimas: sistema palaikoma šiose naršyklėse: Google Chrome 72.0.3626.121 versija, Mozilla Firefox 65.0.2 versija, Internet Explorer 10 / 11.

42 lentelė. NFR-7 testavimo suvestinė

Eil. Nr.	Testavimo duomenys	Laukiamas rezultatas	Gautas rezultatas
1.	**Veiksmas**: naudotojas tikrina, ar sistema funkcionuoja Google Chrome naršyklės 72.0.3626.121 versijoje.	Sėkmingas	Sėkmingas
2.	**Veiksmas**: naudotojas tikrina, ar sistema funkcionuoja Mozilla Firefox naršyklės 65.0.2 versijoje.	Sėkmingas	Sėkmingas
3.	**Veiksmas**: naudotojas tikrina, ar sistema funkcionuoja Internet Explorer naršyklės 10 / 11 versijose.	Sėkmingas	Sėkmingas

6 priedas. Programinės įrangos testavimo metodikos eksperimentinės sistemos naudotojo instrukcija

Testavimo proceso valdymo sistemoje realizuoti septyni testavimo vykdymui būtini procesai: projekto etapo inicijavimo, analizės etapo, projektavimo etapo, programavimo etapo, testavimo etapo, priėmimo etapo. Sistema sudaryta iš trijų posistemių: administratoriaus, užduočių sąrašo, bei vykdomo projekto stebėjimo posistemės. Sistema skirta šiems naudotojų tipams:

- **Administratorius** – administratoriaus rolę turintis asmuo yra atsakingas už naudotojų duomenų valdymą, bei teisių priskyrimą.
- **Projektų vadovas** – projekto vadovo teisę turintis asmuo yra atsakingas už projekto etapų inicijavimą, bei vykdomo projekto stebėjimą.
- **Testavimo komandos vadovas** – komandos vadovo rolę turintis asmuo yra atsakingas už projekto etapo užduočių kūrimą, priskyrimą, tikrinimą, perskyrimą, bei vykdomo projekto stebėjimą.
- **Specialistas/testuotojas** – projekto komandos narys yra atsakingas už užduočių rezultatų pildymą, bei taisymą. Komandos narys taip pat gali atšaukti jam priskirtą užduotį.

Detalus sistemos funkcionalumas aprašomas pagrindinėms sistemos funkcijoms, kuriomis naudojasi testuotojai, projektų vadovai, bei testuotojų grupės vadovas.

Prisijungimas prie sistemos

Bet kokie veiksmai sistemoje gali būti atlikti tik prisijungus prie sistemos.

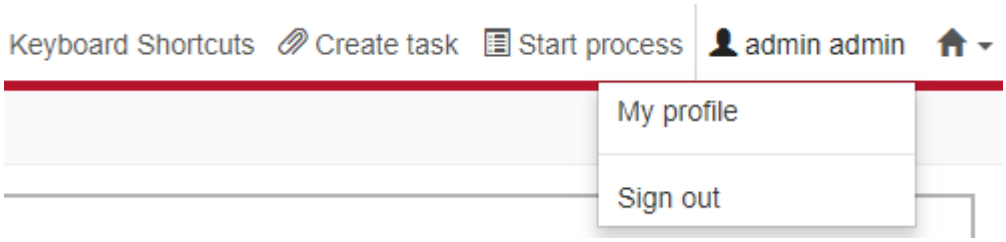
91 pav. Naudotojo prisijungimo langas

Jei norite prisijungti prie sistemos, atlikite šiuos veiksmus:
1. Interneto naršyklėje įveskite sistemos internetinį adresą.
2. Naudotojo prisijungimo lange įveskite naudotojo vardą ir slaptažodį;
3. Spauskite mygtuką [Sign in].

Rezultatas
Naudotojas prijungtas prie sistemos.

Atsijungimas nuo sistemos

Baigti darbą su sistema galima atsijungiant iš sistemos.



92 pav. Atsijungimo nuo sistemos meniu

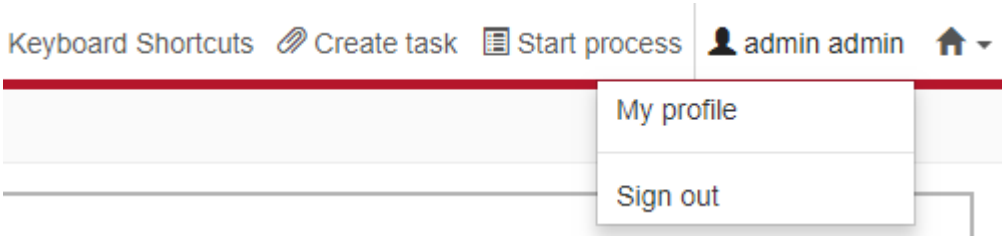
Jei norite atsijungti nuo sistemos, atlikite šiuos veiksmus:

1. Sistemos viršutiniame meniu paspauskite profilio simbolį.
2. Išsiskleidusiame pagalbiniame meniu paspauskite mygtuką [Sign out].

Rezultatas
Naudotojas atjungtas nuo sistemos.

Profilio redagavimas

Atnaujinti asmeninius duomenis galima pakeičiant profilio informaciją.



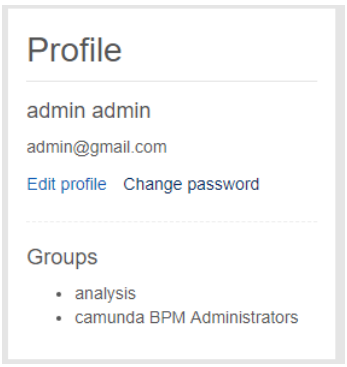
93 pav. Profilio redagavimo meniu

Jei norite pakeisti profilio duomenis, atlikite šiuos veiksmus:
1. Užduočių sąrašo posistemės viršutiniame meniu paspauskite profilio simbolį.
2. Išsiskleidusiame pagalbiniame meniu paspauskite mygtuką [My profile].
3. Atsidariusiame lange spauskite nuorodą [Edit profile].
4. Pakeiskite norimą informaciją.
5. Spauskite mygtuką [Update].

Rezultatas
Naudotojo profilio duomenys atnaujinti.

Slaptažodžio keitimas

Atnaujinti prisijungimo duomenis galima pakeičiant paskyros slaptažodį.



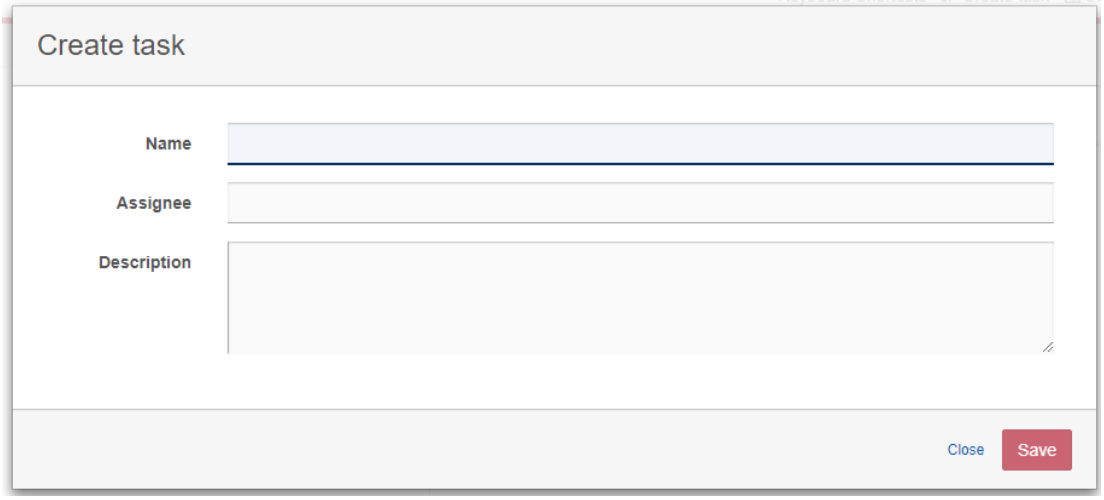
94 pav. Slaptažodžio redagavimas

Jei norite pakeisti slaptažodį, atlikite šiuos veiksmus:
1. Užduočių sąrašo posistemės viršutiniame meniu paspauskite profilio simbolį.
2. Išsiskleidusiame pagalbiniame meniu paspauskite mygtuką [My profile].
3. Atsidariusiame lange spauskite nuorodą [Change password].
4. Įveskite esamą ir būsimą slaptažodžius.
5. Spauskite mygtuką [Change password].

Rezultatas
Naudotojo slaptažodis atnaujintas.

Pagalbinės užduoties kūrimas

Įvykdyti procese nenumatytus darbus galima sukuriant pagalbinę užduotį.



