KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

DAINIUS KNIUKŠTA

DARBŲ SEKOMIS GRINDŽIAMŲ INFORMACINIŲ SISTEMŲ KŪRIMO METODIKA

Magistro darbas

Vadovas
doc. dr. T. Skersys

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

DAINIUS KNIUKŠTA

DARBŲ SEKOMIS GRINDŽIAMŲ INFORMACINIŲ SISTEMŲ KŪRIMO METODIKA

Magistro darbas

Vadovas:
doc. dr. T. Skersys
2013-05-21

Recenzentas:
lekt. dr. S. Drąsutis
2013-05-21

Atliko:
IFM-1/4 gr. studentas
Dainius Kniukšta
2013-05-21

KAUNAS, 2013

AUTORIŲ GARANTINIS RAŠTAS
DĖL PATEIKIAMO KŪRINIO

2013 -Gegužės- 21 d.
Kaunas

Autoriai,_____Dainius Kniukšta_____
(vardas, pavardė)

_____,

patvirtina, kad Kauno technologijos universitetui pateiktas baigiamasis magistro darbas (toliau vadinama – Kūrinys)_____ **,,**Darbų sekomis grindžiamų informacinių sistemų kūrimo metodika“_____

pagal Lietuvos Respublikos autorių ir gretutinių teisių įstatymą yra originalus ir užtikrina, kad

1) jį sukūrė ir parašė Kūrinyje įvardyti autoriai;
2) Kūrinys nėra ir nebus įteiktas kitoms institucijoms (universitetams) (tiek lietuvių, tiek užsienio kalba);
3) Kūrinyje nėra teiginių, neatitinkančių tikrovės, ar medžiagos, kuri galėtų pažeisti kito fizinio ar juridinio asmens intelektinės nuosavybės teises, leidėjų bei finansuotojų reikalavimus ir sąlygas;
4) visi Kūrinyje naudojami šaltiniai yra cituojami (su nuoroda į pirminį šaltinį ir autorių);
5) neprieštarauja dėl Kūrinio platinimo visomis oficialiomis sklaidos priemonėmis.
6) atlygins Kauno technologijos universitetui ir tretiesiems asmenims žalą ir nuostolius, atsiradusius dėl pažeidimų, susijusių su aukščiau išvardintų Autorių garantijų nesilaikymu;
7) Autoriai už šiame rašte pateiktos informacijos teisingumą atsako Lietuvos Respublikos įstatymų nustatyta tvarka.

Autoriai

_____Dainius Kniukšta_____ _____
(vardas, pavardė) (parašas)

DARBO SANTRAUKA

Šiame darbe siekiama sudaryti metodiką, kuri leistų lengviau kurti informacines sistemas grindžiamas darbų sekų vykdymu. Siekiant sudaryti metodiką skirtą darbų sekų vykdymu grindžiamų informacinių sistemų kūrimui reikia išanalizuoti esančias metodikas skirtas informacinėms sistemos kurti ir atsirinkti geriausias jų savybes, kurias panaudoti sudarant naująją metodiką.

Sudarius metodiką atliekamas eksperimentinės informacinės sistemos skirtos studijų modulių valdymui sukūrimas remiantis sudaryta metodika. Sistemos realizacijai atlikti pasirinkta Bonita studio programa, kuri puikiai atitinka visus sistemai keliamus reikalavimus. Sukūrus sistemą ja naudojant Bonita studio ir išbandoma paleidžiant darbų sekų vykdymą.

Naudojant naujai sudarytą metodiką yra daug paprasčiau kurti darbų sekų vykdymų grindžiamas informacines sistemas, kadangi metodika labiausiai į tai orientuojasi. Taip pat metodika stengiasi padėti išvengti pagrindinių sistemos žlugimo priežasčių.

SUMMARY

In this work we want to design methodology, which allows as developing informational systems based on workflow execution easily. In order to design methodology we need to analyze other methodologies which are used to develop informational systems. After analysis we try to pick up best parts from these methodologies and combine to design one methodology which allows as easily to develop workflow-based informational systems.

When we finished designing methodology decided to develop experimental system using methodology which was designed. To develop system we choose Bonita Studio. After we develop system using Bonita Studio we can execute it and see how it works.

It is easier to develop systems based on workflow execution because it oriented to workflow execution. Also methodology tries to help avoid all main risks why systems fails.

TURINYS

DARBO SANTRAUKA ... 4
SUMMARY ... 5
TURINYS ... 6
PAVEIKSLĖLIŲ TURINYS ... 8
LENTELIŲ TURINYS ... 9
TERMINŲ IR SANTRUMPŲ ŽODYNĖLIS ... 10
1 ĮVADAS ... 11
2 TYRIMO OBJEKTAS, SRITIS IR PROBLEMA ... 11
2.1 Tyrimo objektas ... 11
2.2 Tyrimo sritis ... 11
2.3 Tyrimo problema ... 11
3 TYRIMO TIKSLAS ... 12
3.1 Uždaviniai ... 12
4 ANALIZĖS TIKSLAS ... 13
5 ANALIZĖS METODAI ... 13
6 TYRIMO OBJEKTO METODŲ ANALIZĖ ... 13
6.1 DSDM metodika ... 13
6.1.1 DSDM metodikos principai ... 13
6.1.2 DSDM metodikos naudojimosi taisyklės ... 14
6.1.3 Kūrimo etapai naudojant DSDM metodiką ... 14
6.1.4 Kaip DSDM padeda išvengti sunkumų ... 16
6.1.5 DSDM komandos rolės ... 16
6.1.6 MoSCoW prioritizavimas ... 17
6.2 Crystal metodikų šeima ... 17
6.2.1 Crystal šeimos metodikų principai ... 18
6.2.2 Crystal Clear metodika ir kūrimo ciklas ... 18
6.2.3 Crystal Orange metodika ir komandos rolės ... 19
6.3 Agile Scrum metodika ... 20
6.3.1 Agile Scrum principai ... 20
6.3.2 Projekto komandos sudėtis naudojant Agile Scrum metodiką ... 20
6.3.3 Sistemos kūrimo gyvavimo ciklas naudojant Agile Scrum metodiką ... 21
6.4 RUP - Rational Unified Process metodika ... 21
6.4.1 Pagrindiniai RUP metodikos principai ... 21
6.4.2 Sistemos kūrimo gyvavimo ciklas naudojant RUP metodiką ... 22
6.4.3 RUP metodikos komandos rolės ... 24
6.5 Metodikų apibendrinimas ... 24
6.6 Agile Scrum prieš RUP ... 27
7 ESAMŲ IS SISTEMŲ REALIZAVIMO IR MODELIAVIMO ĮRANKIŲ ANALIZĖ ... 28
7.1 Bonita studio 5.6.1 ... 29
8 SIEKIAMAS SPRENDIMAS ... 31
9 ANALIZĖS IŠVADOS ... 31
10 DARBŲ SEKOMIS GRINDŽIAMŲ INFORMACINIŲ SISTEMŲ KŪRIMO METODIKA ... 31
10.1 Pradinė IS projekto analizė ... 34
10.2 Informacinės sistemos reikalavimų analizė ir projektavimas ... 37
10.3 Informacinės sistemos realizacijos kūrimas ... 40
10.4 Informacinės sistemos testavimas ... 42
10.5 Informacinės sistemos palaikymas ... 44
11 EKSPERIMENTINIS TYRIMAS ... 44
11.1 Pradinė projekto analizė ... 44
11.1.1 Sistemos kūrimo galimybių analizė ... 45
11.1.2 Sistemos veiklos procesų analizė ... 48
11.1.3 Sistemos įsipareigojimų dokumentas ... 49
11.2 Sistemos reikalavimų analizė ir projektavimas ... 49
11.2.1 Funkciniai sistemos reikalavimai ... 49

11.2.2 Nefunkciniai sistemos reikalavimai 50
11.2.3 Sistemos grafinių modelių modeliavimas 51
11.3 Sistemos realizacijos kūrimas 52
11.4 Sistemos testavimas 53
11.4.1 Sistemos testavimo atvejai 54
11.5 Sistemos palaikymas 54
IŠVADOS 55
LITERATŪRA 56

PAVEIKSLĖLIŲ TURINYS

2.1 pav. IS orientuota į informacijos pasiekiamumą remiantis [22] 11
2.2 pav. Veiklos taisyklių vykdymu pagrįsta IS remiantis [22] 12
6.1 pav. DSDM metodikos veikimo procesas remiantis [23] 15
6.2 pav. Scrum karkaso procesas remiantis [6] 21
7.1 pav. Sumodeliuotas darbų sekų modelis 29
7.2 pav. Sekos žingsniui nurodyti kintamieji ir jų tipai 30
7.3 pav. Sumodeliuota sistema paleista naudojant Bonita execution engine 30
10.1 pav. Metodikos fazių tarpusavio ryšiai 33
10.2 pav. Metodikos fazės sudedamosios dalys 34
10.3 pav. Pradinės sistemos analizės fazės procesų modelis 34
10.4 pav. Sistemos pradinių reikalavimų surinkimo proceso modelis 35
10.5 pav. Sistemos galimybių analizės proceso modelis 35
10.6 pav. Sistemos verslo poreikių analizės modelis 36
10.7 pav. Sistemos prototipo kūrimo modelis 36
10.8 pav. Sistemos įsipareigojimo dokumento parengimo modelis 37
10.9 pav. Sistemos reikalavimų analizės ir projektavimo fazės veiklos proceso modelis 38
10.10 pav. Sistemos modeliavimo metu modeliuojamų UML diagramų sąrašas 39
10.11 pav. Sistemos realizacijos kūrimo fazės proceso modelis 40
10.12 pav. Sistemos realizacijos kūrimo proceso modelis 41
10.13 pav. Sistemos testavimo fazėje surinktų klaidų taisymo proceso modelis 42
10.14 pav. Sistemos testavimo fazės proceso modelis 42
10.15 pav. Sistemos testavimo proceso modelis 43
10.16 pav. Sistemos palaikymo fazės proceso modelis 44
11.1 pav. Sistemos administratoriaus panaudojimo atvejų modelis 46
11.2 pav. Dėstytojo ir studento panaudojimo atvejų modelis 47
11.3 pav. Užduoties būsenų kitimo modelis 48
11.4 pav. Dėstytojo prašymo pateikimo veiklos proceso modelis 48
11.5 pav. Pasirinkto modulio veikimo proceso modelis 49
11.6 pav. Sistemos diegimo modelis 51
11.7 pav. Dėstytojo ir administratoriaus bendravimo veiklos procesas 51
11.8 pav. Dėstytojo ir studento veiklos proceso modelis 52
11.9 pav. Sistemos vykdymo lango vaizdas 52
11.10 pav. Sistemos vartotojo sąskaitos vaizdas 53
11.11 pav. Prisijungimo prie sistemos langas 53

LENTELIŲ TURINYS

6.1 LENTELĖ PROJEKTO VYKDYMO KARKASŲ APŽVALGA....25
6.2 LENTELĖ AGILE SCRUM IR RUP METODIKŲ PALYGINIMAS REMIANTIS [15]....27
7.1 LENTELĖ ESAMŲ IS SISTEMŲ REALIZAVIMO IR MODELIAVIMO ĮRANKIŲ ANALIZĖ....28
10.1 LENTELĖ METODIKOS TEIGIAMOS SAVYBĖS....32
11.1 LENTELĖ DARBŲ SĄRAŠO PAVYZDYS....44
11.2 LENTELĖ PRADINIAI SISTEMOS REIKALAVIMAI....45
11.3 LENTELĖ GALIMYBIŲ ANALIZĖ....45
11.4 LENTELĖ SISTEMOS VARTOTOJŲ GRUPĖS....46
11.5 LENTELĖ ADMINISTRATORIAUS PANAUDOJIMO ATVEJŲ PAAIŠKINIMAI....46
11.6 LENTELĖ DĖSTYTOJO IR STUDENTO PANAUDOJIMŲ ATVEJŲ PAAIŠKINIMAS....47
11.7 LENTELĖ DARBŲ SĄRAŠO PAVYZDYS....49
11.8 LENTELĖ FUNKCINIS REIKALAVIMAS NR. 1....49
11.9 LENTELĖ FUNKCINIS REIKALAVIMAS NR.2....50
11.10 LENTELĖ FUNKCINIS REIKALAVIMAS NR.3....50
11.11 LENTELĖ NEFUNKCINIS REIKALAVIMAS NR.1....50
11.12 LENTELĖ NEFUNKCINIS REIKALAVIMAS NR.2....50
11.13 LENTELĖ NEFUNKCINIS REIKALAVIMAS NR.3....50
11.14 LENTELĖ NEFUNKCINIS REIKALAVIMAS NR.4....51
11.15 LENTELĖ DARBŲ SĄRAŠO PAVYZDYS....52
11.16 LENTELĖ DARBŲ SĄRAŠO PAVYZDYS....53
11.17 LENTELĖ SISTEMOS TESTAVIMO SUVESTINĖ....54
11.18 LENTELĖ DARBŲ SĄRAŠO PAVYZDYS....54

TERMINŲ IR SANTRUMPŲ ŽODYNĖLIS

ERP – Enterprise resource planning (liet. Verslo valdymo sistema)

SAP – Systems Applications and Products (liet. Duomenų apdorojimo sistemos)

LDAP – Lightweight Directory Access Protocol (liet. Lengvasvorio katalogo prieigos protokolas)

CRM – Customer Relationship Management (liet. Klientų valdymo sistema)

XML – Extensible Markup Language (liet. Duomenų aprašymo kalba)

PDF – Portable Document Format (liet. Atviro standarto dokumentas)

VGS – Vartotojo grafinė sąsaja

DSDM – Dynamic systems development method (liet. Dinamiškas sistemų kūrimo metodas)

RAD – Rapid application development (liet. Iteracinis sistemos kūrimas)

RUP – Rational Unified Process (liet. Kartotinio programinės įrangos kūrimo metodika)

UML – Unified Modeling Language (liet. Vieninga modeliavimo kalba)

EJB – Enterprise JavaBeans (liet. Įmonių JavaBeans)

HTTP – HyperText Transfer Protocol (liet. Hyper teksto perdavimo protokolas)

JAXRS – Java API for RESTful Web Services (liet. Java programėlė padedanti sukurti internetinius servisus remiantis architektūriniu šablonu)

CSS – Cascading Style Sheets (liet. Kalba skirta nusakyti kita struktūrine kalba aprašyto dokumento vaizdavimą)

BPMN – Business Process Model and Notation (liet. Veiklos procesų modeliavimo notacija)

SQL – Structured Query Language (liet. Struktūrizuota užklausų kalba)

MB – Megabite

1 ĮVADAS

Kuriant informacines sistemas susiduriama su problemomis, kurios kyla šiuo metu daugiausia informacinės sistemos kuriamos, kurios orientuojasi informacijos pasiekiamumą. Dažnai informacijos pasiekiamumas kiša koją sistemos vartotojams, kai jie nori naudotis vienu failu tuo pat metu. Dėl tokių problemų dažnai stoja darbai. Geriausiai šią problemą gali padėti išspręsti darbų sekų vykdymu grindžiamos informacinės sistemos, kurios orientuojasi į veiklų vykdymą ir padeda užtikrinti, kad tuo pat metu nesinaudos vienu failu keli sistemos vartotojai.

Tam, kad galėtumėme kurti darbų sekų vykdymu grindžiamas informacines sistemas, reikia sudaryti metodiką, kuri padėtų jas kurti. Pasitelkiant geriausias šiuo metu esančių sistemų kūrimo metodikų savybes ir jas apjungiant į vieną visumą ir sudaryti metodiką, kuri bus skirta darbų sekų vykdymų grindžiamų informacinių sistemų kūrimui.

