KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Vidas Toleikis

CASE ĮRANKIO PLĖTINYS TESTAVIMO ATVEJŲ KŪRIMUI AUTOMATIZUOTI

Baigiamasis magistro projektas

Vadovas
prof. L. Nemuraitė

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

CASE ĮRANKIO PLĖTINYS TESTAVIMO ATVEJŲ KŪRIMUI AUTOMATIZUOTI

Baigiamasis magistro projektas

Informacinių sistemų inžinerijos studijų programa (kodas 621E15001)

Vadovas

prof. L. Nemuraitė

2015-05-25

Recenzentas

dr. S. Pavalkis

2015-05-25

Projektą atliko

Vidas Toleikis

2015-05-25

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

(Fakultetas)

(Studento vardas, pavardė)

Informacinių sistemų inžinerijos studijų programa, 621E15001

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Pavadinimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano, **Vido Toleikis**, baigiamasis projektas tema „*CASE* įrankio plįtinys testavimo atvejų kūrimui automatizuoti“ 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.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

Toleikis, V. *CASE Tool Extension for Automating the Creation of Test Cases*. *Final Degree Project of Master of Information Systems Engineering* / Supervisor Prof. Lina Nemuraitė; Kaunas University of Technology, Faculty of Informatics.

Kaunas, 2015. 52 p.

SUMMARY

Test cases determination and theirs documentation in software life cycle is not sufficiently developed stage, compared to the others. For non-automated test cases generation, this stage required time consuming and makes the human factor errors. The goal of this work - to create the conditions to reduce labor costs and human factory errors amount and increase the integrity and traceability of project artifacts by creating automated generation of test cases from use cases and their scenarios decision and realization for CASE tool, enabling to capture results of testing phase.

Done analysis of similar tools and techniques showed that there is no a complete and re-usable solution for automated generation of test case. For removing these deficiencies created profile, that extend UML Testing Profile by adding parameters for capturing testing data. This plugin is realized in MagicDraw plugin. An experimental research of this plugin confirmed that the goals of this work are met.

Realized plugin recommended for acceptance testing.

Keywords: UTP, UML Testing Profile, Automated Test Case Generation, Test Context, MagicDraw Plugin

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

- **CASE** - *Computer-Aided Software Engineering* – automatizuotas kompiuterinis programinės įrangos projektavimas.
- **OMG** – *Object Management Group* – tai tarptautinė, atviros narystės, technologijų standartų konsorciumas.
- **PA** – Panaudojimo atvejų diagrama.
- **SD** – Sekų diagrama.
- **TTCN** – *Testing and Test Control Notation* – testavimo ir testų kontrolės žymėjimas.
- **UML** - *Unified Modeling Language* – vieninga modeliavimo kalba.
- **UTP** – *UML Testing Profile* – UML testavimo profilis.

ĮVADAS

Šis magistro darbas priklauso Informacinių sistemų inžinerijos studijų programai.

Programinės įrangos gyvavimo cikle testavimo atvejų nustatymas ir jų dokumentavimas yra nepakankamai išplėtotas etapas, lyginant su kitais. Dėl neautomatizuoto testavimo atvejų generavimo šis etapas išsitempia laiko atžvilgiu, bei įneša žmogiškojo faktoriaus klaidų tikimybę. Automatizavus šį procesą sumažėtų klaidų kiekis, sutrumpėtų testavimo laikas ir taip sumažėto testavimo etapo kaina. Taip pat būtų užtikrintas rezultatų vientisumas ir atsekamumas.

UML Testing Profile (*UTP*) yra *UML* plėtinys, kuris naudoja *UML* kalbą apibrėžti, dokumentuoti ir kurti testavimo atvejus programinės įrangos sistemoms. Šis profilis yra *OMG* standartas, todėl integruojamas su kitais *UML* modeliais, jame apibrėžiamos viešai pripažintos ir suprantamos sąvokos ir jų naudojimo taisyklės. Todėl tikslinga jį panaudoti automatizuojant testavimo atvejų kūrimą.

Šį profilis yra realizuotas *MagicDraw* įrankyje, kuris yra apdovanojimų pelnęs verslo procesų, architektūros, programinės įrangos ir sistemos modeliavimo įrankis. Realizuoti automatinį testavimo atvejų generavimą šiame įrankyje būtų gerokai lengviau, nes reikėtų automatizuoti jau įgyvendintą rankinį testavimo atvejų kūrimą.

Siekiama sukurti įskiepį, kuris automatizuotų testavimo atvejų generavimą MagicDraw įrankiui, būtų pagrįstas UTP standartu ir palaikytų testavimo iteracijas, jog būtų galima matyti, kaip progresuoja klaidų ištaisymas ar naujų klaidų atsiradimas testavimo etapo gyvavimo cikle. Šis iteracijų palaikymas suteiktų galimybę automatiniam testavimo ataskaitų generavimui, ko pasigendama realizuotose automatizuotų testavimo atvejų generavimo įrankiuose.

Yra keletas įrankių kuriuose realizuotas automatinis testavimo atvejų generavimas, tačiau jie nėra paremti *UTP* standartu. Vienas iš tokių įrankių yra *Enterprise Architect*. Jo principas yra panašus į siekiamą realizuoti *MagicDraw* įrankyje. Panašumas tas, jog testavimo atvejų generavimui naudojami panaudojimo atvejai ir jų struktūrizuoti scenarijai. Šiuo principu ar jo dalimi paremti dauguma modeliais grindžiamų testavimo atvejų kūrimas, ne tik įrankiuose realizuotų, bet ir teoriniuose metoduose.

Darbo problematika ir aktualumas

Esamuose *CASE* įrankiuose nėra išnaudotos visos galimybės projekto rezultatų vientisumui ir atsekamumui užtikrinti. Testavimo atvejai turėtų būti kuriami pagal reikalavimus. Neautomatizuotas testavimo atvejų kūrimas įneša papildomų klaidų tikimybę ir reikalauja didelių darbo sąnaudų. Automatizuotas testavimo atvejų kūrimas leistų analizuoti testavimo atvejų išsamumą, projekto artefaktų padengimą testais, testavimo rezultatų kokybę. Kadangi reikalavimai aprašomi *CASE* įrankiu, yra visos galimybės generuoti testavimo scenarijus ir testavimo dokumentus iš reikalavimų (panaudojimo atvejų) specifikacijų, susiejant juos su sistemos reikalavimais.

Darbo tikslas ir uždaviniai

Darbo tikslas – sudaryti sąlygas sumažinti testavimo darbo sąnaudas ir žmogiškųjų klaidų tikimybę bei padidinti projekto artefaktų vientisumą ir atsekamumą, sukuriant automatizuoto testavimo atvejų generavimo iš panaudojimo atvejų ir jų scenarijų sprendimą ir jo realizaciją *CASE* įrankyje bei priemones fiksuoti testavimo etapo rezultatus.

Uždaviniai:

1. Išanalizuoti:
    1. 1 Esamus modeliais grindžiamo testavimo atvejų kūrimo metodus (*OMG UML* testavimo profilį ir kt.)
    1. 2 Esamus testavimo atvejų kūrimo sprendimus *CASE* įrankiuose
    1. 3 *CASE* įrankių naudotojų poreikius testavimo procesui automatizuoti
2. Sudaryti *UML* testavimo profiliu paremtą pasirinkto projekto testavimo modelį ir automatizuoto testavimo atvejų kūrimo algoritmą
3. Sudaryti automatizuoto testavimo atvejų generavimo įskiepio projektą

4. Realizuoti automatizuoto testavimo atvejų generavimo įskiepio prototipą
5. Atlikti eksperimentą sukurto sprendimo tinkamumui įvertinti

Darbo rezultatai ir jų svarba

Atlikta panašių įrankių ir metodų analizė parodė, kad nėra išbaigto ir pakartotinai panaudojamo sprendimo testavimo atvejams generuoti, bei nėra paremto *OMG* standartu.

Atlikus *UML testavimo profilio* analizę paaiškėjo jog tikslinga naudoti šį profilį kuriant automatizuotą testavimų atvejų generavimui, dėl jo išbaigtumo, pripažinimo ir tai jog jis yra *OMG* standartas. Norint sudaryti galimybę fiksuoti testavimo etapo rezultatus, bei reikalingus duomenis testavimui vykdyti, reikia sukurti profilį, kuris išplėstų *UTP* profiliui priklausantį *TestCase* stereotipą papildomais parametrais.

Norint įgyvendinti darbo tikslus reikalingas *CASE* įrankis, kuriame yra realizuotas *UTP* profilis, suteikiantis galimybę išplėsti profilį ir sukurti naują funkcionalumą. Atlikus *MagicDraw* įrankio analizę paaiškėjo, kad jis griežtai laikosi standartų, jame yra realizuotas *UTP* ir sudaryta galimybė išplėsti įrankio funkcionalumą, kuriant įskiepius ir profilius. Šios galimybės reikalingos norint įgyvendinti darbo tikslą, tad nuspręsta įskiepį realizuoti būtent šiam įrankiui.

Realizuotas įskiepis ir atliktas eksperimentas patvirtino darbo tikslų įgyvendinamumą.

Darbo struktūra

Darbą sudaro 7 skyriai.

Pirmame skyriuje analizuojamas *UML Testing Profile* profilis, kuris skirtas *UML* kalbą apibrėžti, dokumentuoti ir kurti testavimo atvejus programinės įrangos sistemoms, bei *MagicDraw* įrankis, kuriame realizuotas šis profilis. Taip pat analizuojami realizuoti automatizuoto testavimo atvejų generavimo prototipai/įrankiai (*UC-System*, *Enterprise Architect*), *IBM* siūlomas teorinis modelis, bei pateikiamas metodų palyginimas. Po analizės dalies pateikiamas darbo tikslas, uždaviniai, siekiamo sprendimo apibrėžimas ir padarytos išvados kaip įgyvendinti darbo tikslą.

Antrame skyriuje pateikta MagicDraw, įskiepio ir naudotojo tarpusavio sąveika, automatizuoto testavimo atvejų generavimo modelis, kuriame matyti kokie komponentai įeis į automatizuojamą veiklos proceso dalį. Pateiktas sukurtas algoritmas pagrindiniams ir alternatyviems scenarijams iš panaudojimo atvejų diagramos išgauti. Taip pat pateiktas sukurtas profilis, kuris išplečia *TestCase* stereotipą parametrais skirtais testavimo duomenims fiksuoti.

Trečiame skyriuje pateikta įskiepio architektūra, detalus projektas, nusakantis kaip įskiepis bus realizuotas.

Ketvirtame skyriuje pateikti realizuoto sprendimo testavimas, t. y. kokie testavimai buvo atlikti ir kokie rezultatai gauti.

Penktame skyriuje pateiktas atliktas eksperimentas, kurio metu norima patvirtinti jog pavyko įgyvendinti išsikeltus tikslus.

Šeštame skyriuje pateikiamos atliktą darbą apibendrinančios išvados.

Septintame skyriuje pateikiamas literatūros sąrašas, kuri buvo analizuojama rašant ši darbą.

1. PROBLEMINĖS SRITIES ANALIZĖ

Analizės tikslas – išanalizuoti esamus modeliais grindžiamo testavimo atvejų kūrimo metodus bei testavimo atvejų kūrimo sprendimus *CASE* įrankiuose, naudotojų poreikius ir esamas *MagicDraw* įrankio galimybes, kurias bus galima panaudoti realizuojant automatizuotą testavimo atvejų generavimą iš panaudojimo atvejų:

- Panaudojimo atvejų ir jų scenarijų vaizdavimą
- Testavimo atvejų kūrimą ir aprašymą

Tyrimo objektas – automatizuoto testų kūrimo procesas, leidžiantis sugeneruoti testavimo atvejus ir dokumentus iš reikalavimų specifikacijų.

Tyrimo sritis –modeliais grindžiamo kūrimo ir testų kūrimo automatizavimo metodai *CASE* įrankių aplinkoje.

Tyrimo problema yra ta, kad esamuose *CASE* įrankiuose nėra išnaudotos visos galimybės projekto rezultatų vientisumui ir atsekamumui užtikrinti. Testavimo atvejai turėtų būti kuriami pagal reikalavimus. Neautomatizuotas testavimo atvejų kūrimas įneša papildomų klaidų tikimybę ir reikalauja didelių darbo sąnaudų. Automatizuotas testavimo atvejų kūrimas leistų analizuoti testavimo atvejų išsamumą, projekto artefaktų padengimą testais, testavimo rezultatų kokybę. Kadangi reikalavimai aprašomi *CASE* įrankiu, yra visos galimybės generuoti testavimo scenarijus ir testavimo dokumentus iš reikalavimų (panaudojimo atvejų) specifikacijų, susiejant juos su sistemos reikalavimais.

1.1. *UML* testavimo profilio ir jo taikymo *MagicDraw CASE* įrankyje analizė

UML Testing Profile (*UTP*) yra *UML* plėtinys, kuris naudoja *UML* kalbą apibrėžti, dokumentuoti ir kurti testavimo atvejus programinės įrangos sistemoms. Šis profilis yra *Object Management Group* (*OMG*) standartas. *OMG* yra tarptautinis, atviros narystės, technologijų standartų konsorciumas. *OMG* standartai paremti pardavėjų, galutinių vartotojų, akademinių institucijų ir vyriausybinių agentūrų. *OMG* darbo grupės sukurti verslo integravimo standartai įvairiuose technologijose ir pramonės šakose. *OMG* modeliavimo standartai, įskaitant *UML*, suteikia galimybę vizualiai dokumentuoti programinės įrangos kūrimą, priežiūrą ir kitus procesus.

UTP numatytas testavimo integravimas su modeliavimu. *UTP*, kaip ir *UML* bei *SysML* reikalavimų modeliai yra tarpusavyje integruoti *OMG* standartai. Jame pateikiamas testavimo ekspertų sudarytas standartinis, viešai pripažintas ir suprantamas testavimo sąvokų ir jų naudojimo apibrėžimas. Todėl tikslinga jį panaudoti automatizuojant testavimo atvejų kūrimą.

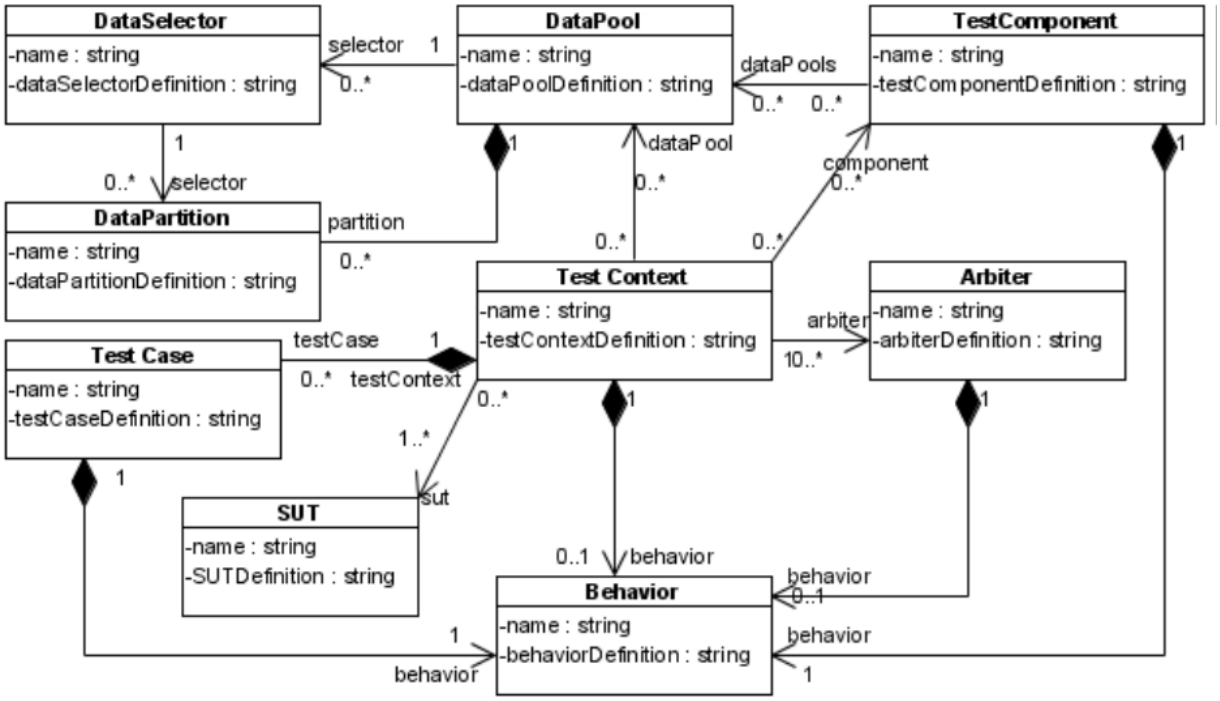
Profilis (angl. *Profile*) išplečia ir riboja originalią *UML* kalbą. *UTP* nustato testavimo specifikacijų ir testavimo modelių sąvokas, apimančias testavimo architektūrą, elgseną, duomenis ir laiką. Kartu šios sąvokos apibrėžia modeliavimo kalbą, kuri skirta vizualizuoti, specifikuoti, analizuoti, kurti ir dokumentuoti testavimo sistemos artefaktus. Filosofijoje, kuri pritaikyta testavimo koncepcijų kūrimui, panaudoti *UML 2.0* sąvokos ir taip sumažintas naujų konceptų įvedimas. Analizuojant esamas testavimo specifikacijas ir įgyvendintus metodus testavimui, nustatytos papildomos testavimo profilio sąvokos. 1.1 paveikslėlyje pavaizduotas šio profilio metamodelis. Tai modeliavimo kalbos modelis, kuriame apibrėžiamos pagrindinės kalbos savybės.

UTP įveda sąvokas darbui su nenumatytu elgesiu, suteikia priemones apibrėžti išsamesnius, dar abstraktesnius testavimo modelius. Tai supaprastina tikrinimą ir pagerina testavimo modelio aiškumą.

Kitas svarbus testavimo specifikacijos aspektas yra testavimo duomenų apibrėžimas ir kodavimas. Tam *UTP* palaiko simbolius (angl. *Wildcards*), duomenų telkinius (angl. *Data Pool*), duomenų skaidinius (angl. *Data Partitions*), duomenų selektorius (angl. *Data Selector*) ir kodavimo taisykles. Simboliai yra labai naudingi tvarkant netikėtus įvykius ar įvykius, kuriuose yra daug įvairių reikšmių.

Aptartos sąvokos suteikia reikalingas galimybes sudaryti tikslią testavimo specifikaciją naudojant *UML 2.0*. Testavimo profilis apima tiek struktūrines tiek elgesio elementus ir suteikia

pagrindines testavimo domenų sąvokas, kurios padaro testavimo specifikaciją efektyvesnę ir veiksmingesnę.



1.1 pav. *UML-TP* metamodelis [1]

Metamodelio elementai:

- Testavimo kontekstas (angl. *Test Context*) leidžia grupuoti testavimo atvejus, aprašyti testavimo konfigūracija, ir apibrėžti testavimo kontrolę, tai yra reikalinga testavimo atvejų vykdymo tvarka.
- Testavimo atvejis (angl. *Test Case*) yra testavimo konteksto operacija, nurodanti, kaip bendradarbiaujančių komponentų rinkinys sąveikauja su *SUT*, kad įgyvendintų testavimo tikslą.
- Testavimo komponentai (angl. *Test Component*) apibrėžiami kaip testavimo sistemos objektai, kurie gali bendrauti su *SUT* ar kitais komponentais realizuojančiais testavimo elgseną.
- Elgsena (angl. *Behavior*) apibrėžia veiksmus ir operacijas, kurios reikalingos testavimo objektui įvertinti.
- *SUT – System Under Test* – sistema po testavimo.
- Duomenų telkinyje (angl. *Data Pool*) pateikiamas reikšmių ar suskirstymų rinkinys, kuris susijęs su konkrečiu testavimo kontekstu ir jo testavimo atveju.
- Duomenų skaidinys (angl. *Data Partition*) yra naudojamas apibrėžti lygiavertes klases ir duomenų rinkinius.
- Duomenų selektorius (angl. *Data Selector*) apibrėžia skirtingas pasirinkimo sąlygas duomenų rinkiniams.
- Arbitras (angl. *Arbiter*) suteikia galimybę įvertinti testavimo rezultatus , gautus su įvairiais objektais, siekiant nustatyti bendrą testavimo atvejo ar testavimo konteksto sprendimą.

UML testavimo profilis yra realizuotas *MagicDraw CASE* įrankyje.