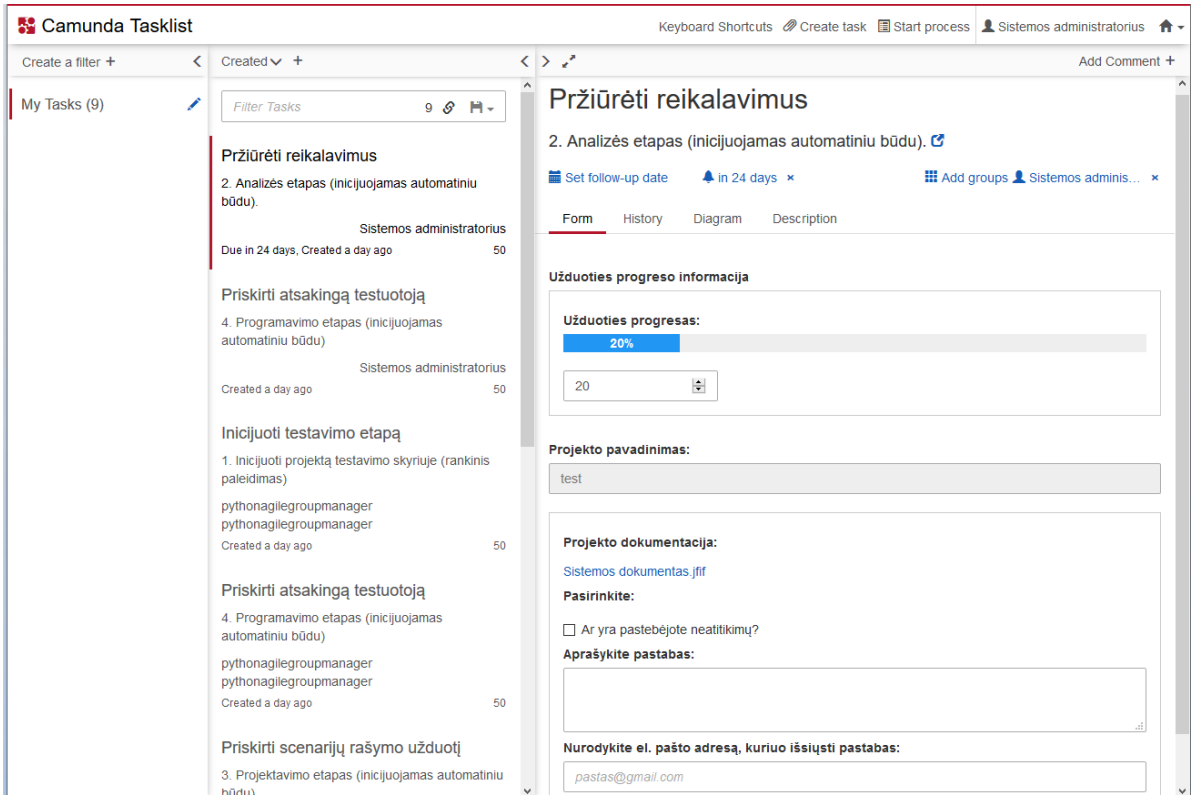
95 pav. Pagalbinės užduoties kūrimo langas

Jei norite sukurti pagalbinę užduotį, atlikite šiuos veiksmus:
1. Sistemos viršutiniame meniu paspauskite mygtuką [Create task].
2. Atsidariusiame lange įveskite užduoties pavadinimą, apibūdinimą ir priskirkite užduotį sistemos naudotojui.
3. Spauskite mygtuką [Save].

Rezultatas
Pagalbinė užduotis sukurta ir priskirta naudotojui.

Užduočių duomenų valdymas

Peržiūrėti artėjančius darbus ir užpildyti atliktų užduočių rezultatus galima atsidarius užduočių sąrašą.



96 pav. Užduočių sąrašo langas

Jei norite peržiūrėti užduotis, atlikite šiuos veiksmus:

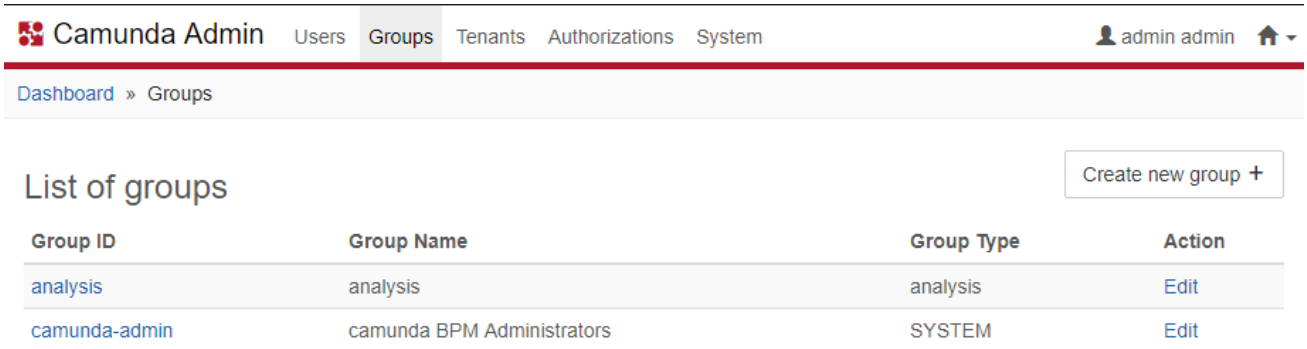
1. Užduočių sąrašo posistemėje atidarykite „My Tasks“.
2. Sąraše pasirinkite norimą užduotį ir peržiūrėkite jos duomenis.
   2.1. Jeigu norite užpildyti užduoties rezultatus pasirinkite [Form], užpildykite duomenis ir spauskite mygtuką [Complete].
   2.2. Jeigu norite peržiūrėti užduoties istoriją spauskite [History].
   2.3. Jeigu norite peržiūrėti proceso diagramą, spauskite mygtuką [Diagram].
   2.4. Jeigu norite peržiūrėti užduoties aprašymą spauskite [Description].

Rezultatas

Užduoties duomenys peržiūrėti.

Valdyti sistemos naudotojų grupių duomenis

Dirbti su sistemos grupių duomenimis galima keičiant, trinant, kuriant naujas grupes.



97 pav. Sistemos grupių sąrašo langas

Jei norite keisti grupių duomenis, atlikite šiuos veiksmus:

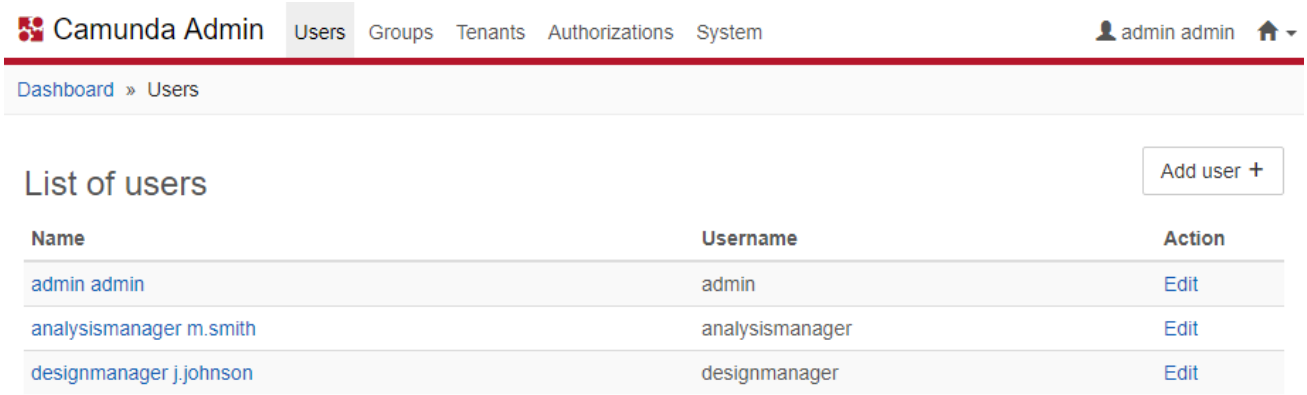
1. Administratoriaus posistemės pagrindiniame lange pasirinkite [Groups].
2. Sąraše pasirinkite norimą grupę ir peržiūrėkite jos duomenis.
   2.1. Jeigu norite redaguoti esamos grupės duomenis, spauskite sąraše prie norimo įrašo [Edit], pakeiskite norimą informaciją ir spauskite [Update Group].
   2.2. Jeigu norite pašalinti esamos grupės duomenis, spauskite sąraše prie norimo įrašo [Edit], o atsidariusiame lange spauskite [Delete Group].

Rezultatas

Sistemos grupių duomenys pertvarkyti.

Valdyti sistemos naudotojų duomenis

Dirbti su sistemos naudotojų duomenimis galima keičiant, trinant, kuriant naujus naudotojus.



98 pav. Sistemos naudotojų sąrašo langas

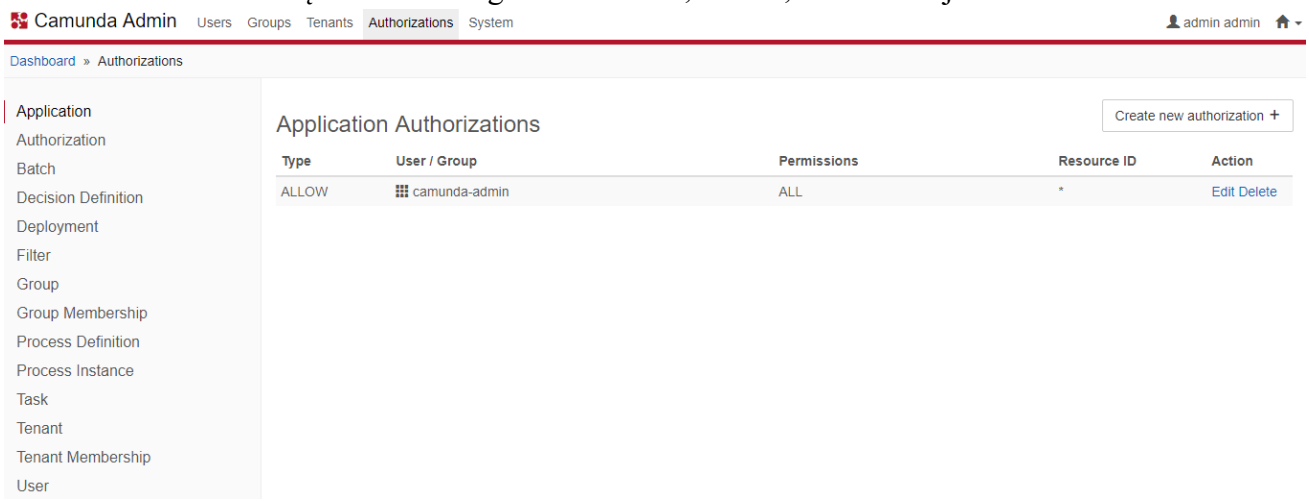
Jei norite keisti naudotojų duomenis, atlikite šiuos veiksmus:

1. Administratoriaus posistemės pagrindiniame lange pasirinkite [Users].
2. Sąraše pasirinkite norimą naudotoją ir peržiūrėkite jo duomenis.
   2.1. Jeigu norite redaguoti esamo naudotojo duomenis, spauskite sąraše prie norimo įrašo [Edit], pakeiskite norimą informaciją ir spauskite [Update Profile].
   2.2. Jeigu norite pašalinti esamo naudotojo duomenis, spauskite sąraše prie norimo įrašo [Edit], o atsidariusiame lange spauskite [Accaunt] ir [Delete User].