2 TYRIMO OBJEKTAS, SRITIS IR PROBLEMA

2.1 Tyrimo objektas

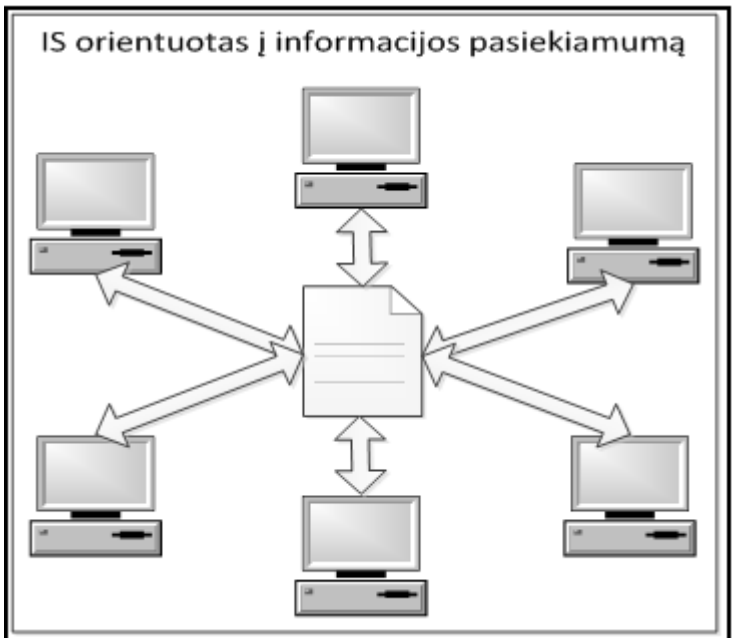
Darbų sekų vykdymu grindžiamos informacinės sistemos kūrimo metodika. Naudojant šią metodiką, kuriant informacines sistemas bus galima sukurti informacinę sistemą, kuri veiks darbų sekomis, kurios yra nurodomos kūrimo metu.

2.2 Tyrimo sritis

Darbų sekų vykdymu grindžiamos informacinės sistemos kūrimo metodai ir įrankiai. Atliekant tyrimo srities analizę apžvelgti metodus, kurie yra naudojami kuriant informacines sistemas pagrįstas darbų sekų vykdymu. Taip pat apžvelgti ir išbandyti produktus, kuriais būtų galimą atlikti tyrimą.

2.3 Tyrimo problema

Dauguma informacinių sistemų, kurios yra kuriamos yra orientuotos į informacijos pasiekiamumą, kas iliustruota 2.1 pav.

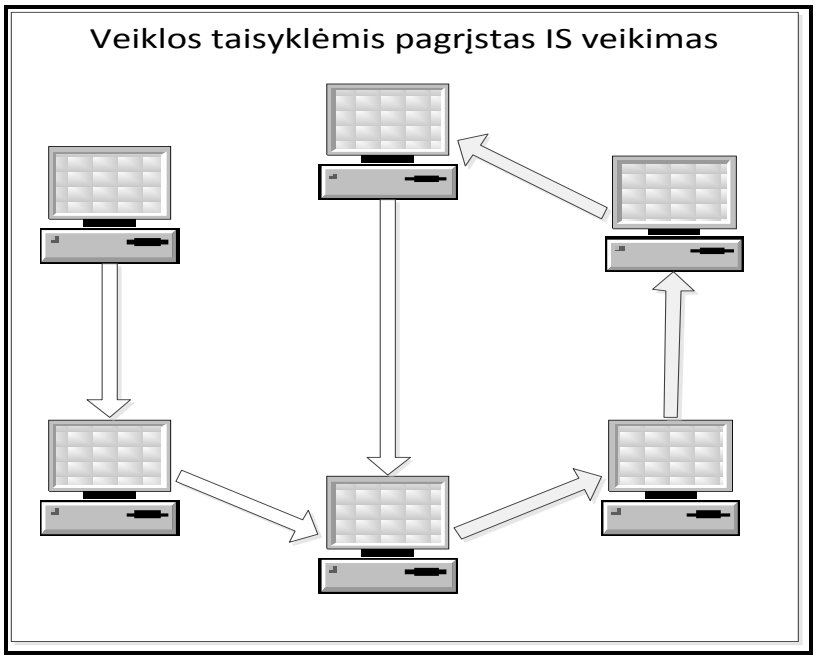


2.1 pav. IS orientuota į informacijos pasiekiamumą remiantis [22]

Tokiose sistemose, visi vartotojai gali, bet kuriuo laiko momentu pasiekti reikalingus dokumentus. Todėl dažniausiai susiduriama su problema, kad visi darbuotojai gali koreguoti

dokumentą ir taip pažeidžiamas dokumento pilnumas. Yra sukurta galimybė kontroliuoti koreguojamus dokumentus, leidžiant tik vienam žmogui koreguoti vienu metu, bet tai kitiems sistemos vartotojams sudaro diskonforta, kai jie negali atlikti savo darbų susijusių su dokumento koregavimu ir turi laukti kol baigs korekcijas pirmas žmogus. Taip pat būna, kad korekcijas atlikęs žmogus pamiršta patvirtinti padarytas korekcijas ir padaryti dokumentą prieinama kitiems sistemos vartotojams taip sudarydamas nereikalinga laiko gaišatį.

Šiai susidariusiai problemai išspręsti geriausiai gali padėti veiklos procesų vykdymu grindžiamos sistemos. Veiklos procesų vykdymu grindžiamos sistemos nėra grindžiamos informacijos pasiekiamumu. Jos stengiasi visus darbus suskirstyti į procesus, kuriuos turi atlikti pavieniai sistemos vartotojai ir tokiu būdu yra išvengiama laiko gaišaties, dėl to kad nėra galimybės pasiekti ir koreguoti reikiamą dokumentą.



2.2 pav. Veiklos taisyklių vykdymu pagrįsta IS remiantis [22]

Kaip pavaizduota 2.2 paveiksle, informacinė sistema veikia vartotojui atsiųsdama užduotį, kurią jis turi padaryti ir kai ją jis atlieka, užduoties atlikimas yra toliau siunčiamas kitam sistemos vartotojui ir taip visa užduotis yra atliekama padarius dalį užduoties ir persiunčiant gautą rezultatą kitiems sistemos vartotojams. Sistemai veikiant veiklos procesu vykdymo pagrindu nėra galimybės keliems darbuotojams naudotis vienu dokumentu tuo pačiu laiku. Ir tai sutrumpina vykdomų darbų laiką reikalingą jiems atlikti, nes darbai yra vykdomi negaišinant komandos narių, kurie tą užduotį atlieka.

3 TYRIMO TIKSLAS

Populiarinti darbų sekų vykdymu grindžiamas informacines sistemas ir tokių sistemų kūrimą, parodant jų kūrimo pranašumą.

3.1 Uždaviniai

1. Išanalizuoti darbų sekomis vykdomų informacinių sistemų kūrimo metodikas
2. Išanalizuoti darbų sekų vykdymą palaikančių sistemų veikimą
3. Naujos, darbų sekomis vykdomų informacinių sistemų kūrimo metodikos, ar rekomendacijų sudarymas
4. Suprojektuoti, sukurti ir ištestuoti sistemą pagrindžiančią sukurtą metodiką.

4 ANALIZĖS TIKSLAS

Išanalizuoti šiuo metu pateikiamus literatūros šaltinius, egzistuojančius sprendimus kilusiai problemai spręsti. Išanalizavus literatūros šaltinius ir egzistuojančius programinius sprendimus, atlikus jų lyginamąją analizę, bus galimą pagal gautus rezultatus pastebėti geriausias visų metodų puses ir panaudojimo atvejus, kuriuos bus galima aptarti kuriant savąja metodiką.

5 ANALIZĖS METODAI

Atliekant tyrimo analizę bus panaudota mokslinės literatūros analizės metodas, kurio pagalba bus išsiaiškinti darbų sekomis grindžiamų informacinių sistemų kūrimo metodai. Taip pat bus naudojamas stebėjimo metodas, kurio metu bus bandoma stebint informacinės sistemos kūrimo eigą analizuoti, kaip pasikeistų kūrimo eiga jeigu pritaikytume mokslinės literatūros analizės metu rastus kūrimo metodus.

6 TYRIMO OBJEKTO METODŲ ANALIZĖ

Remiantis [14] pateiktais duomenimis, kokios priežastys dažniausiai būdavo, dėl ko projektai būdavo nesėkmingi ir joms išspręsti labiausiai gali pasitarnauti tinkamai pasirinktas sistemos kūrimo metodikas.

Kiekviena metodika pasirinkta gali būti kažkiek pritaikoma esamos komandos kompetencijoms išryškinti ir trūkumams sumažinti. Ne visos metodikos gali būti pritaikomos kiekvienam projektui, nes viena metodika labiau orientuojasi į vieną kryptį, kaip dokumentavimą ir sistemos projektavimą. Kiti labiau į greitą produkto kūrimo procesą, mažiau dėmesio skirdami projektavimui ir reikalavimų analizei.

6.1 DSDM metodika

DSDM [23] paremtas **RAD** metodikos pagrindu naudojantis nuolatinį vartotojo įtraukimą į laipsnišką sistemos kūrimo procesą, kuris leistų tinkamai reaguoti į besikeičiančius reikalavimus ir juos suvaldyti, kad būtų sukurta sistema, atitinkanti visus verslo poreikius neviršijant nei laiko, nei biudžeto.

6.1.1 DSDM metodikos principai

Kertiniai principai ir pagrindinės taisyklės **DSDM** metodikos teisingam panaudojimui:

1. Tiek sistemos vartotojas, tiek sistemos kūrėjas dirba išvien, kad būtų kuo geriau sukurta sistema. Kai kartu prie sistemos kūrimo dirba kūrėjas ir naudotojas, galima daug efektyviau ir greičiau kurti sistemą, nes vartotojas matydamas sistemą gali greitai išreikšti kylančius reikalavimus ar jų neatitikimus.
2. Kūrimo komanda turi priimti sprendimus, svarbius projekto tinkamam progresui, nelaukiant patvirtinimo iš vadovų.
3. DSDM orientuojasi į dažną sukurto produkto pristatymą klientui su mintimi, kad geriau anksčiau pristatyti produktą „pakankamai gerą“ nei „puikų“ projekto gale. Pristatinėjant produktą dažniau nuo anksčiausios kūrimo proceso iteracijos, taip produktas gali būti testuojamas ir greičiau pastebėtos padarytos klaidos, kurios sekančios iteracijos metu būtų pataisomos. Sistemos dokumentacija galėtų būti pateikiama kiekvienos iteracijos metu su padarytais pakeitimais.
4. DSDM pagrindiniai produkto atlikimo kriterijai yra ne įgyvendinti visus verslui reikalingus poreikius, o padaryti tik tuos, kurie yra patys svarbiausi.

5. Kūrimo procesas yra iteracinis kas kartą klientui pateikiant papildytą sistemos versiją ir kūrimo iteracijos metu yra atsižvelgiamą į iš kliento gautas pastabas taip bandant kuo geriau įgyvendinti kliento lūkesčius.
6. Visi kūrimo metu padaryti pakeitimai yra paprastai atstatomi.
7. Pagrindiniai sistemos reikalavimai ir projekto apimtys prieš prasidedant projektui turi būti nustatyta.
8. Bendravimas ir bendradarbiavimas tarp visų projekto dalyvių yra būtinas, kad būtų pasiektas tikslas.
9. **20% / 80% taisyklė** – DSDM metodika teigia, kad per 20% projekto laiko įmanoma padaryti 80% projekto plano. Todėl DSDM metodika orientuojasi į tą 80% projekto plano, teikdama kad per likusį laiką išaiškės likę reikalavimai ir bus galima įgyvendinti likusius 20% plano.

6.1.2 DSDM metodikos naudojimosi taisyklės

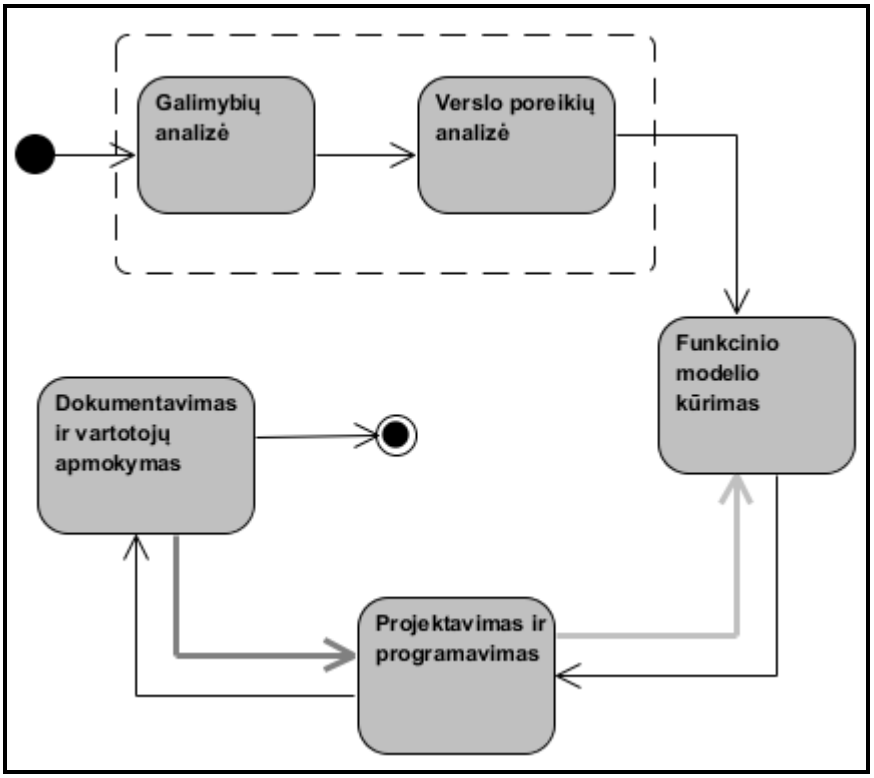
Tam, kad naudojant DSDM metodiką vykdomas projektas būtų sėkmingas, turi būti keletas sąlygų įvykdytų. Pirmiausia, turi būti nuolatinis bendravimas tarp kūrimo komandos ir būsimų sistemos vartotojų. Taip pat, turi būti tinkamas projekto valdymas, nes tai yra viena iš priežasčių, kodėl daugybė projektų žlunga. Kita sąlyga, kuri turi būti įgyvendinta, yra, kad visas projektas turi būti suskaidomas į kuo mažesnes veiklas, kurias būtų paprasta prioretizuoti ir būtų galima iteratyviai jas vykdyti, nes nepakankamai išsmulkintos veiklos dažniausiai būna priežastys, kodėl projektuose atsiranda vėlavimas.

Projektai, kuriuose yra perpanaudojamas kodas yra nelabai suderinama su DSDM metodika, nes metodika reikalauja tobulumo ir tai dažniausiai kertasi su metodikos svarbiausia taisykle 80%/20%, kuri buvo anksčiau aprašyta.

6.1.3 Kūrimo etapai naudojant DSDM metodiką

DSDM metodikoje aprašyti 3 informacinės sistemos kūrimo etapai. Tai yra pasiruošimas projektui, projekto gyvavimo ciklas ir projekto palaikymas. Žemiau pateikiamas 6.1 paveiksle.

Naudojant DSDM metodiką, kūrimo procesas susidaro iš 7 kūrimo proceso fazių. Pirmoji fazė daroma pačioje projekto pradžioje, kaip analizuojamos projekto galimybės ir verslo poreikiai. Toliau seka 3 fazės susijusios su projekto prototipo kūrimu, projektavimu ir programavimu, dokumentavimu ir vartotojų apmokymais. Ir galiausiai vyksta projekto palaikymas, kai sistema yra galutinai įdiegiama kliento kompiuteriuose ir atliekamas realus sistemos testavimas.