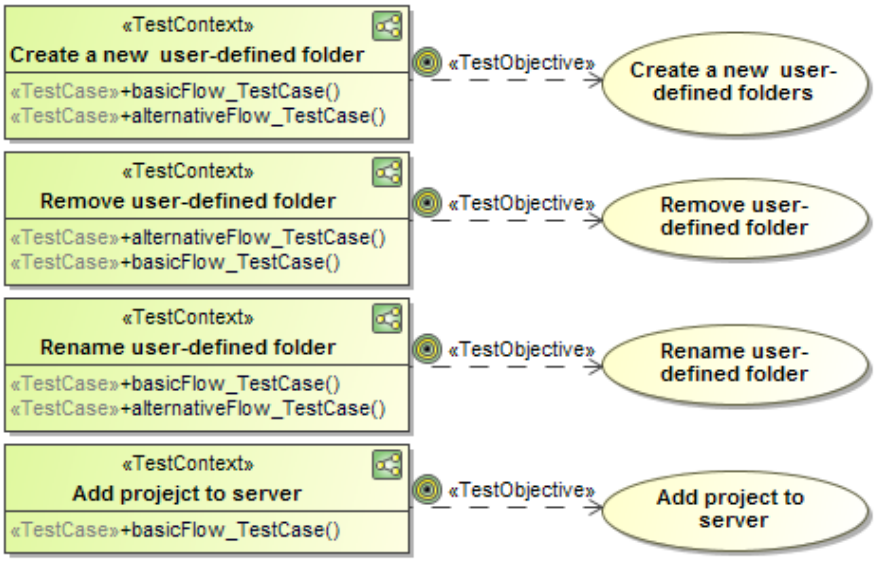
MagicDraw yra verslo procesų, architektūros, programinės įrangos ir sistemos modeliavimo įrankis. Jis palaiko komandinį darbą ir yra skirtas verslo, programinės įrangos analitikams, programuotojams, kokybės užtikrinimo inžinieriams, dokumentacijos rašytojams. Tai dinamiškas ir

universalus projektavimo įrankis, palengvinantis objektinių sistemų ir duomenų bazių analizę ir projektavimą. *MagicDraw™* atitinka naujausius *Java* bei *UML* technologijų standartus, turi vieną iš patikimiausių išeities kodo inžinerijos mechanizmų *Java, C#, C++* ir *CORBA IDL* programavimo kalboms, gali vykdyti atvirkštinę šių kalbų kodo ir duomenų bazių schemų inžineriją, duomenų bazių schemų generavimą. *MagicDraw™* naudoja ciklinę (angl. *Roundtrip*) technologiją, leidžiančią keisti tiek objektinį modelį, tiek programos kodą bet kokia tvarka, juos nuolat sinchronizuojant.

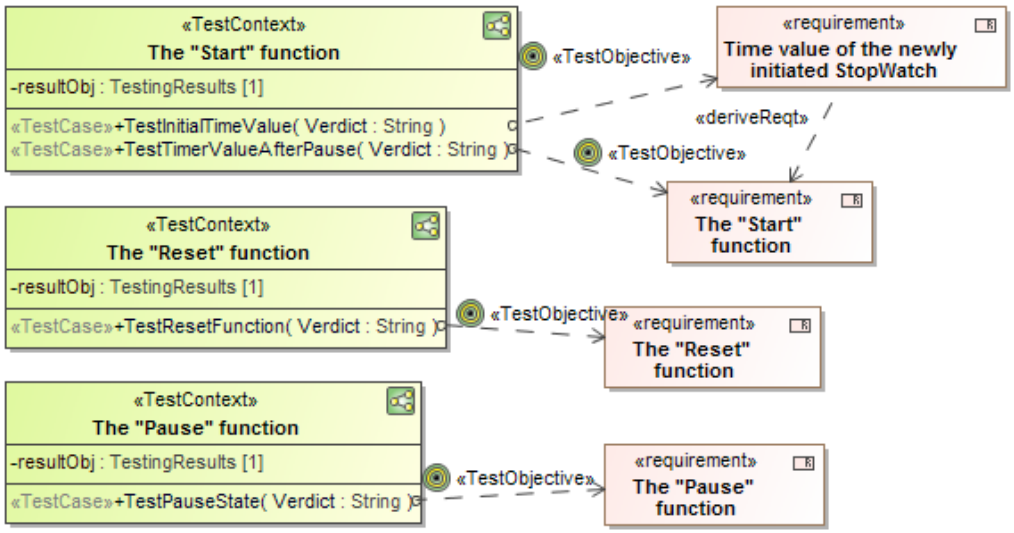
Su *UML* profiliais ir pritaikytomis diagramomis (angl. *custom diagrams*) galima išplėsti *UML 2* standartą. Naudojant *Open API,* galima išplėsti funkcionalumą įtraukiant naujus dizaino modelius, metrikas, transformacijas ar kitus įskiepius. Su adaptuojamu *MagicDraw* ataskaitų generavimo mechanizmu naudotojai gali pritaikyti dokumentus savo įmonės vidaus kūrimo procesams.

MagicDraw įrankyje, panaudojimo atvejų profilyje, jau yra realizuotas panaudojimo atvejų scenarijų aprašymas, kuris palengvintų testų generavimo iš panaudojimo atvejų automatizavimą. Sukurtas profilis, leidžiantis kurti testavimo atvejus, paremtus *UML Testing Profile* (*UTP*) standartu. *MagicDraw* įrankyje testavimo architektūros diagramos pavaizduotos 1.2 ir 1.3 paveikslėlyje.

Norint sudaryti galimybę fiksuoti testavimo etapo rezultatus, bei reikalingus duomenis testavimui vykdyti reikia sukurti profilį, kuris išplėstų *TestCase* stereotipą papildomais parametrais.

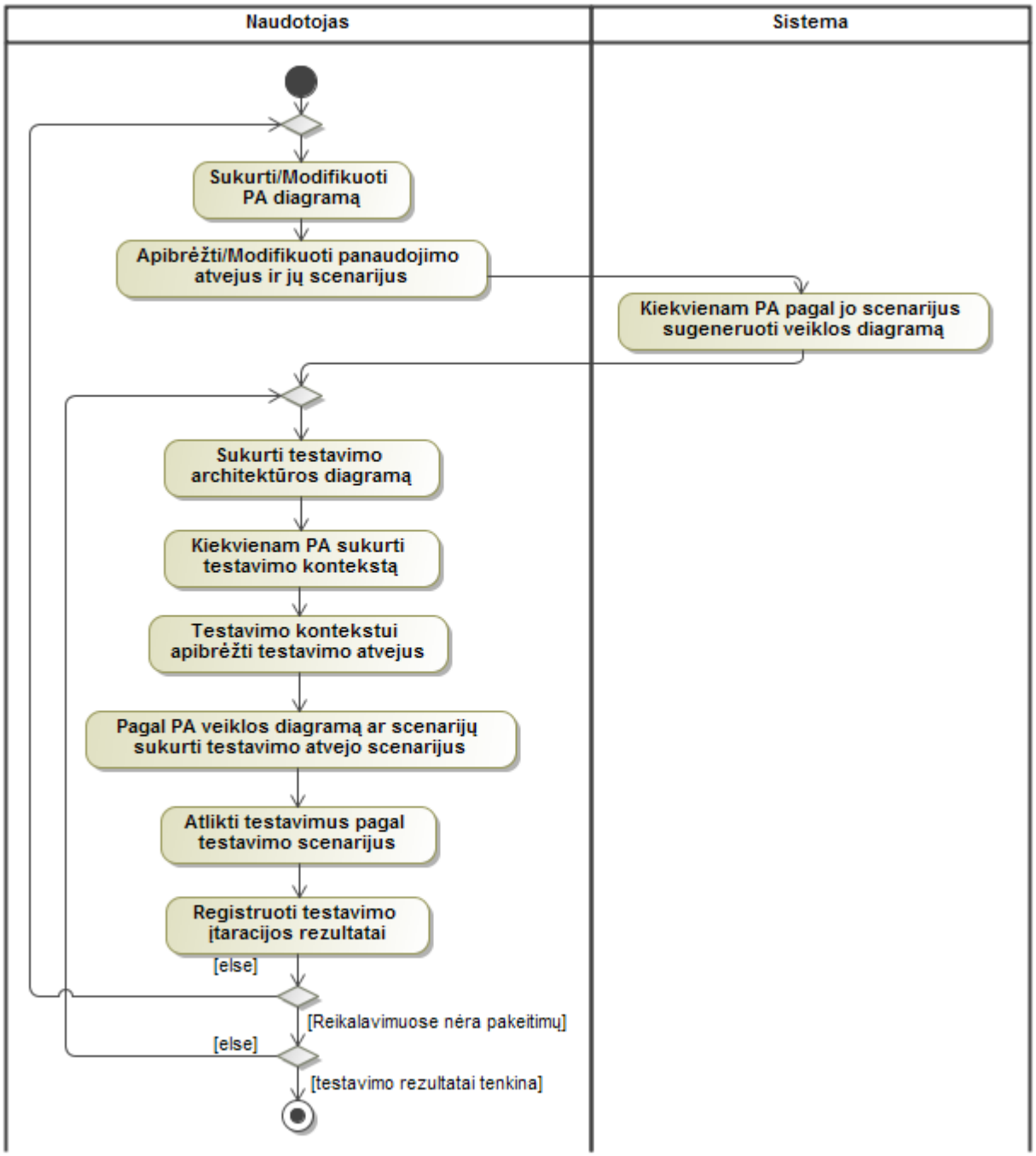


1.2 pav. Testavimo architektūros diagrama (testavimo atvejai susieti su panaudojimo atvejais)



1.3 pav. Testavimo architektūros diagrama (testavimo kontekstai susieti su reikalavimais)

Esamas procesas, t. y., veiksmų seka, kurią reiki norint sukurti testavimo architektūros diagramą pagal panaudojimo atvejų diagramą, pavaizduotas **1.4** pav.



1.4 **pav. Esamas testavimo procesas** *MagicDraw* **įrankyje**

1.2. Automatizuoto testavimo atvejų generavimo metodų literatūros šaltiniuose analizė

IBM siūlomas teorinis metodas [3] sudaryti testavimo atvejus iš panaudojimo atvejų susideda iš trijų etapų:

1. Kiekvienam panaudojimo atvejui sudaromas pilnas panaudojimo atvejų scenarijų rinkinys.
2. Kiekvienam scenarijui nustatomas bent vienas testavimo atvejis ir jo vykdymo sąlygą.
3. Kiekvienam testavimo atvejui nustatomos duomenų reikšmės su kuriomis bus testuojama.

Šiuos tris etapus panagrinėsime pagal konkretų pavyzdį.

Pirmame etape iš panaudojimo atvejo ir aprašyto tekstinio scenarijaus nustatomos pagrindinio ir alternatyvių srautų (angl. *main and alternate flows*) kombinacijos ir sukuriama scenarijų matrica (pvz. 1.1 lentelėje).

1.1 lentelė. Dalinio scenarijaus matrica kurso registracijos panaudojimo atvejui [3]

Scenarijaus pavadinimas	Pradedantysis srautas	Alternatyvus
Scenarijus 1 - Sėkminga registracija	Pagrindinis srautas	
Scenarijus 2 - Neidentifikuotas studentas	Pagrindinis srautas	A1
Scenarijus 3 - Naudotojas išėjo	Pagrindinis srautas	A2
Scenarijus 4 - Kurso katalogo sistema nepasiekiama	Pagrindinis srautas	A4
Scenarijus 5 - Registracija uždaryta	Pagrindinis srautas	A5
Scenarijus 6 – Negali įrašyti	Pagrindinis srautas	A3

Antrame etape testavimo atvejus nustatome analizuojant scenarijus ir peržiūrint panaudojimo atvejų tekstinius aprašymus. Toliau iš panaudojimo atvejų tekstinio aprašymo nustatomos sąlygos ir reikalingų duomenų elementai įvairių scenarijų vykdymui. Testavimo atvejų dokumentavimui panaudojamas matricos formatas (pvz. 1.2 lentelėje).

1.2 lentelė. Testavimo atvejaus matrica kurso registracijos panaudojimo atvejui [3]

Testavimo atvejaus ID	Scenarijus/ Sąlyga	Studento ID	Slaptažodis	Pasirinktas kursas	Būtina sąlyga įvykdymui	Kursas atidarytas	Tvarkaraštis atidarytas	Laukiamas rezultatas
RC 1	Scenarijus 1- sėkminga registracija	V	V	V	V	V	V	Parodomas tvarkaraštis ir patvirtinimo numeris
RC 2	Scenarijus 2- neidentifikuotas studentas	I	n/a	n/a	n/a	n/a	n/a	Klaidos pranešimas; Grįžtama į prisijungimo langą
RC 3	Scenarijus 3- galiojantis naudotojas išeina	V	V	n/a	n/a	n/a	n/a	Pasirodo prisijungimo langas
RC 4	Scenarijus 4- kurso registracijos sistema nepasiekiama	V	V	n/a	n/a	n/a	n/a	Klaidos pranešimas; grįžtama į 2 žingsnį
RC 5	Scenarijus 5- registracija uždaryta	V	V	n/a	n/a	n/a	n/a	Klaidos pranešimas; grįžtama į 2 žingsnį
RC 6	Scenarijus 6- negali įrašyti - kursas pilnas	V	V	V	V	I	V	Klaidos pranešimas; grįžtama į 3 žingsnį
RC 7	Scenarijus 6- negali įrašyti - būtina sąlyga nėra įvykdyta	V	V	V	I	V	V	Klaidos pranešimas; grįžtama į 4 žingsnį
RC 8	Scenarijus 6- negali įrašyti - tvarkaraščio konfliktas	V	V	V	V	V	I	Klaidos pranešimas; grįžtama į 4 žingsnį

„V“ nurodo teisingą, „I“ – neteisingą reikšmę, o „n/a“ reiškia, kad nereikia nurodyti duomenų šitame atvejyje.

Be testavimo duomenų testavimo atvejai negali būti vykdomi . Todėl trečiame etape nustatomos faktinės duomenų reikšmės reikalingos „V“ ir „I“ kintamiesiems, kurios bus naudojamos vykdant baigiamuosius testus. 1.3 lentelėje pateikiama pavyzdinė testavimo atvejų matrica su nustatytomis faktinėmis reikšmėmis.

1.3 lentelė. Testavimo atvejaus matrica su duomenų reikšmėmis [3]

Testavimo atvejaus ID	Scenarijus/ Sąlyga	Studento ID	Slaptažodis	Pasirinktas kursas	Būtina sąlyga įvykdymui	Kursas atidarytas	Tvarkaraštis atidarytas	Laukiamas rezultatas
RC 1	Scenarijus 1- sėkminga registracija	jheumann	abc123	M101 E201 S101	Yes	Yes	Yes	Parodomas tvarkaraštis ir patvirtinimo numeris
RC 2	Scenarijus 2- neidentifikuotas studentas	Jheumann1	n/a	n/a	n/a	n/a	n/a	Klaidos pranešimas; Grįžtama į prisijungimo langą
RC 3	Scenarijus 3- galiojantis naudotojas išeina	jheumann	abc123	n/a	n/a	n/a	n/a	Pasirodo prisijungimo langas
RC 4	Scenarijus 4- kurso registracijos sistema nepasiekiama	jheumann	abc123	n/a	n/a	n/a	n/a	Klaidos pranešimas; grįžtama į 2 žingsnį
RC 5	Scenarijus 5- registracija uždaryta	jheumann	abc123	n/a	n/a	n/a	n/a	Klaidos pranešimas; grįžtama į 2 žingsnį
RC 6	Scenarijus 6- negali įrašyti - kursas pilnas	jheumann	abc123	M101 E201 S101	Yes	M101 full	Yes	Klaidos pranešimas; grįžtama į 3 žingsnį
RC 7	Scenarijus 6- negali įrašyti - būtina sąlyga nėra įvykdyta	jheumann	abc123	M101 E201 S101	No for E201	Yes	Yes	Klaidos pranešimas; grįžtama į 4 žingsnį
RC 8	Scenarijus 6- negali įrašyti - tvarkaraščio konfliktas	jheumann	abc123	M101 E201 S101	Yes	Yes	E202 and S101 conflict	Klaidos pranešimas; grįžtama į 4 žingsnį

Šio metodo esmė – sudaryti testavimo scenarijus, kuriems identifikuojami visi reikalingi kintamieji, bei teims kintamiesiems apibrėžiamos konkrečiuos reikšmės, su kuriomis turi būti vykdomas testavimas. Kadangi naudinga testuojant apibrėžti testavimo duomenis kuriamame sprendime tikslinga sudaryti sąlygas juos apibrėžti.

1.3. Automatizuotų testavimo atvejų generavimo realizacijų analizė

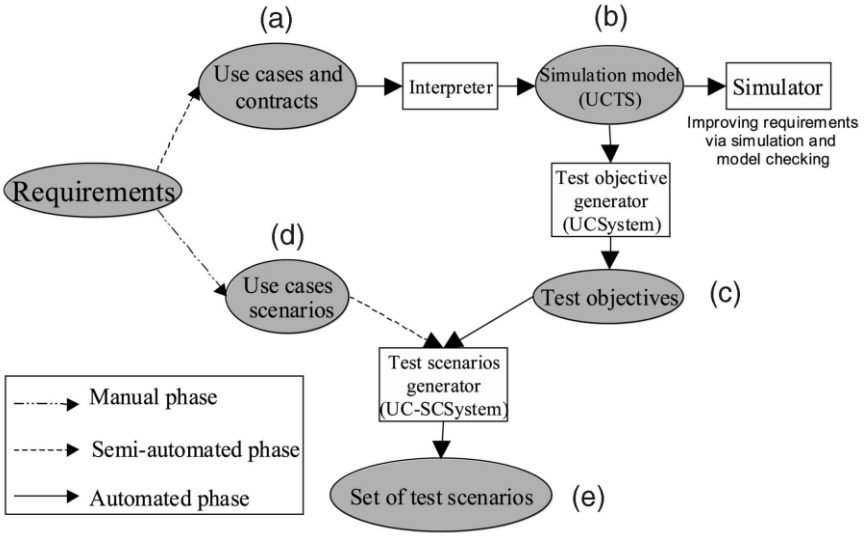
1.3.1. *UC-System* prototipas

Automatic Test Generation: A Use Case Driven Approach [2] straipsnyje aprašomas metodas sudarytas iš 2 etapų:

1. Testavimo tikslų generavimas iš panaudojimo atvejų (etapai (a)-(c) 1.5 pav.).
2. Testavimo scenarijų generavimas iš testavimo tikslų (etapai (c)-(e) 1.5 pav.).

Etape (a) (žr. 1.5 pav.) panaudojimo atvejams nustatomi parametrai. Parametrais gali būti tiek aktoriai (kaip dalyviai), tiek programos pagrindinės sąvokos (verslo sąvokos reikalavimų analizėje). Kai parametrai panaudojimo atvejams yra apibrėžti, kontraktai (angl. *contracts)* išreiškiami prieš ir po sąlygų formą įtraukiant šiuos parametrus. Panaudojimo atvejų kontraktai yra pirmo laipsnio loginė išraiška derinant predikatus su loginiais operatoriais. Predikatai yra naudojami apibūdinti sistemos faktus (t.y. aktorių būseną, pagrindinių koncepcijų būseną arba roles).

Prototipo įrankis (*UC-System*) iš panaudojimo atvejų ir jų kontraktų sukuria imitacijos modelį ((b) etapas 1.5 pav.) ir generuoja korektiškas panaudojimo atvejų sekas ((c) etapas 1.5 pav.), kurios vadinamos testavimo tikslais. Kaip parodyta 1.5 pav. panaudojimo atvejų modelis gali būti imituojamas ir taip patikrinami ir jei reikia pataisomi reikalavimai prieš testų generavimą iš jų.