Rezultatas
Sistemos naudotojų duomenys pertvarkyti.

Valdyti sistemos teises

Dirbti su sistemos teisių duomenimis galima keičiant, trinant, kuriant naujas teises.



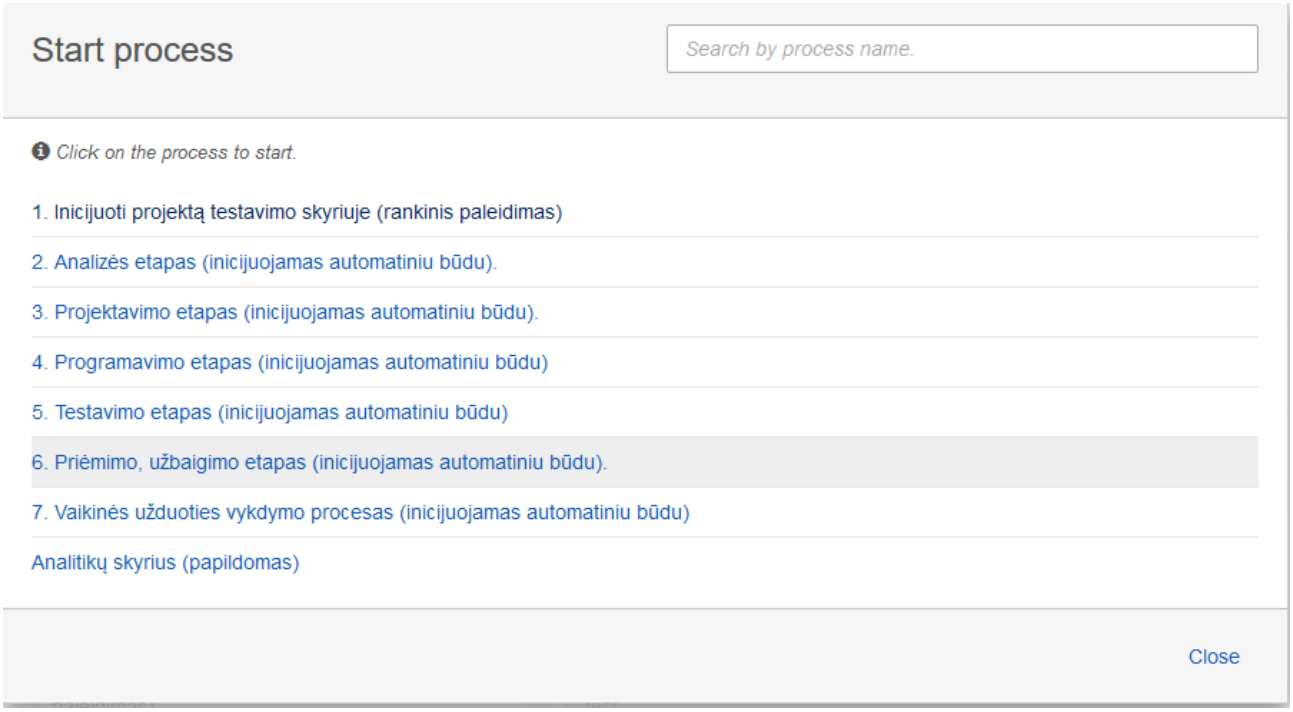
99 pav. Sistemos teisių sąrašo langas

Jei norite keisti teisių duomenis, atlikite šiuos veiksmus:
1. Administratoriaus posistemės pagrindiniame lange pasirinkite [Authorizations].
2. Sąraše pasirinkite norimą funkcionalumą ir suteikite prie jo prieigą naudotojui/grupei.

Rezultatas
Sistemos naudotojų teisės pertvarkytos.

Inicijuoti projekto etapo pradžią

Pradėti testuotojų komandos darbą galima inicijuojant projektą testuotojų skyriuje.



100 pav. Sistemos projekto inicijavimo langas

Jei norite inicijuoti projektą testuotojų skyriuje, atlikite šiuos veiksmus:

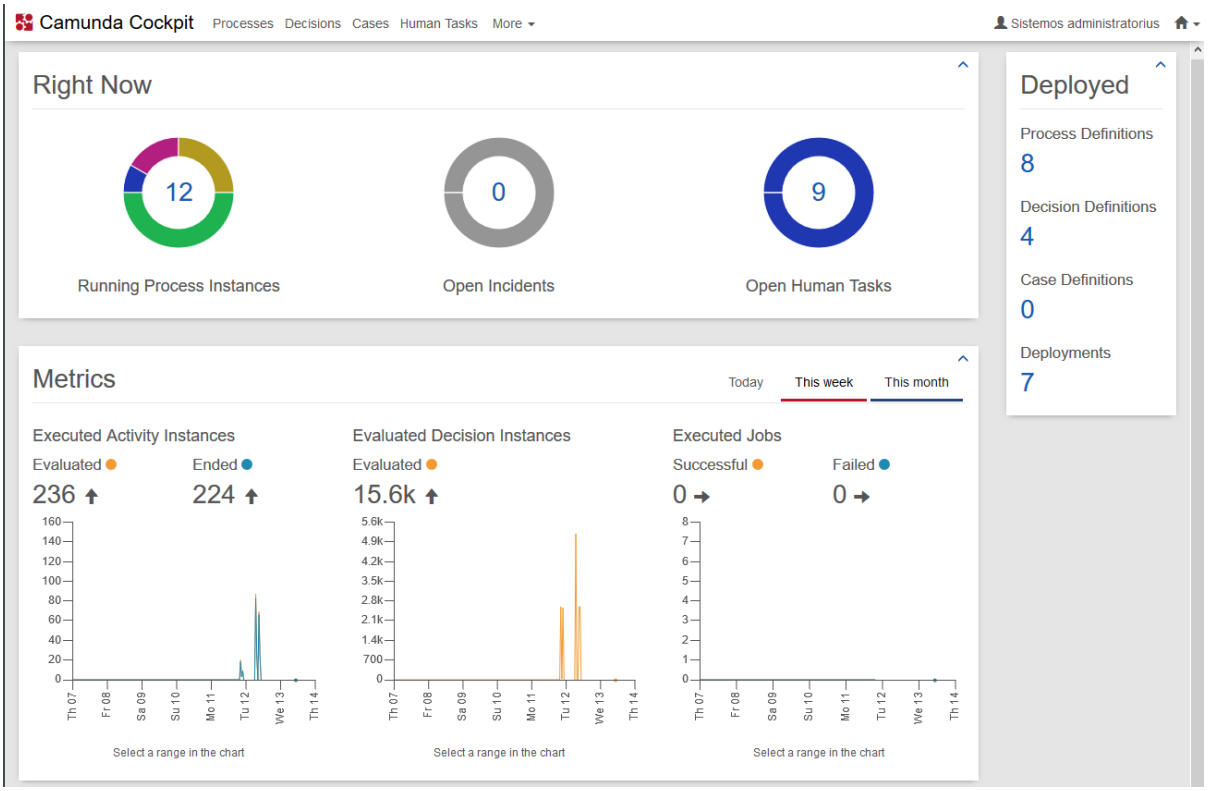
1. Užduočių sąrašo viršutiniame meniu pasirinkite [Start process].
2. Pasirinkite rankiniu būdu paleidžiamą procesą „Inicijuoti projektą testavimo skyriuje (rankinis paleidimas)“.
3. Įkelkite projekto vykdymui reikalingą dokumentą, įveskite projekto pavadinimą, pasirinkite programavimo kalbą, metodiką.
4. Spauskite mygtuką [Start].

Rezultatas

Projekto etapas inicijuotas.

Stebėti vykdomą projektą realiu laiku

Atlikti vykdomo projekto stebėjimą galima stebėjimo posistemėje.



101 pav. Vykdomo projekto stebėjimo langas

Jei norite projekto etapą, atlikite šiuos veiksmus:

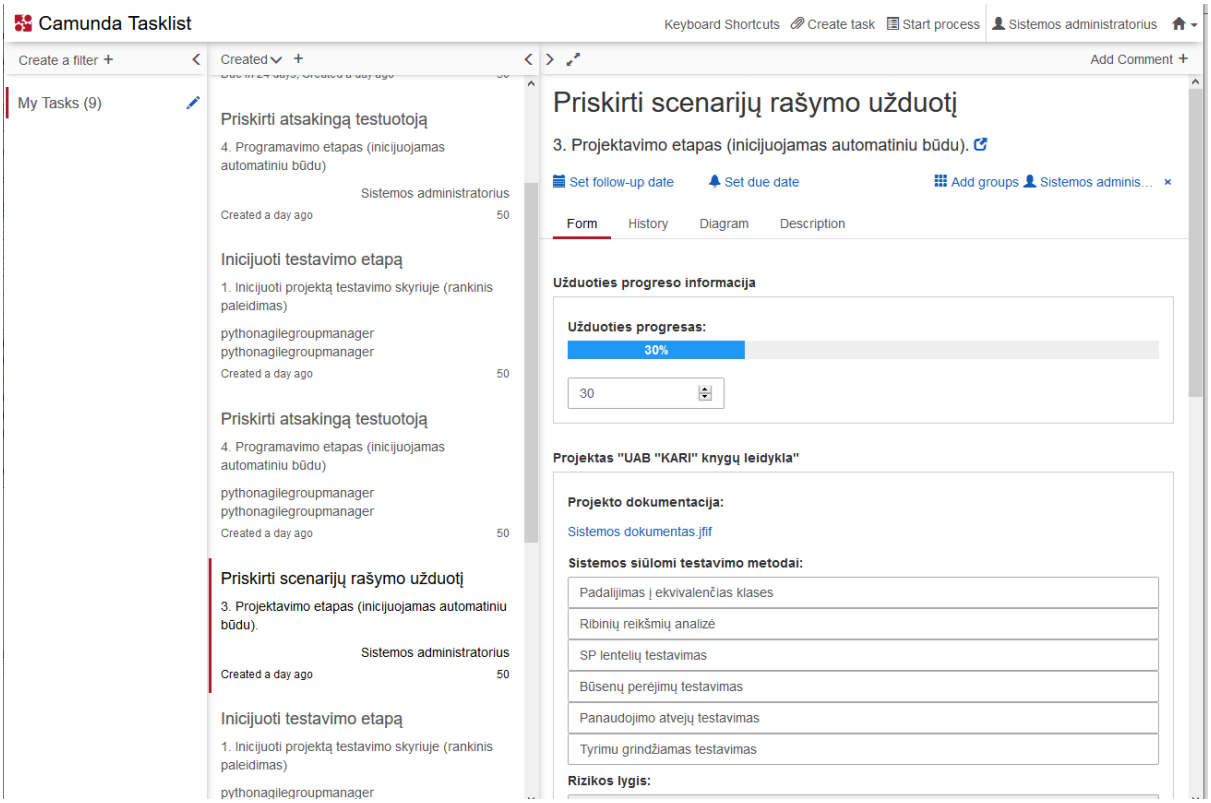
1. Prisijunkite prie posistemė „Cockpit“.
2. Pagrindiniame puslapyje galite atlikti norimus veiksmus:
   2.1. Jeigu norite peržiūrėti konkretų vykdomą procesą, pasirinkite meniu „Processes“ ir išsirinkite norimą procesą.
   2.2. Jeigu norite peržiūrėti naudojamas DMN lenteles ir jų duomenis pasirinkite „Decisions“.
   2.3. Jeigu norite generuoti ataskaitas, peržiūrėti užduočių statistiką spauskite „Human Tasks“.