6.1 pav. DSDM metodikos veikimo procesas remiantis [23]

Trečiajame paveikslėlyje matome trijų skirtingų spalvų rodykles tarp proceso veiklų. Plonos rodyklės parodo, kaip turėtų projektas idealiu atveju vykti. Šviesiai pilkos rodyklės parodo projekto veiklos kryptis, jeigu iškyla kažkokios problemos, o tamsesnė rodyklė parodo kryptį, tik tuo atveju jeigu jau suprogramuotas produktas neatitinka reikalavimų.

6.1.3.1 Pasiruošimas projektui naudojant DSDM metodiką

Šio etapo metu yra pasiruošiama projekto vykdymui. Paruošiamas projekto finansavimo planas, taip pat suderinamas įsipareigojimų dokumentas, kuris vėlesniuose etapuose yra labai svarbus norint tinkamai toliau valdyti projektą.

Atliekamos dvi analizės šiame etape: sistemos galimybių analizė ir verslo poreikių analizė. Galimybių analizės metu analitikai turėdami pagrindinius reikalavimus bando išsiaiškinti ar su turimais resursais ir nurodyta projekto trukme spėtume atlikti. Ši analizė atliekama kiek galima greičiau, nes DSDM metodika naudojama, kai projekto vykdymui yra skirtas mažas laiko intervalas. Verslo poreikių analizės metu analitikai stengiasi išsiaiškinti visus verslo aspektus projekte ir atsakyti į klausimus:

1. Ar šis sprendimas turės įtakos verslui?
2. Kokie yra verslo dalyviai, kurie naudosis sistema?
3. Kokie yra tinkamiausi verslo procesai?
4. Kuriuos verslo procesus reikia realizuoti, testuoti, įdiegti ir palaikyti?
5. Kokias technologijas naudoti kuriant ir diegiant?

6.1.3.2 Projekto gyvavimo ciklas naudojant DSDM metodiką

Atlikus pirmajame etape galimybių ir verslo analizę pereinama prie sistemos kūrimo. Šiame etape kūrimo procesas vyksta 3 etapais: funkcinio modelio kūrimas, projektavimas ir programavimas, dokumentavimas ir vartotojų apmokymas.

1. **Funkcinio modelio kūrimas** – šio etapo metu yra kuriamas sistemos funkcinis prototipas, kuris yra pateikiamas klientui įvertinti. Kuriamas prototipas turi padėti

greičiau išsiaiškinti, kokių funkcijų reikia klientui ir kaip jos turi veikti. Pateikiant prototipą klientui yra išsiaiškinama ar prototipas gali būti tinkamai naudojamas, ar paprasta juo naudotis, ar sistema pajėgi aptarnauti visus sistemos vartotojus.

2. **Projektavimas ir programavimas** – šio etapo metu sistema yra projektuojama ir kuriama iteracijomis, o suprogramavus testuojama.
3. **Dokumentavimas, vartotojų apmokymas** – šio etapo metu atliekamas sukurtos sistemos dalies dokumentavimas ir vartotojo naudojimosi gido darymas. Taip pat šiame etape atliekamas sistemos dalies priėmimo testavimas peržiūrint ar suprogramuota dalis atitinka visus nusistatytus reikalavimus ir funkcionalumą, kurio reikia. Pristatant padarytą dalį klientui yra apmokomi kliento pusės vartotojai, kaip naudotis sukurta sistemos dalimi.

6.1.3.3 Sistemos palaikymas naudojant DSDM metodiką

Šio etapo metu sukurta sistemos dalis yra peržiūrima ir tikrinama ar veikia teisingai ir tinkamai. Pastebėtos klaidos yra taisomos, jeigu rastos klaidos yra susijusios su padaryta klaida ankstesniuose žingsniuose ši iteracija gali būti grąžinta, į bet kurį prieš tai buvusi etapą.

6.1.4 Kaip DSDM padeda išvengti sunkumų

Daugybė kūrimo metu atsiradusių sunkumų gali privesti projektą prie žlugimo. Pateikiama keletas būdų, kuriuos siūlo DSDM metodas sunkumams išvengti.

1. **Nesugebėjimas įvykdyti / išspęsti problemos, kuriai spręsti buvo kuriama sistema** – DSDM metodika suteikia galutiniams sistemos vartotojams nuolatos nuo pat pirmo kūrimo etapo testuoti sistemą. Tokiu būdu vartotojas testuodamas greičiau duodą grįžtamąjį ryšį kūrėjams apie sistemos veikimo tinkamumą ir poreikių atitikimą.
2. **Kaina būna didesnė nei sistemos teikiama nauda** – baigus DSDM verslo poreikių analizę projekto pradžioje žymiai sumažina galimybę sulaukti tokios pasekmės kuriant sistemą.
3. **Prastas bendradarbiavimas tarp projekto susijusių šalių** – DSDM metodika stengiasi, kad tarp visų suinteresuotų projekto šalių būtų nuolatinis bendradarbiavimas.
4. **Vartotojui per sunku ja naudotis, nes ji veikia ne taip kaip vartotojas tikisi** - DSDM suteikia galimybę viso programavimo proceso metu leisti vartotojams testuoti sistemą ir taip gauti vartotojo atsiliepimus iš karto, kurie leistų išvengti tokių sistemos problemų.
5. **Trūkumai vartotojo sąsajoje** – DSDM prototipo kūrimo etapas leidžia tokių problemų išvengti, nes galutiniam vartotojui taip pat yra pateikiama ir vartotojo sąsaja, kurią jis matys realizavus sistemą.
6. **Sistemos nelankstumas ir sudėtingumas prižiūrint** – naudojant DSDM metodiką ši problema negali įvykti, nes metodika orientuojasi į sistemos lankstumą.

6.1.5 DSDM komandos rolės

DSDM metodika apibrėžia komandos roles, kurios reikalingos tinkamam projekto valdymui.

1. **Ambasadorius** (angl. Ambassador) – žmogus, kuris atlieka jungiamąjį darbą tarp kliento ir kūrimo komandos. Jis koordinuoja kūrimo komandos darbą ir turi gerai žinoti, kaip kuriama sistema turėtų veikti.

2. **Aiškiaregys** (angl. Visionary) – pagrindinė varomoji projekto jėga padedanti išlikti kelyje siekiant projekto tikslų. Dažnai šis žmogus ir būna pradėjęs/ sumanęs projektą.
3. **Patarėjas** (angl. Advisor)– žmogus, kuris turi praktinės patirties srityse, kurios yra kompiuterizuojamos arba išmano technologijas, kurių pagalba kompiuterizuojama.
4. **Techninės srities koordinatorius** (angl. Technical Co-ordinator**)** – atsakingas už daugelį sistemos techninių aspektų ir užtikrina korektišką ir tinkamą techninės pusės veikimą.
5. **Sponsorius** (angl. Executive Sponsor**)** – tai asmuo, kuris skiria projektui reikalingas lėšas.
6. **Projekto vadovas** (angl. Project manager) – prižiūri sistemos prototipo kūrimą ir galutinės sistemos realizacija.
7. **Komandos vadovas** (angl. Team leader)– valdo komandos narių poreikius ir užtikrina, kad visa komanda dirbtų, kaip viena komanda.
8. **Vyriausias programuotojas** (angl. Senior developer)– žmogus, kuris turi patirties ir žinių apie projekto planą ir reikalavimus paversti veikiančiu kodu.
9. **Dokumentuotojas** (angl. Scribe) – žmogus, kuris atsakingas už sistemos dokumentavimą.

6.1.6 MoSCoW prioritizavimas

Sistemos funkcionalumo svarba, nusakoma pagal priskirtus prioritetus. Iš viso yra keturios grupės: privalo būti, turėtų būti, galėtų būti ir gali ir nebūti.

1. **Privalo būti** (angl. Must have) - tai funkcionalumas, kuris privalo būti įgyvendintas, nes tai yra labai svarbios funkcijos.
2. **Turėtų būti** (angl. Should have) – tai funkcijos, kurios yra svarbios verslui.
3. **Galėtų būti** (angl. Could have) – tai funkcijos, kurios reikalingos, bet sistema gali būti padaryta ir be jų, bet ateityje būtų planuojama jas pridėti.
4. **Gali ir nebūti** (angl. Won‘t have) – funkcijos, be kurių visiškai nesunku dirbti ir galima jų pridėjimą nukelti neribotam laikui.

Prioritizavimas sistemos kūrime yra labai svarbus dalykas, nes neturime pakankamai laiko viskam padaryti laiku. Todėl suprioretizavus lengviau atsirinkti funkcijas, kurias būtina atlikti.

6.2 Crystal metodikų šeima

Crystal metodiką [18] yra nuspręsta laikyti „lengvasvore metodika“ informacinėms sistemoms kurti. Crystal pavadinimas parinktas dėl panašumo į brangakmenį. Panašumas atsiranda pažvelgus į brangakmenį ir pamatomos jo sienelės. Kiekviena sienelė Crystal metodikoje atitinka informacinės sistemos naudojamas technikas, įrankius, standartus ir reikalingų darbuotojų roles.

Cockburn sukūrė keletą skirtingų metodikų, kurios patenkintų poreikius, skirtingų dydžių komandoms, sprendžiant skirtingas problemas. Crystal metodikų šeima naudoja skirtingas spalvas tam, kad būtų paprasta atskirti, kam skirta metodika. Kaip pavyzdys, jeigu projektas yra mažas galima naudoti Clear (bespalvė), Orange (oranžinė) ar Yellow (geltona) metodikas. Jeigu projektas yra labai svarbus, gali būti naudojamos Diamondor (deimantinė) ar Sapphire (safyrinė) metodikos.

Šiuo metu Crystal metodikų šeimoje yra 8 metodikos: Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Orange Web, Crystal Red, Crystal Maroon, Crystal Diamond, Crystal Sapphire.

6.2.1 Crystal šeimos metodikų principai

Tarp visų Crystal šeimos metodikų yra 7 vyraujančios bendros principai. Cocburn nustatė, kad kuo daugiau šių principų yra laikomasi projekte, tuo didesnė tikimybė, kad projektas pavyks. Šie septyni principai yra:

1. **Dažnas atlikto darbo pristatymų** – dažno pristatymo idėja paimta iš agile metodikų. Projektuotojai ir programuotojai nusprendžia, kurias sistemos funkcijas sukurti šios iteracijos metu, tada susidaro testavimo planus kiekvienos realizacijos testavimui. Iteracijų trukmė gali kisti nuo savaitės iki ketvirčio, priklausomai nuo projekto dydžio ir trukmės. Išleidžiant dažnai iteracijas būsimi sistemos vartotojai gali greičiau pastebėti sistemos trūkumus ir taip sutaupyti daug laiko juos taisant. Taip pat klientas gali greičiau pamatyti, kai projektas pradedamas vykdyti ne taip, kaip buvo tartasi. Iteracijų skaičius projekto metu nėra nurodytas, bet kuo jų daugiau, tuo paprasčiau kontroliuoti projektą.
2. **Sistemos patobulinimai atliekami atsižvelgiant į komentarus gautus** - atsižvelgimas į pateiktus komentarus, kaip turėtų būti daroma, priverčia programuotojus atsitraukti nuo paprasto programavimo ir pasižiūrėti į tai kitaip. Apgalvoti, kaip reiktų padaryti, kad veiktų atsižvelgiant į komentarus.
3. **Dažnas komandos komunikavimas** – bendravimas komandoje yra labai svarbus. Remiantis metodika, geriausia, kad kūrimo komanda dirbtų viename kabinete ir iškilus klausimams nereiktų niekur eiti, nes vaikštant kartais būna pametamos kitos idėjos. Kai visa komanda dirba viename kabinete, paprasčiau uždavinėti klausimus, nes klausimus girdi visa komanda ir atsakymus gali visi pateikti arba pataisyti, ir visi tada žinotų visą informaciją. Dirbant vienam šalia kito patogu ir tai, kad galima, bet kada pasiklausti klausimo, kuris iškilo.
4. **Asmeninė sauga** – tai yra susiję su komandos narių pasitikėjimu kitais ir savimi, kalbant prieš kitus kolegas, klausiant klausimus ar keliant turimas idėjas.
5. **Susitelkimas** – Crystal metodikoje komandos nariai turi susitelkti į du dalykus: tai yra jiems paskirtą užduotį, kad spėtų ją atlikti laiku ir į tai, kaip vyksta visas projektas. Remiantis metodika, programuotojai per dieną turėtų gauti dvi valandas, kai jų niekas negali pertraukinėti ir jis galėtų visiškai susikoncentruoti į darbą. Taip pat projekto vadovas turėtų.
6. **Lengvas sistemos priėjimas ekspertams prie sistemos, kūrimo metu** – šis metodas verčia programuotoją dirbti kartu su srities ekspertu, kuris galėtų atsakyti į visus kylančius klausimus ir padėti išspręsti kylančias problemas.
7. **Techninė aplinka su automatiniais testais ir konfigūracijų valdymas, bei dažnas versijų diegimas ir integravimas** – pagrindinė mintis šiame veiksme yra ta, kad nuolatos vis integruodami ir testuodami versijas, galime anksčiau pastebėti esančias klaidas sistemos. Kol tai yra daroma, rastas klaidas yra lengviau pataisyti, nei tai reiktų daryti, kai sistema visiškai baigta kurti.

6.2.2 Crystal Clear metodika ir kūrimo ciklas

Šis metodas skirtas mažiausios apimties projektams valdyti. Naudojant šia metodiką informacinę sistemą kuria viena nedidelė komanda, kuri sėdi viename kambaryje. Komandą sudaro: sponsorius, vyr. projektuotojas - programuotojas, projektuotojas – programuotojas ir vartotojas.

Kuriama sistema klientui pristatoma palaipsniui, iteracijomis, vis ką nors padarant naujo ir ištaisant pastebėtas klaidas. Iteracijos trukmė nuo 2 savaičių iki 3 mėnesių. Visas projekto vykdymo procesas yra prižiūrimas pildant projekto plano dokumentus ir sistemos dokumentacijas. Kaip ir anksčiau aprašyta yra vykdomas nuolatinis vartotojo įtraukimas į sistemos kūrimą, prašant, kad jis naudotųsi sistema ir praneštu apie pastebėtus trūkumus ir neatitikimus.

Crystal clear (bespalvė) metodika reikalauja, kad būtų kuriama sistemos dokumentacija. Kas turėtų būti dokumentacijoje rašoma, tiksliai metodikoje nenurodoma ir paliekama laisvė patiems dokumentacijos rengėjams.

6.2.3 Crystal Orange metodika ir komandos rolės

Ši metodika [20] skirta didesniems projektams vykdyti. Kūrimo komanda sudaryta nuo 10 iki 40 prie projekto dirbančių asmenų. Projektai, kurių vykdymui naudojama ši metodika, trunka nuo 1 iki 2 metų. Taip pat vykdant projektą jį atlikti kuo pigiau ir kuo greičiau.

Projekto komanda sudarytą iš: sponsoriaus, verslo eksperto, panaudojimo eksperto, verslo procesų analitiko, projektų vadovo, sistemų architekto, sistemos dizainerio, vyr. projektuotojo – programuotojo, projektuotojo – programuotojo, vartotojo sąsajos projektuotojo, dokumentuotojo, testuotojo. Visi šie skirtingų specializacijų darbuotojai suskirstyti į kelias skirtingas komandas tai: sistemos planavimo, projekto priežiūros, sistemos architektų, testavimo, sistemos infrastuktūros, funkcijų ir technologijų.

Didžiausia funkcijų komanda išskirstoma į grupeles pasinaudojant holistine įvairovės strategija, (holistic diversity strategy) aprašyta Cocburn‘o 1998 metais. Kiekviena nauja grupelė sudaryta iš verslo procesų analitiko – projektuotojo, vartotojo sąsajos projektuotojo ir nuo 1 iki 3 programuotojų – projektuotojų. Taip pat grupelėje yra duomenų bazės projektuotojas ir testuotojas.