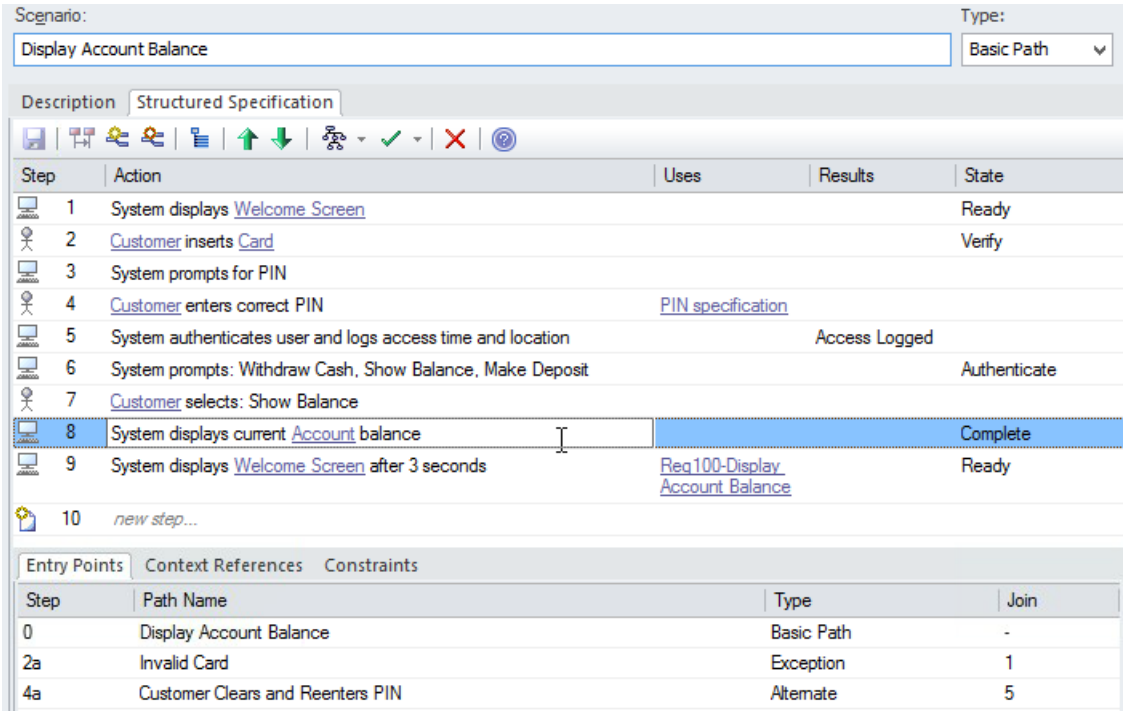


1.5 pav. Reikalavimais grindžiamų testavimo scenarijų generavimo metodika [2]

Antrame metodo etape generuojami testavimo scenarijai iš testavimo tikslų. Testavimo scenarijai gali būti tiesiogiai naudojami kaip testavimo atvejai arba gali reikėti kažkokių testuotojo papildymų jei vis dar trūksta tam tikrų pranešimų ar parametrų. Ši informacija gali būti pridedama iš panaudojimo atvejų, veiklos, būsenų diagramų. Tačiau paprastumo dėlei naudojama sekų diagrama, kuri vadinama panaudojimo atvejų scenarijumi. Šie scenarijai sukuriami 1.5 pav. (d) etape.

1.3.2. Enterprise Architect CASE įrankis

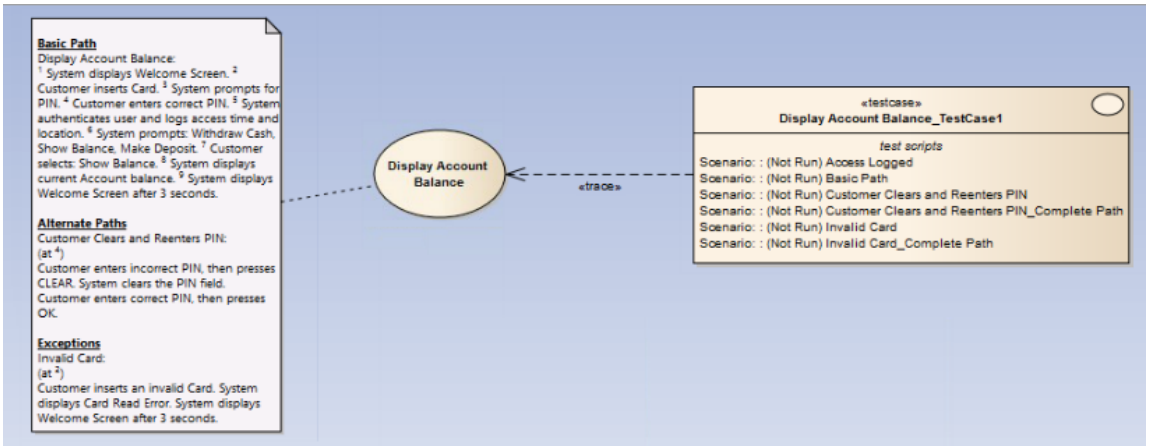
Šiame įrankyje yra realizuotas automatizuotas testavimo atvejų generavimas iš panaudojimo atvejų struktūrizuotų scenarijų. 1.6 pav. pateikta kaip atrodo pilnai struktūrizuotas scenarijus iš kurio bus generuojamas testavimo atvejis.



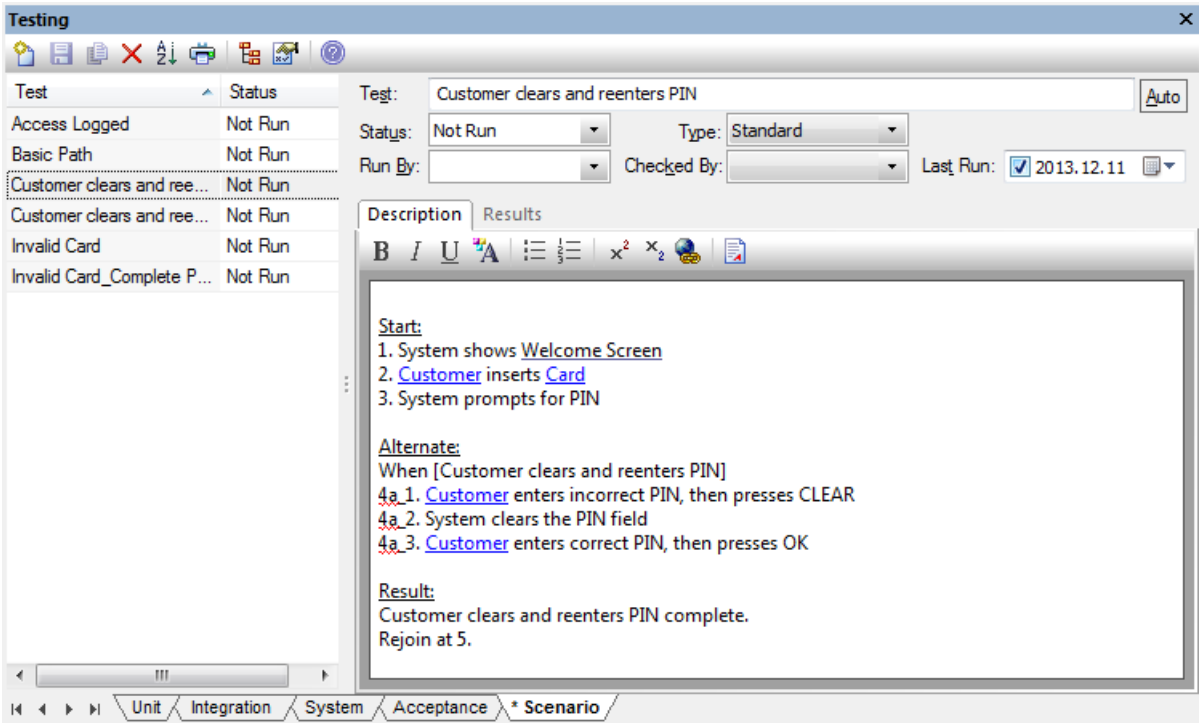
1.6 pav. Baigtas struktūrinis scenarijus [10]

Panaudojimo atvejo struktūrizuotą scenarijų sudaro pagrindinis, alternatyvūs ir išimties scenarijai. Bet kurį scenarijų sudaro veiksmų seka, kuri turi būti įvykdyta. Kiekvienam veiksmui galima nurodyti papildomus parametrus: veiksmo iniciatorių (naudotojas, sistema), bet kokį naudojamą (angl. *Uses*) modelio elementą , rezultatą ir sistemos būseną. Alternatyviems ir išimties scenarijams nurodomas pradžios žingsnis (angl. *Step*) ir kuriame žingsnyje (angl. *Join*) sistema atsidurs po įvykdyto scenarijaus. Apribojimų (angl. *Constrains*) kortelėje nurodomos prieš ir po sąlygos.

1.7 pav. pateiktas sugeneruotas testavimo atvejis iš 1.6 pav. pateikto panaudojimo atvejo. Testavimo atvejyje saugomi testavimo scenarijai, kurie yra nukopijuoti iš panaudojimo atvejo. Alternatyvių ir išimties scenarijai pateikiami dviem variantais. Pirmasis (1.8 pav.) vykdantis tik iki alternatyvaus ar išimties scenarijaus paskutinio žingsnio, o antrasis (1.9 pav.) – išplėstą scenarijų, t.y. užbaigus alternatyvų ar išimties scenarijus grįžta į pagrindinį ir užbaigia scenarijų iki galo.



1.7 pav. Testavimo atvejis [10]



1.8 pav. Naudotojas išvalo ir iš naujo įveda slaptažodį [10]

Start:
1. System shows Welcome Screen
2. Customer inserts Card
3. System prompts for PIN

Alternate:
When [Customer clears and reenters PIN]
4a 1. Customer enters incorrect PIN, then presses CLEAR
4a 2. System clears the PIN field
4a 3. Customer enters correct PIN, then presses OK

Continues:
5. System authenticates user and logs access time and location
6. System prompts: Withdraw Cash, Show Balance, Make Deposit
7. Customer selects - Show Balance
8. System display current Account balance
9. System display Welcome screen after 3 seconds
Uses: Req100 - Display Account Balance

Result:
Use case ends.

1.9 pav. Naudotojas išvalo ir iš naujo įveda slaptažodį (pilnas kelias) [10]

Testavimo scenarijus turi šiuos parametrus:

1. Statusas (nevykdytas, sėkmingas, nesėkmingas, atidėtas)
2. Tipas (standartinis, regresinis)
3. Kieno sukurtas
4. Kieno vykdytas
5. Kada paskutinį kartą vykdytas.

Enterprise Architect CASE įrankyje realizuotas automatizuotas testavimo atvejų generavimas neatitinka UTP standarto. Pagal UTP metamodelį, kuris pateiktas 1.1 paveikslėlyje, testavimo atvejis turi atitikti vieną scenarijų ir priklausyti testavimo kontekstui, kuriame saugomi

vieno panaudojimo atvejo visi scenarijai. O šiame įrankyje panaudojimo atvejo visi testavimo scenarijai saugomi testavimo atvejyje.

1.3.3. Esamų testavimo scenarijų sudarymo ir generavimo metodų palyginimas

Išanalizavus analizėje pateiktus metodus paaiškėjo, jog ne vienas realizuotas įrankyje metodas nepalaiko automatinio ataskaitų ar apibendrinančių lentelių generavimo apie testavimą. Panaudojimo atvejų diagrama yra pagrindas testavimo atvejų generavimo šiuose metoduose. Metodai skiriasi papildomos informacijos šaltiniais. *IBM* teoriniame metode įrankyje naudojami panaudojimo atvejo tekstiniai scenarijai, aprašyti dokumentuose. *Enterprise Architect CASE ir MagicDraw* įrankiuose naudojami struktūrizuoti tekstiniai scenarijai. Tik *MagicDraw* įrankyje papildomai naudojamos veiklos diagramos, kurios sugeneruotos iš struktūrizuotų tekstinių scenarijų. O *Automatic Test Generation: A Use Case Driven Approach* [2] straipsnyje aprašytame prototipe naudojamos sekų diagramos.

MagicDraw įrankis vienintelis palaiko *UTP* standartą. Apibendrintas palyginimas pateiktas 1.4 lentelėje.

1.4 lentelė. Išanalizuotų metodų palyginimas

Metodas / Kriterijus	IBM	[2] straipsnyje siūlomas metodas	Enterprise Architect	MagicDraw	Siekiamas sprendimas
1. Ar atitinka UTP standartą	-	-	-	+	+
2. Ar realizuotas	-	+	+	+	+
3. Ar automatizuotas testavimo atvejų generavimas	-	+	+	-	+
4. Ar palaiko iteracijas	-	-	-	-	+
5. Ar automatiškai generuojamos testavimą apibendrinančios lentelės/ ataskaitos	-	-	-	-	+

1.4. Darbo tikslas, uždaviniai ir siekiami privalumai

Darbo tikslas – sudaryti sąlygas sumažinti testavimo darbo sąnaudas ir žmogiškųjų klaidų tikimybę bei padidinti projekto artefaktų vientisumą ir atsekamumą, sukuriant automatizuoto testavimo atvejų generavimo iš panaudojimo atvejų ir jų scenarijų sprendimą ir jo realizaciją *CASE* įrankyje bei priemones fiksuoti testavimo etapo rezultatus..

Uždaviniai:

1. Išanalizuoti:
   1.1. Esamus modeliais grindžiamo testavimo atvejų kūrimo metodus (*OMG UML* testavimo profilį ir kt.)
   1.2. Esamus testavimo atvejų kūrimo sprendimus *CASE* įrankiuose
   1.3. *CASE* įrankių naudotojų poreikius testavimo procesui automatizuoti
2. Sudaryti automatizuoto testavimo atvejų kūrimo metodiką ir *UML* testavimo profiliu paremtą pasirinkto projekto testavimo modelį
3. Suprojektuoti automatizuoto testavimo atvejų generavimo įskiepio projektą
4. Realizuoti automatizuoto testavimo atvejų generavimo įskiepio prototipą
5. Atlikti eksperimentą sukurtos metodikos ir įskiepio tinkamumui įvertinti

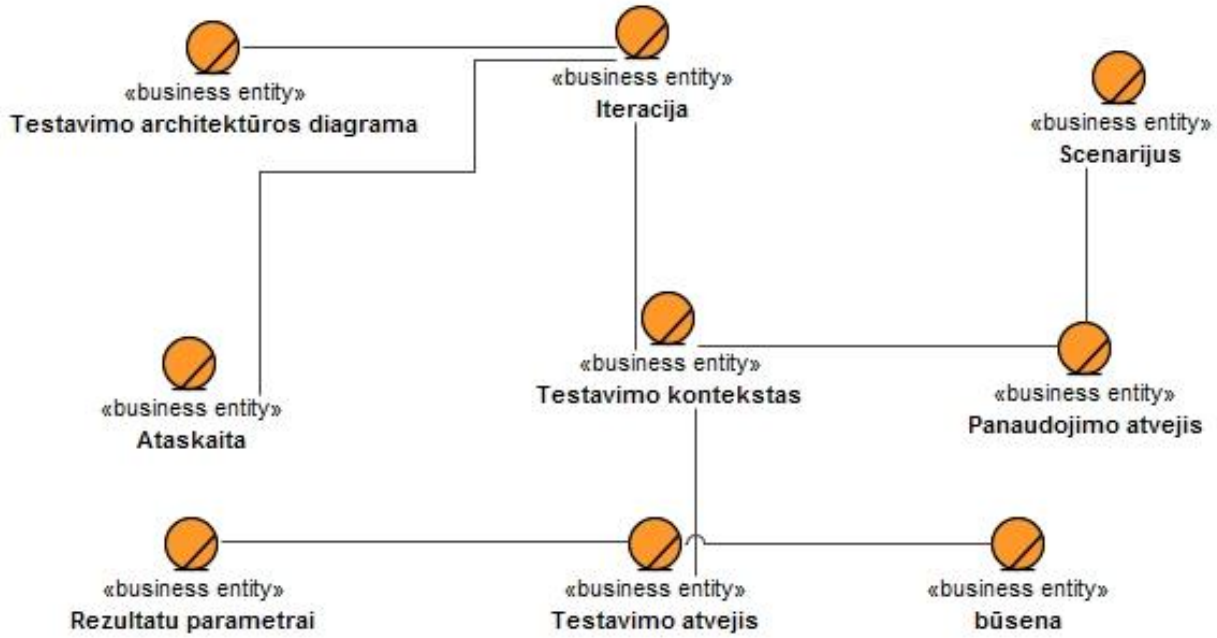
Pagrindinis testavimo atvejų kūrimo automatizavimo **naudotojas** yra kokybės inžinierius. Tačiau tai nėra vienintelis naudotojas. Su testavimo atvejais susiduria programinės įrangos ir sistemų projektuotojai, projektų ir kokybės vadovai.

Testavimas yra svarbus etapas programinės įrangos kūrimo gyvavimo cikle ir užima nemažą jo dalį. Viena iš svarbiausių, imli laikui ir įvelianti klaidų testavimo veiklų yra testavimo atvejų nustatymas ir jų dokumentavimas. Automatizavus šį etapą, būtų pasiekti šie rezultatai:

1. Mažesnės testavimo darbo sąnaudos;
2. Mažesnė žmogiškųjų klaidų tikimybė;
3. Geresnis projekto artefaktų atsekamumas.

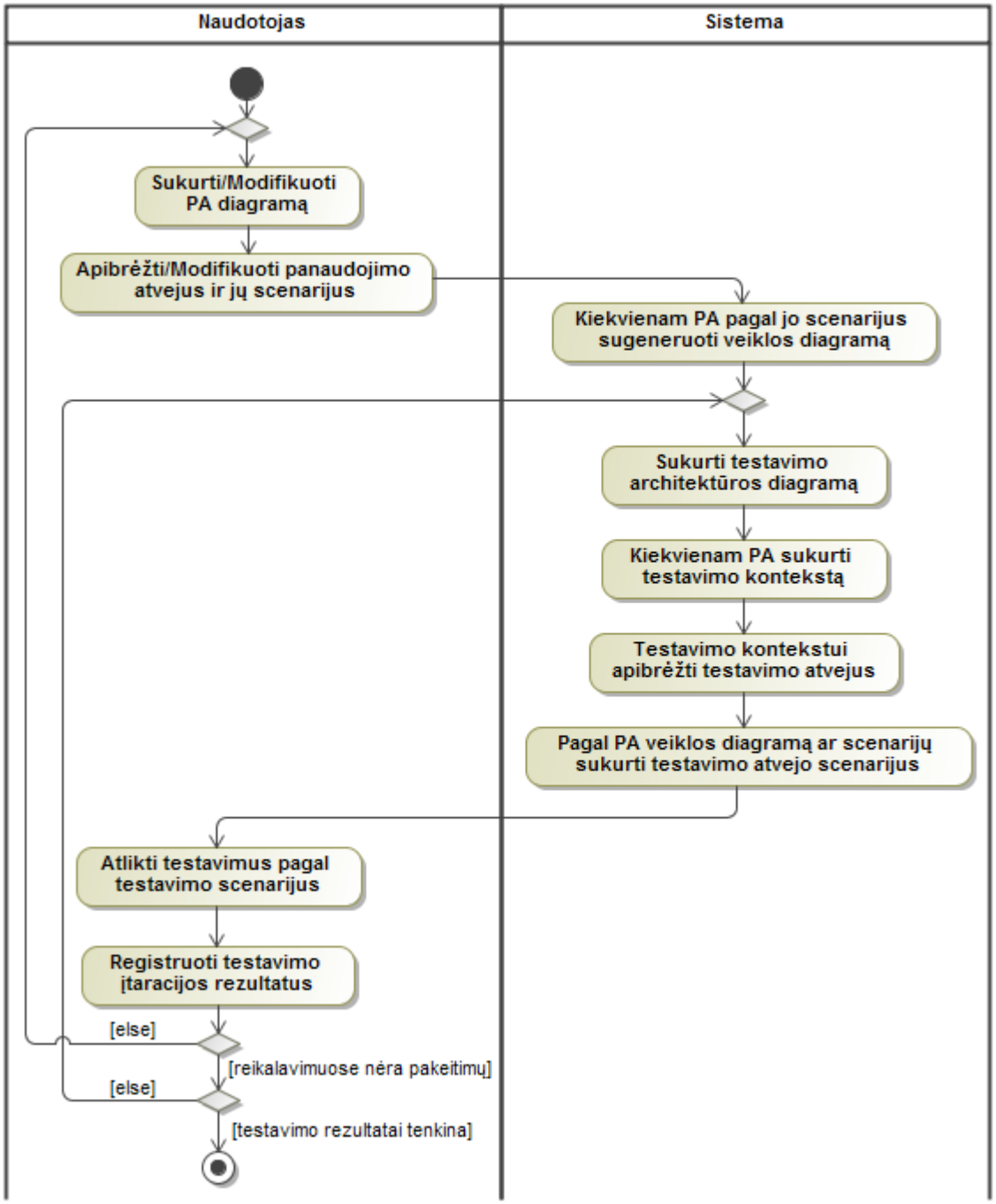
1.5. Siekiamo sprendimo apibrėžimas

1.10 pav. Panaudojimo atvejai išpildantys darbo tikslą



1.11 pav. Koncepcinis dalykinės srities esybių modelis

1.12 paveikslėlyje pavaizduotas siekiamas sprendimas, palyginus su esama situacija (**1.4** pav.) matyti kaip pasikeis veiklos procesas ir kuri dalis bus automatizuota. Automatizuoti procesai bus šie: sukurti testavimo architektūros diagramą, kiekvienam PA sukurti testavimo kontekstą, kiekvienam testavimo kontekstui apibrėžti testavimo atvejus, pagal PA veiklos diagramą sukurti testavimo atvejų scenarijus.