Rezultatas

Projekto vykdymas peržiūrėtas.

Sukurti (priskirti) užduotį

Inicijuoti komandos narių darbą galima sukuriant jiems užduotis.



102 pav. Užduoties kūrimo langas

Jei norite sukurti užduotį, atlikite šiuos veiksmus:

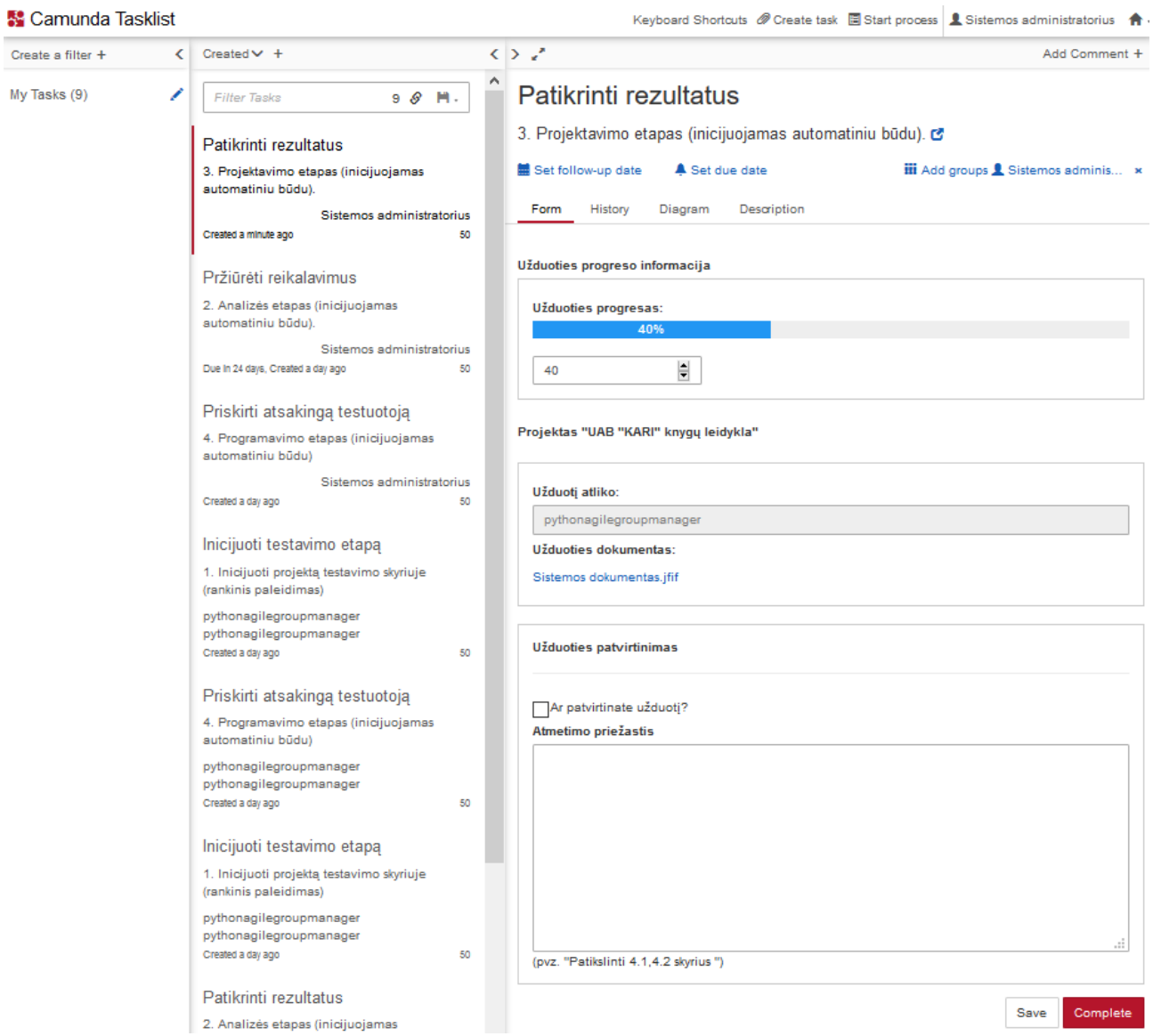
1. Užduočių sąrašo posistemėje atidarykite „My Tasks“.
2. Sąraše pasirinkite užduotį „Priskirti <> užduotį“.
3. Užpildykite užduoties duomenis.
4. Priskirkite užduotį darbuotojui.
5. Spauskite mygtuką [Complete].

Rezultatas

Užduotis sukurta ir perduota atsakingam komandos nariui.

Tvirtinti užduoties rezultatus

Užbaigti užduoties vykdymą galima patvirtinant gautus užduoties rezultatus.



103 pav. Užduoties tvirtinimo langas

Jei norite patvirtinti užduotį, atlikite šiuos veiksmus:

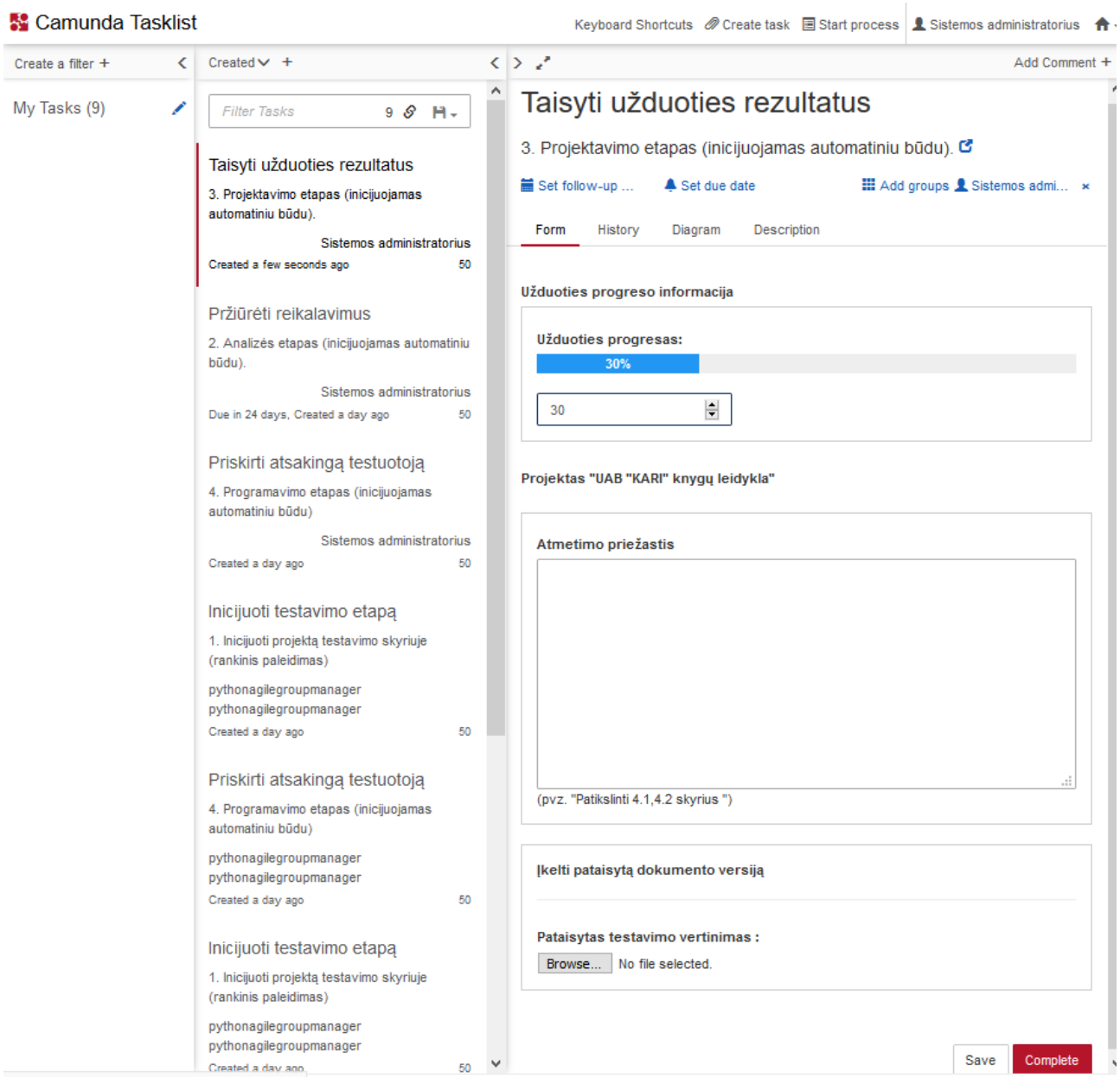
1. Užduočių sąrašo posistemėje atidarykite „My Tasks“.
2. Sąraše pasirinkite užduotį „Patikrinti rezultatus“.
3. Peržiūrėkite gautus užduoties rezultatus.
4. Pažymėkite varnelę „Ar patvirtinate užduotį?“
5. Spauskite mygtuką [Complete].