Kuriamas produktas yra sudarytas iš reikalavimų dokumento, būsenos aprašo, vartotojo grafinės sąsajos dokumentacijos, kiekvienos grupelės kuriamos vienos dokumentacijos, vartotojo vadovo, programos kodo, testavimo atvejų ir migracijos kodo.

6.2.3.1 Crystal Orange (oranžinė) kūrimo procesas

Projektas kuriamas suskaidžius visas projekto užduotis į iteracijas, kurių trukmė yra nuo 2 iki 4 savaičių. Pasibaigus iteracijai komanda susitinka, kad aptarti, kaip pavyko realizuoti iteracijos užduotis, kas ne taip pavyko, kaip tikėtasi ir ką reikėtų padaryti kitos iteracijos metu.

Crystal Orange metodika stengiasi suorganizuoti viską: kas ką darys projekto metu, kad nebūtų persidengimų ar tuščių vietų atliekamų užduočių sąraše. Kiekvienas komandos narys gauna tam tikras užduotis, kurias reikia atlikti. Visas procesas suskirstomas į kelis etapus:

1. Verslo atstovas padaro verslo procesų ir sistemos panaudojimo atvejų dokumentaciją pirminį galimo projekto aprašą.
2. Remiantis padaryta dokumentacija technologijų komandos nariai įvertina, kiek reikės įdėti darbo, kad sukurti reikiamas funkcijas. Komandos vadovai atsižvelgdami į panaudojimo atvejus, nustato kokia galėtų būti visų darbų kaina, kurią praneša klientui prieš galutinai sutariant vykdyti projektą. Sutarus kainą, vartotojo sąsajos projektuotojai pradeda darbą su marketingo ir verslo procesų komandomis, kad sukurti puslapio prototipą, kuris apimtų visus reikiamus reikalavimus.
3. Verslo procesų analitikai padaro detalius panaudojimo atvejų ir naudojamų duomenų aprašus, kuriuos pateikia vartotojo sąsajos kūrėjams ir programuotojams. Programuotojai dirba su vartotojo sąsaja prie jos prijunkdami sukurtus atskirus panaudojimo atvejus. Tuo metu kiti programuotojai kuria regresinius testus parašytam

kodui testuoti. Kai parašytas kodas praeina regresinius testus, tada jis siunčiamas testuotojams, kurie patikrina kodą, ištestuoja visus testavimo atvejus ir baigę testavimą siunčia visas likusias klaidas programuotojams, kad jie jas ištaisytų.

4. Ištaisius likusias klaidas padaryta sistemos dalis integruojama su jau esama ir atliekami iš naujo testavimai, tikrinant ar po integracijos neatsirado naujų klaidų.
5. Kai po integracijos atlikus testus klaidų nerandama, sistema duodama vartotojams testuoti ir jeigu jie randa klaidų sistemoje, yra sudaryta SWAT komanda, kuri kas dvi iteracijas keičiasi ir greitai sprendžia iškilusias klaidas.

6.3 Agile Scrum metodika

Agile Scrum [4] yra iteratyvus ir priaugantis informacinių sistemų kūrimo metodas. Scrum ne tik pakeitė projektų vykdymą, bet ir pakeitė nusistovėjusias idėjas apie informacinių sistemų kūrimo metodikas. Scrum susikoncentravo į projektų vykdymą, kur yra suplanuoti į priekį. Scrum informacinių sistemų kūrime pirmą kartą aprašytas „New Product Development Game“ (Harvard Business Review 86116:137–146, 1986).

6.3.1 Agile Scrum principai

Taikant Scrum projekto vykdymo valdymui orientuojamasi į principus:

1. Daugiausia dėmesio kūrimo periodu skiriama patenkinti klientą reguliariai jam parodant vis naujai sukurtos sistemos dalį;
2. Ramų reagavimą klientui pateikiant naujus reikalavimus net ir sistemos kūrimo pabaigoje. Taip stengiamasi padidinti kliento konkurencingumą įsidiegus sistemą;
3. Skatinamas verslo atstovų ir kūrėjų glaudus bendradarbiavimas viso kūrimo proceso metu, kad užtikrinti kuo geriau atliktą darbą;
4. Pats efektyviausias ir naudingiausias bendravimas yra bendravimas gyvai;
5. Pagrindinis matas, kuriuo matuojamas atliktas darbas yra padarytas progresas;

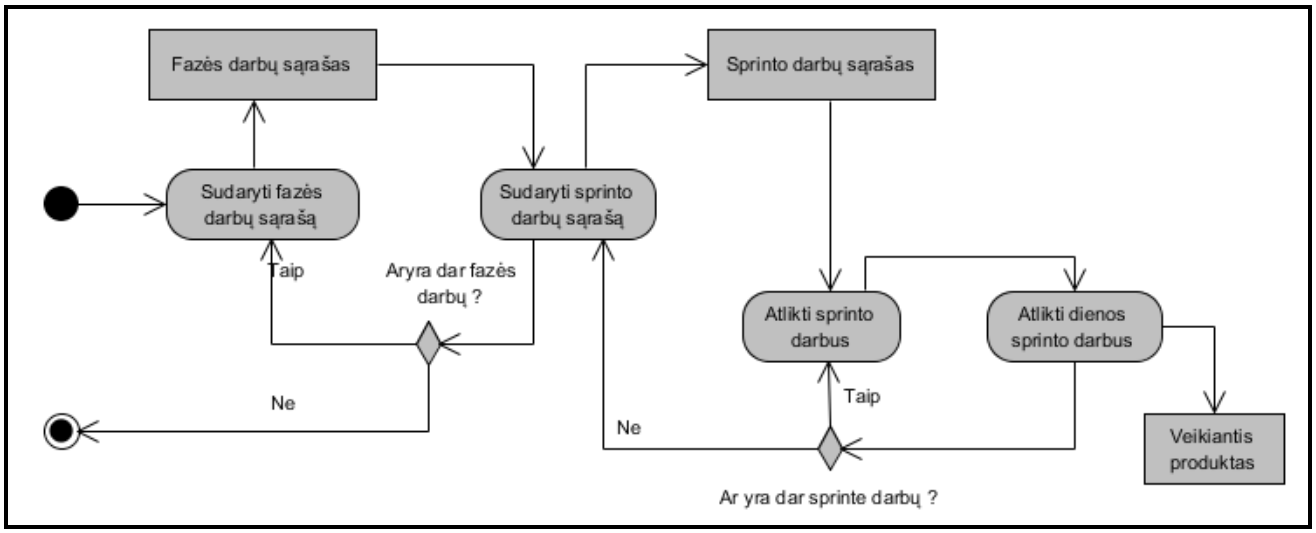
6.3.2 Projekto komandos sudėtis naudojant Agile Scrum metodiką

Projekto komandą sudaro 3 pagrindinės rolės ir spektras papildančių rolių darbuotojų.

1. **Produkto savininkas** – produkto savininkas atstovauja užsakovą ir yra jo balsas kuriant sistemą. Jis yra atsakingas už komandos kuriamos sistemos teikiamą naudą klientui. Produkto savininkas kuria darbų sąrašus ir juos talpiną produkto darbų sąraše. Scrum komanda turi turėti tik vieną produkto savininką.
2. **Kūrimo komanda** – kūrimo komanda yra atsakinga už veikiančio produkto pristatymą kiekvienos iteracijos pabaigoje. Komandą sudaro nuo 3 iki 9 darbuotojų, kurie ir atlieką pagrindinį darbą sistemos kūrime. Kūrimo komanda atlieką sistemos analizę, projektavimą, programavimą, testavimą, sistemos architektūros realizavimą ir dokumentacijos kūrimą. Komanda yra savarankiška ir jos darbui nereikalingas projekto vadovų įsikišimas. Dažniausiai komandos nariai yra kelių sričių specialistai ir gali atlikti kelis skirtingus darbus, taip atsiranda profesijų persidengimas, kuris padeda, kai yra reikalinga pagalba kitam komandos nariui atlikti jo darbą.
3. **Scrum meistras** – scrum meistras rūpinasi, kad komandos darbui būtų kuo mažiau trukdžių. Taip pat rūpinasi, kad projektas vyktų, kaip buvo planuota.
4. **Projekto vadovas** – projekto vadovas rūpinasi darbuotojų interesais.
5. **Savininkas** – tai klientas, kuris užsakė projektą ir kuriam kuriama sistema sukurs tam tikrą sutartą naudą. Jie įtraukiami tiesiogiai į projekto veiklą tik tada, kai yra peržiūrima iteracijos metu atlikta pažanga.

6.3.3 Sistemos kūrimo gyvavimo ciklas naudojant Agile Scrum metodiką

Naudojant Scrum kūrimo karkasą sistemos kūrimo procesui valdyti, reikia projekto pradžioje padaryti sistemos detalų projektavimą ir darbų nustatymą. Scrum kūrimo procesas pavaizduotas 6.2 paveikslėlyje.



6.2 pav. Scrum karkaso procesas remiantis [6]

Pirmiausia, visa kūrimo komanda turėtų susidaryti projekto reikiamų įvykdyti sąrašą, kurio kiekviena dalis gali būti dar smulkinama iki atskyrų darbų ar užduočių. Tai vadinama - pradinių sprintų sudarymas. Kaip anksčiau minėta, kad kiekvienas sprintas gali būti sudarytas iš daugelio mažų užduočių, kurios yra sudedamos į sprinto darbų sąrašą. Šį sąrašą pasibaigus ankstesniam sprintui sudaro susirinkusi kūrimo komanda. Susitikimo metu komandos nariai apsitaria ką galima padaryti per sekanti sprintą, kokie darbai iš sąrašo sudėtingiausi ir kam skirti daugiausia laiko. Sudaromi sprintai trunka nuo savaitės iki daugiausiai vieno mėnesio. Pradėjus sprintą ir atlikinėjant sprinto darbus, komandos nariai susirenka kas dieną ir pasipasakoja, ką per vakar dieną nuveikė ir ką planuoja per šią padaryti, taip yra kontroliuojamas darbuotoju darbas, kad jie netinginiautų. Taip pat, kasdienių susitikimų metu, galima užsiminti apie iškilusius sunkumus atliekant užduotį ir po susitikimo su atitinkamu komandos nariu pasitarti, kaip problema galėtų būti išspręsta. Kasdieniai susitikimai neturėtų trukti ilgiau nei 15 min. Kaip minėta, sprintas turėtų trukti nuo vienos savaitės iki daugiausiai vieno mėnesio, po jo,klientui turi būti parodomi per sprintą atlikti darbai. Atliktas darbas turėtu būti veikiantis produktas, kurį klientas galėtų pabandyti.

6.4 RUP - Rational Unified Process metodika

RUP [7] tai ne vien pavienis adaptuotas procesų karkasas, tinkantis programinės įrangos kūrėjų komandoms. Komandos, pagal esančius poreikius, gali pasirinkti tam tikrus elementus iš procesų. Produktas apima susijusias žinias grindžiamas artefaktais ir daugelio skirtingų produkte veikiančių veiklų detalizuotų aprašymų. RUP įtrauktas į „IBM Rational Method Composer“ produktą, kuris leidžia derinti procesus. Suvienytas procesas buvo sukurtas, kad apimti viešą dalykinės srities procesą (žinomą, kaip jungtinis procesas) ir detalesnę specifikaciją, žinomą kaip „Rational Unified Process“, kurią galima pateikti kaip produktą.

6.4.1 Pagrindiniai RUP metodikos principai

Naudojant RUP yra laikomasi 6 pagrindinių principų, skirtų projektuojant sistemą sumažinti klaidų skaičių ir padidinti produktyvumą. Šie 6 naudingi principai yra: iteratyvus sistemos kūrimas,

reikalavimų valdymas, komponentinė architektūra, sistemos modeliavimas, sistemos testavimas, pasikeitimų sistemoje kontrolė.

1. **Iteratyvus sistemos kūrimas** – šiais laikais beveik neįmanoma sukurti sistemos, viską vykdant paeiliui, suplanuoti sistemą, ją suprojektuoti, suprogramuoti ir galiausiai ištestuoti ir atiduoti klientui veikiantį produktą. Šiuo metu patogiausia kurti sistemas mažais etapais, kurie leidžia geriau reaguoti į besikeičiančius reikalavimus. Kuriant sistemą naudinga ją kurti nedideliais etapais ir per etapą sukurtą sistemos dalį duoti klientui testuoti ir įvertinti padarytą pažangą ir pateikti komentarus apie atliktą darbą bei pasikeitusius reikalavimus. Taip kuriant sistemą yra sumažinama rizikos galimybė, kad projektas nepasiseks.
2. **Reikalavimų valdymas** - RUP procesas aprašo, kaip reikia surinkti, valdyti, kaip dokumentuoti ir sekti projekto eigoje besikeičiančius reikalavimus. Panaudojimo atvejų naudojimas renkant reikalavimus įrodė, kad tai yra labai veiksminga priemonė, nes tai leidžia užtikrinti, kad visi reikiami funkciniai reikalavimai yra surinkti, reikiamos dizaino dalys suprojektuotos. Testavimas naudojantis dokumentacijoje, pateikiama informacija leidžia greitai nustatyti ar vykdomas projektas teisinga kryptimi eina.
3. **Komponentinė architektūra** – taikant komponentinį sistemos kūrimą palengvinamas sistemos testavimas ir koregavimas radus klaidų. Taip pat esant reikalui daug paprasčiau yra perpanaudoti jau sukurtus komponentus, nei iš naujo juos kurti. Taip pat tai yra patogesnis kūrimo būdas didesnėms komandoms.
4. **Sistemos modeliavimas** – sistemos modeliavimas ir projektavimas naudojant UML įrankį. Šis įrankis padeda susimodeliuoti sistemos architektūrą ir jos veikimą grafinėmis priemonėmis. Susimodeliavus sistemą, grafiškai galima pamatyti, kaip sistemos dalys tarpusavyje bendrauja. Pasinaudojant grafiniais sistemos modeliais paprasčiau stebėti, kaip sistema realizuota ir kaip buvo sumodeliuota.
5. **Sistemos testavimas** – prastas sistemos veikimas ir žemas sistemos patikimumas yra vienas pagrindinių faktorių, kodėl sistemos nebūna priduodamos laiku ir tinkamai. Tam kad neįvyktų tokios klaidos, kuriama sistema turi būti nuolatos testuojama ir taisomos rastos klaidos, kad būtų pagerintas sistemos veikimas ir patikimumas. Testuojama turėtų būti remiantis susikurtais testavimo planais, kurie pasiruošiami remiantis sistemos dokumentacija.
6. **Pasikeitimų sistemoje kontrolė** – šiuo metu sistemoje padaryti pakeitimus nėra sunku, todėl atsiranda poreikis sekti visus daromus sistemoje pakeitimus. Dirbant prie vieno projekto keliems programuotojams yra poreikis kontroliuoti programuotojų atliekamus darbus sekti, kad nebūtų daromi pakeitimai kito programuotojo užduotyje.

6.4.2 Sistemos kūrimo gyvavimo ciklas naudojant RUP metodiką

RUP kūrimo karkasas yra ganėtinai panašus į krioklio metodą. RUP procesą sudaro keturios pagrindinės dalys, tai yra:

Pradinio etapo metu reikia nustatyti vykdomo projekto ribas, taip pat reikia nustatyti visas išorines sistemas ir kitas priemones su kuriomis sistema bendraus. Taip pat reikia nustatyti ir galimus sistemos rizikos taškus, kurie gali pakenkti projektui. Taip pat reikia nustatyti visus sistemos panaudojimo atvejus, nustatyti visus reikalingus resursus, kad būtų projektas įvykdytas ir svarbiausias projekto dalykas, kurį reikia pasidaryti tai susikurti projekto planą su pagrindiniais atsiskaitymų taškais ir jų datomis, kad būtų aiškūs darbų terminai.