1.12 pav. Siekiamo veiklos proceso modelis

1.6. Analizės išvados

1. Programinės įrangos gyvavimo cikle testavimo atvejų nustatymas ir jų dokumentavimas yra nepakankamai išplėtotas etapas, lyginant su kitais. Dėl neautomatizuoto testavimo atvejų generavimo šis etapas reikalauja didelių laiko sąnaudų bei įneša žmogiškojo faktoriaus klaidų tikimybę. Automatizavus šį procesą sumažėtų klaidų kiekis, sutrumpėtų testavimo laikas ir taip sumažėto testavimo etapo kaina, būtų užtikrintas rezultatų vientisumas ir atsekamumas.
2. Atlikus *Enterprise Architect CASE* įrankio analizę paaiškėjo, jog tai gerai apmastytas ir kokybiškas automatinio testavimo atvejų generavimo metodas. Yra parametrai, kurių pilnai pakanka testavimo atvejams ir jų dabartinei būsenai aprašyti. Tad dalis šių parametrų bus panaudota kuriamame sprendime. Taip pat realizuotas patogus būdas paruošti panaudojimo atvejus testavimo atvejų generavimui, t. y., struktūrizuoto tekstinio

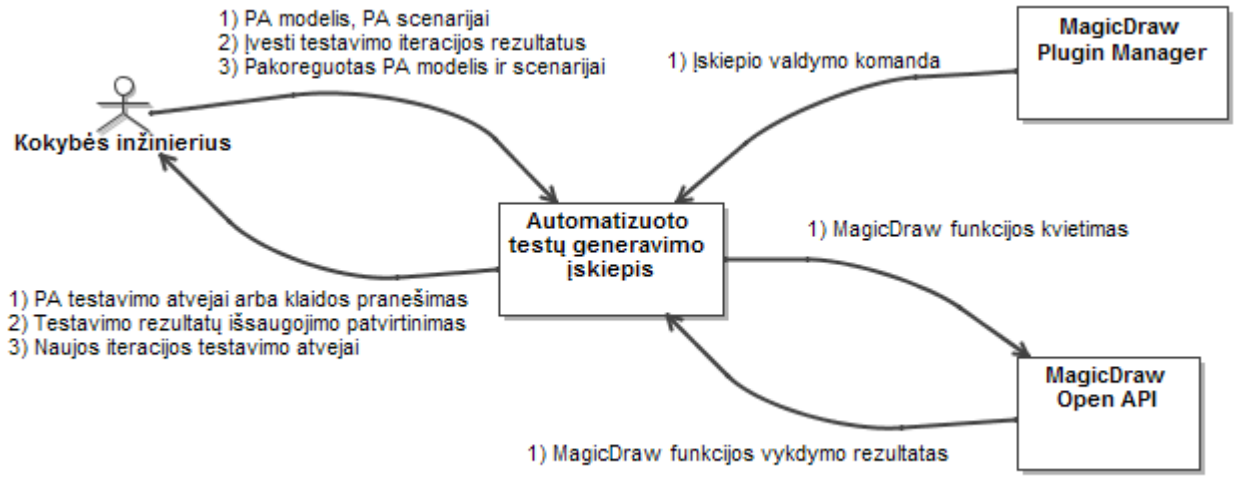
panaudojimo atvejų scenarijaus aprašymas. Vieninteliai šio įrankio trūkumai būtų tie, jog nėra pagrįstas standartu ir neturi testavimus apibendrinančių lentelių ar ataskaitų.

3. Tuo tarpu [1] straipsnyje aprašytas metodas yra sudėtingesnis ir tai tėra tik prototipas. Sunku spręsti kokia šio metodo kokybė, kai negalima išbandyti realiai. Neaišku, ar jis palaiko ir kokius testavimo atvejų parametrus, ar yra testavimus apibendrinančių lentelių ar ataskaitų.
4. *IBM* siūlomas teorinis metodas yra geras tuo, kad ne tik sukuriami pagrindiniai ir alternatyvūs testavimo scenarijai, jų parametrai, bet ir apibrėžiami konkrečios parametrų reikšmės. Kadangi naudinga testuojant apibrėžti testavimo duomenis kuriamame sprendime tikslinga sudaryti sąlygas juos apibrėžti.
5. Atlikus *UTP* analizę, paaiškėjo jog šį profilį tikslinga naudoti automatizuojant testavimų atvejų generavimą dėl jo išbaigtumo, pripažinimo ir dėl to, kad tai yra *OMG* standartas. Norint sudaryti galimybę fiksuoti testavimo etapo rezultatus, bei reikalingus duomenis testavimui vykdyti, reikia sukurti profilį, kuris išplėstų *UTP* profiliui priklausantį *TestCase* stereotipą papildomais parametrais.
6. Norint įgyvendinti darbo tikslus reikalingas *CASE* įrankis, kuriame yra realizuotas *UTP* profilis, suteikiantis galimybę išplėsti profilį ir sukurti naują funkcionalumą.
7. Atlikus *MagicDraw* įrankio analizę, paaiškėjo, kad jis griežtai laikosi standartų, jame yra realizuotas *UTP* ir sudaryta galimybė išplėsti įrankio funkcionalumą, kuriant įskiepius ir profilius. Tad automatizuotą testavimo atvejų generavimą tikslinga realizuoti šiame įrankyje.

2. SPRENDIMO REIKALAVIMŲ SPECIFIKACIJA IR PROJEKTAS, FORMALUS APRAŠAS

2.1. Reikalavimų specifikacija

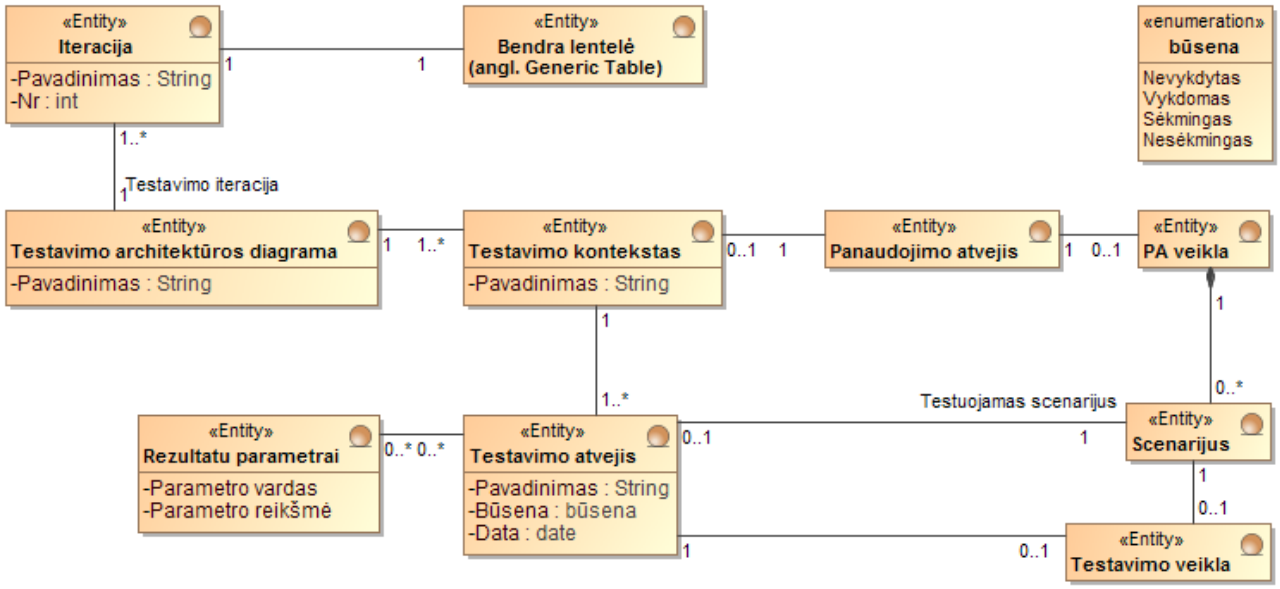
2.1 paveiksle pavaizduota kontekstinė diagrama, kurioje atsispindi įskiepio, *MagicDraw* ir naudotojo tarpusavio sąveika.



2.1 pav. MagicDraw, įskiepio ir naudotojo tarpusavio sąveikos kontekstinė diagrama

2.2. Dalykinės srities modelis

2.2 paveikslėlyje pavaizduotas dalykinės srities modelis, kuriame matyti kokie komponentai įeis į automatizuojamą veiklos proceso dalį.

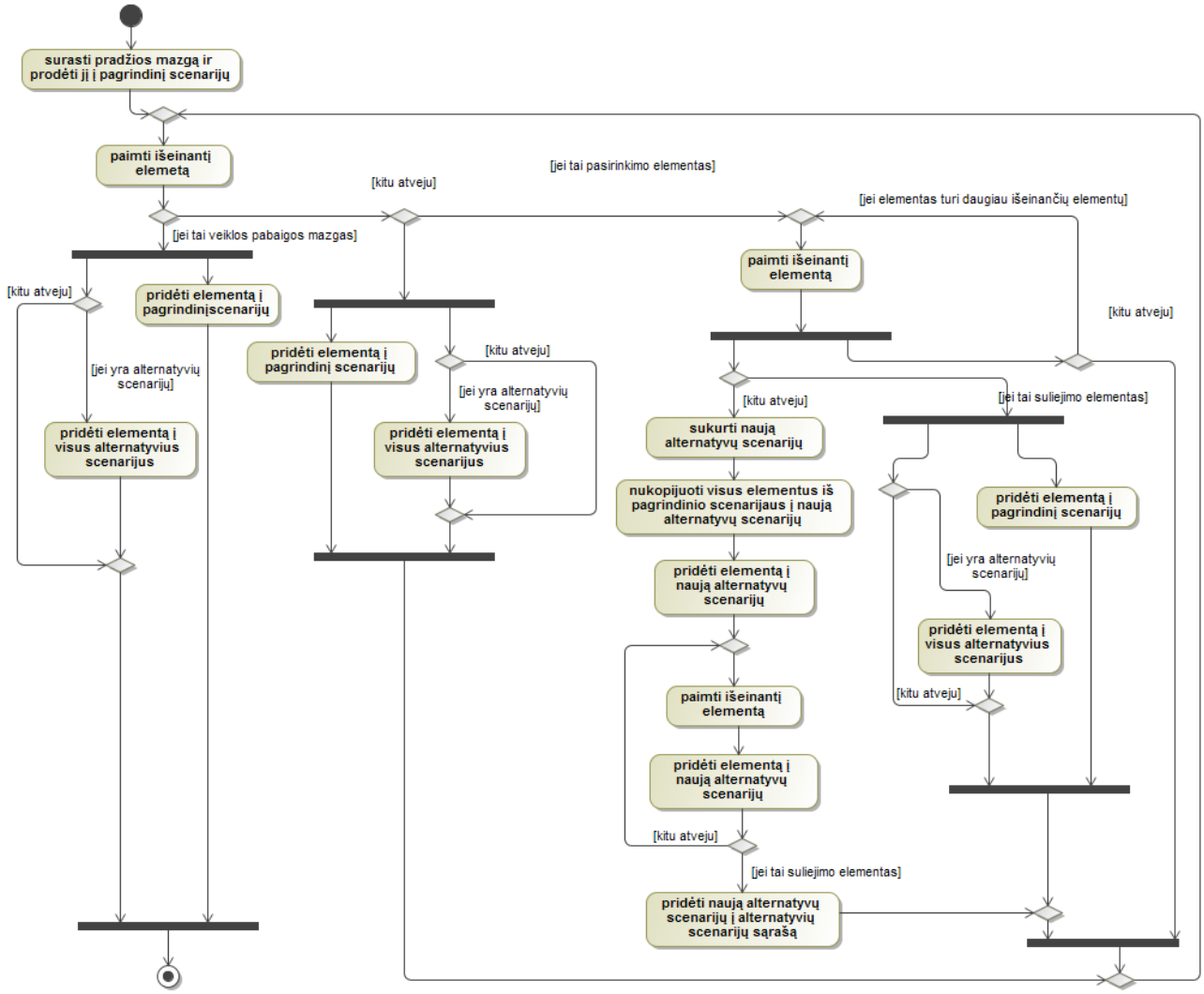


2.2 pav. Automatizuoto testavimo atvejų generavimo modelis

2.3. Formalus sprendimo aprašas

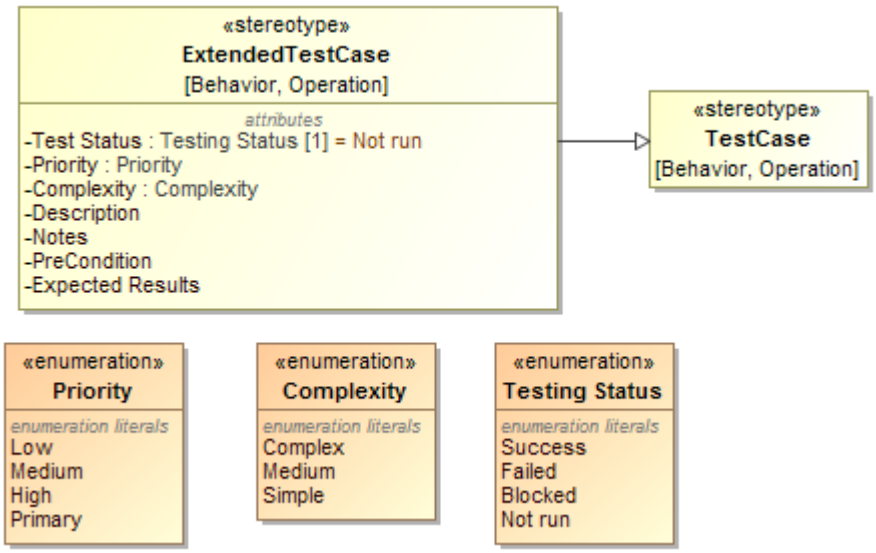
Įrankį formaliai aprašo išplėstas profilis ir testavimo atvejų analizavimo algoritmas.

Konkretaus panaudojimo atvejų pagrindinis, alternatyvūs ir išimties scenarijai atvaizduojami vienoje veiklos diagramoje, kuri *MagicDraw* įrankyje sugeneruojama iš tekstinio scenarijų aprašo. Veiklos diagramoje apibrėžtiems pagrindiniams ir alternatyviems scenarijams atskirti sukurtas algoritmas, kuris pavaizduotas 2.3 paveikslėlyje. Išimties scenarijai neįtraukti į algoritmą, nes jie atvaizduojami atskiruose *Structured Activity Node* elementuose.



2.3 pav. Pagrindinio ir alternatyvaus scenarijaus analizavimo algoritmo veiklos diagrama

Tam kad būtų galima žymėti kiekvienos iteracijos testavimo rezultatus sukurtas profilis – *Extended UML Testing Profile* (2.4 paveikslėlį), kuris išplečia standartu pripažintą *UML Testing Prifile* profilį. Sukurtas profilis papildo testavimo atvejį atributais, kurie detalizuoti 2.5 lentelėje.



2.4 pav. *Extended UML Testing Profile* struktūra

2.5 lentelė. *Extended UML Testing Profile* atributai

Pavadinimas	**Aprašymas**	**Duomenų tipas**
Test Status	Testavimo statusas	Pasirinkimo sąrašas, kurį sudaro šie elementai: sėkmingas (angl. *Success*), nesėkmingas (angl. *Failed*), blokuotas (angl. *Blocked*), nevykdytas (angl. *Not run*)
Priority	Prioritetas, nusako kokios svarbos yra testavimo atvejis	Pasirinkimo sąrašas, kurį sudaro šie elementai: žemo (angl. *Low*), vidutinio (angl. *Medium*), aukšto (angl. *High*), pirminis (angl. *Primary*)
Complexity	Testavimo atvejo sudėtingumas	Pasirinkimo sąrašas, kurį sudaro šie elementai: sudėtingas (angl. *Complex*), vidutinio sudėtingumo (angl. *Medium*), paprastas (angl. *Simple*)
Description	Pateikiamas testavimo atvejo aprašymas	Tekstas
Notes	Pateikiamos testavimo atvejo pastabos	Tekstas
PreCondition	Aprašomos kokios sąlygos reikalingos norint įvykdyti testavimo atvejį	Tekstas
Expected Results	Aprašomi kokie rezultatai laukiami įvykdžius testavimo atvejį	Tekstas

2.4. Reikalavimų apibendrinimas

Kuriamam sprendimui iškelti tokie reikalavimai:

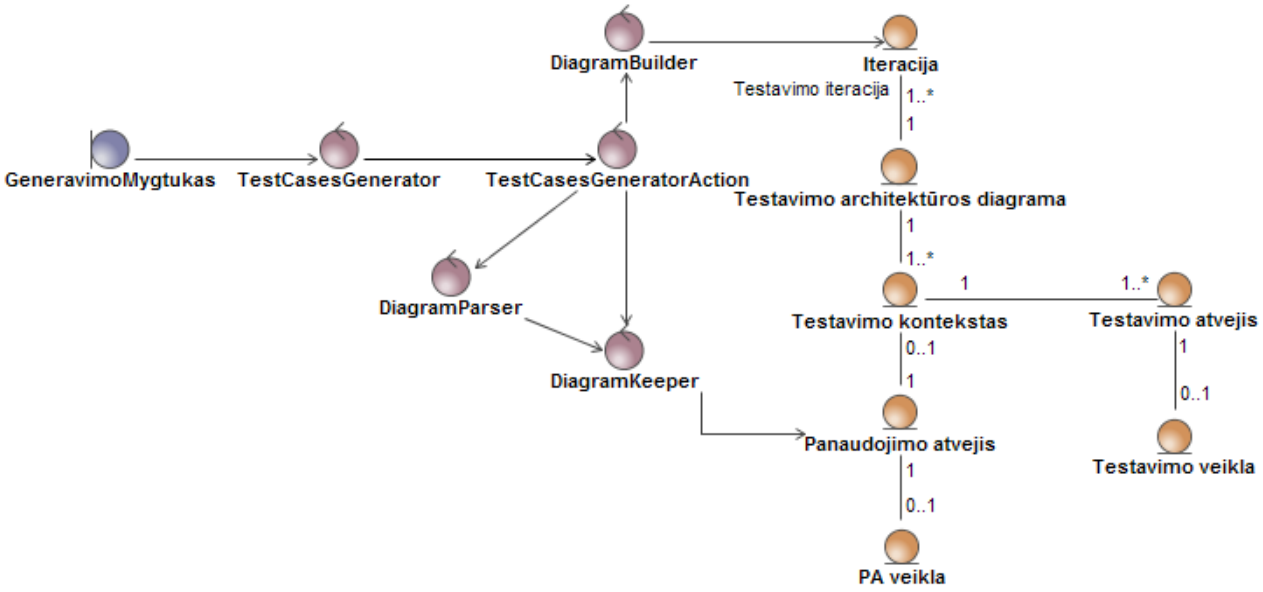
1. Sukurti profilį, kuris išplėstų *UTP* profiliui priklausantį *TestCase* stereotipą papildomais parametrais testavimo duomenims fiksuoti.
2. Sukurti įskiepį, kuris automatiškai generuotų testavimo atvejus iš panaudojimo atvejų ir jų scenarijų.
3. Kiekvieną testavimo atvejų generavimą grupuoti į iteracijas, kurias nuosekliai numeruoti.
4. Kiekvienoje iteracijoje sugeneruoti apibendrinimo lentelę, kurioje pateikiami visi testavimo atvejai su parametrais, testavimo duomenims fiksuoti.
5. Generuojant naują testavimo iteraciją perkelti testavimo duomenis užfiksuotus prieš tai buvusioje iteracijoje į naujai kuriamą.

3. AUTOMATIZUOTO TESTAVIMO ATVEJŲ GENERAVIMO SPRENDIMO PROJEKTAS

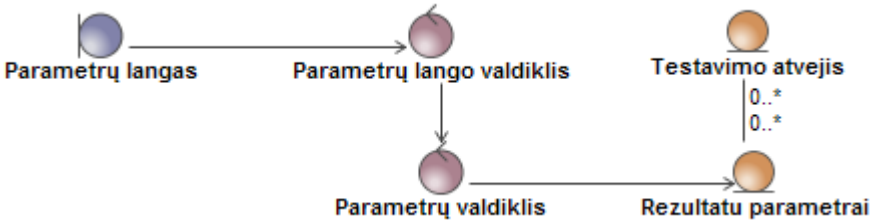
3.1. Sistemos architektūra

3.1.1. Reikalavimų analizė

Analizės (robastiškumo) diagramos parodo, kokios vartotojo sąsajos, programinės ir esybių (duomenų) klasės turi būti sukurtos, norint realizuoti kompiuterizuojamus panaudojimo atvejus. 3.1 ir 3.2 paveikslėliuose pavaizduoti kompiuterizuojamų panaudojimo atvejų analizės (robastiškumo) diagramos.