Rezultatas

Užduoties rezultatai patvirtinti.

Taisyti užduoties rezultatus

Jeigu užduoties rezultatai buvo atmesti, juos galima ištaisyti.



104 pav. Užduoties rezultatų taisymo langas

Jei norite patvirtinti užduotį, atlikite šiuos veiksmus:

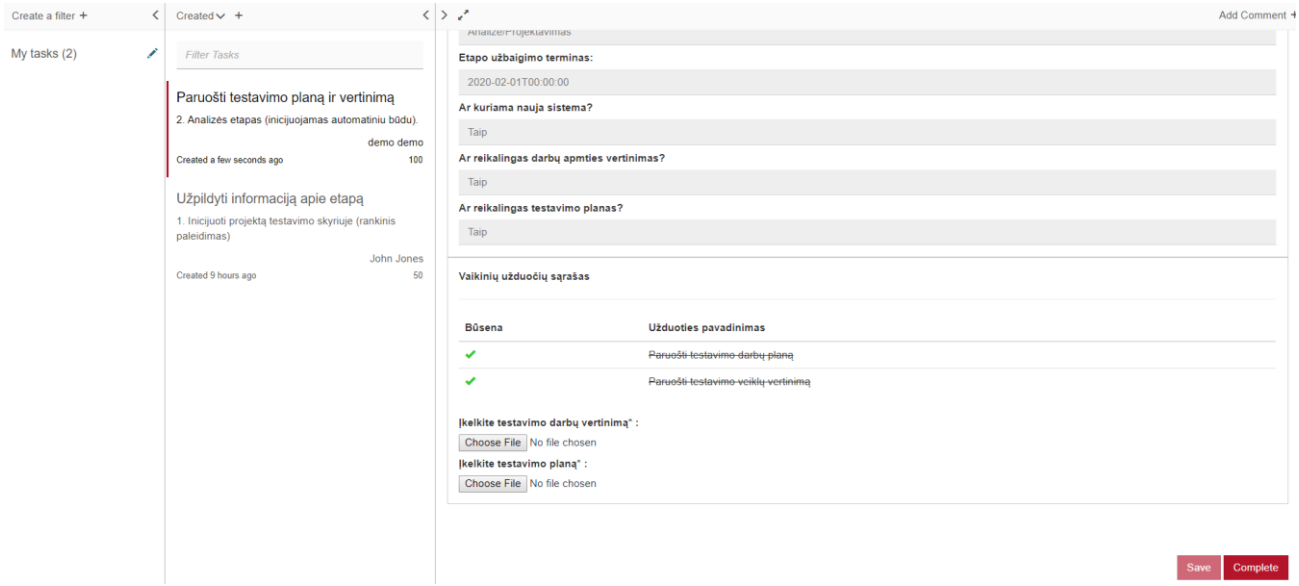
1. Užduočių sąrašo posistemėje atidarykite „My Tasks“.
2. Sąraše pasirinkite užduotį „Taisyti užduoties rezultatus“.
3. Peržiūrėkite gautą pastabą.
4. Įkelkite naują užduoties rezultatų failą.
5. Spauskite mygtuką [Complete].

Rezultatas

Užduoties rezultatai pataisyti.

Įvykdyti užduotį, kuri turi sub-užduočių

Kai užduotis turi sub-užduočių reikia atlikti pirma sub-užduočių įvykdymą, vėliau pagrindinės užduoties.



105 pav. Sub-užduočių vykdymo langas

Jei norite įvykdyti sudėtinę užduotį, atlikite šiuos veiksmus:

1. Užduočių sąrašo posistemėje atidarykite „My Tasks“.
2. Sąraše pasirinkite aukščiausio lygmens užduotį.
3. Sąraše „Vaikinių užduočių sąrašas“ po vieną pasirinkite užduotį ir ją įvykdykite.
4. Įvykdžius visas sub-užduotis, užpildykite pagrindinės užduoties rezultatus.
5. Spauskite mygtuką [Complete].

Rezultatas

Sudėtinė užduotis įvykdyta.

7 priedas. Apklausos anketa

Programinės įrangos testavimo metodikos eksperimentinės sistemos vertinimo apklausa (testuotojams)

Apklausos tikslas

Apklausa skirta testuotojams siekiant įvertinti, ar sukurta programinės įrangos testavimo metodika ir eksperimentinė sistema gali būti pritaikoma IT įmonėje. Taip pat siekiama surinkti kuo daugiau informacijos apie pastebėtus privalumus bei trūkumus, dalyvaujant eksperimentinės sistemos bandomojoje eksploatacijoje.

1. Ar manote, kad eksperimentinė sistema palengvino Jūsų kasdienius darbus?

☐ Taip | ☐ Ne

1.1. Jeigu taip, įvardinkite konkrečius sistemos teikiamus privalumus:

2. Ar dirbant su sistema pastebėjote trūkumų? Jeigu taip, įvardinkite kokių:

3. Ar buvo sudėtinga naudotis sistema? Kaip vertinate sudėtingumą:

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Labai sudėtinga — Iškilo keletas klausimų — Nesudėtinga

4. Ar naudojantis sistema pastebėjote klaidų (sisteminių/loginių)?

☐ Taip | ☐ Ne

4.1. Jeigu taip, įvardinkite nekorektiškai veikiančias vietas:

5. Eksperimentinė sistema pateikia siūlomus testavimo metodus. Kaip dažnai jais naudojotės?

☐ Niekada ☐ Kartais ☐ Visada

5.1. Jeigu niekada nesinaudojote arba naudojotės tik kartais, įvardinkite priežastis:

6. Ar pasikeitė reikiamų atlikti užduočių skaičius?

☐ Sumažėjo ☐ Išliko toks pat ☐ Padidėjo

7. Kaip manote, ar komunikacija tarp komandos narių pasikeitė? Jeigu manote, kad pasikeitė, pagrįskite atsakymą:

8. Kiek laiko dirbate testavimo srityje?

☐ Iki 3mėn. ☐ Iki 6 mėn. ☐ Iki 1m. ☐ Iki 2m. ☐ Daugiau nei 2m.

Programinės įrangos testavimo metodikos eksperimentinės sistemos vertinimo apklausa (testuotojų grupės vadovui)

Apklausos tikslas

Apklausa skirta testuotojų grupės vadovui siekiant įvertinti, ar sukurta programinės įrangos testavimo metodika ir eksperimentinė sistema gali būti pritaikoma IT įmonėje. Taip pat siekiama surinkti kuo daugiau informacijos apie pastebėtus privalumus bei trūkumus, dalyvaujant eksperimentinės sistemos bandomojoje eksploatacijoje.

1. Ar manote, kad eksperimentinė sistema palengvino Jūsų kasdienius darbus?

☐ Taip | ☐ Ne

1.1. Jeigu taip, įvardinkite konkrečius sistemos teikiamus privalumus:

2. Ar dirbant su sistema pastebėjote trūkumų? Jeigu taip, įvardinkite kokių:

3. Ar buvo sudėtinga naudotis sistema? Kaip vertinate sudėtingumą:

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Labai sudėtinga — Iškilo keletas klausimų — Nesudėtinga

4. Ar naudojantis sistema pastebėjote klaidų (sisteminių/loginių)?

☐ Taip | ☐ Ne

4.1. Jeigu taip, įvardinkite nekorektiškai veikiančias vietas:

5. Eksperimentinė sistema pateikia siūlomas testavimo veiklas. Kaip dažnai jomis naudojotės?

☐ Niekada ☐ Kartais ☐ Visada

5.1. Jeigu niekada nesinaudojote arba naudojotės tik kartais, įvardinkite priežastis:

6. Ar pasikeitė reikiamų atlikti užduočių skaičius?

☐ Sumažėjo ☐ Išliko toks pat ☐ Padidėjo

7. Kaip manote, ar komunikacija tarp komandos narių pasikeitė? Jeigu manote, kad pasikeitė, pagrįskite atsakymą:

8. Eksperimentinė sistema suteikia galimybę realiu laiku stebėti vykdomus projektus bei veiklas. Kaip dažnai naudojotės funkcionalumu?

☑ Niekada ☐ Kartais ☐ Visada

8.1. Jeigu niekada nesinaudojote arba naudojotės tik kartais, įvardinkite priežastis:

9. Kaip manote, ar pasikeitė naujų darbuotojų mokymo procesas?

☐ Sulėtėjo ☐ Išliko toks pat ☐ Pagreitėjo

9.1. Jeigu manote, kad pasikeitė, įvardinkite galimas priežastis:

10. Kaip manote, ar skyriuje pagerėjo testavimo proceso kokybė?

☐ Pablogėjo ☐ Išliko tokia pati ☐ Pagerėjo

10.1. Jeigu manote, kad pasikeitė, įvardinkite galimas priežastis:

11. Kiek laiko dirbate testavimo srityje?

☐ Iki 3mėn. ☐ Iki 6 mėn. ☐ Iki 1m. ☐ Iki 2m. ☐ Daugiau nei 2m.