Šio etapo metu taip pat pradedama kurti dokumentacija, kurioje aprašoma projekto reikalavimai, pagrindiniai sistemos funkciniai taškai ir pagrindiniai projekto apribojimai. Taip pat atliekamas pradinis projekto projektavimas naudojant įvairias projektavimo priemones. Sukuriama, jeigu reikia, ir keletas sistemos prototipų.

Šio etapo metu, tūrėtų būti padaroma apie 10 % - 20% sistemos panaudojimo atvejų, pradinis verslo proceso modelis. Taip pat pateikiamas pradinis rizikos atvejų planavimas. Paruošiamas pradinis projekto planas, kuriame pateikiamos projekto fazės ir iteracijos, pagal kurias bus dirbama ir pateikiami sistemos veikiančios versijos. Ir svarbiausia dalis, kad tūrėtų būti pateikiami keli sistemos prototipai, iš kurių klientas galėtų rinktis jam labiausiai patinkantį.

Baigiantis šiam etapui, galima įsivertinti, kaip pavyko įvykdyti etapo planus. Jeigu įvertinant etapo atliktus darbus ir nepalankiai įsivertinus etapo darbus gali būti, kad etapą reikia pakartoti arba nutraukti projekto vykdymą. Etapo rezultatui įsivertinti yra 5 kriterijai:

1. Klientas įvertina projekto etapo metu nuveiktą darbą peržiūrėdamas ir įvertindamas kainos suskaičiavimą ir plano sudarymą;
2. Reikalavimų supratimas iš pirminių panaudojimo atvejų;
3. Projekto kaštų ir vykdymo plano paskaičiavimo, prioritetų, rizikų ir kūrimo proceso patikimumas;
4. Sukurtų prototipų architektūrinis detalumas;
5. Realios išlaidos prieš planuotas išlaidas.

Pasirengimo etapo metu yra svarbiausi uždaviniai: išanalizuoti pagrindinę problemą, susikurti sistemos architektūrą, pasitobulinti projekto planą ir pasistengti eliminuoti didžiausias rizikas turinčias užduotis. Kuriant sistemos architektūrą turi būti jau žinoma sistemos apimtis, pagrindiniai funkcionalumai ir nefunkciniai sistemos reikalavimai.

Pasibaigus šiam etapui jau beveik būna baigti sistemos projektavimo darbai, surinkti ir suspecifikuoti sistemos reikalavimai, tiek funkciniai, tiek nefunkciniai. Turi būti jau sukurtas pasileidžiantis sistemos architektūrinis sprendimas. Kiekvienos iteracijos pabaigoje turėtų būti įvertinama, kaip pavyko iteracijos planus pasiekti ir kur dar reiktų pasitemti.

Šio etapo metu turėtų būti jau padaryta:

1. Apie 80% panaudojimo atvejų modelių sukurta, visi panaudojimo atvejai ir aktoriai turi būti identifikuoti ir dauguma panaudojimo atvejų aprašytų dokumentacijoje;
2. Sistemos architektūros aprašas paruoštas;
3. Veikiantis sistemos prototipas;
4. Peržiūrėti rizikų ir verslo procesų sąrašai;
5. Preliminarus vartotojo vadovas paruoštas;
6. Paruoštas kūrimo planas, kuriame pateiktos kiekvienos iteracijos užduotys ir kiekvienos iteracijos įvertinimo kriterijai.

Kūrimo etapo metu yra sukuriami sistemos atskiri komponentai ir jie integruojami į sistemą. Taip pat sistema nuolatos yra testuojama ir taip einama prie išbaigtos sistemos. Šio etapo metu turėtų būti koreguojami projektų planai atitinkamai į susiklosčiusias situacijas kuriant projektą. Esant dideliam projektui, galima padaryti, kad kelios komandos lygiagrečiai kurtų atitinkamas sistemos dalis, taip paspartinant sistemos kūrimą.

Pasibaigus šiam etapui, sistemos kūrimas baigiamas ir klientui duodama sistema testavimui. Tokia veikla vadinama - beta sistemos testavimu. Taip pat jau būna pabaigti daryti vartotojų naudojimosi gidai ir sistemos dokumentacija.

Šio etapo pabaigoje įvertinant, kaip pavyko projektas reikia atkreipti didžiausia dėmesį į tai ar sistema yra pakankamai išbaigta, kad būtų įdiegiama kliento naudojimui. Taip pat reikia įvertinti ar klientas yra pasiruošęs priimti sukurtą sistemą naudojimui. Vienas svarbiausių kriterijų vertinant projektus yra, kokia realioji projekto kaina lyginant su planuotąja kaina.

Pereinamojo etapo metu yra pabaigiama kurti sistemos realizacija, jeigu buvo kokių nors funkcijų, kurių kūrimas buvo sustabdytas. Taip pat, kliento testavimo metu rastų klaidų taisymas. Pabaigus visiškai kurti sistemą ir ją įdiegus kliento kompiuteriuose, galima kliento serverius perkelti į sistemos suprojektuotą duomenų bazę, bei apmokyti sistemos vartotojus. Taip pat galima pradėti naują projekto vykdymo etapą - palaikymas, kuris gali trukti neribotą laiką, kol klientas naudosis sistema. Palaikymo metu klientas gali paprašyti sukurti naują funkcionalumą sistemai, kurio jai trūksta.

6.4.3 RUP metodikos komandos rolės

RUP komandą sudaro:

1. **Sistemų architektas** – atsakingas už sistemos architektūros projektavimą ir paruošimą diegimui;
2. **Testuotojai** – atsakingi už sistemos klaidų radimą sistemoje ir jų taisymo sekimą;
3. **Programuotojai** – atsakingi už sistemos kūrimą;
4. **Verslo procesų analitikai** – atsakingi už kompiuterizuojamos įmonės verslo procesų analizę ir dokumentaciją, verslo procesų modeliavimą;
5. **Analitikai** – atsakingi už sistemos reikalavimų surinkimą ir analizę, sistemos projektavimą;
6. **Vartotojo sąsajos projektuotojai** – atsakingi už vartotojo sąsajos projektavimą ir kūrimą.

6.5 Metodikų apibendrinimas

Kaip ir kiekvienam dideliam ar mažam projektui vykdyti ir sėkmingai užbaigti yra reikalingas vykdymo karkasas, pagal kurį dirbant yra pasiekiami geri rezultatai. Taip ir kuriant darbų sekų vykdymų grindžiamas sistemas, reikalingas karkasas sėkmingam tokio projekto įgyvendinimui. Analizės metu buvo aptarti pagrindiniai šiuo metu naudojami karkasai tokio tipo projektams vykdyti, o gauti rezultatai pateikiami žemiau esančioje lentelėje.

Kiekviena lentelėje pateikiama metodika įvertinta pagal 4 atrinktus kriterijus, kurie atrinkti remiantis internetiniais šaltiniais rašančiais apie priežastis, kodėl dažniausiai žlunga informacinių sistemų kūrimo projektai. Atrinkti kriterijai yra - dokumentacijos svarba, projekto planavimas, reikalavimų valdymas ir komandos pilnumas.

Dokumentacijos svarba pasirinkta, nes remiantis parašytu **Frank Winters** straipsniu [23], kuriame teigiama, kad dažniausiai dėl dokumentacijos prasto pildymo ir kyla pagrindinės bėdos, kurios priveda projektus prie žlugimo. Neskiriant pakankamai laiko ir pastangų dokumentacijos pildymui, atsiranda bėdos dėl pasikeitusių reikalavimų, kuriems atsiradus funkcinė specifikacija ar sistemos nefunkciniai reikalavimai lieka neaprašyti, kas projektui einant į pabaigą gali iššaukti didelius poreikius keisti sistemos kodą, kas kainuotų daugybe pinigų ir laiko. Taip pat nepilna ir nevisiškai detaliai parašyta specifikacija gali sudaryti galimybę atsirasti nesusipratimams programuojant, kai programuotojas supratęs vienaip padaro visai kitaip nei buvo tikimasi aprašant. Tokios klaidos ateityje gali kainuoti taip pat daugybe laiko ir pinigų, nes reikės viską perdaryti.

Frank Winters tame pačiame straipsnyje [23] toliau rašo, kad taip pat didelė problema yra blogas projekto plano sudarymas ir tolimesnis projekto plano vykdymas ir esant reikalui jo

perplanavimas. **Autoriaus** teigimu [23] prastas projekto planavimas gali įvykti net gi pačioje projekto pradžioje, kai projekte reikia įvertinti rizikas ir sudaryti projekto veiksmų planą. Blogai įvertinus rizikas ir netinkamai sudėliojus projekto planą, galimą didžiausią riziką turinčias veiklas pasilikti projekto pabaigai ir taip atsiranda didelė tikimybė, kad nepavyks projekto įvykdyti laiku. Taip pat, sudarinėjant projekto planus svarbus kas tam tikrą laiko tarpą peržiūrėti projekto planą ir esant poreikiui jį perplanuoti, nes gal komandai pavyko greičiau dirbti nei buvo planuota ar komanda užstrigusi buvo prie kurios nors užduoties ir taip sugaišo šiek tiek laiko. Perplanavimas leidžia matyti realų projekto vykdymo grafiką, kuris parodo kiek dar liko funkcijų įgyvendinti, kol projektas bus baigtas.

Autorius [23] teigia, kad yra ir dar viena priežastis, kuri gali skatinti projektus žlugti. Tai yra informacinės sistemos reikalavimų pokyčiai ir reikalavimų pakeitimų valdymas. Sunku rasti informacinės sistemos projektą, kurio vykdymo metu nekistų reikalavimai. Reikalavimų pokyčių valdymas būtinas užtikrinti, kad naujai atsiradę reikalavimai būtų išanalizuojami, dokumentuojami ir galiausiai suprogramuojami. Kas kart atsiradus vis naujam reikalavimui, reikalinga jam priskirti prioritetą, kad būtų galima greičiau nuspręsti kada reiktų šį reikalavimą įgyvendinti, nes tik atsiradus naujiems reikalavimas juos vykdyti yra negerai, nes tai gaišina laiką ir atitraukia programuotojus nuo jų darbų, bei yra tikimybė, kad projekto eigoje jis gali pasikeisti. Pasikeitus reikalavimui būtų atliekamas nereikalingas darbas taisant kodą, kad atitiktų reikalavimus.

Paskutinis vertinimo kriterijus pasirinktas remiantis patalpinto **Zern Liew** straipsniu [25], kuris teigia kad komanda turi būti sukomplektuota taip, kad darbuotojai galėtų koncentruotis į vieną darbų specializaciją. Remiantis straipsniu [25], jeigu komandos nariai norėdami atlikti užduotį turi atlikti daugybę skirtingų darbų, kurių specifikos skiriasi, tai teigiama, kad darbuotojas negali visiškai tinkamai susikoncentruoti į užduotį ir dėl to nukenčia užduoties atlikimo kokybė. Taip pat dažniausiai darbuotojai turi vienoje srityje daugiau patirties ir kai jiems reikia atlikti užduotis, kurioms atlikti jis neturi pakankamai patirties, tai ta užduotis gali labai ilgai užtrukti, kol darbuotojas įsigilins į užduoties specifiką.

Lentelėje pateikiamų kriterijų ir metodikų vertinimui reikalinga vertinimo sistema, kuria remiantis būtų galima įvertinti aptartas metodikas pagal pasirinktus kriterijus. Pasirinkta vertinimo sistema labai paprasta:

1. „ - “ – vertinama, kai metodika neatitinka pasirinkto vertinimo kriterijaus. Įvertinimo vertė skaičiuojant galutinį balą lygi 0;
2. „ +- “ – vertinama, kai metodika atitinka vertinimo kriterijų, bet turi truputį trūkumų. Įvertinimo vertė skaičiuojant galutinį balą lygi 0,5;
3. „ + “ – vertinama, kai metodika atitinka vertinimo kriterijų. Įvertinimo vertė skaičiuojant galutinį balą lygi 1;

6.1 lentelė Projekto vykdymo karkasų apžvalga

Metodo pavadinimas	Komandos pilnumas	Projekto planavimas	Dokumentacijos svarba	Reikalavimų valdymas	Adaptacija darbų sekomis grindžiamų sistemų kūrimui	Galutinis
Agile Scrum	+	+	+-	+	+-	**4**
RUP	+	+	+	+	+-	**4,5**
DSDM	+-	+-	+	+-	+-	**3**
Crystal Clear	-	+-	+	+-	+-	**2,5**
Crystal Orange	+	+-	+	+-	+-	**3,5**

Pirmasis vertinimo kriterijus pasirinktas yra komandos pilnumas, tai yra kaip pilnai yra užpildomos visos reikalingos projekto vykdymui komandos rolės. Komandos pilnumas reikalingas, kad būtų atlikti visi reikalingi veiksmai, kad projektas pasiektų užsibrėžtą tikslą. Tam, kad būtų įvykdyta, komandoje reikalingi: projekto vadovas, VGS dizaineris, testuotojas, programuotojai, veiklos procesų analitikai, analitikai, dokumentuotojai ir savininkas, kuris rūpinasi, kad klientas neapleistų vykdomo projekto.

Remiantis įvairiais šaltiniais, komandas turėtų sudaryti savo srities specialistai, kurie geriausiai žino, kaip atlikti savo darbą. Jeigu komandoje yra sričių, kur vienos specializacijos žmogus atlieka dviejų skirtingų sričių darbus, jis bus nepajėgus atlikti tų darbų puikiai. Todėl yra svarbi komandos pilnatvė, kuri užtikrina visų reikiamų komandos specializacijų žmonių buvimą ir savo srities darbų vykdymą.

Atlikus analizę pasirinktose metodikose, analizuojant, kaip komandos pilnumo sąlygą išpildo metodikos, pastebėta, kad RUP, Agile Scrum ir Crystal Orange metodikos geriausiai atitinka komandos pilnumo sąlygas. Todėl, kad jos geriausiai atitinka kriterijų yra įvertinamos pliusu.

DSDM metodika gavusi įvertinimą „plius minus“, nes tarp aprašomų komandos rolių nėra paminima gan svarbių komandoje rolių, tai yra: testuotojas ir analitikas. Jų trūkumas galėtų pasireikšti silpnesniu reikalavimų išsiaiškinimu ir jų specifikavimu, taip pat programuotojams būtų sunkiau atlikti savo užduotis, nes nebūtų ko paklausti, kaip turėtų būti daroma ar išsiaiškintų programuotojams kilusius klausimus bendraujant su klientu. Testuotojo trūkumas atsilieptų produkto kokybėje, nes nebūtų kam tikrinti klaidas, remiantis esama dokumentacija ir paruoštais testavimo planais. Nors testavimus galėtų atlikti ir patys programuotojai, bet tai būtų nelogiška, nes tokiu atveju jie yra atitraukiami nuo tiesioginio darbo. Taip pat, testuotojas turėtų būti tarsi paprastas sistemos vartotojas, kuris žino apie sistemos veikimą tiek, kiek parašyta dokumentacijoje.

Mažiausią įvertinimą gauna Crystal Clear metodika, nes komanda sudaryta tik iš programuotojų – projektuotojų. Crystal Clear metodikoje nurodomos komandos narių rolės tinkamos tik vykdyti labai mažus projektus, kurie trunka trumpą laiką, nes programuotojai - projektuotojai patys turi atlikti sistemos reikalavimų analizę, projektavimą, programavimą ir testavimą. Kaip anksčiau minėta, atliekant kitų sričių specialistų funkcijas yra nesunku padaryti klaidų, kurios gali atsiliepti galutinio produkto kokybėje.

Gerai suplanuotas ir tinkamai vykdomas projekto planas gali būti raktas į sėkmę vykdant projektus. Remiantis [14] pateikta statistika, kodėl dažniausiai žlunga informacinių sistemų kūrimo projektai, tai net 39% projektų žlunga dėl blogo projektų planavimo ir vykdymo. Norint tinkamai valdyti projektą, reikia projekto pradžioje įsivertinus grėsmes susidaryti projekto planą ir jį projekto eigoje vis koreguoti, kad būtų galima sekti, kaip sekasi vykdyti, kur iškilo problemų.