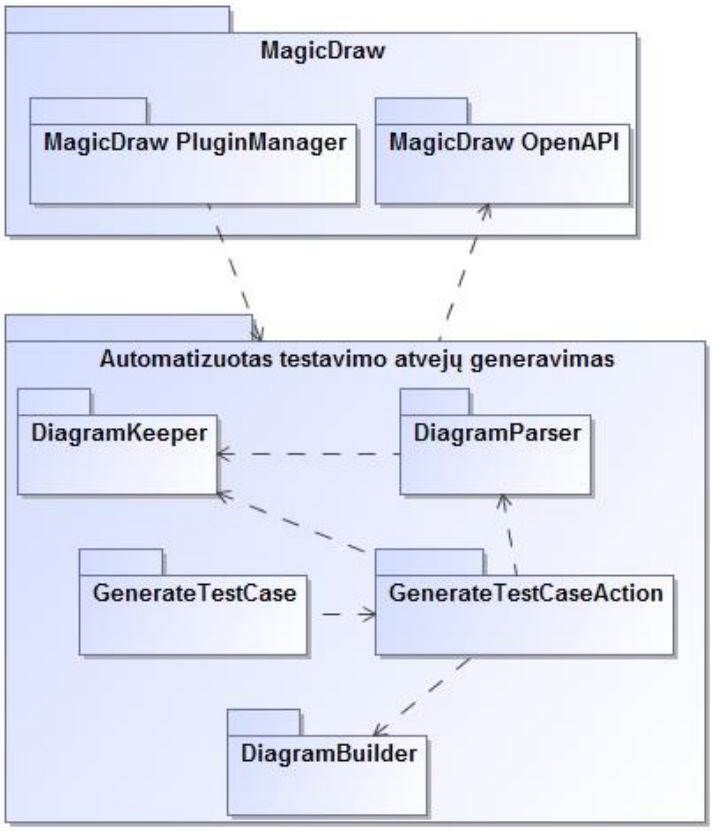
3.1 pav. Generuoti testavimo atvejus



3.2 pav. Peržiūrėti / registruoti testavimo rezultatus

3.1.2. Loginė visos sistemos architektūra

3.3 paveiksle pateikta kuriamo įskiepio ir bendravimo su *MagicDraw* įrankiu loginė architektūra.



3.3 pav. Įskiepio loginė architektūra

3.1.3. Vartotojo sąsajos klasių modelis

Specifinės vartotojo sąsajos nebus, bus naudojama *MagicDraw* naudotojo sąsaja. Bus papildyta tik funkcionalumu, kurį naudotojas galės įvykdyti paspaudus atitinkamą mygtuką.

3.2. Detalus projektas

3.4 paveikslėlyje pavaizduota suprojektuota klasių diagrama, t. y. kokios klasės sudarys įskiepį. Taip pat kokia klasė kokius atributus ir operacijas turės. 3.1 - 3.38 lentelėse pateikti kiekvienos klasės operacijų aprašymai.

3.1 lentelė. *init* operacija

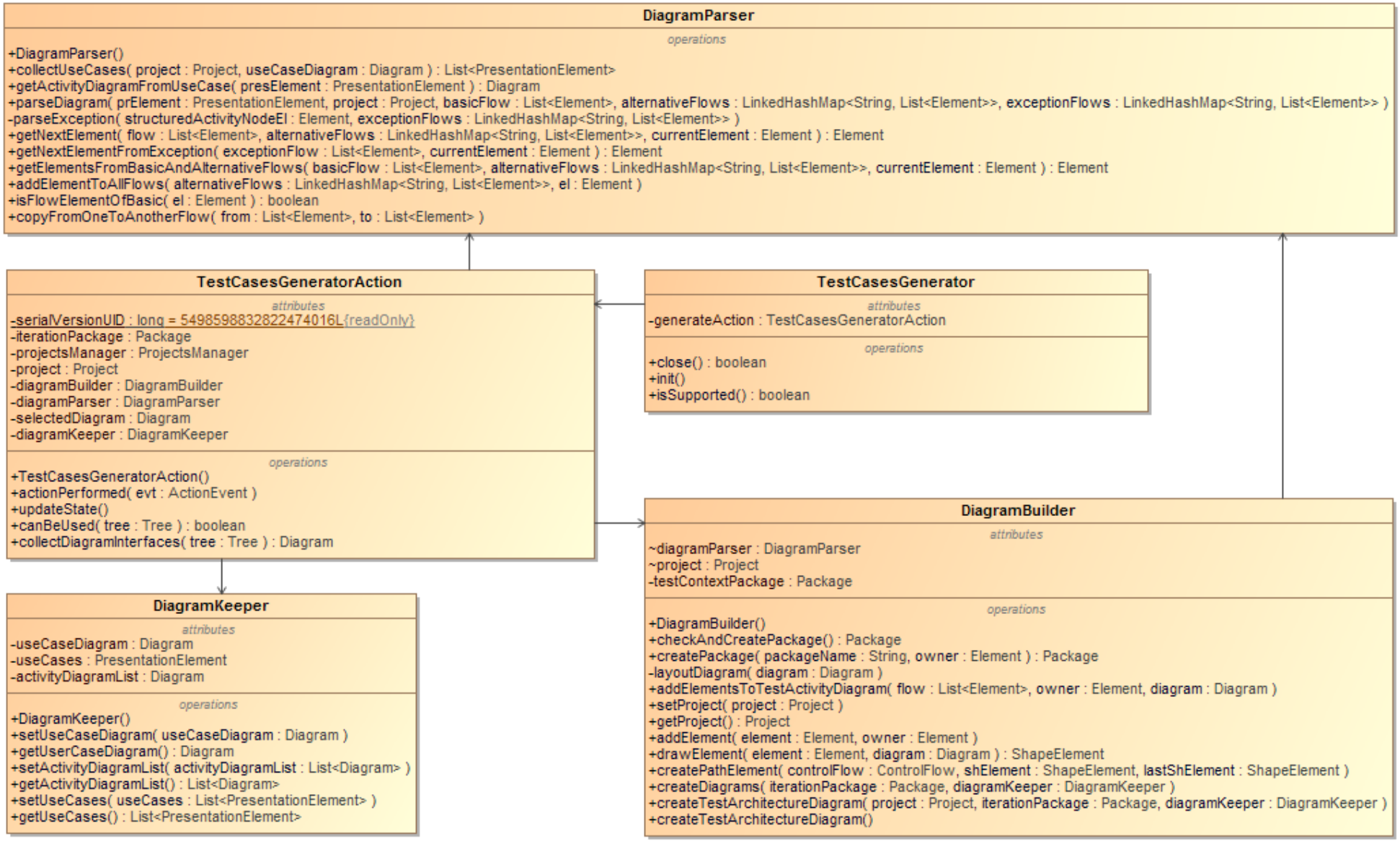
Operacijos pavadinimas	init
Operacijos paskirtis	Operacija iškviečiamas inicijuojant įskiepį i MagicDraw. Kviečiamas tik jei metodas isSuported gražina loginę reikšmę true. Apibrėžia įskiepio funkcionalumą.
Paduodami parametrai	-
Gražinama reikšmė	-

3.2 lentelė *isSuported* operacija

Operacijos pavadinimas	isSuported
Operacijos paskirtis	Nusako ar įskiepis bus įkeltas į MagicDraw ar ne.
Paduodami parametrai	-
Gražinama reikšmė	Loginė reikšmė (true / false)

3.3 lentelė. *close* operacija

Operacijos pavadinimas	close
Operacijos paskirtis	Apibrėžia veiksmus uždarant įskiepį.
Paduodami parametrai	-
Gražinama reikšmė	Loginė reikšmė (true / false)



3.4 pav. Įskiepio klasių diagrama

3.4 lentelė. *TestCasesGeneratorAction* operacija

Operacijos pavadinimas	TestCasesGeneratorAction
Operacijos paskirtis	TestCasesGeneratorAction klasės konstruktorius
Paduodami parametrai	-
Gražinama reikšmė	-

3.5 lentelė. *actionPerformed* operacija

Operacijos pavadinimas	actionPerformed
Operacijos paskirtis	Apibrėžia „Generate Test Cases“ mygtuko funkcionalumą.
Paduodami parametrai	ActionEvent
Gražinama reikšmė	-

3.6 lentelė. *updateState* operacija

Operacijos pavadinimas	updateState
Operacijos paskirtis	Apibrėžia „Generate Test Cases“ mygtuko aktyvumą.
Paduodami parametrai	-
Gražinama reikšmė	-

3.7 lentelė. *canBeUsed* operacija

Operacijos pavadinimas	canBeUsed
Operacijos paskirtis	Nusako ar „Generate Test Cases“ mygtukas turi būti aktyvus ar ne.
Paduodami parametrai	Tree
Gražinama reikšmė	Loginė reikšmė (true / false)

3.8 lentelė. *collectDiagramInterfaces* operacija

Operacijos pavadinimas	collectDiagramInterfaces
Operacijos paskirtis	Nusako ar pasirinktas elementas yra panaudojimo atvejų diagrama
Paduodami parametrai	Tree
Gražinama reikšmė	Diagrama

3.9 lentelė. *DiagramReader* operacija

Operacijos pavadinimas	DiagramReader
Operacijos paskirtis	DiagramReader klasės konstruktorius
Paduodami parametrai	-
Gražinama reikšmė	-

3.10 lentelė. *setUseCaseDiagram* operacija

Operacijos pavadinimas	setUseCaseDiagram
Operacijos paskirtis	Priskirti lokaliam kintamajam reikšmę iš išorės.
Paduodami parametrai	Panaudojimo atvejų diagrama
Gražinama reikšmė	-

3.11 lentelė. *getUseCaseDiagram* operacija

Operacijos pavadinimas	getUseCaseDiagram
Operacijos paskirtis	Prieiti prie lokalaus kintamojo iš išorės.
Paduodami parametrai	-
Gražinama reikšmė	Panaudojimo atvejų diagrama

3.12 lentelė. *setActivityDiagramList* operacija

Operacijos pavadinimas	setActivityDiagramList
Operacijos paskirtis	Priskirti lokaliam kintamajam reikšmę iš išorės.
Paduodami parametrai	Veiklos diagramų sąrašas
Gražinama reikšmė	-

3.13 lentelė. *getActivityDiagramList* operacija

Operacijos pavadinimas	getActivityDiagramList
Operacijos paskirtis	Prieiti prie lokalaus kintamojo iš išorės.
Paduodami parametrai	-
Gražinama reikšmė	Veiklos diagramų sąrašas

3.14 lentelė. *setUseCasesList* operacija

Operacijos pavadinimas	setUseCasesList
Operacijos paskirtis	Priskirti lokaliam kintamajam reikšmę iš išorės.
Paduodami parametrai	Panaudojimo atvejų sąrašas
Gražinama reikšmė	-

3.15 lentelė. *getUseCasesList* operacija

Operacijos pavadinimas	getUseCasesList
Operacijos paskirtis	Prieiti prie lokalaus kintamojo iš išorės.
Paduodami parametrai	-
Gražinama reikšmė	Panaudojimo atvejų sąrašas

3.16 lentelė. *DiagramParser* operacija

Operacijos pavadinimas	DiagramParser
Operacijos paskirtis	DiagramParser klasės konstruktorius
Paduodami parametrai	-
Gražinama reikšmė	-

3.17 lentelė. *collectUsesCases* operacija

Operacijos pavadinimas	collectUseCases
Operacijos paskirtis	Išrinkti panaudojimo atvejus iš panaudojimo atvejų diagramos
Paduodami parametrai	Panaudojimo atvejų diagrama
Gražinama reikšmė	Panaudojimo atvejų sąrašas

3.18 lentelė. *getActivityDiagramFromUseCase* operacija

Operacijos pavadinimas	getActivityDiagramFromUseCase
Operacijos paskirtis	Iš panaudojimo atvejo išgaunamas scenarijus (panaudojimo atvejų diagrama)
Paduodami parametrai	Panaudojimo atvejis
Gražinama reikšmė	Veiklos diagrama

3.19 lentelė. *parseDiagram* operacija

Operacijos pavadinimas	parseDiagram
Operacijos paskirtis	Valdyti kitiem procesams, kurie susiję su diagramos analizavimu
Paduodami parametrai	Panaudojimo atvejis, projektas, pagrindinio scenarijaus, alternatyvių ir išimties scenarijų sąrašai
Gražinama reikšmė	-

3.20 lentelė. *parseException* operacija

Operacijos pavadinimas	parseException
Operacijos paskirtis	Išgauti išsities scenarijaus elementus
Paduodami parametrai	Struktūrizuotos veiklos mazgo elementas, išimties scenarijų sąrašas
Gražinama reikšmė	Elementų sąrašas

3.21 lentelė. *getNextElement* operacija

Operacijos pavadinimas	getNextElement
Operacijos paskirtis	Išgauti iš elemento po jo einantį sekantį elementą
Paduodami parametrai	Pagrindinio ir alternatyvių scenarijų sąrašai, elementas
Gražinama reikšmė	Elementas

3.22 lentelė. *getNextElementFromException* operacija

Operacijos pavadinimas	getNextElementFromException
Operacijos paskirtis	Išgauti iš elemento po jo einantį sekantį elementą išimties scenarijuje
Paduodami parametrai	Išimties scenarijų sąrašai, elementas
Gražinama reikšmė	Elementas

3.23 lentelė. *getElementsFromBasicAndAlternativesFlow* operacija

Operacijos pavadinimas	getElementsFromBasicAndAlternativeFlow
Operacijos paskirtis	Išgauti elementams ir atskirti kurie yra pagrindinio scenarijaus, kurie alternatyvaus scenarijaus
Paduodami parametrai	Pagrindinio ir alternatyvių scenarijų sąrašai, pasirinkimo elementas
Gražinama reikšmė	Elementas

3.24 lentelė. *addElementsToAllFlows* operacija

Operacijos pavadinimas	addElementsToAllFlows
Operacijos paskirtis	Pridėti elementą į visus alternatyvius scenarijus
Paduodami parametrai	Alternatyvių scenarijų sąrašas ir elementas
Gražinama reikšmė	-

3.25 lentelė. *isFlowELementOfBasic* operacija

Operacijos pavadinimas	isFlowElementOfBasic
Operacijos paskirtis	Atskirti ar elementas priklauso pagrindiniam ar alternatyviam scenarijui
Paduodami parametrai	Elementas
Gražinama reikšmė	Loginė reikšmė (true / false)

3.26 lentelė. *copyFromOneTOAnotherFlow* operacija

Operacijos pavadinimas	copyFromOneToAnotherFlow
Operacijos paskirtis	Nukopijuoti elementus iš vieno scenarijaus į kitą
Paduodami parametrai	Scenarijai iš kurio ir į kurį vyksta kopijavimas
Gražinama reikšmė	-

3.27 lentelė. *DiagramBuilder* operacija

Operacijos pavadinimas	DiagramBuilder
Operacijos paskirtis	DiagramBuilder klasės konstruktorius
Paduodami parametrai	-
Gražinama reikšmė	-

3.28 lentelė. *checkAndCreatePackage* operacija

Operacijos pavadinimas	checkAndCreatePackage
Operacijos paskirtis	Analizuoja ar projekte yra sukurtas testavimo paketas ir jei taip, analizuoja kiek iteracijų buvo sugeneruota.
Paduodami parametrai	-
Gražinama reikšmė	Iteracijos aplankalas

3.29 lentelė. *createPackage* operacija

Operacijos pavadinimas	createPackage
Operacijos paskirtis	Sukurti aplankalams
Paduodami parametrai	Aplankalo pavadinimas ir savininkas
Gražinama reikšmė	Aplankalas

3.30 lentelė. *layoutDiagram* operacija

Operacijos pavadinimas	layoutDiagram
Operacijos paskirtis	Išlygiuoti diagramos elementus
Paduodami parametrai	Diagrama
Gražinama reikšmė	-

3.31 lentelė. *addElementsToTestActivityDiagram* operacija

Operacijos pavadinimas	addElementsToTestActivityDiagram
Operacijos paskirtis	Užpildyti testavimo veiklų diagramą elementais
Paduodami parametrai	Scenarijus, savininkas ir diagrama
Gražinama reikšmė	-

3.32 lentelė. *setProject* operacija

Operacijos pavadinimas	setProject
Operacijos paskirtis	Priskirti lokaliam kintamajam reikšmę iš išorės.
Paduodami parametrai	Projektas
Gražinama reikšmė	-

3.33 lentelė. *getProject* operacija

Operacijos pavadinimas	getProject
Operacijos paskirtis	Prieiti prie lokalaus kintamojo iš išorės.
Paduodami parametrai	-
Gražinama reikšmė	Projektas

3.34 lentelė. *addElement* operacija

Operacijos pavadinimas	addElement
Operacijos paskirtis	Pridėti elementą į projektą
Paduodami parametrai	Elementas ir savininkas
Gražinama reikšmė	-

3.35 lentelė. *drawElement* operacija

Operacijos pavadinimas	drawElement
Operacijos paskirtis	Nupiešti elementą diagramoje
Paduodami parametrai	Elementas ir diagrama
Gražinama reikšmė	-

3.36 lentelė. *createPathElement* operacija

Operacijos pavadinimas	createPathElement
Operacijos paskirtis	Nupiešti ryšį tarp elementų
Paduodami parametrai	Du elementai
Gražinama reikšmė	-

3.37 lentelė. *createDiagram* operacija

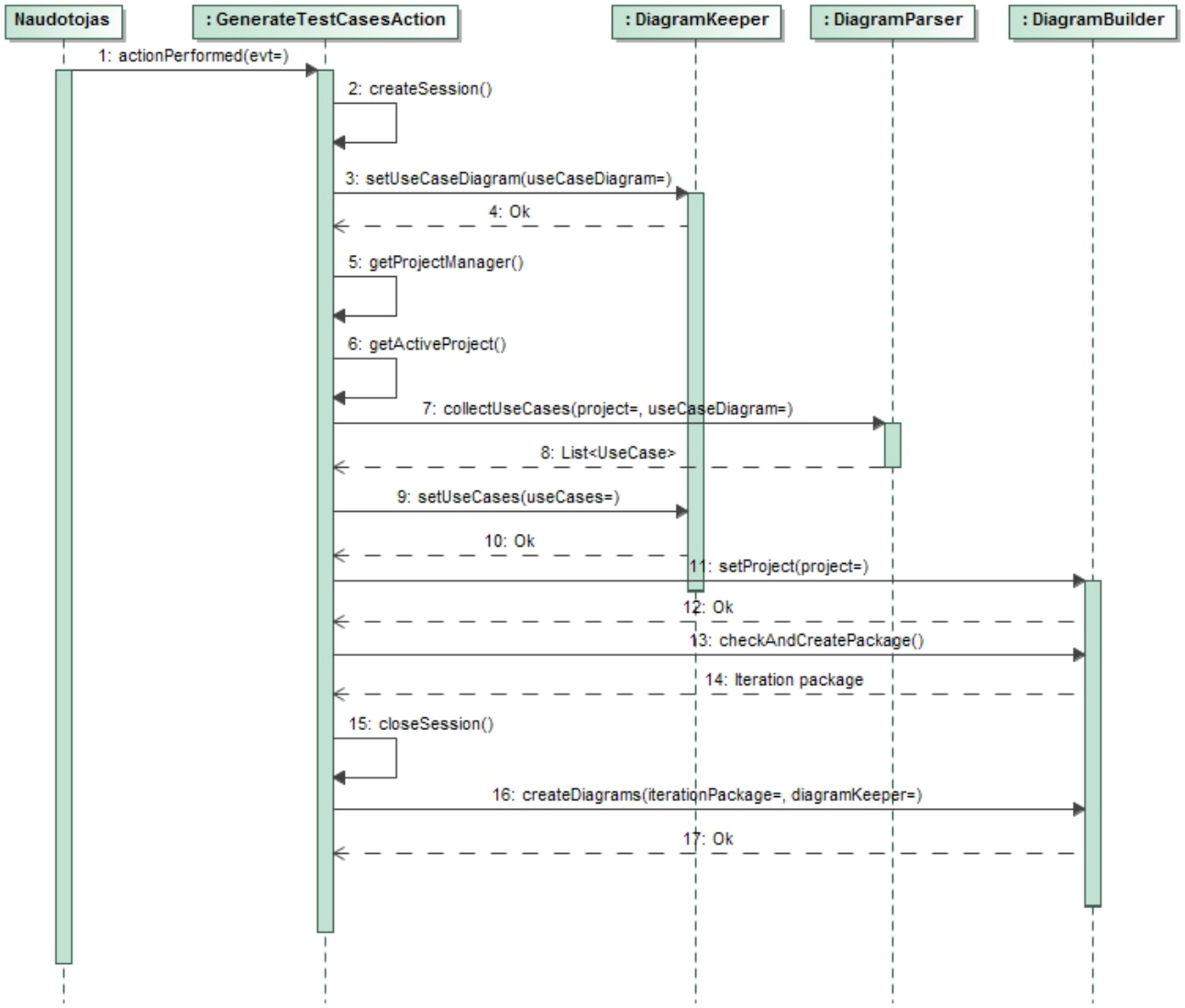
Operacijos pavadinimas	createDiagram
Operacijos paskirtis	Valdyti procesus, kurie susiję su iteracijos ir testavimo architektūros modelio sukūrimu
Paduodami parametrai	PA sąrašas, pagrindinis, alternatyvūs ir išimties scenarijai su elementų sąrašu, sudarančių scenarijų, bei PA identifikatoriumi, nurodančių kuriam PA priklauso scenarijus
Gražinama reikšmė	-