8 priedas. Testavimo proceso įvertinimo klausimynas

		Iki šiol		Pritaikius metodiką	
	Testavimo aplinkos planavimo vertinimo kriterijai	**Taip**	**Ne**	**Taip**	**Ne**
1.	Ar įmonėje yra apibrėžta testavimo politika?		+		+
2.	Ar įmonėje yra apibrėžta testavimo strategija?		+	+	
3.	Ar įmonėje yra apibrėžtas testavimo procesas, kuris būtų grindžiamas įmonės testavimo strategija?		+	+	
4.	Ar įmonėje testavimo procesas apima tiek statinį, tiek dinaminį testavimą?		+	+	
5.	Ar testavimo strategijoje yra apibrėžtos rolės, kurias testavimas gali apimti ir įvardinama, kokios rolės iš visų apibrėžtų yra naudojamos būtent Jūsų įmonėje? Pvz., išbandyti sistemos naudotojų poreikius nesinaudojant turima specifikacija.		+	+	
	Suma:	0		4	

		Iki šiol		Pritaikius metodiką	
	Testavimo proceso valdymo vertinimo kriterijai	**Taip**	**Ne**	**Taip**	**Ne**
1.	Ar vadovai skiria reikiamus išteklius (įskaitant laiką) mokymui, planavimui, užduočių vykdymui, bei rezultatų vertinimui?	+	+	+	
2.	Ar testuotojai yra įtraukiami į projekta nuo projekto inicijavimo etapo siekiant, kad testavimas būtų vykdomas nenutraukiamai viso projekto vykdymo metu?		+	+	
3.	Ar vadovai skiria tiek resursų, kiek reikalauja sistemos kūrimo procesas?	+		+	
4.	Ar vadovai įsitraukia į testavimo planavimą ir vykdymą tiek pat aktyviai kaip ir į kitas projekto vykdymo veiklas?		+	+	
5.	Ar vadovai pakankamai išmano testavimo teoriją, procesus ir įrankius, kad galėtų efektyviai valdyti testavimo planavimą, vykdymą ir galėtų pakeisti testavimo rezultatus?	+		+	
	Suma:	3		5	

		Iki šiol		Pritaikius metodiką	
	Testavimo proceso naudojimo įmonėje vertinimo kriterijai	**Taip**	**Ne**	**Taip**	**Ne**
1.	Ar testuotojai dirba pagal iš anksto numatytą testavimo procesą, ruošia testavimo duomenis, vykdo testus ir aprašo gautus rezultatus?		+	+	
2.	Ar testuotojai gali vienareikšmiškai suprasti aprašytą testavimo procesą ir jį vykdyti pagal numatytas taisykles?		+	+	
3.	Ar apibrėžtas testavimo procesas padengia visas kokybiškam testavimui reikiamas veiklas?		+	+	
4.	Ar buvo paruoštas planas, kaip pagerinti testavimo procesą, kad testavimas taptų efektyvesnis ir būtų vykdomas laiku?		+	+	
5.	Ar testavimo procesą kuria testuotojai/proceso dalyviai?	+		+	
	Suma:	1		5	

		Iki šiol		Pritaikius metodiką	

	Naudojamų testavimo įrankių vertinimo kriterijai	Taip	Ne	Taip	Ne
1.	Ar testuotojai naudoja automatinius įrankius testavimo duomenų generavimui?	+		+	
2.	Ar testavimo įrankiai pasirenkami remiantis logine maniera?	+		+	
3.	Ar testuotojai naudoja įrankius tik po to, kai yra pakankamai išmokinti naudotis atitinkamais įrankiais?		+		+
4.	Ar testavimo įrankia aprašyti testavimo plane?		+	+	
5.	Ar yra numatytas procesas, kuris apibrėžia, kaip naudotis testavimo įrankiu pateikiant detalias instrukcijas?		+		+
	Suma:	2		3	
		Iki šiol		**Pritaikius metodiką**	
	Testavimo mokymų vertinimo kriterijai	**Taip**	**Ne**	**Taip**	**Ne**
1.	Ar yra numatytas testuotojų kvalifikacijos kėlimo planas, kuris apima visus etapas nuo pradedančio testuotojo iki patyrusio testuotojo pozicijos?	+		+	
2.	Ar testuotojai yra pakankamai mokinami prieš pradedant vykdyti testavimo procesą?	+		+	
3.	Ar testuotojai mokinami testavimo teorijos apie galimų rizikų analizę ir įvairius testavimo tipus, bei testuotojai supranta, kodėl reikia atlikti konkrečoas testavimo veiklas?	+		+	
4.	Ar testuotojai mokinami apie tai, kokią įtaką naudotojų pasitenkinimui daro skirtingi testavimo metodai ir kaip reikėtų interpretuoti gautus rezultatus?	+		+	
5.	Ar testuotojai mokinami, kaip išmatuoti sistemos efektyvumą ir testuotojai naudoja gautus duomenis siekiant pagerinti testavimo procesą?	+		+	
	Suma:	5		5	
		Iki šiol		**Pritaikius metodiką**	
	Naudotojų pasitenkinimo vertinimo kriterijai	**Taip**	**Ne**	**Taip**	**Ne**
1.	Ar galutiniai sistemos naudotojai gauna pakankamai informacijos, kaip galima sekti testavimo progresą ir įvertinti testavimo rezultatus?		+		+
2.	Ar vykdomos naudotojų apklausos siekiant nustatyti naudotojų pasitenkinimo lygį šiose srityse: testavimo planavimas, vykdymas, rezultatų pateikimas, komunikacija ir kt.?	+		+	
3.	Ar naudotojai dalyvauja testavime, kurio metu nustatoma, ar sitemą galima priimti?	+		+	
4.	Ar vartotojams pateikiamas testavimo planas ir ar jie pateikia patvirtinimą, jog sutinka, jog jeigu procesas vyks pagal planą, tuomet testavimas patenkins visus reikalavimus?	+		+	
5.	Ar naudotojo pagrindinės veiklos (duomenų įvedimas, išvedimas, naudotojų vadovai irk t.) yra numatytos testavimo procese?	+		+	
	Suma:	4		4	
		Iki šiol		**Pritaikius metodiką**	
	Testavimo įverčių vertinimo kriterijai	**Taip**	**Ne**	**Taip**	**Ne**
1.	Ar įmonėje apibrėžti testavimo vertinimo kriterijai ir jie yra naudojami įvertinti testavimo proceso efektyvumą?	+		+	

2.	Ar yra įdiegtas procesas, skirtas išmatuoti testavimo efektyvumą?	+	+
3.	Ar vertinamas biudžeto ir grafiko laikymąsis?	+	+
4.	Ar vertinamas naudojamų įrankių ir automatinių testų teikiama nauda?	+	+
5.	Ar procentaliai įvertinama, kiek klaidų buvo pašalinta lyginant su visomis rastomis klaidomis, kurios buvo priskirtos PĮ kūrimo procesui?	+	+
	Suma:	3	3

		Iki šiol		**Pritaikius metodiką**	
	Testavimo kokybės valdymo vertinimo kriterijai	**Taip**	**Ne**	**Taip**	**Ne**
1.	Ar testuotojų padarytos klaidos testavimo metu yra fiksuojamos?	+		+	
2.	Ar testavimo planas yra tikrinamas ir po patvirtinimo naudojamas kaip vidinis skyriaus standartas?		+	+	
3.	Ar testavimo planas įtraukia procedūras, kuriomis remiantis būtų galima teigti, jog testavimo procesas įvykdytas pagal planą?		+		+
4.	Ar reguliariai ruošiams ataskaitos, atspindinčios testuotojų vykdomų veiklų statusus?	+		+	
5.	Ar kokybės kontrolės ataskaitos yra periodiškai ruošiamos ir aptariamos, siekiant įvertinti testavimo proceso efektyvumą visos įmonįs mastu?	+		+	
	Suma:	3		4	

9 priedas. Projekto vykdymo metu atliekamų veiksmų skaičiaus tyrimo rezultatai

Projekto inicijavimo etapas	**Istoriniai duomenys**	**Pritaikius metodiką ir demonstracinę sistemą**
Inicijuoti projektą	1. Projekto vadovas „OpenProject“ sistemoje sukuria naują projektą. 2. Projekto vadovas parašo laišką testuotojų grupės vadovui.	1. Projekto vadovas užpildo projekto inicijavimo formą eksperimentinėje sistemoje (sistema automatiškai sukuria priskiria komandą, sukuria užduotį testuotojų vadovui).
Priskirti testuotojų komandą	1. Testuotojų vadovas peržiūri, kuri komanda dirba su projekte naudojamomis technologijomis ir metodika. 2. Testuotojų vadovas „OpenProject“ sistemoje priskiria atsakingą testuotoją prie projekto, prisega projekto dokumentaciją.	- (sistema automatiškai priskiria komandą pagal sprendimo priėmimo lentelėje suvestus duomenis, perduoda duomenis atsakingam testuotojui).
Projekto etapo inicijavimas testuotojų grupėje	1. Perskaityti gautą dokumentaciją, nustatyti, koks etapas turės būti vykdomas. 2. „OpenProject“ sistemoje sukurti skiltį etapo užduotims. 3. Sukurtoje etapo skiltyje užpildyti duomenis apie projekto etapą ir pateikti nurodymus.	1. Perskaityti dokumentaciją. 2. Užpildyti formą, kuri skirta surinkti pagrindinę informaciją apie projektą. 3. Užpildyti formą, kurioje reikia nurodyti inicijuojamą etapą.
Suma (atliktų veiksmų):	**7**	**4**

Analizės etapas	**Istoriniai duomenys**	**Pritaikius metodiką ir demonstracinę sistemą**
Sukurti ir priskirti testuotojams užduotis	1. Sukurti testavimo scenarijų rašymo užduotį. 2. Priskirti testavimo scenarijų užduotį.	1. Priskirti testavimo scenarijų pildymo užduotį (užpildyti testuotojo