Peržiūrėjus metodikas geriausiai galima įvertinti tik **Agile Scrum** ir **RUP** metodikas jas įvertinant pliusu. **RUP** metodika kreipia nemažą dėmesį projekto plano sudarymui ir vykdymui. Įsivertinusi galimas rizikas, pradedamas sudarinėti projekto planas su preliminariomis užduočių įvykdymo datomis, kurios sekančioje kūrimo fazėje. Kai yra baigiama informacinės sistemos problemos ir reikalavimų analizė ir galima tiksliai jau nurodyti užduočių atlikimo datas. Sekančiomis fazėmis, kūrimo projekto planas yra koreguojamas nežymiai. Kita pliusu įvertinta metodika yra **Agile Scrum**. Naudojant šią metodiką, pačioje projekto pradžioje yra sudaromas preliminarus projekto planas, pagal kurį ruošiamasi vykdyti projektą. Kiekvieno naujo sprinto pradžioje yra sudaromas sprinto planas ir nustatoma sprinto pabaigos data. Sprinto detalų planą galima laikyti tarsi projekto plano vieną iš sudedamųjų dalių. Likusios metodikos įvertintos „plius minus“, nes projektų

planai sudaromi projekto pradžioje ir projekto eigoje tik peržiūrimas planas, kad žinoti, kokioje padėtyje yra projektas.

Remiantis[14] pateiktais duomenimis, kodėl dažniausiai žlunga vykdomi IT projektai yra dokumentacijos trūkumas. Kai nėra ruošiama dokumentacija, sunku sekti projekto kintančius reikalavimus, sunkiau į komandą įsilieti žmogui, kai jis apie vykdomą projektą nieko nenusimano ir nėra jam apie ką pasiskaityti, kad žinotu apie ką projektas. Taip pat, pateikiant programuotojui užduotį, nėra kur nurodyti jam pasiskaityti plačiau apie užduotį ir kaip tai turėtų veikti. Testuotojai neturėdami dokumentacijos su reikalavimų specifikacija ir sistemos aprašu, negali susidarinėti testavimo planų.

Žemiausią įvertinimą gavo **Agile Scrum** metodika, nes metodikoje nėra kreipiama dėmesio į sistemos dokumentacijos kūrimą ir tobulinimą keičiantis sistemos reikalavimams. Metodika labiau orientuojasi į dokumentacijos kūrimą sau, kad žinotų kas padaryta, o ne į dokumentaciją, kurią būtų galima duoti vartotojams skaityti. Visos likusios aptariamos metodikos įvertinamos pliusu, nes projekto vykdymo metu skatina dokumentacijos kūrimą ir tobulinimą, kas padeda vykdant projektą matyti, kaip sukurta sistema, kas joje numatoma atlikti ir kaip tai planuojama atlikti. Taip pat, dokumentacijoje matoma informacija, kokie reikalavimai buvo surinkti analizės metu.

Taip pat svarbus punktas projektų valdyme yra reikalavimų valdymas projekto metu. Šį vertinimo kriterijų geriausiai atitiko **RUP** ir **Agile Scrum** metodikos, kurios įvertinamos pliusais. RUP metodika visus naujai atsirandančius ar senesnius reikalavimus keičiant yra stengiamasi iš karto ir dokumentacijoje pakeisti, kad visi reikalavimų pokyčiai būtų dokumentuoti ir būtų galima ateityje sekti, kaip keitėsi reikalavimai. Tuo tarpu **Agile Scrum** metodika nesistengia dokumentuoti besikeičiančių reikalavimų, kaip tai daro **RUP** metodika. Scrum metodika remiantis visus naujus reikalavimų pasikeitimus stengiasi sekančiu sprintu realizuoti, kas leidžia naujoje sistemos versijoje turėti visus naujausius sistemos reikalavimus realizuotus. **DSDM, Crystal Orange ir Crystal Clear** metodikos įvertintos plius minus įvertinimu, nes nėra tokios lanksčios reikalavimų pasikeitimams. **DSDM** metodika naudojant reikalavimų pokyčiai leidžiami, kol iki projekto pabaigos lieka apie 20% suplanuoto laiko, nes tikima pagal metodiką, kad per 80% laiko išsigrynins visi reikalavimai iki nurodyto laiko. **Crystal** metodikos reikalavimų pasikeitimus gauna iš pastabų pateikiant produktą klientui, kuris apžiūros ir testavimo metu pateikia komentarus. Neigiamas dalykas, kad programuotojus atitraukia nuo jų programavimo darbų ir reikalauja bendrauti su klientu, tam, kad išsiaiškinti, kaip reiktų tinkamai realizuoti sistemą, kad būtų realizuotas ir naujas reikalavimas.

Kaip vienas iš esminių vertinimo kriterijų yra metodikos galimybė pritaikyti ją sistemų kūrimui, kurių veikimas grindžiamas darbų sekų vykdymu. Visos apžvelgtos metodikos nesikoncentruoja į tokių sistemų kūrimą, todėl jos labiau pritaikytos paprastų sistemų kūrimui, nors atlikus ilgus adaptacijos darbus ir būtų galima metodikas pritaikyti prie tokių sistemų kūrimo.

6.6 Agile Scrum prieš RUP

Šios dvi metodikos dažniausiai yra taikomos kuriant informacines sistemas. Metodikos iš esmės yra labai panašios ir skiriasi tik keliais nežymiais aspektais. Vienas iš ryškiausių aspektų yra tas, kad **RUP** visas vykdymo procesas suskaidytas yra į 4 fazes, kai Scrum metodika visą procesą suskaido į smulkias užduotis, kurios susideda į darbų sąrašą ir kiekvienos iteracijos pradžioje susidaro darbų sąrašą, kurį reikia atlikti. Detalesnis metodikų palyginimas pateikiamas 6.2 lentelėje.

6.2 lentelė Agile Scrum ir RUP metodikų palyginimas remiantis [15]

	RUP	Agile Scrum
Kūrimo ciklas	Formaliai visas kūrimo ciklas sudarytas iš 4 fazių. Kai kurios fazių veiklos gali būti	Kiekviena iteracija yra dalis viso kūrimo ciklo

	ir kartojamos	
Planavimas	Projekto planavimas susietas su keletu iteracijų, kurių metu yra sudaromi projekto planai. Sudaromas planas turi keletą tarpinių tikslų.	Nėra sudarinėjami jokie projektų planai. Kiekvienos naujos iteracijos pradžioje yra nustatomi planai iteracijos. Ir klientas pasako kada projektas yra baigtas.
Apimtis	Projekto apimtys yra nustatomos projekto pradžioje ir aprašomos dokumentacijoje. Projekto eigoje projekto apimtys gali būti peržiūrimos ir tikslinamos.	Vietoje projekto apimties skaičiavimų naudojamas darbų sąrašas, kuris kiekvienos iteracijos pradžioje yra peržiūrimas.
Tikslas	Dokumentacija, funkcinių reikalavimų specifikacija, sistemos architektūros dokumentacija, kūrimo planas, testavimo planai, veikiantis produktas.	Vienintelis projekto tikslas yra pilnai veikianti informacinė sistema.

Įvertinus šias dvi metodikas galima prieiti išvados, kad naudojant Scrum metodiką projektai vykdomi nežinant kada jie bus baigti, kai RUP metodika projekto pradžioje sudaryto plano laikosi ir žino visas datas. Scrum metodika suteikia klientui laisvę kaitaliojant sistemos reikalavimus, kai RUP metodika viską kruopščiai dokumentuoja kiekvieną reikalavimų pasikeitimą, kurių nuolatinis kitimas gali vėlesniu metu brangiai kainuoti. Įvertinant anksčiau nusistatytus projekto reikalavimus gaunama, kad geriausiai tinkama metodika yra RUP, nes atsižvelgia į dauguma keliamų reikalavimų ir yra paprasta pritaikyti įvairaus dydžio projektams.

7 ESAMŲ IS SISTEMŲ REALIZAVIMO IR MODELIAVIMO ĮRANKIŲ ANALIZĖ

Šiuo metu rinkoje yra daug produktų, kuriais naudojantis galėtume atlikti darbų sekų modeliavimą, vykdymą, vartotojo sąsajos sukūrimą ir galutinio projekto sugeneravimą. Dauguma produktų gali atlikti tik tam tikras funkcijas, bet ne visas kartu. Įgyvendinant šį projektą norima, kad produktas galėtų atlikti, kuo daugiau funkcijų su kuo mažesnėmis sąnaudomis. Labiausiai tinkamas programas įvertinsim žemiau pateiktoje lentelėje.

Lentelėje pateikiami vertinimo kriterijai atsirinkti atsižvelgus į darbo specifiką ir į kokius darbus reikia atlikti, kad sistema būtų realizuota. Darbų sekų modeliavimas ir jų verifikavimas yra vienos svarbiausios veiklų darbe. Visos kuriamos sistemos esmė yra ta, kad sistema turėtų vykdyti sumodeliuotas darbų sekas. Tam, kad sistema galėtų naudotis kiekvienas norintis vartotojas, reikia padaryti lengvai suprantamą vartotojo sąsają. Didelis privalumas yra tai, kad galima viską padarius, vienu mygtuko paspaudimu susigeneruoti veikiančia sistemą. Tai ne tik paspartina darbą ir sumažina galimybes atsirasti klaidoms kuriant sistemą. Kiekvieną projektą vykdant yra labai svarbus faktorius - kaina. Visada stengiamasi taupyti, kad projektą pavyktų užbaigti su teigiamu balansu, tai produkto kainos kriterijus yra svarbus veiksnys.

7.1 lentelė Esamų IS sistemų realizavimo ir modeliavimo įrankių analizė

	AccuProcess Modeler	**Bizagi Process Modeler**	**Magic Draw Cameo Business Modeler**	**Bonita studio 5.6.1**
Darbų sekų modeliavimas	+	+	+	+
Sumodeliuotu darbų sekų verifikavimas	-	+	+	+
Darbų sekų vykdymas	+-	+	+	+
Vartotojo sąsajos kūrimas	-	+	+-	+

Galutinio projekto generavimas	-	+	-	+
Produkto kaina	-	-	-	+

Įvertinus atrinktus programinius paketus, kurie galėtų tikti sistemos realizavimui, gauti rezultatai pateikiami antroje lentelėje. Kaip matoma, žemiausią įvertinimą galutinį gavo AccuProcess Modeler programa, kuri yra mokama ir leidžia tik keletą dienų išbandyti bandomąją versiją. Išbandžius programos bandomąją versiją pastebėta, kad sistema leidžia modeliuoti darbų sekas ir jas bandyti simuliuoti, bet simuliacija leidžiama tik turint pilną licenciją.

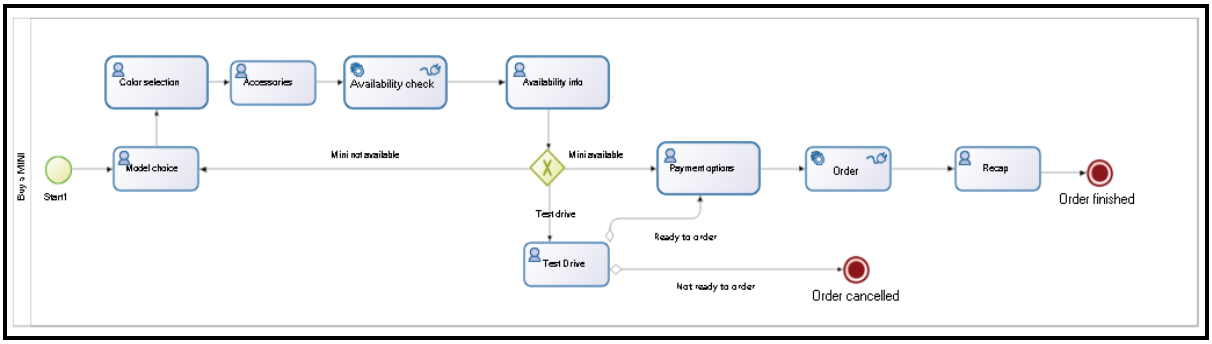
Sekantis produktas gavęs mažą įvertinimą yra Magic Draw Cameo Business Modeler. Šis produktas surinko 3,5 balo. Nepaisant puikiai įgyvendinto veiklų procesų modeliavimo, verifikavimo ir suprojektuotos sistemos testavimo modulių, kurie yra geriausiai įgyvendinti iš kitų produktų. Naudojantis šiuo produktu, negalėsime sukurti veiklos procesams vartotojo grafinius interfeisus, kuriuos būtų galima naudoti galutinės sistemos realizacijoje. Magic draw leidžia tik modeliuoti grafinę vartotojo sąsają. Taip pat, nebus galimybės sugeneruoti galutinio produkto į kelis failus, kuriais pasinaudojus būtų galima paleisti sistemą veikti.

Kitas produktas, kuris surinko net 5 balus yra Bizagi Process Modeler. Šis produktas beveik atitiko visus keliamus reikalavimus, kol nebuvo prieitas punktas „Kaina“. Norint naudotis visomis produkto funkcijomis, reikia nusipirkti licenciją. Nemokama produkto versija leidžia sumodeliavus veiklos procesų modelį patikrinti ar nepadaryta jokių projektavimo klaidų. Norint atlikti kitus veiksmus, tokius kaip testavimą, vartotojo sąsajos kūrimą, galutinės versijos generavimą, reikia nusipirkti licenciją.

Apžvelgus esamus sprendimus, kurie leidžia kurti informacines sistemas pagrįstas darbų sekų vykdymu, geriausiai atitiko reikalavimus Bonita studio 5.6.1 programa, kuri surinko 6 balus iš 6 galimų. Bonita studio yra atviro kodo programa, kuria galima modeliuoti veiklos procesus, o po to sumodeliuoti būsimos sistemos pateikiamus langus ir galiausiai sugeneruoti suprojektuotą sistemą ir ją išbandyti bandomuoju režimu.

7.1 Bonita studio 5.6.1

Bonita studio darbų sekų projektavimas vyksta naudojant lengvai suprantamą ir paprastą naudoti OMG standartą BPMN2. Naudojant BPMN [2] sumodeliuojamas darbų sekų modelis, pagal kurį projektuojama sistema veiks (žr. 7.1 pav.).

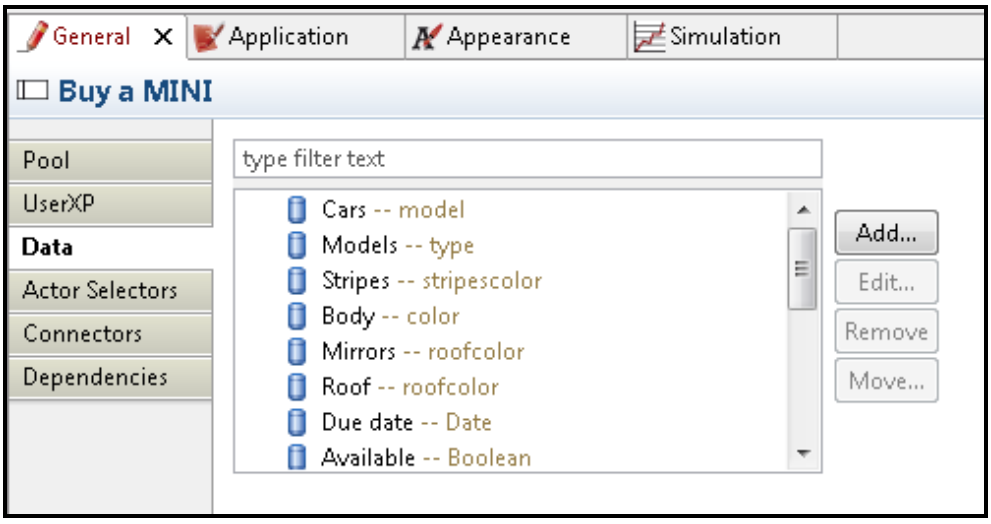


7.1 pav. Sumodeliuotas darbų sekų modelis

Vartotojas gali modeliuoti naudodamas pateikiamą ekrane BPMN elementų paletę, iš kurios gali paspausti ant norimo elemento ir nutemti į norimą ekrano vietą. Taip pat, gali paspausti ant jau esančių sumodeliuotos sekos žingsnio ir gauti sekančio žingsnio elementą. Kai norimas procesas yra sumodeliuotas, galima atlikti sumodeliuotos sekos simuliavimą.