3.38 lentelė. *createTestArchitectureDiagram* operacija

Operacijos pavadinimas	createTestArchitectureDiagram
Operacijos paskirtis	Sukurti testavimo architektūros diagramą su testavimo atvejais, bei jų scenarijais
Paduodami parametrai	Iteracijos paketas, projektas, DiagramKeeper klasės kintamasis
Gražinama reikšmė	-

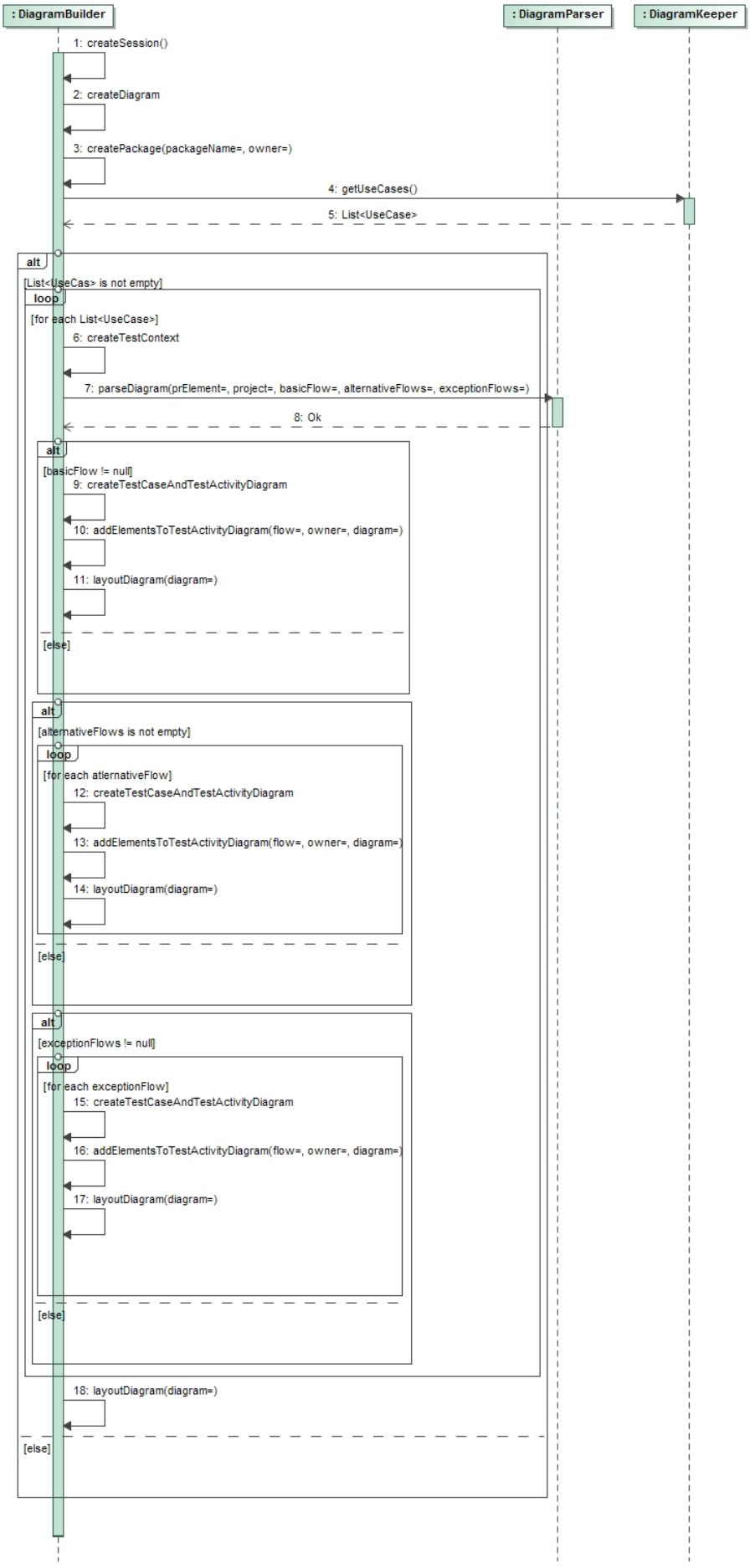
3.3. Sistemos elgsenos modelis

3.5 paveikslėlyje pavaizduota „*GenerateTestCasesAction*“ klasės „*actionPerformed*“ operacija, kurioje atsispindi abstrakti veiksmų eiga aktyvavus testavimo atvejų generavimą.

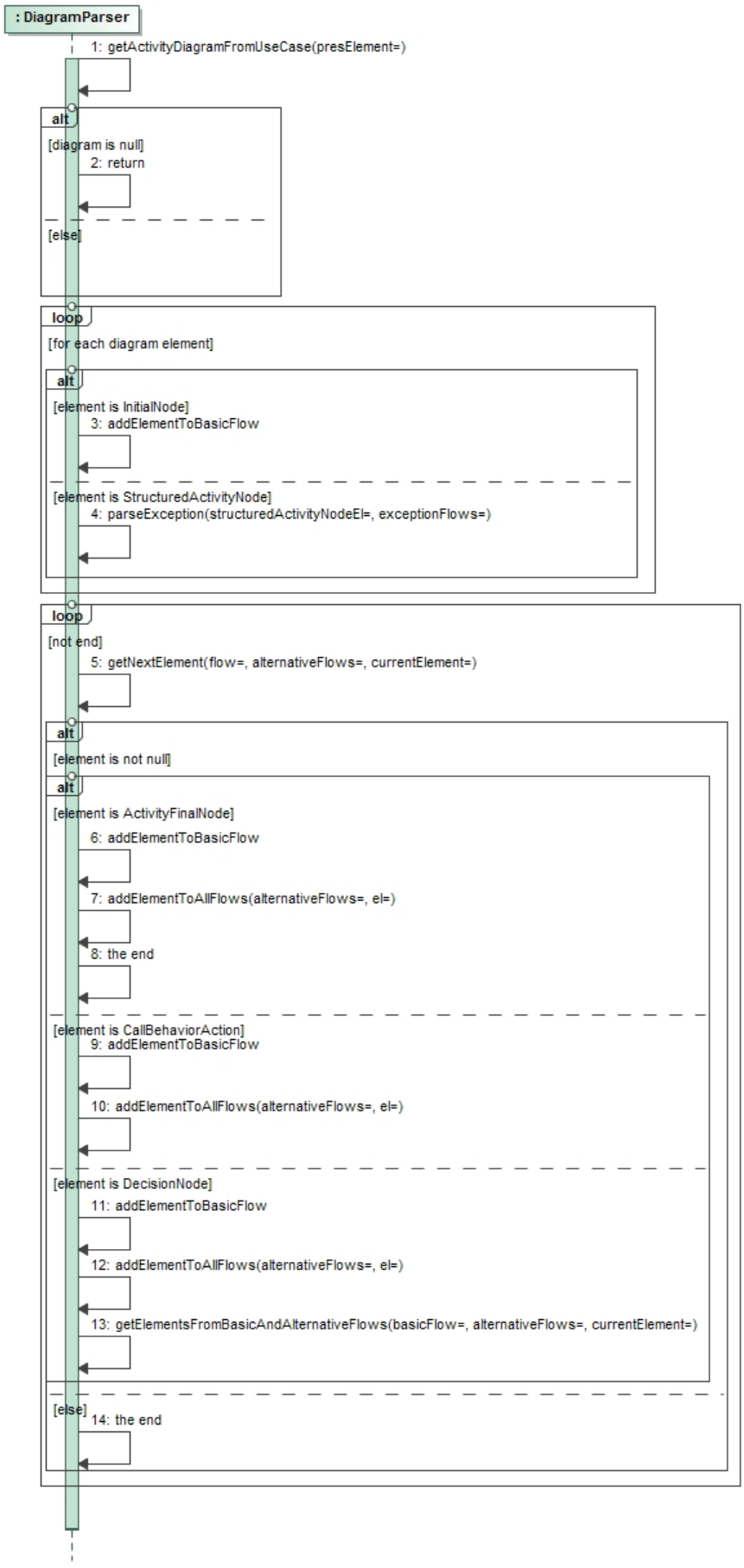


3.5 pav. *actionPerformed* operacijos sekų diagrama

3.6 paveikslėlyje detalizuota „*createDiagram*“ operacija, t. y. kokios kitos operacijos ir kokia seką vykdomos, o 3.7 paveikslėlyje detalizuota „*parseDiagram*“ operacijos logika.

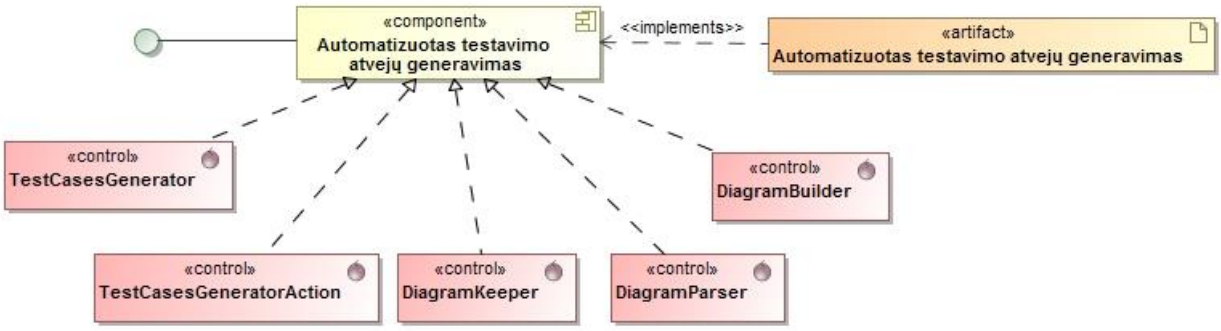


3.6 pav. *createDiagrma* operacijos sekų diagrama

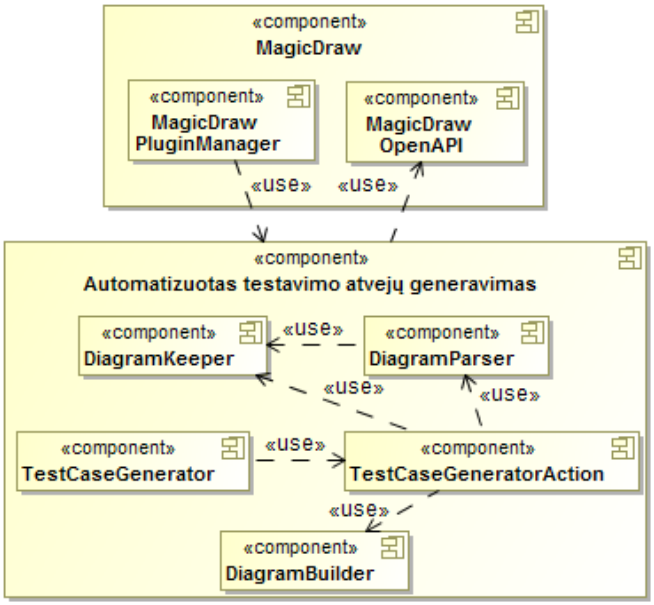


3.7 pav. *parseDiagram* operacijos sekų diagrama

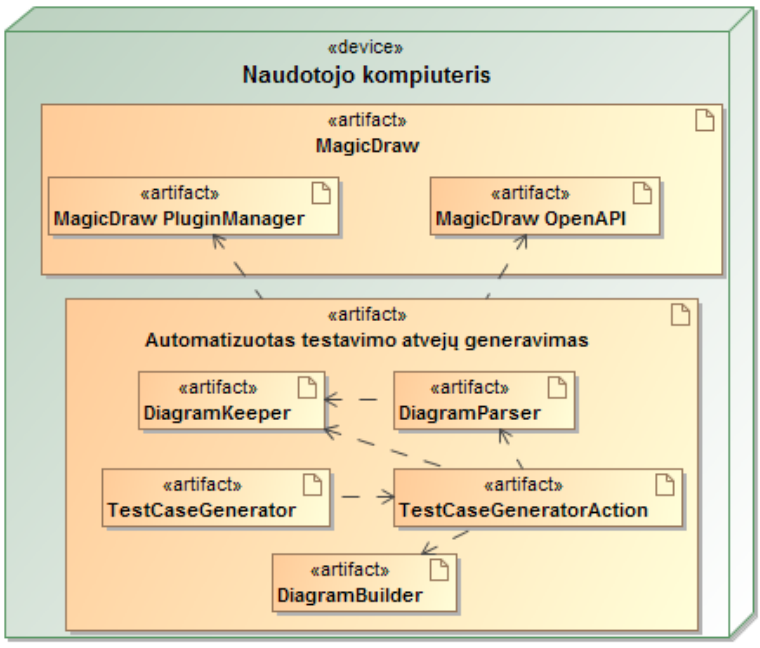
3.4. Realizacijos modelis



3.8 pav. Įskiepio realizacijos modelio komponentų diagrama



3.9 pav. Įskiepio fizinės architektūros komponentų diagrama



3.10 pav. Įskiepio diegimo modelis

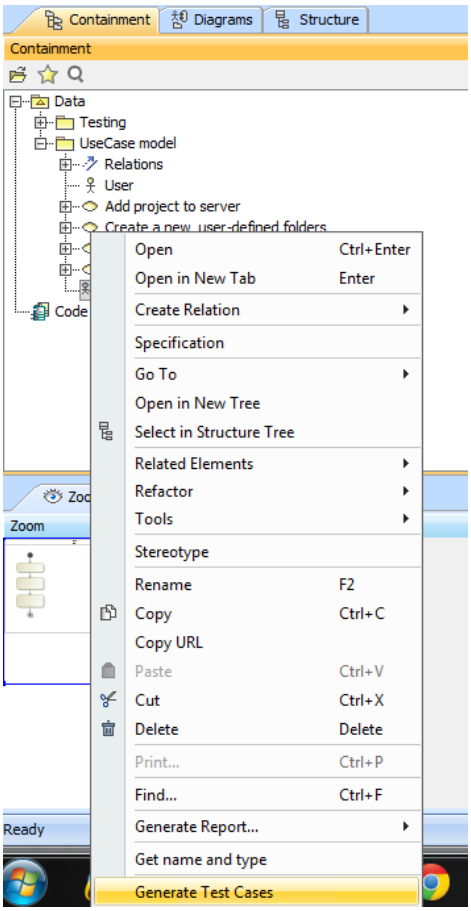
4. SPRENDIMO REALIZACIJA IR TESTAVIMAS

4.1. Sprendimo realizacijos ir veikimo aprašas

Įskiepis realizuotas atsižvelgiant į fizinę architektūra, kuri nurodyta 3.9 pav., bei kitus projektavimo etape suprojektuotus modelius ir aprašytas specifikacijas. Įskiepis realizuotas Eclipse atviro kodo įrankiu pagrįstu Java programavimo kalba.

Norint išplėsti *MagicDraw* funkcionalumą realizuoto įskiepio funkcionalumu reikia importuoti įskiepį į *MagicDraw*. Tai galima padaryt naudojant MagicDraw *Resourse/Plugin Manager*. Paspaudus „*Import*“ mygtuką ir nurodžius įskiepio paketą jis bus suimportuotas į MagicDraw direktoriją.

Suimportavus įskiepį galima naudotis jo išplečiamu funkcionalumu. Norint iškviesti automatizuotą testavimo atvejų generavimą, reikia „*Conteinment*“ kortelėje ant panaudojimo atvejų diagramos paspausti dešinį pelės klavišą ir pasirinkti „*Generate Test Cases*“ meniu punktą (4.1 pav.).



4.1 pav. Įskiepio automatizuoto testavimo atvejų generavimo meniu

4.2. Testavimo modelis, duomenys, rezultatai

Testavimas buvo vykdomas visos realizacijos etapo metu, po kiekvienos realizuotos įskiepio funkcionalumo dalies. Ištestuoti po mažą realizacijos dalį yra daug lengviau ir turint nuolat stabilią ir veikiančia realizacijos dalį mažesnė tikimybė jog prireiks kardinalių pakeitimų, jei kas nors veiks nekorektiškai. Priešingu nei realizuojant visą funkcionalumą ir po to testuojant.

Testavimas skirstomas į šiuos etapus:

- Iteracijos paketų analizė ir iteracijos paketo sukūrimas projekte
- Automatizuotas testavimo atvejų generavimas
- Testavimo rezultatų įvedimas ir išsaugojimas

Norint patikrinti ar realizacija atitinka nustatytus reikalavimus, reikia susidaryti testavimo modelį, kuris ir padės užtikrinti reikalavimų išpildymą. Apibrėžti testavimo atvejai, bei jų testavimo rezultatai, nurodyti žemiau pateiktuose lentelėse.

4.2.1. Iteracijos paketų analizės ir iteracijos paketo sukūrimas projekte testavimas

4.1 lentelė. Testavimo atvejis 1.1

ID	1.1
Trumpas aprašymas	Iteracijos aplanko sukūrimas pirmą kartą
Prieš sąlyga	Projekte neturi būti sukurto nė viena testavimo iteracija, bei apibendrinantis „Testing“ aplankas.
Vykdymo žingsniai	1. Panaudojimo atvejų diagramos pasirinkimas „*Containment*“ kortelėje 2. Pelės dešinio klavišo paspaudimas 3. „*Generate Test Cases*“ meniu pasirinkimas
Laukiami rezultatai	Pagrindiniame MagicDraw (Model Data) aplanke sukurtas aplankas pavadinimu „Testing“, o jame dar vienas aplankas – „Iteration_1“
Būsena	Įvykdytas – rezultatas teigiamas
Vykdymo rezultatas	Sukurtas „Testing“ aplankas, o jame „Iteration_1“

4.2 lentelė. Testavimo atvejis 1.2

ID	1.2
Trumpas aprašymas	Pirmos iteracijos aplanko sukūrimas kai egzistuoja „Testing“ aplankas
Prieš sąlyga	Projekte neturi būti sukurto nė viena testavimo iteracija, bet turi būti sukurtas „Testing“ aplankas.
Vykdymo žingsniai	1. Panaudojimo atvejų diagramos pasirinkimas „*Containment*“ kortelėje 2. Pelės dešinio klavišo paspaudimas 3. „*Generate Test Cases*“ meniu pasirinkimas
Laukiami rezultatai	Pagrindiniame MagicDraw „*Model Data*“ aplanke sukurtas aplankas pavadinimu „Testing“, o jame dar vienas aplankas – „Iteration_1“
Būsena	Įvykdytas – rezultatas teigiamas
Vykdymo rezultatas	Sukurtas „Testing“ aplankas, o jame „Iteration_1“

4.3 lentelė. Testavimo atvejis 1.3

ID	1.3
Trumpas aprašymas	N-osios iteracijos aplanko sukūrimas
Prieš sąlyga	Projekte egzistuoja dvi testavimo iteracijos
Vykdymo žingsniai	1. Panaudojimo atvejų diagramos pasirinkimas „*Containment*“ kortelėje 2. Pelės dešinio klavišo paspaudimas 3. „*Generate Test Cases*“ meniu pasirinkimas
Laukiami rezultatai	„Testing“ aplanke sukurtas aplankas pavadinimu „Iteration_3“
Būsena	Įvykdytas – rezultatas teigiamas
Vykdymo rezultatas	„Testing“ aplanke sukurtas „Iteration_3“ aplankas

4.2.2. Automatizuoto testavimo atvejų generavimo testavimas

4.4 lentelė. Testavimo atvejis 2.1

ID	2.1
Trumpas aprašymas	Testavimo architektūros diagramos sukūrimas
Prieš sąlyga	-
Vykdymo žingsniai	1. Panaudojimo atvejų diagramos pasirinkimas „*Containment*“ kortelėje 2. Pelės dešinio klavišo paspaudimas 3. „*Generate Test Cases*“ meniu pasirinkimas
Laukiami rezultatai	Iteracijos pakete sukuriamas testavimo architektūros diagrama
Būsena	Įvykdytas – rezultatas teigiamas
Vykdymo rezultatas	Iteracijos pakete sukurta testavimo architektūros diagrama

4.5 lentelė. Testavimo atvejis 2.2

ID	2.2
Trumpas aprašymas	Testavimo kontekstų sukūrimas
Prieš sąlyga	Panaudojimų atvejų diagramoje turi būti bent vienas panaudojimo atvejis
Vykdymo žingsniai	1. Panaudojimo atvejų diagramos pasirinkimas „*Containment*“ kortelėje 2. Pelės dešinio klavišo paspaudimas 3. „*Generate Test Cases*“ meniu pasirinkimas
Laukiami rezultatai	Kiekvienam panaudojimo atvejui testavimo architektūros diagramoje sukurtas testavimo kontekstas, kurio pavadinimas atitinka panaudojimo atvejo pavadinimą
Būsena	Įvykdytas – rezultatas sėkmingas
Vykdymo rezultatas	Kiekvienam panaudojimo atvejui testavimo architektūros diagramoje sukurtas testavimo kontekstas, kurio pavadinimas atitinka panaudojimo atvejo pavadinimą

4.6 lentelė. Testavimo atvejis 2.3

ID	2.3
Trumpas aprašymas	Ryšio tarp testavimo konteksto ir panaudojimo atvejo sukūrimas
Prieš sąlyga	Panaudojimų atvejų diagramoje turi būti bent vienas panaudojimo atvejis
Vykdymo žingsniai	1. Panaudojimo atvejų diagramos pasirinkimas „*Containment*“ kortelėje 2. Pelės dešinio klavišo paspaudimas 3. „*Generate Test Cases*“ meniu pasirinkimas
Laukiami rezultatai	Testavimo architektūros diagramoje atvaizduojami panaudojimo atvejai ir iš jam sukurto testavimo konteksto nubrėžtas priklausomybės ryšys.
Būsena	Įvykdytas – rezultatas sėkmingas
Vykdymo rezultatas	Testavimo architektūros diagramoje atvaizduojami panaudojimo atvejai ir iš jam sukurto testavimo konteksto nubrėžtas priklausomybės ryšys.