		priskyrimo formą eksperimentinėje sistemoje).
Pildyti progresą	1. Įvesti procentus, kurie atspindėtų progresą.	1. Įvesti procentus, kurie atspindėtų progresą.
Pildyti prie užduoties praleistą laiką	1. Susirasti užduotį. 2. Atidaryti laiko pildymo langą. 3. Parašyti komentarą ir praleistas valandas. **Pastaba**: atliekama kasdien! Tikslus skaičius priklauso nuo projekto dydžio	- (sistema automatiškai fiksuoja užduoties vykdymo laiką ir jį atvaizduoja stebėjimo posistemėje).
Rezultatų tikrinimo užduotis	1. Įkelti failą, pakeisti užduoties būseną.	1. Užpildyti rezultatų pateikimo formą.
Patikrinti užduoties rezultatus	1. Susikurti tikrinimo užduotį. 2. Užpildyti praleistą laiką.	- (užduotis sukuriama automatiškai, praleistas laikas fiksuojamas automatiškai).
Pateikti atsakymą dėl užduoties rezultatų	1. Parašyti pastabas, prisegti dokumentą.	1. Užpildyti atsakymo pateikimo formą (pastabos, dokumentas).
Suma (atliktų veiksmų):	**10**	**4**
Projektavimo etapas	**Istoriniai duomenys**	**Pritaikius metodiką ir demonstracinę sistemą**
Sukurti ir priskirti testuotojams užduotis	1. Sukurti testavimo scenarijų rašymo užduotį. 2. Priskirti testavimo scenarijų užduotį.	1. Priskirti testavimo scenarijų pildymo užduotį (užpildyti testuotojo priskyrimo formą eksperimentinėje sistemoje).
Pildyti progresą	1. Įvesti procentus, kurie atspindėtų progresą.	1. Įvesti procentus, kurie atspindėtų progresą.
Pildyti prie užduoties praleistą laiką	1. Susirasti užduotį. 2. Atidaryti laiko pildymo langą. 3. Parašyti komentarą ir praleistas valandas. **Pastaba**: atliekama kasdien! Tikslus skaičius priklauso nuo projekto dydžio	- (sistema automatiškai fiksuoja užduoties vykdymo laiką ir jį atvaizduoja stebėjimo posistemėje).
Pateikti užduoties rezultatus	1. Įkelti failą, pakeisti užduoties būseną.	1. Užpildyti rezultatų pateikimo formą.
Rezultatų tikrinimo užduotis	1. Susikurti tikrinimo užduotį. 2. Užpildyti praleistą laiką.	- (užduotis sukuriama automatiškai, praleistas laikas fiksuojamas automatiškai).
Pateikti atsakymą dėl užduoties rezultatų	1. Parašyti pastabas, prisegti dokumentą.	1. Užpildyti atsakymo pateikimo formą (pastabos, dokumentas).
Suma (atliktų veiksmų):	**10**	**4**
Programavimo etapas	**Istoriniai duomenys**	**Pritaikius metodiką ir demonstracinę sistemą**
Sukurti ir priskirti testuotojams užduotis	1. Sukurti testavimo scenarijų rašymo užduotį. 2. Priskirti testavimo scenarijų užduotį. 3. Sukurti statinės kodo analizės užduotį. 4. Priskirti statinės kodo analizės užduotį. 5. Sukurti testavimo užduotį. 6. Priskirti testavimo užduotį.	1. Priskirti atsakingą etapo testuotoją (užpildyti testuotojo priskyrimo formą eksperimentinėje sistemoje). Visos reikiamos užduotys sukuriamos automatiškai pagal iš anksto numatytą procesą.
Pildyti progresą	1. Įvesti procentus, kurie atspindėtų progresą.	1. Įvesti procentus, kurie atspindėtų progresą.

Pildyti prie užduoties praleistą laiką	1. Susirasti užduotį. 2. Atidaryti laiko pildymo langą. 3. Parašyti komentarą ir praleistas valandas. **Pastaba**: atliekama kasdien! Tikslus skaičius priklauso nuo projekto dydžio	- (sistema automatiškai fiksuoja užduoties vykdymo laiką ir jį atvaizduoja stebėjimo posistemėje).
Pateikti užduoties rezultatus	1. Įkelti failą, pakeisti užduoties būseną.	1. Užpildyti rezultatų pateikimo formą.
Klaidų registravimas	1. Susikurti klaidos užduotį. 2. Užpildyti užduoties informaciją (aprašymas, programuotojas ir kt.).	1. Užpildyti informaciją apie klaidas, programuotojo el. paštą.
Pateikti atsakymą dėl užduoties rezultatų	1. Parašyti pastabas, prisegti dokumentą.	1. Užpildyti atsakymo pateikimo formą (pastabos, dokumentas).
Rezultatų tikrinimo užduotis	1. Susikurti tikrinimo užduotį. 2. Užpildyti praleistą laiką.	- (užduotis sukuriama automatiškai, praleistas laikas fiksuojamas automatiškai).
Suma (atliktų veiksmų):	**16**	**5**

Testavimo etapas	**Istoriniai duomenys**	**Pritaikius metodiką ir demonstracinę sistemą**
Sukurti ir priskirti testuotojams užduotis	1. Sukurti testavimo užduotį. 2. Priskirti testavimo užduotį. 3. Sukurti regresinio testavimo užduotį. 4. Priskirti regresinio testavimo užduotį.	1. Priskirti testavimo scenarijų pildymo užduotį (užpildyti testuotojo priskyrimo formą eksperimentinėje sistemoje).
Pildyti progresą	1. Įvesti procentus, kurie atspindėtų progresą.	1. Įvesti procentus, kurie atspindėtų progresą.
Pildyti prie užduoties praleistą laiką	1. Susirasti užduotį. 2. Atidaryti laiko pildymo langą. 3. Parašyti komentarą ir praleistas valandas. **Pastaba**: atliekama kasdien! Tikslus skaičius priklauso nuo projekto dydžio	- (sistema automatiškai fiksuoja užduoties vykdymo laiką ir jį atvaizduoja stebėjimo posistemėje).
Pateikti užduoties rezultatus	1. Įkelti failą, pakeisti užduoties būseną.	1. Užpildyti rezultatų pateikimo formą.
Klaidų registravimas	1. Susikurti klaidos užduotį. 2. Užpildyti užduoties informaciją (aprašymas, programuotojas ir kt.).	2. Užpildyti informaciją apie klaidas, programuotojo el. paštą.
Suma (atliktų veiksmų):	**11**	**4**

Priėmimo etapas	**Istoriniai duomenys**	**Pritaikius metodiką ir demonstracinę sistemą**
Sukurti ir priskirti testuotojams užduotis	1. Sukurti nefunkcinio testavimo užduotį. 2. Priskirti testavimo užduotį. 3. Sukurti testavimo proceso vertinimo užduotį. 4. Priskirti testavimo vertinimo užduotį.	1. Priskirti atsakingą etapo testuotoją (užpildyti testuotojo priskyrimo formą eksperimentinėje sistemoje).
Pildyti progresą	1. Įvesti procentus, kurie atspindėtų progresą.	1. Įvesti procentus, kurie atspindėtų progresą.
Pildyti prie užduoties praleistą laiką	1. Susirasti užduotį. 2. Atidaryti laiko pildymo langą. 3. Parašyti komentarą ir praleistas valandas.	- (sistema automatiškai fiksuoja užduoties vykdymo laiką ir jį atvaizduoja stebėjimo posistemėje).

	Pastaba: atliekama kasdien! Tikslus skaičius priklauso nuo projekto dydžio	
Pateikti užduoties rezultatus	1. Įkelti failą, pakeisti užduoties būseną.	1. Užpildyti rezultatų pateikimo formą.
Klaidų registravimas	1. Susikurti klaidos užduotį. 2. Užpildyti užduoties informaciją (aprašymas, programuotojas ir kt.).	1. Užpildyti informaciją apie klaidas, programuotojo el. paštą.
Pateikti atsakymą dėl užduoties rezultatų	1. Parašyti pastabas, prisegti dokumentą.	1. Užpildyti atsakymo pateikimo formą (pastabos, dokumentas).
Pateikti sistemos atidavimo klientui informaciją	1. Parašyti el. laišką	1. Užpildyti sistemoje perdavimo informacijos formą
Suma (atliktų veiksmų):	**13**	**6**

10 priedas. Sukurtų realizacijos failų aprašymas

Pavadinimas	Paskirtis/tipas	Aprašymas
assigntester.html	HTML forma	Forma naudojama proceso „Analizės etapas (inicijuojamas automatiniu būdu)“ veikoje „Priskirti testuotoją“. Šioje formoje užpildomi testuotojo duomenys, kuriam priskiriama užduotis.
assigntester_requirements.html	HTML forma	Forma naudojama proceso „Analizės etapas (inicijuojamas automatiniu būdu)“ veikoje „Priskirti reikalavimų peržiūros užduotį“. Šioje formoje užpildomi testuotojo duomenys, kuriam priskiriama užduotis.
dofinalacceptancedata.html	HTML forma	Forma naudojama proceso „Priėmimo, užbaigimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Užpildyti informaciją apie sistemos perdavimą klientui“. Šioje formoje užpildomas faktas, jog sistema buvo perduota kliento testavimui.
dofinaldata.html	HTML forma	Forma naudojama proceso „Testavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Pateikti galutinę testavimo ataskaitą“. Šioje formoje pateikiama galutinė vidinio testavimo ataskaita.
dononfunctionaltesting.html	HTML forma	Forma naudojama proceso „Priėmimo, užbaigimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Užpildyti nefunkcinio testavimo rezultatus“. Šioje formoje pateikiami nefunkcinio testavimo rezultatai.
dotestcase.html	HTML forma	Forma naudojama proceso „Priėmimo, užbaigimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Analizės etapas (inicijuojamas automatiniu būdu)“. Šioje formoje pateikiami nefunkcinio testavimo rezultatai.
dotestcase_design.html	HTML forma	Forma naudojama proceso „Projektavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Užpildyti scenarijų rašymo užduoties rezultatus“. Šioje formoje pateikiami sudaryti testavimo scenarijai.
dotestcase_programming.html	HTML forma	Forma naudojama proceso „Programavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Užpildyti scenarijų rašymo užduoties rezultatus“. Šioje formoje pateikiami sudaryti testavimo scenarijai.
dotestcase_programming_data.html	HTML forma	Forma naudojama proceso „Programavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Užpildyti informaciją apie paruoštus testavimo duomenis“.