Prieš pradedant simuliavimą, kiekvienam sekos žingsniui galima nurodyti reikalingus nustatymus ir kintamuosius(žr. 7.2 pav.), kuriuos reikia gauti žingsnio metu. Nurodant visus kintamuosius ir apribojimus sekų žingsniams, lengviau galima simuliavimo metu nustatyti, kur yra

sumodeliuotos sekos trūkumai ir kur reikia taisyti, kad norima darbų seka veiktų gerai. Simuliavimų metu galima nustatyti, kurie kintamieji ar apribojimai yra mažiausiai naudojami ir kaip jie įtakoja darbų seką.

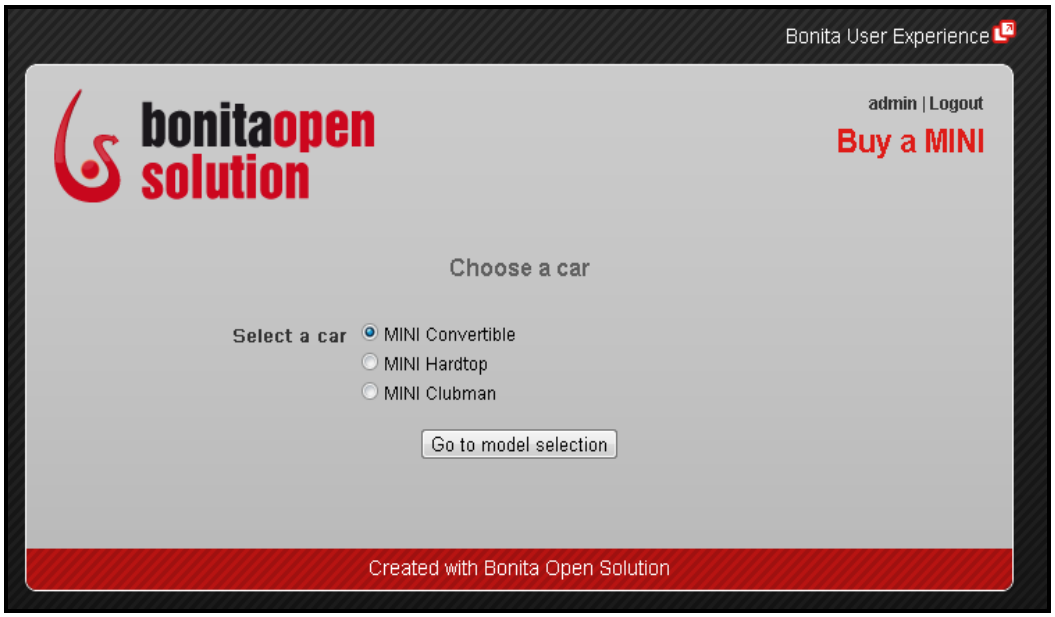


7.2 pav. Sekos žingsniui nurodyti kintamieji ir jų tipai

Atlikus sumodeliuotos darbų sekos simuliavimą ir ištaisius visus trūkumus galima pradėti kurti duomenų įvedimo formas naudojantis Bonita studio paketo formų kūrimo įrankiu. Naudojantis formų kūrimo įrankiu, galima lengvai ir greitai sukurti norimas duomenų pateikimo ir įvedimo formas kiekvienam sekos žingsniui.

Paketas taip pat palaiko daugiau nei 80 įvairiausių prisijungimo šablonų prie duomenų bazių (**Oracle, MySQL, MSSQL**) taip pat **ERP, SAP, LDAP, CRM, Google, Twitter, Facebook** ir kitų paprastai nurodant tik prisijungimo duomenis.

Atlikus visus reikiamus veiksmus projektuojant darbų sekų modelį ir formuojant duomenų pateikimo formas galima pasinaudoti Bonita execution engine funkcija.



7.3 pav. Sumodeliuota sistema paleista naudojant Bonita execution engine

Visą sukurtą projektą sugeneruos naršyklėje pasileidžiančią darbų sekomis grindžiamą sistemą pavaizduota 7.3 paveiksle.

8 SIEKIAMAS SPRENDIMAS

Šio magistro darbu siekiama sudaryti metodiką, kuria naudojantis būtų galima lengvai kurti informacines sistemas, kurių veikimas pagrįstas veiklos procesų vykdymu. Šiam sprendimui pasiekti reikia išanalizuoti šiuo metu esamus panašius sprendimus ir literatūros šaltinius, iš jų atsirinkti geriausias savybes, kurias būtų galima panaudoti naujos metodikos kūrime.

9 ANALIZĖS IŠVADOS

1. Išanalizavus mokslinę literatūrą buvo rasti keturi tinkami metodai, kuriuos galima pritaikyti kuriant naują metodiką. Atlikus rastų metodų veikimo analizę, rastos silpnosios ir stipriosios metodų veikimo pusės.
2. Atliekant paiešką, esamų programinių sprendimo paketų rasta keletas, kurie atitinka keliamus reikalavimus metodikos realizavimui. Nors atlikus detalesnę rastų sprendimų analizę paaiškėjo, kad nevisi rasti paketai yra tinkami realizuoti metodikai.
3. Atliekant vartotojų analizę rasti asmenys, kurie yra glaudžiai susiję su tyrimo objekto vykdymu. Atliekant detalesnę vartotojų analizę rastos jų savybės, kuriomis jie yra susiję su metodikos taikymu.

10 DARBŲ SEKOMIS GRINDŽIAMŲ INFORMACINIŲ SISTEMŲ KŪRIMO METODIKA

Kuriant informacines sistemas, taikant įvairias kūrimo valdymo metodikas, sistemos kūrimas pradžioje yra suskaidomas į atskiras dalis. Suskaidytos dalys dažniausiai turi būti atliekamos paeiliui viena po kitos, kai kurios ir lygiagrečiai arba net kartojant keletą kartų, kol pasiekiamas reikiamas tikslas. Kadangi vykdant projektus labai svarbus veiksnys laikas, tai sudaroma metodika stengiasi sudarinėjant teorinius veiklų išdėstymus kuo labiau veiklas, kurias galima atlikti tuo pat metu ir taip sutaupyti laiko sekančioms veiklos.

Pagal analizės dalyje padarytas išvadas, kad geriausios metodikos sistemų kūrimo valdymui yra RUP ir Agile Scrum abi metodikos turi nemažai bendrų pliusų. Sudarant naują metodiką galima remtis aptartų metodikų teigiamomis savybėmis. Kuriant sistemą visada įdomu žinoti, kurioje kūrimo fazėje vykdomas projektas šiuo momentu yra, tai paprasčiausiai būtų nustatyti pasinaudojant RUP metodikoje aprašomomis sistemos kūrimo fazėmis ir prijungiant keletą DSDM metodikoje aprašomų fazių, padedančių užtikrinti gerą projekto vykdymą. Fazės būtų suskirstomos į šešias dideles sistemos kūrimo fazes:

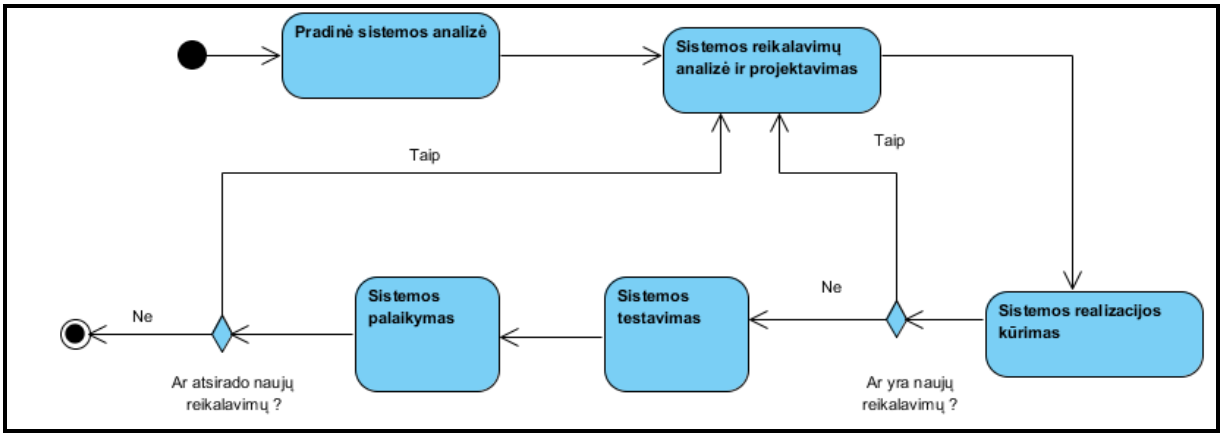
1. **Pradinė projekto analizė** – remiantis atlikta analize DSDM metodika, sistemos kūrimo pradžioje atlieka analizę ar kūrimo komanda turi visas reikiamas kompetencijas atlikti sistemos kūrimui reikiamus darbus. Taip pat įvertina projekto sudėtingumą ir galimas rizikas. Atlikus pradinę analizę, sukuriamas sistemos nedidelis prototipas, kuriuo remiantis pateikiama sistemos vizija klientui ir apskaičiuota sistemos pradinė sąmata;
2. **Sistemos reikalavimų analizė ir projektavimas** – surenkami sistemos visi reikalavimai, kuriais remiantis bus kuriama sistema ir atliekama analizė. Atliekant sistemą bus visi surinkti reikalavimai dokumentuojami ir specifikuojami. Atliekamas sistemos projektavimas grafinėmis priemonėmis, pateikiamos sistemos veikimo diagramos, panaudojimo atvejų diagramos, veiklos procesų diagramos, sistemos architektūros diagrama. Atlikus sistemos projektavimą visi grafiniai sistemos modeliai aprašomi, kad būtų aiškesni;

3. **Sistemos realizacijos kūrimas** – remiantis dokumentacija ir sistemos projekto grafiniais modeliais atliekama sistemos realizacija;
4. **Sistemos testavimas** – pradžioje, kai baigiama ruošti dokumentacija pradedami daryti sistemos testavimo planai, pagal kuriuos bus atliekami testavimai. Kai sistemos realizacijos fazė pradedama atlikinėti pradedami ir testavimo darbai. Kadangi sistema kuriama moduliais tai ir testavimai atliekami suprogramuotiems moduliams atskirai ir po to pakartotinai testuojama, kai sistemos moduliai apjungiami į vieną bendrą sistemą;
5. **Sistemos palaikymas** – kai sistemos testavimas ir klaidų taisymas baigiamas, sistema perduodama klientui. Abipusiu susitarimu nustatytą laiko tarpą sistema stebima ir iškilus sistemos klaidoms jos greitai yra taisomos.

10.1 lentelė Metodikos teigiamos savybės

Metodika	**Pasirinkimas**	**Priežastis**
DSDM	Pradinė projekto analizė	Šis etapas pasirinktas, nes padeda apsisaugoti nuo projektų, kurių kūrimo komanda nesugebėtų įgyvendinti
RUP	Sistemos reikalavimų analizė ir projektavimas	Kadangi kuriama metodika orientuojasi į sistemos reikalavimų surinkimą ir jų valdymą, bei į sistemos projektavimo svarbą pasirinktas RUP metodikos siūlomi veiklos procesai
RUP	Sistemos realizacijos kūrimas	RUP metodikos siūlomas realizacijos kūrimas pilnai atitinką kuriamos metodikos realizacijai keliamus reikalavimus
RUP	Sistemos testavimas	Sistemos testavimo etapui pasirinkta RUP metodikos siūloma metodika, kaip turėtų būti testuojama sistema ir kaip jos testavimui turėtų būti ruošiamasi
DSDM	Sistemos palaikymas	Sistemos palaikymo etapas pasirinktas, nes remiantis straipsniu [14] sistemos palaikymo etapas svarbus tinkamam sistemos perdavimui klientui ir jo apmokymams
Agile Scrum	Veiklos ciklas	Scrum metodikos veiklos procesų valdymo mechanizmas puikiai padeda organizuoti darbus ir kiekvienos iteracijos metu iš naujo įvertinti likusias veiklas ir prioritetus perskirstyti naujoms veikloms.
	Reikalavimų specifikavimo lentelė	Modifikavus truputi Volerė šablono pateikiamą reikalavimų aprašymui skirtą lentelę [27]

Visos sistemos fazės ir jų eiliškumas pateiktas 10.1 paveikslėlyje**,** kuriame pavaizduoti galimi ryšiai tarp sistemos kūrimo fazių.



10.1 pav. Metodikos fazių tarpusavio ryšiai

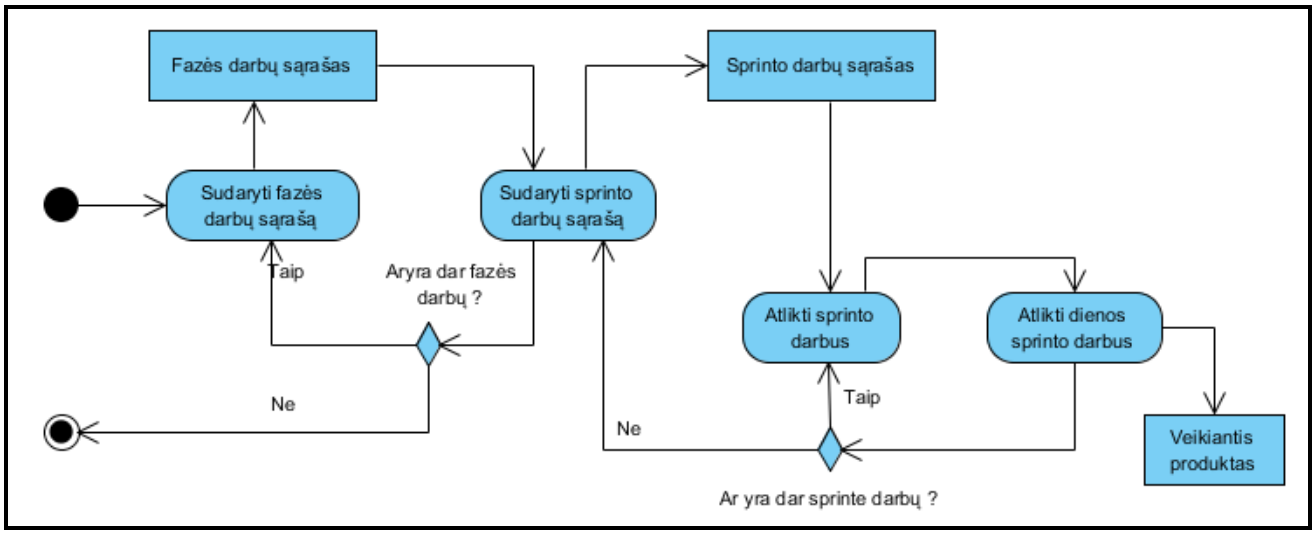
Pagal pateiktą 10.1 paveikslėlyje grafinį modelį matoma, kaip sistemos kūrimo fazės sąveikauja tarpusavyje. Pirmoji sistemos kūrimo fazė yra **pradinė sistemos analizė**, kurią atlikus bus galima pereiti į sistemos reikalavimų analizės fazę. Antroje fazėje daroma sistemos dokumentacija, po to ji bus dar pildoma sekančioje fazėje, tai yra sistemos projektavimas. Taip pat sistemos palaikymo, sistemos realizacijos ir sistemos projektavimo fazėse gali atsirasti poreikis atnaujinti arba pridėti naujų sistemos reikalavimų, kurie turi būti dokumentuojami ir kruopščiai sekami jų pasikeitimai, kad užtikrinti reikalavimų pasikeitimų valdymą. Pagal analizės metu paruoštą **dokumentaciją** ir **sukurtą prototipą** atliekamas sistemos projektavimas. Projektavimo metu sukurti sistemos grafiniai modeliai yra dokumentuojami aprašant grafinį modelį. Atlikus sistemos projektavimo etapą, pradedama sistemos realizacijos kūrimas, kurio metu gali atsirasti poreikių keisti sistemos reikalavimus. Kadangi sistemos kūrimas vyksta moduliniu būdu, tai yra kuriama sistema atskiromis nedidelėmis dalimis, kurios atskirai gali veikti ir kai yra apjungiamos į vieną bendrą sistemą, galima realizavus sistemos modulį galima jį perduoti testuotojams ir lygiagrečiai pradėti vykdyti sistemos testavimo fazę. Testuotojai atlikdami testavimą registruoja klaidas. Testuotojai sistemos **testavimo fazės** metu testuodami naudojasi anksčiau pasiruoštais testavimo planais paruoštais pagal sistemos dokumentaciją. Kai baigiama sistemos realizacija ir testavimas, sistema perduodama klientui ir pradedama sistemos palaikymo fazė, kurios metu prižiūrima naujai sukurta sistema ir taisomos klaidos, kurios atsiranda sistemos vartotojams naudojantis sistema.