4.7 lentelė. Testavimo atvejis 2.4

ID	2.4
Trumpas aprašymas	Testavimo atvejų sukūrimas
Prieš sąlyga	Panaudojimų atvejų diagramoje vienas panaudojimo atvejis ir aprašytas jo scenarijus, kurį sudaro pagrindinis, 2 alternatyvūs ir išimties panaudojimo atvejai
Vykdymo žingsniai	1. Panaudojimo atvejų diagramos pasirinkimas „*Containment*“ kortelėje 2. Pelės dešinio klavišo paspaudimas 3. „*Generate Test Cases*“ meniu pasirinkimas
Laukiami rezultatai	Testavimo kontekste sukuriami vienas pagrindinis, du alternatyvūs ir vienas išimties testavimo atvejis.
Būsena	Įvykdytas – rezultatas sėkmingas
Vykdymo rezultatas	Testavimo kontekste sukuriami vienas pagrindinis, du alternatyvūs ir vienas išimties testavimo atvejis.

4.8 lentelė. Testavimo atvejis 2.5

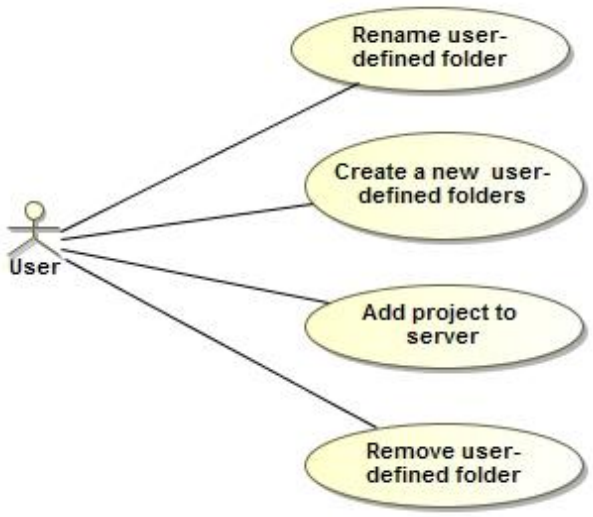
ID	2.5
Trumpas aprašymas	Testavimo atvejų diagramos sukūrimas ir priskirimas testavimo atvejui
Prieš sąlyga	Panaudojimų atvejų diagramoje turi būti bent vienas panaudojimo atvejis ir aprašytas jo scenarijus
Vykdymo žingsniai	1. Panaudojimo atvejų diagramos pasirinkimas „*Containment*“ kortelėje 2. Pelės dešinio klavišo paspaudimas 3. „*Generate Test Cases*“ meniu pasirinkimas
Laukiami rezultatai	Kiekvienam testavimo atvejui sukuriama testavimo atvejų diagrama ir priskiriama atitinkamam testavimo atvejui esančiame testavimo kontekste
Būsena	Įvykdytas – rezultatas sėkmingas
Vykdymo rezultatas	Kiekvienam testavimo atvejui sukuriama testavimo atvejų diagrama ir priskiriama atitinkamam testavimo atvejui esančiame testavimo kontekste

4.9 lentelė. Testavimo atvejis 2.6

ID	2.5
Trumpas aprašymas	*Generic Table sukūrimas*
Prieš sąlyga	-
Vykdymo žingsniai	1. Panaudojimo atvejų diagramos pasirinkimas „*Containment*“ kortelėje 2. Pelės dešinio klavišo paspaudimas 3. „*Generate Test Cases*“ meniu pasirinkimas
Laukiami rezultatai	Sukurtame iteracijos aplankale sukurta *Generic Table* lentelė. Kurioje sukuriami stulpeliai: *Name*, *Owner*, *Description*, *Priority*, *Notes*, *Expected Results*, *PreCondition*, *Complexity*, *Test Status*. Taip pat lentelė užpildoma iteracijos testavimo atvejais.
Būsena	Įvykdytas – rezultatas sėkmingas
Vykdymo rezultatas	Sukurtame iteracijos aplankale sukurta *Generic Table* lentelė. Kurioje sukuriami stulpeliai: *Name*, *Owner*, *Description*, *Priority*, *Notes*, *Expected Results*, *PreCondition*, *Complexity*, *Test Status*. Taip pat lentelė užpildoma iteracijos testavimo atvejais. (4.3 paveikslėlį)

4.2.3. Kontrolinis pavyzdys

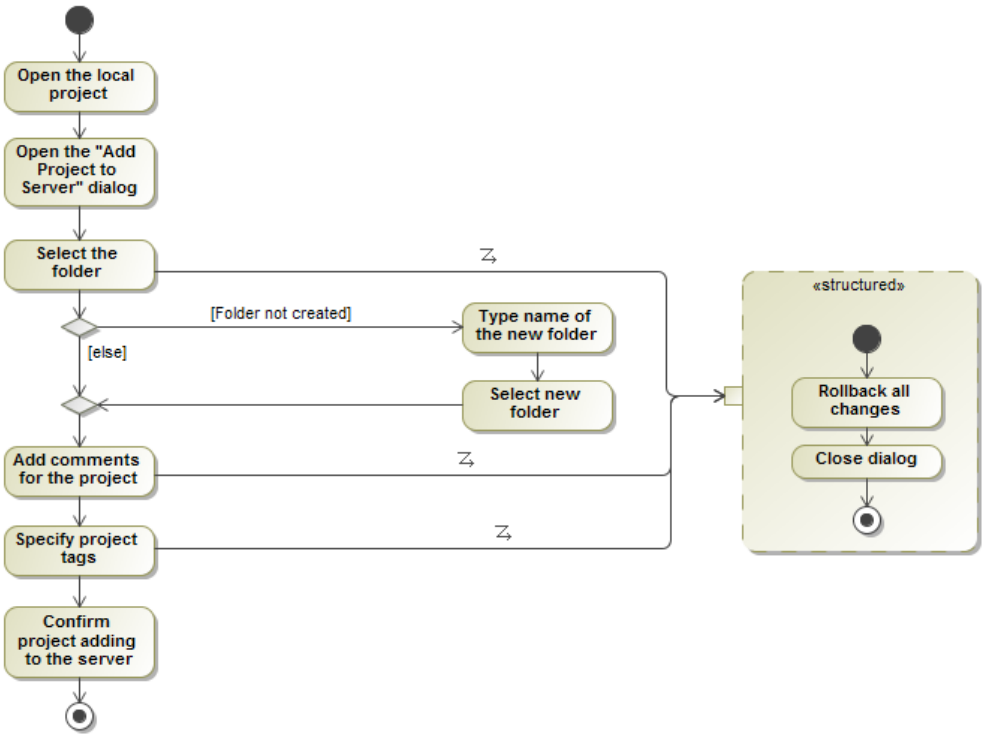
Turimi duomenys pavaizduoti 4.2 ir 4.4 paveiksluose ir gauti rezultatai įvykdžius automatizuotą testavimo atvejų generavimą pavaizduoti 4.3, 4.5 ir 4.6 paveiksluose.



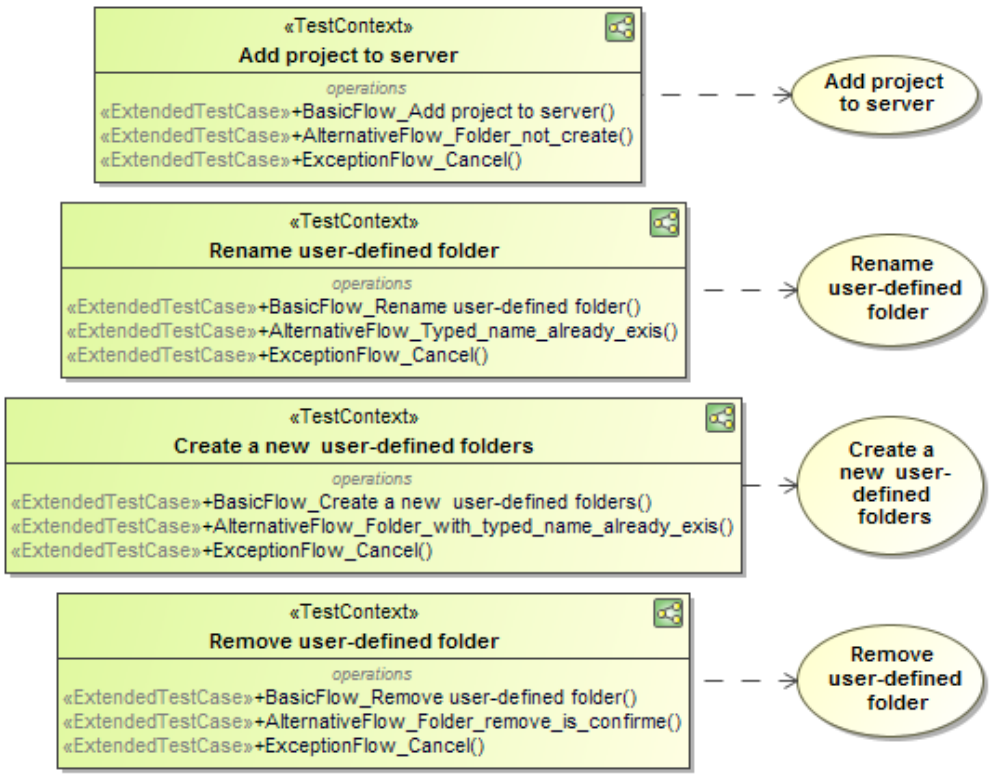
4.2 pav. Pavyzdinė panaudojimo atvejų diagrama testavimo atvejų generavimui

#	Name	Owner	Description	Priority	Notes	Expected Results	Pre Condition	Complexity	Test Status
1	BasicFlow_Rename	Rename user-...							
2	AlternativeFlow_T	Rename user-...							
3	ExceptionFlow_Cai	Rename user-...							
4	BasicFlow_Remove	Remove user-...							
5	AlternativeFlow_F	Remove user-...							
6	ExceptionFlow_Cai	Remove user-...							
7	BasicFlow_Create	Create a new ...							
8	AlternativeFlow_F	Create a new ...							
9	ExceptionFlow_Cai	Create a new ...							
10	BasicFlow_Add pro	Add project to...							
11	AlternativeFlow_F	Add project to...							
12	ExceptionFlow_Cai	Add project to...							

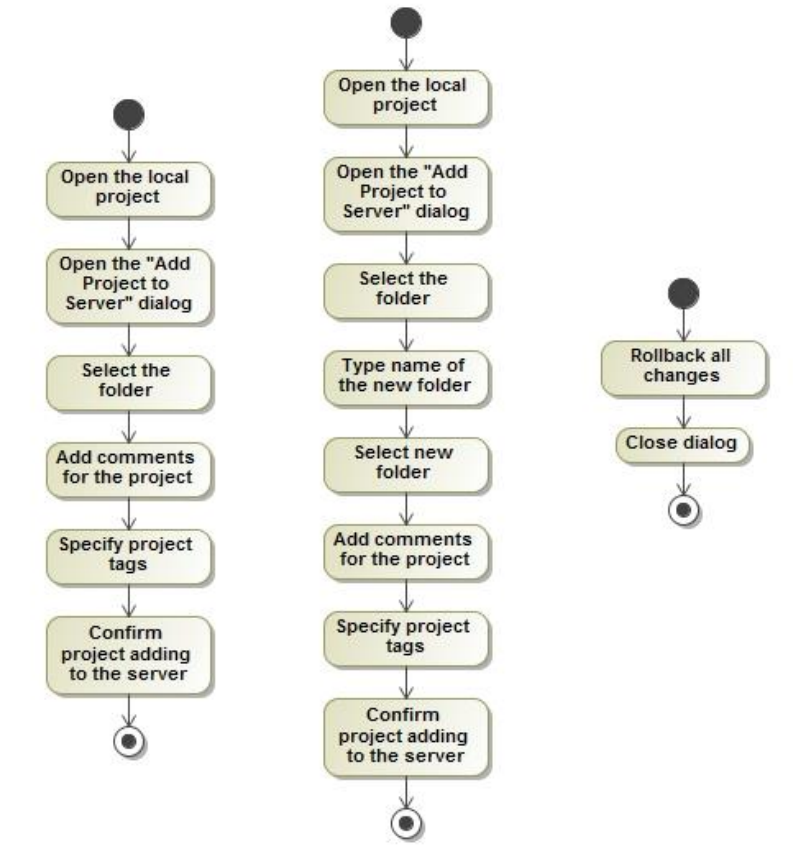
4.3 pav. Testavimo atveju generavimo metu iteracijoje sukurta „*Generic Table*“ lentelė



4.4 pav. *Add Project to Server* scenarijus



4.5 pav. Sugeneruota testavimo architektūros diagrama



4.6 pav. ***Add project to Server*** **sugeneruoti testavimo atvejai**

5. EKSPERIMENTINIS SPRENDIMO REALIZACIJOS TYRIMAS

Šiame skyriuje aprašomas eksperimentinis tyrimas, kurio metu siekiama patikrinti, ar *CASE* įrankio plėtinys įgyvendina šio darbo tikslus, analizuojamos jo taikymo galimybės.

5.1. Eksperimento apibrėžimas

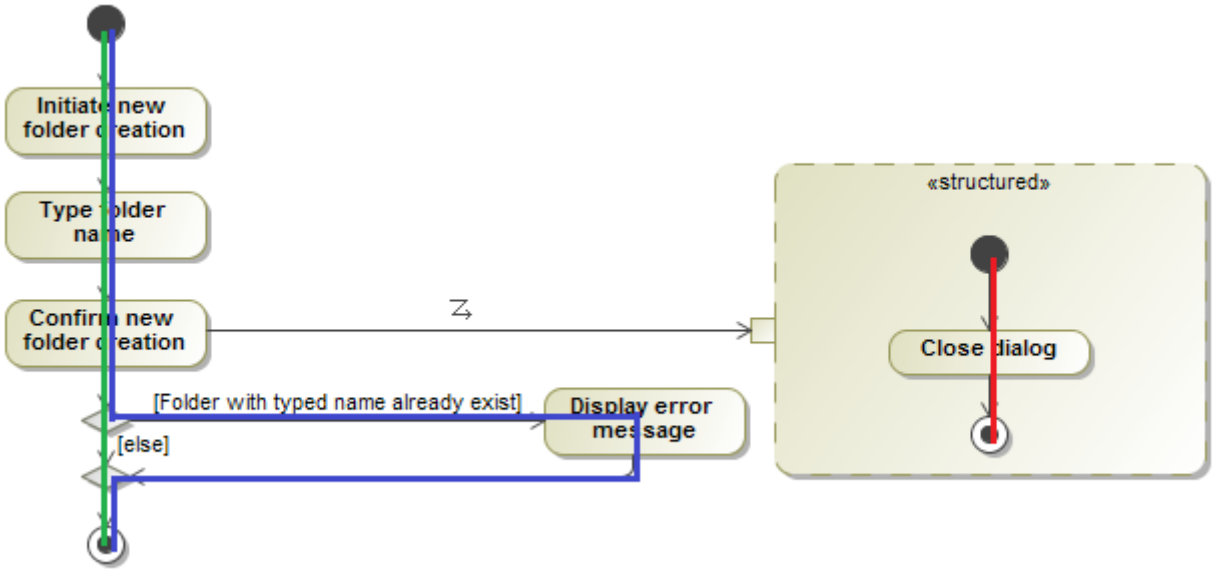
Eksperimento tikslas – patvirtinti, kad sukurtas *CASE* įrankio plėtinys testavimo atvejų kūrimui automatizuoti įgyvendina šio darbo tikslus. Eksperimente bus vertinami šie kriterijai:

1. Mažesnės darbo sąnaudos
2. Mažesnė žmogiškųjų klaidų tikimybė
3. Geresnis projekto artefaktų atsekamumas

5.2. Eksperimento vykdymo rezultatai

Norint patvirtinti, kad realizuotas įskiepis sumažina darbo sąnaudas, atliktas eksperimentas, kuris parodo žmogaus darbo sąnaudų Y, išreiškiamų *MagicDraw* įrankio naudotojo atliekamais veiksmais, priklausomybę nuo panaudojimo atvejų scenarijuose atliekamų veiksmų skaičiaus X.

Veiksmų skaičius X susideda iš kiekvieno panaudojimo atvejo pagrindinio, alternatyvių ir išimtinių scenarijų veiksmų skaičiaus (veiksmo atitikmuo veiklos diagramoje – veiksmo (angl. *Action*) elementas). Veiksmų skaičiaus nustatymo pavyzdys pagal 5.1 paveiksle pateiktą panaudojimo atvejo scenarijų veiklos diagramą pateikiamas 5.1 lentelėje.



5.1 Panaudojimo atvejo scenarijų veiklos diagrama

5.1 lentelė. Panaudojimo atvejų scenarijuose atliekamų veiksmų skaičiaus nustatymas

Scenarijus	**Veiksmų skaičius**
Pagrindinis (pažymėtas žalia linija 5.1 paveikslėlyje)	3
Alternatyvus (pažymėtas žalia linija **Error! Reference ource not found.**5.1 paveikslėlyje)	4
Išimties (pažymėtas žalia linija 5.1 paveikslėlyje)	1
Iš viso:	8

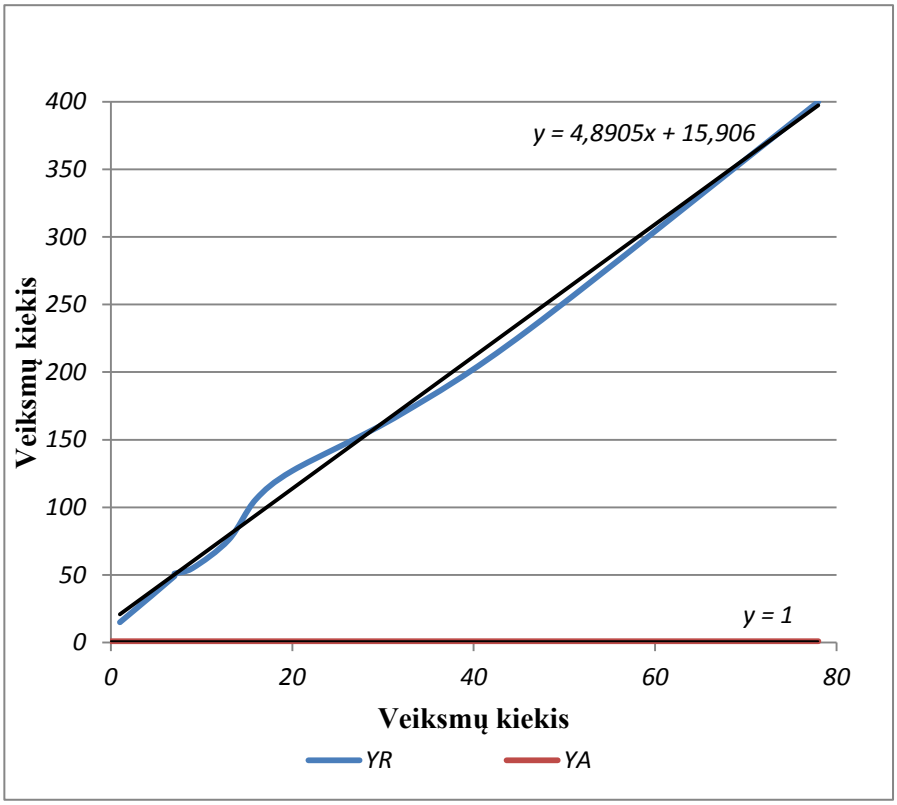
Žmogaus *MagicDraw* įrankiu atliekamų veiksmų skaičius Y_R susideda iš:

- Diagramos sukūrimo
- Diagramos elemento sukūrimo

- Diagramos ar jos elemento pavadinimo užrašymo
- Ryšio tarp elementų sukūrimo
- Operacijos metodo nurodymo
- „*Generate Test Case*“ meniu paspaudimo

Automatizuoto testavimo atvejų generavimo atveju žmogaus veiksmų skaičius $Y_A = 1$, t. y., jam reikia tik paspausti mygtuką „Generuoti testavimo atvejus“.