Pavadinimas	Paskirtis/tipas	Aprašymas
dotestcase_programming_results.html	HTML forma	Forma naudojama proceso „Programavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Užpildyti informaciją apie atliktą vienetų testavimą“.
dotestcase_static.html	HTML forma	Forma naudojama proceso „Programavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Užpildyti statinės kodo analizės rezultatus“.
dotestingprocess.html	HTML forma	Forma naudojama proceso „Testavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Užpildyti testavimo rezultatus“.
do_planning.html	HTML forma	Forma naudojama proceso „Analizės etapas (inicijuojamas automatiniu būdu)“ veikoje „Paruošti testavimo planą ir vertinimą“.
fillprojectinfo.html	HTML forma	Forma naudojama proceso „Inicijuoti projektą testavimo skyriuje (rankinis paleidimas)“ veikoje „Užpildyti informaciją apie projektą“.
nonfunctionaltestingprocess.html	HTML forma	Forma naudojama proceso „Priėmimo, užbaigimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Atlikti priėmimo etapo planavimą“.
review_requirements.html	HTML forma	Forma naudojama proceso „Analizės etapas (inicijuojamas automatiniu būdu)“ veikoje „Peržiūrėti reikalavimus“.
selectprojectphase.html	HTML forma	Forma naudojama proceso „Inicijuoti projektą testavimo skyriuje (rankinis paleidimas)“ veikoje „Inicijuoti testavimo etapą“.
start.html	HTML forma	Forma, naudojama visuose procesuose, kurie inicijuojami automatiniu būdu. Ši forma skirta pateikti klaidos pranešimą, kai procesą bandoma inicijuoti rankiniu būdu.
static_programming.html	HTML forma	Forma naudojama proceso „Programavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Priskirti atsakingą testuotoją“.
testcase.html	HTML forma	Forma naudojama proceso „Analizės etapas (inicijuojamas automatiniu būdu)“ veikoje „Patikrinti rezultatus“. veikoje „Priskirti scenarijų rašymo užduotį“.
testcase._design.html	HTML forma	Forma naudojama proceso „Projektavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Priskirti scenarijų rašymo užduotį“.
testcase_programming.html	HTML forma	Forma naudojama proceso „Programavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Priskirti testavimo užduotį“.
testingevaluation.html	HTML forma	Forma naudojama proceso „Priėmimo, užbaigimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Užpildyti atlikto testavimo vertinimo rezultatus“.
testingprocess.html	HTML forma	Forma naudojama proceso „Testavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Priskirti testavimo užduotį“.
assigntester.html	HTML forma	Forma naudojama proceso „Analizės etapas (inicijuojamas automatiniu būdu)“ veikoje „Priskirti testuotoją“.

Pavadinimas	Paskirtis/tipas	Aprašymas
assigntester_requirements.html	HTML forma	Forma naudojama proceso „Analizės etapas (inicijuojamas automatiniu būdu)“ veikoje „Priskirti reikalavimų peržiūros užduotį“.
dofinalacceptancedata.html	HTML forma	Forma naudojama proceso „Priėmimo, užbaigimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Užpildyti atlikto testavimo vertinimo rezultatus“.
dofinaldata.html	HTML forma	Forma naudojama proceso „Testavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Pateikti galutinę testavimo ataskaitą“.
approve-task-design.html	HTML forma	Forma naudojama proceso „Projektavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Patikrinti rezultatus“.
approve-task.html	HTML forma	Forma naudojama proceso „Analizės etapas (inicijuojamas automatiniu būdu)“ veikoje „Patikrinti rezultatus“.
approve-task2-design.html	HTML forma	Forma naudojama proceso „Projektavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Patikrinti pataisytus rezultatus“.
approve-task2.html	HTML forma	Forma naudojama proceso „Analizės etapas (inicijuojamas automatiniu būdu)“ veikoje „Patikrinti pataisytus rezultatus“.
correct-task-design.html	HTML forma	Forma naudojama proceso „Projektavimo etapas (inicijuojamas automatiniu būdu)“ veikoje „Taisyti užduoties rezultatus“.
correct-task.html	HTML forma	Forma naudojama proceso „Analizės etapas (inicijuojamas automatiniu būdu)“ veikoje „Taisyti užduoties rezultatus“.
sub-task.html	HTML forma	Forma naudojama, kai sukuriama vaikinė užduotis.
accetancephase.bpmn	Vykdomasis BPM procesas	Priėmimo etapo vykdomasis procesas.
analysisphase.bpmn	Vykdomasis BPM procesas	Analizės etapo vykdomasis procesas.
designphase.bpmn	Vykdomasis BPM procesas	Projektavimo etapo vykdomasis procesas.
programmingphase.bpmn	Vykdomasis BPM procesas	Programavimo etapo vykdomasis procesas.
softwaretesting_initiate.bpmn	Vykdomasis BPM procesas	Projekto inicijavimo etapo vykdomasis procesas.
subtask.bpmn	Vykdomasis BPM procesas	Vaikinių užduočių vykdymo vykdomasis procesas.
testingphase.bpmn	Vykdomasis BPM procesas	Testavimo etapo vykdomasis procesas.
assign-method.dmn	Sprendimo priėmimo lentelė	Juodosios dėžės metodų atrinkimo sprendimo priėmimo lentelė.
assign-method1.dmn	Sprendimo priėmimo lentelė	Baltosios dėžės metodų atrinkimo sprendimo priėmimo lentelė.
assign-method2.dmn	Sprendimo priėmimo lentelė	Praktika grindžiamų metodų atrinkimo sprendimo priėmimo lentelė.
assign-team.dmn	Sprendimo priėmimo lentelė	Testavimo komandos atrinkimo sprendimo priėmimo lentelė.

Pavadinimas	Paskirtis/tipas	Aprašymas
assignee.class	Java klasė	Klasė, skirta atrinkti visus ta tikros grupės naudotojus. Funkcionalumas naudojamas priskiriant užduotį.
cleardata.class	Java klasė	Klasė, skirta proceso kintamiesiems ištrinti, kuriems priskiriamos tam tikros reikšmės vykdomo proceso apimtyje.
CollectVariableValuesDelegate.class	Java klasė	Klasė, skirta išsaugoti proceso kintamųjų reikšmes.
CreateChildTaskListener.class	Java klasė	Klasė skirta vaikinės užduoties sukūrimui ir susiejimui su tėvine užduotimi.
CreateSubTasksListener.class	Java klasė	Klasė skirta vaikinės užduoties duomenims užpildyti iš JSON struktūros.
InitProcessVariables_createTask.class	Java klasė	Klasė, skirta sugeneruoti rekomenduojamų testavimo metodų sąrašą, kurį būtų galima naudoti formose.
SendBug.class	Java klasė	Klasė, skirta išsiųsti el. laišką apie rastas testavimo klaidas.
SendComments.class	Java klasė	Klasė, skirta išsiųsti el. laišką su pastebėtais neatitikimais reikalavimuose.
Sendmail.class	Java klasė	Klasė, skirta išsiųsti el. laišką naują priskirtą užduotį, kuri turi nurodytą atlikimo terminą.
SendMail_task.class	Java klasė	Klasė, skirta išsiųsti el. laišką apie naują priskirtą užduotį.
SendSignal_acceptance.class	Java klasė	Klasė, skirta inicijuoti priėmimo testavimo etapą ir perduoti visas vykdytam procese priskirtas kintamųjų reikšmes.
SendSignal_acceptanceends.class	Java klasė	Klasė, skirta inicijuoti priėmimo etapo pabaigą ir perduoti visas vykdytam procese priskirtas kintamųjų reikšmes.
SendSignal_analysis.class	Java klasė	Klasė, skirta inicijuoti analizės etapą ir perduoti visas vykdytam procese priskirtas kintamųjų reikšmes.
SendSignal_analysisends.class	Java klasė	Klasė, skirta inicijuoti analizės etapo pabaigą ir perduoti visas vykdytam procese priskirtas kintamųjų reikšmes.
SendSignal_design.class	Java klasė	Klasė, skirta inicijuoti projektavimo etapą ir perduoti visas vykdytam procese priskirtas kintamųjų reikšmes.
SendSignal_designends.class	Java klasė	Klasė, skirta inicijuoti projektavimo etapo pabaigą ir perduoti visas vykdytam procese priskirtas kintamųjų reikšmes.
SendSignal_programming.class	Java klasė	Klasė, skirta inicijuoti programavimo etapą ir perduoti visas vykdytam procese priskirtas kintamųjų reikšmes.
SendSignal_programmingends.class	Java klasė	Klasė, skirta inicijuoti programavimo etapo pabaigą ir perduoti visas vykdytam procese priskirtas kintamųjų reikšmes.
SendSignal_taskcreated.class	Java klasė	Klasė, skirta sukurti tėvinę užduotį ir perduoti visą reikiamą informaciją.
SendSignal_testing.class	Java klasė	Klasė, skirta inicijuoti testavimo etapą ir perduoti visas vykdytam procese priskirtas kintamųjų reikšmes.
SendSignal_testingends.class	Java klasė	Klasė, skirta inicijuoti testavimo etapo pabaigą ir perduoti visas vykdytam procese priskirtas kintamųjų reikšmes.
SetManager.class	Java klasė	Klasė, skirta atrinkti ir nustatyti testuotojų komandos vadovą.

Pavadinimas	Paskirtis/tipas	Aprašymas
test.class	Java klasė	Klasė, skirta testavimo metodų priskyrimui konkretiems proceso kintamiesiems.
VerifySubtasksEndedListener.class	Java klasė	Klasėm skirta tikrinti, ar visos vaikinės užduotys yra įvykdytos. Jeigu ne visos, tuomet pateikiamas klaidos pranešimas.

11 priedas. Eksperimentinės sistemos diegimo aktas

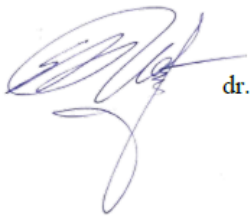
Kauno Technologijos Universitetui

DIEGIMO AKTAS

2020 m. gegužės 18 d.

Kaunas

Patvirtiname, kad studentės Brigitos Baršauskaitės magistro darbe „Programinės įrangos testavimo metodika ir jos taikymas veiklos procesų valdymo sistemoje“ aprašyta ir realizuota testavimo proceso valdymo eksperimentinė sistema įdiegta ir naudojama testuotojų skyriaus reikmėms.

Programavimo padalinio vadovas  dr. Ernestas Vyšniauskas

UAB „ATEA“, J. Rutkausko g. 6, LT-05132, Vilnius ,Lietuva.
Įmonės kodas: 122588443, PVM kodas: LT225884413, Tel. +370 5 239 78 30, Fax. +370 5 239 78 31, el.p.:info@atea.lt
Atsisk. sąsk. : Nordea Bank Finland Plc Lietuvos skyrius. Banko kodas: 21400 SWIFT: NDEALT2X
LT032140030001327814 (LTL), LT412140030001240218 (EUR)
www.atea.lt