Ši priklausomybė grafiškai pavaizduota 5.2 paveikslėlyje, kuriame vaizduojama priklausomybė esant įdiegtam automatizuotam testavimo atvejų generavimui ir be jo. Matome, kad kuo daugiau panaudojimo atvejų ir kuo sudėtingesni jų scenarijai, tuo didesnis žmogaus darbo sąnaudų sumažėjimas.



5.2 pav. Žmogaus darbo sąnaudų priklausomybė nuo panaudojimo atvejų scenarijuose atliekamų veiksmų skaičiaus automatizuoto (Y_A) ir rankinio (Y_R) testavimo atvejų kūrimo atvejais

Sekančio eksperimento metu patikrintas korektiškas duomenų perkėlimas iš ankstesnės testavimo iteracijos į dabartinę. Panaudota 4 skyriuje pateikta *Generic Table* lentele (4.3 paveikslėlį), kuri buvo sukurta vykdant pirmąjį testavimo iteracijos generavimą ir užpildyta testiniais duomenimis (5.3 paveikslėlį).

Įvykdytas antros testavimo etapo iteracijos testavimo atvejų generavimas. Šioje iteracijoje sukurta lentelė su užpildytais duomenimis atitinka ankstesnės iteracijos duomenis, vadinasi šiuo atveju duomenų perkėlimas korektiškas.

#	Name	Owner	Description	Priority	Notes	Expected Results	Pre Condition	Complexity	Test Status
1	BasicFlow_Rename	Rename user-...	Description1	Primary	Notes1	Folder renamed		Simple	Success
2	AlternativeFlow_Ty	Rename user-...		High	Notes2	Show warning msg		Medium	Failed
3	ExceptionFlow_Cai	Rename user-...	Description3	High		Dialog closed	PreCondition	Simple	Blocked
4	BasicFlow_Remove	Remove user-...							Not run
5	AlternativeFlow_F	Remove user-...							Not run
6	ExceptionFlow_Cai	Remove user-...							Not run
7	BasicFlow_Create	Create a new ...							Not run
8	AlternativeFlow_F	Create a new ...							Not run
9	ExceptionFlow_Cai	Create a new ...							Not run
10	BasicFlow_Add pro	Add project to...							Not run
11	AlternativeFlow_F	Add project to...							Not run
12	ExceptionFlow_Cai	Add project to...							Not run

5.3 pav. Pirmos iteracijos *Generic Table* lentelė užpildyta testiniais duomenimis

5.4 pav. Panaudojimo atvejo „*Remove Project from Server*“ pagrindinio scenarijaus veiklos diagrama

Pakoregavus reikalavimuose panaudojimo atvejų diagramą ar konkretaus panaudojimo atvejo scenarijus ir įvykdžius naują testavimo atvejų generavimo iteraciją visi pakeitimai turi atsirasti naujai sugeneruotose diagramose. Šiam funkcionalumui patvirtinti sukurtas dar vienas panaudojimo atvejis – *Remove Project from Server*. Aprašytas pagrindinis scenarijus, kuris pavaizduotas 5.4 paveikslėlyje. Taip pat pakoreguoti testavimo iteracijos duomenys (5.5 paveikslėlį).

#	Name	Owner	Description	Priority	Notes	Expected Results	Pre Condition	Complexity	Test Status
1	BasicFlow_Rename us	Rename user-defi...	Description1	Primary	Notes1	Folder renamed		Simple	Success
2	AlternativeFlow_Type	Rename user-defi...		High	Notes2	Show warning msg		Medium	Success
3	ExceptionFlow_Cance	Rename user-defi...	Description3	High		Dialog closed	PreCondition	Simple	Failed
4	BasicFlow_Remove us	Remove user-defi...	Description4	Medium		Results1		Medium	Success
5	AlternativeFlow_Fold	Remove user-defi...	Description5	Medium		Results2		Medium	Success
6	ExceptionFlow_Cance	Remove user-defi...		Medium		Results3		Medium	Failed
7	BasicFlow_Create a n	Create a new us...							Not run
8	AlternativeFlow_Fold	Create a new us...							Not run
9	ExceptionFlow_Cance	Create a new us...							Not run
10	BasicFlow_Add projec	Add project to se...							Not run
11	AlternativeFlow_Fold	Add project to se...							Not run
12	ExceptionFlow_Cance	Add project to se...							Not run

5.5 pav. Antros iteracijos *Generic Table* lentelė su koreguotais duomenimis

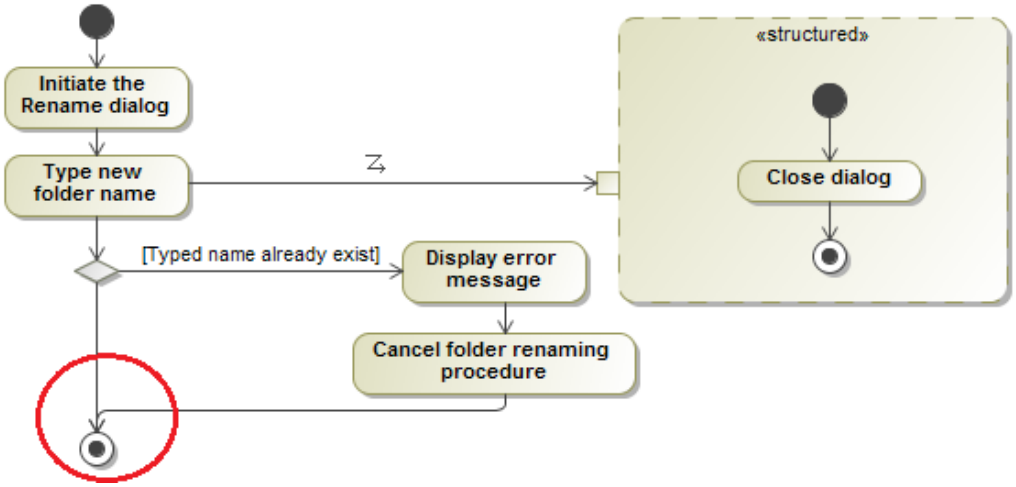
#	Name	Owner	Notes	Pre Condition	Description	Expected Results	Complexity	Priority	Test Status
1	BasicFlow_Rename u	Rename user-defined folder	Notes1		Description1	Folder renamed	Simple	Primary	Success
2	AlternativeFlow_Typ	Rename user-defined folder	Notes2			Show warning msg	Medium	High	Success
3	ExceptionFlow_Can	Rename user-defined folder		PreCondition	Description3	Dialog closed	Simple	High	Failed
4	BasicFlow_Remove u	Remove user-defined folder			Description4	Results1	Medium	Medium	Success
5	AlternativeFlow_Fol	Remove user-defined folder			Description5	Results2	Medium	Medium	Success
6	ExceptionFlow_Can	Remove user-defined folder				Results3	Medium	Medium	Failed
7	BasicFlow_Remove p	Remove project from server							
8	BasicFlow_Create a	Create a new user-define...							Not run
9	AlternativeFlow_Fol	Create a new user-define...							Not run
10	ExceptionFlow_Can	Create a new user-define...							Not run
11	BasicFlow_Add proje	Add project to server							Not run
12	AlternativeFlow_Fol	Add project to server							Not run
13	ExceptionFlow_Can	Add project to server							Not run

5.6 pav. Trečios iteracijos *Generic Table* lentelė

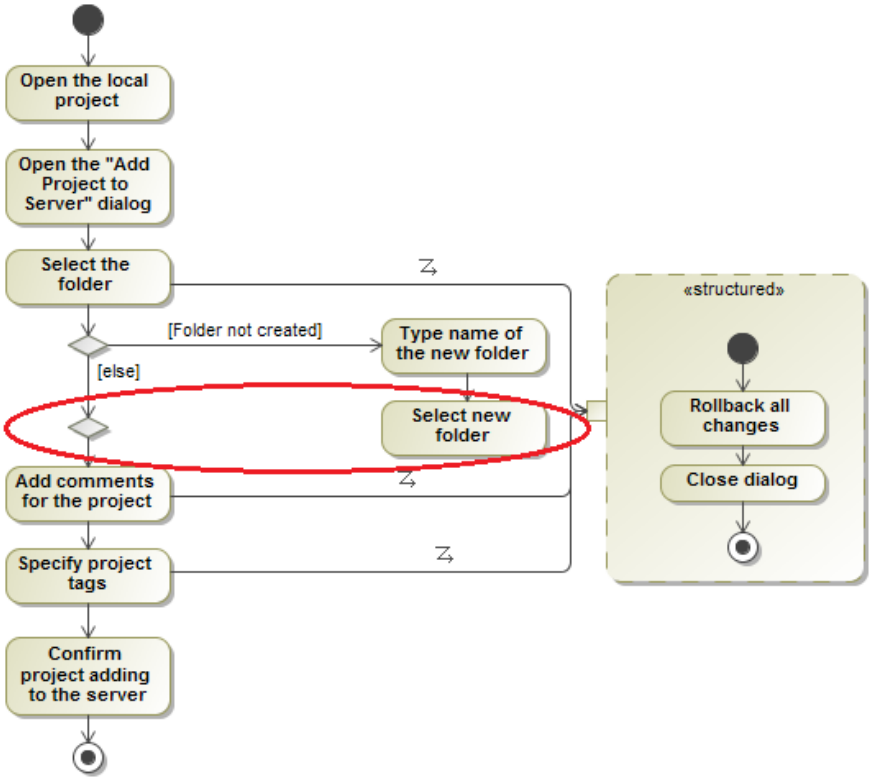
Įvykdytas trečios iteracijos generavimas, kurios duomenys pateikti 5.6 paveikslėlyje. Išanalizavus rezultatus matyti jog padaryti pakeitimai reikalavimuose atsirado trečioje iteracijoje (5.6 paveikslėlį, pakeitimai apvesti raudonai). Taip pat testavimo duomenys atitinka antros iteracijos duomenis.

Ketvirtosios iteracijos generavime patikrinsime kaip įskiepis elgiasi kai veiklos diagrama yra nekorektiška. *Rename user-defined folder* veiklos diagramoje pašalinamas *Merge* elementas (), o *Add Project to Server* veiklos diagramoje pašalinamas ryšys tarp dviejų elementų (5.8 paveikslėlį). Taip pat buvo pašalintas *Remove Project from server* panaudojimo atvejis.

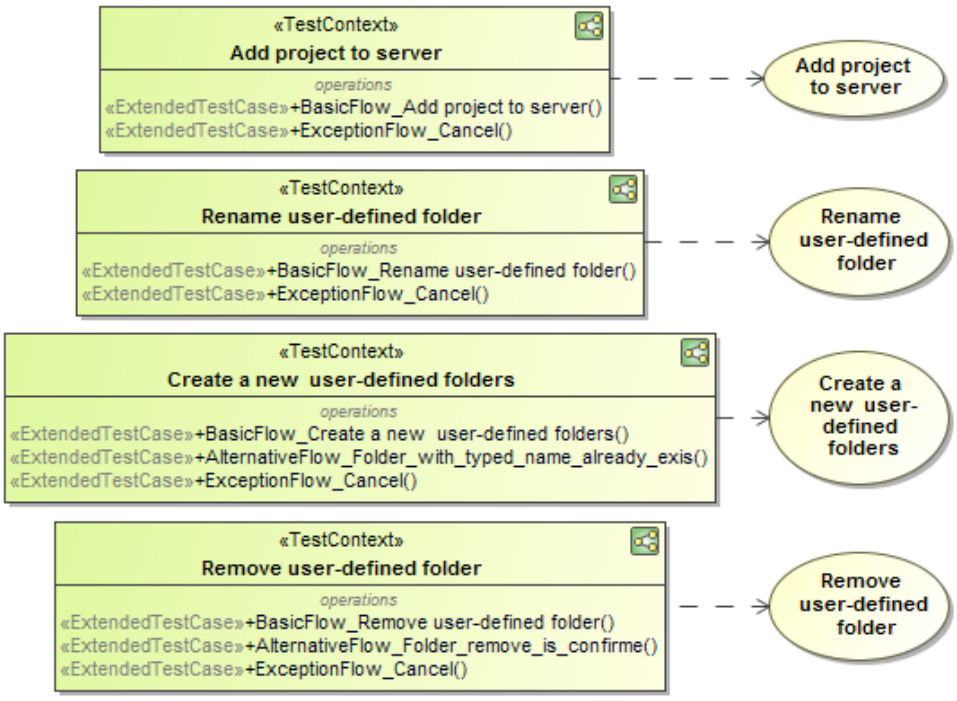
Įvykdytas ketvirtosios iteracijos generavimas. Sugeneruotoje testavimo architektūros diagramoje (5.9 paveikslėlį) matyti jog *Rename user-defined folder* ir *Add Project to Server* panaudojimo atvejams nesugeneruoti alternatyvūs scenarijai. Taip šiems panaudojimo atvejams sugeneruoti nepilni pagrindiniai testavimo atvejų scenarijai (5.10 ir 5.11 paveikslėlius).



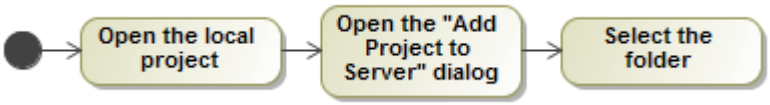
5.7 pav. Nekorektiška *Rename user-defined folder* veiklos diagrama



5.8 pav. Nekorektiška *Add Project to Server* veiklos diagrama



5.9 pav. Ketvirtos iteracijos testavimo architektūros diagrama



5.10 pav. Ketvirtos iteracijos panaudojimo atvejo *Add Project to Server* pagrindinio testavimo scenarijaus veiklos diagrama

5.11 pav. Ketvirtos iteracijos panaudojimo atvejo *Rename user-defined folder* pagrindinio testavimo scenarijaus veiklos diagrama

5.3. Automatizuoto testavimo atvejų generavimo įskiepio veikimo ir savybių analizė, kokybės kriterijų įvertinimas

Atlikus eksperimentus patvirtinta, jog įskiepis sumažina laiko sąnaudas testavimo etape. Kuo daugiau panaudojimo atvejų yra ir kuo daugiau veiksmų sudaro jų scenarijai tuo labiau sumažinamos laiko sąnaudos (5.2 paveikslėlį).

Realizuotos testavimo iteracijos, kuriuose atsispindi reikalavimų, bei testavimo būklė, kurią apibrėžia užpildyti testavimo duomenys, pakitimai laikui bėgant, padidina projekto artefaktų atsekamumą. O automatizuotas testavimo duomenų perkėlimas iš ankstesnės iteracijos sumažina žmogiškųjų klaidų tikimybė perkopijuojant duomenis.

5.4. Įskiepio taikymo rekomendacijos

Įskiepį rekomenduojama naudoti priėmimo testavimui, kuris atliekamas užsakovo reikalavimų atžvilgiu.

6. REZULTATŲ APIBENDRINIMAS IR IŠVADOS

1. Atlikta panašių įrankių ir metodų analizė parodė, kad nėra išbaigto ir pakartotinai panaudojamo sprendimo testavimo atvejams generuoti:
    a. *Enterprise Architect CASE* yra sukurtas kokybiškas automatinio testavimo atvejų generavimo sprendimas, tačiau jis nėra pagrįstas standartu ir neturi galimybės generuoti testavimo ataskaitas
    b. Literatūroje aprašyti testavimo atvejų generavimo sprendimai yra sudėtingesni, jų realizacijos nėra prieinamos ir nėra pagrįstos standartais
    c. *IMB* testavimo atvejų generavimo metodas yra geras tuo, kad ne tik sukuriami pagrindiniai ir alternatyvūs testavimo scenarijai ir jų parametrai, bet ir apibrėžiamos konkrečios parametrų reikšmės. Tačiau metodas tik teoriškai aprašytas.
2. Atlikus *UML* testavimo profilio (*UTP*) ir *UML CASE* įrankių analizę, nustatyta, kad šį profilį tikslinga naudoti, kuriant įskiepį automatizuotam testavimų atvejų generavimui, kadangi jis yra išbaigtas ir plačiai pripažįstamas *OMG* standartas, realizuotas *MagicDraw UML CASE* įrankyje.
3. Sukurtų projektinių modelių realizacija parodė, kad testavimo atvejų sudarymą galima automatizuoti *MagicDraw UML CASE* įrankyje taikant *UTP* profilį.
4. Atliktas eksperimentas patvirtino, kad sukurtas *CASE* įrankio plėtinys testavimo atvejų kūrimui automatizuoti įgyvendina numatytas galimybes.
5. Eksperimentinis sprendimo naudingumo tyrimas parodė, kad rankinio testavimo atvejų kūrimo atveju žmogaus darbo sąnaudos auga tiesine progresija priklausomai nuo reikalavimų sudėtingumo (PA skaičiaus ir scenarijų sudėtingumo). Automatizuotas sprendimas leidžia šių sąnaudų išvengti.
6. Automatizuotas testavimo atvejų kūrimas ypač naudingas esant reikalavimų pokyčiams, nes leidžia užtikrinti projekto artefaktų ir jų testavimo rezultatų atsekamumą.

7. LITERATŪRA

[1] Beatriz Pérez Lamancha; Pedro Reales Mateo; Ignacio Rodríguez de Guzmán; Macario Polo Usaola; Mario Piattini Velthius. Automated Model-based Testing using the UML Testing Profile and QVT. Proceedings of the 6th International Workshop on Model-Driven Engineering, Verification and Validation. 2009, 6.

[2] Cle´mentine Nebut, Franck Fleurey, Yves Le Traon, Jean-Marc Je´ze´quel. Automatic Test Generation: A Use Case Driven Approach. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. 2006, 32(3): 140–155.

[3] Heumann Jim. Generating Test Cases From Use Cases. Iš IBM [interaktyvus]. 2001, birželis [žiūrėta 2013-12-08]. Prieiga per internetą: < http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf>.

[4] Hyungchoul Kim, Sungwon Kang, Jongmoon Baik, Inyoung Ko. Test Cases Generation From UML Activity Diagrams. Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. 2007, 3: 556–561.

[5] Javier J. Gutiérrez; María J. Escalona; Manuel Mejías; Jesús Torres; Generation of test cases from functional requirements. A survey. Prieiga per internetą: < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.6641&rep=rep1&type=pdf >

[6] Kaur, A.; Vig, V. Systematic Review of Automatic Test Case Generation by UML Diagrams. International Journal of Engineering Research & Technology. 2012, 1(6): 1–17.

[7] No Magic. Open API. 2015. Prieiga per internetą: < http://www.nomagic.com/files/manuals/MagicDraw%20OpenAPI%20UserGuide.pdf >

[8] Paul Baker; Zhen Ru Dai; Jens Grabowski; Ųystein Haugen; Ina Schieferdecker; Clay Williams. Model-Driven Testing Using the UML Testing Profile. New York, 2008.

[9] Prasanna, M.; Sivanandam , S.N.; Venkatesan, R.; Sundarrajan, R. A Survey On Automatic Test Case Generation. Academic Open Internet Journal. 2005, 15.

[10] Sparx Systems. Writing Use Case Scenarios For Model Driven Development. Iš Sparx systems [interaktyvus]. 2010, rugsėjis [žiūrėta 2013-11-13]. Prieiga per internetą: < http://www.sparxsystems.com.au/downloads/quick/writing-structured-use-case-scenarios-mdd.pdf>.