Viso projekto metu vykdomos fazės kurios pateiktos 10.1 paveikslėlyje. Kiekvienos fazės metu taip pat yra vykdomas 10.2 paveikslėlyje pateiktas procesas, kuris parengtas remiantis Scrum metodikoje pateikiamu procesu. Prasidėjus fazei prasideda ir pats Scrum metodikoje pateikiamas projekto vykdymo procesas.

Pirmiausia sudaromas veiklų sąrašas, kuris turi būti atliekamas fazės metu. Sudarytam darbų sąrašui sudedami prioritetai, pagal kuriuos turėtų būti atliekamos veiklos. Po to kai prioritetai sudėlioti veikloms galima atsirinkti darbus iš darbų sąrašo, kurie bus atliekami iteracijos metu. Pasirenkamų darbų atlikimo trukmė neturėtų būti ilgesnė nei mėnuo. Turint susidarytą iteracijos darbų sąrašą galima pradėti jį ir įgyvendinti.

Jeigu realizacijos metu atsiranda naujos veiklos, kurios turi būti realizuojamos, tai jos pridedamos į darbų sąrašą. Pasibaigus iteracijai yra baigiama kažkuri sistemos dalis, kurią galima parodyti klientui, kaip progreso įrodymą. Prieš pradedant rinktis naujas veiklas iteracijai, iš naujo įvertinamos ir priskiriami per naujo prioritetai veikloms, kadangi galėjo atsirasti aukštą prioritetą turinčių veiklų realizacijos metu. Iteracijos tęsiasi tol, kol darbų sąraše nebelieka veiklų, kurias būtų galima prisiskirti. Taip veiklos atliekamos kiekvienoje fazėje. Kiekvienos fazės darbų sąrašas yra atskiras.

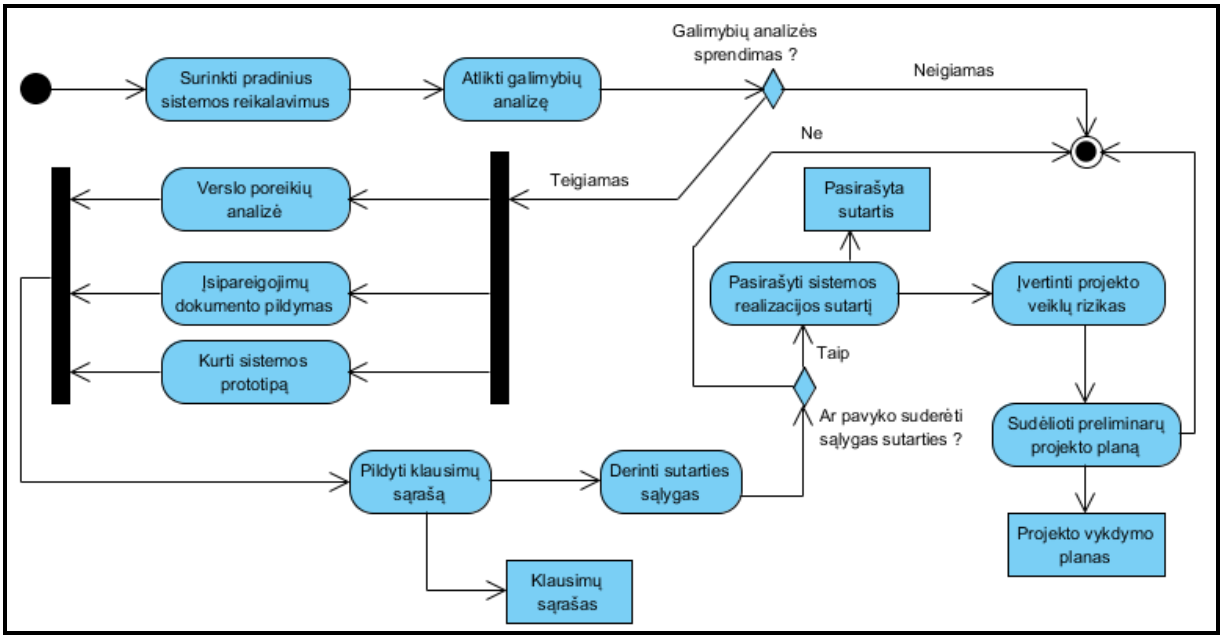
Šis pasirinkimas padarytas toks, kadangi dirbant Scrum metodika, yra paprasta reaguoti į reikalavimų pokyčius, kas yra viena iš priežasčių, dėl ko žlunga projektai remiantis straipsniu [14]. Taip pat leidžia kas kartą įvertinti veiklų prioritetus ir išvengti grėsmių, kurios gali atsirasti dėl technologijų ar komandos pokyčių, kad komandos pajėgumai neįvertinti ir sudėtingos užduotys pasiliekamos projekto pabaigai.



10.2 pav. Metodikos fazės sudedamosios dalys

Šitoks pasirinkimas padarytas remiantis dėl darbų sąrašo vykdymo paprastumo ir tai, kad galima paprastai ir greitai koreguoti vykdomų darbų sąrašus ir jų prioritetus.

10.1 Pradinė IS projekto analizė

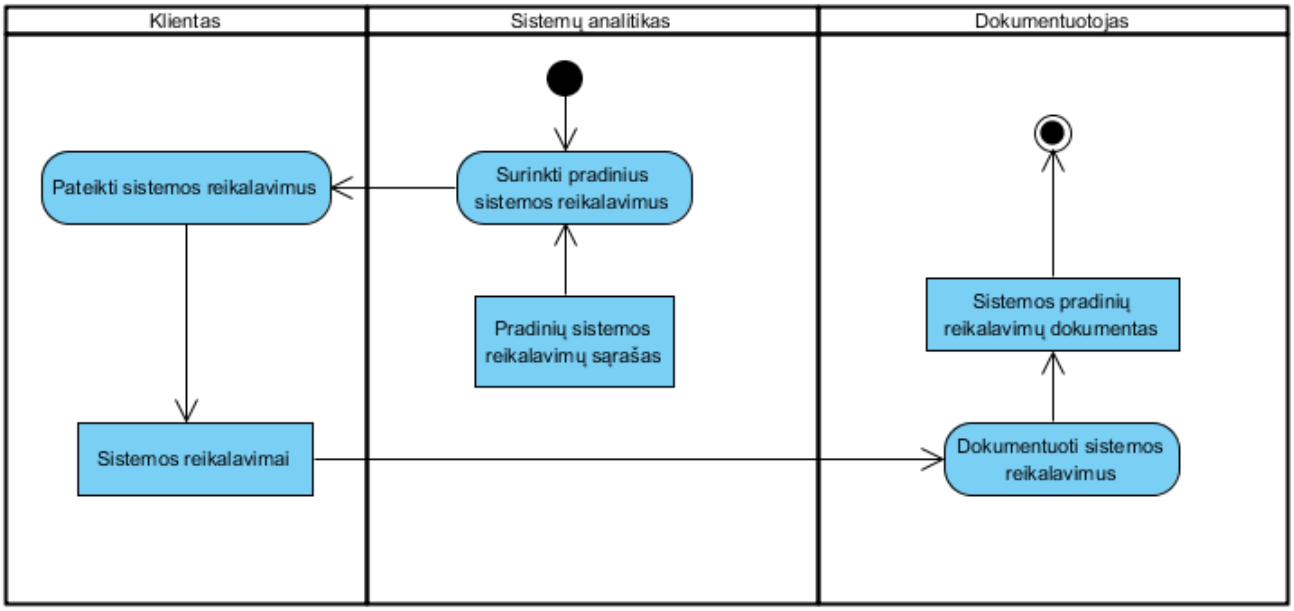


10.3 pav. Pradinės sistemos analizės fazės procesų modelis

Kaip teigiamą kriterijų pasirinkti DSDM metodikoje aprašomą **pirmą sistemos kūrimo fazę** įtakojo internetiniame puslapyje [14] patalpinta informacija apie priežastis, kodėl projektai žlunga. Net **37%** projektų žlunga dėl prasto projekto plano sudarymo, blogo rizikų nustatymo ir neįvertintos galimybės atlikti darbą su turimais resursais.

Tam, kad išvengti tokių pasekmių, pasinaudota DSDM metodikoje naudojama **pradine sistemos kūrimo faze**, kurios metu vykdomos veiklos pavaizduotos 10.3 paveikslėlyje. Šios fazės metu remiantis metodika reikia atlikti sistemos **galimybių analizę**, **verslo poreikių analizę**,

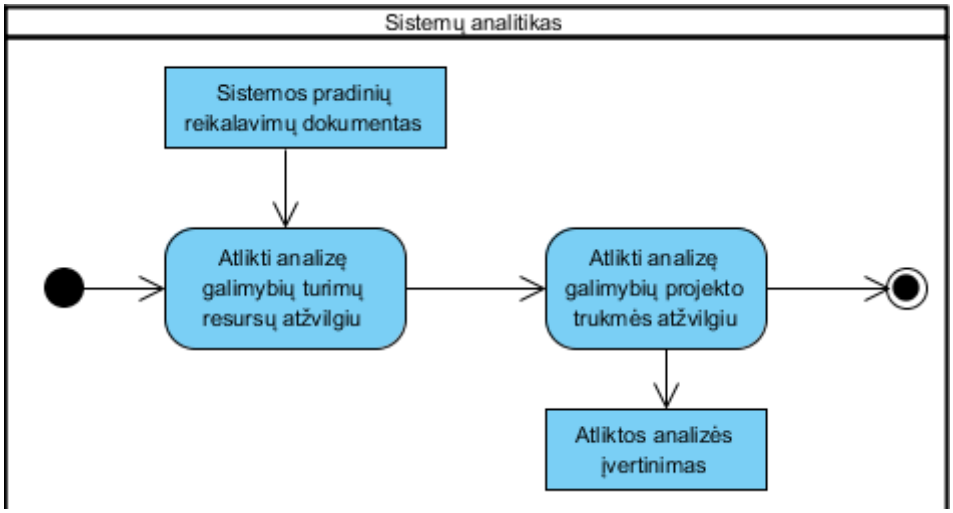
paruošti sistemos kūrimo įsipareigojimų dokumentą, prototipą, pirminį projekto vykdymo planą.



10.4 pav. Sistemos pradinių reikalavimų surinkimo proceso modelis

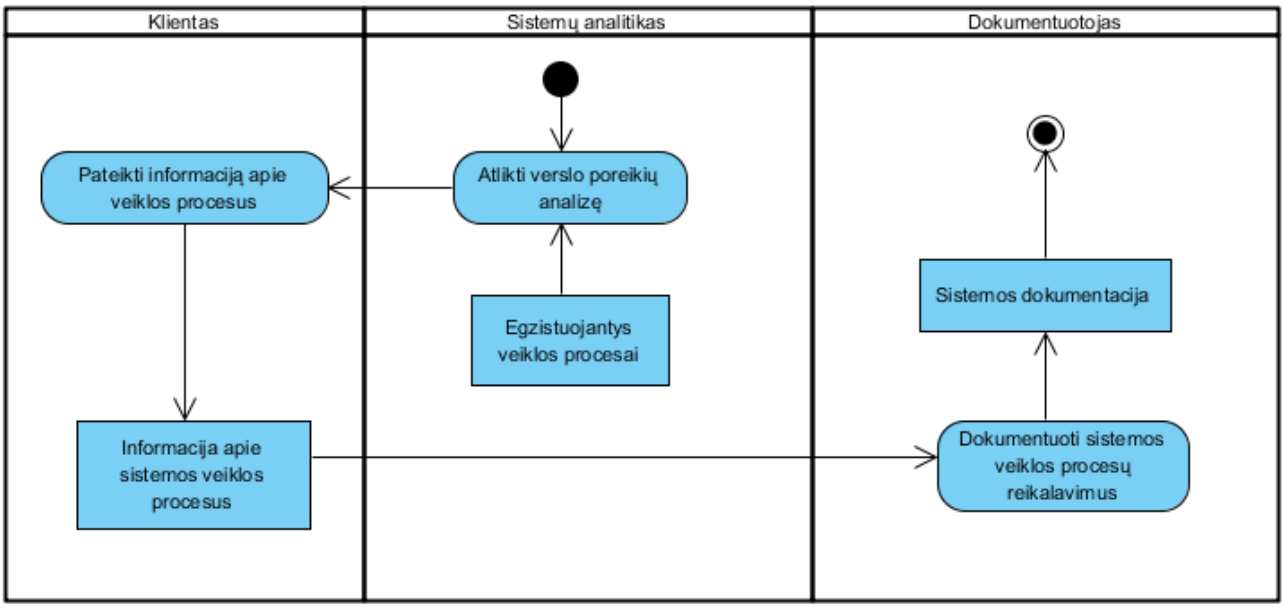
Galimybių analizė atliekama pirmiausiai iš pateikiamo veiklų sąrašo. Analizei atlikti sistemų analitikai susirenka pradinius sistemos reikalavimus ir pasiruošia **pradinių sistemos reikalavimų dokumentą** (žr. 10.4 pav.), kurį po to naudoja atliekant galimybių analizę. Pradiniams reikalavimams surinkti reikalingi sistemų analitikai, dokumentuotojas, nes analitikas išsiaiškina reikalavimus, o dokumentuotojas juos aprašo dokumentacijoje. Sistemos pradinių reikalavimų surinkimo procesas pavaizduotas 10.4 paveikslėlyje, kuriame pateikta, kad reikalavimų surinkime dar dalyvauja ir klientas, pradinių sistemos reikalavimų sąrašas, kuris naudojamas pagrindiniams sistemos reikalavimams patikslinti.

Analitikai susirinkę pagrindinius reikalavimus sistemos stengiasi išsiaiškinti ar su turimais resursais ir projekto įvykdymui skirta laiko trukme įmanoma sukurti norimą informacinę sistemą, galimybių analizės proceso metu (žr. 10.5 pav.).



10.5 pav. Sistemos galimybių analizės proceso modelis

Jeigu analitikų sprendimas atlikus **galimybių analizę** yra teigiamas, tuomet toliau yra bandoma vykdyti **verslo poreikių analizę** ir po truputi ruošti **įsipareigojimų dokumentą**, kuriuo remiantis bus vykdomas projektas. Kitu atveju jeigu atsakymas gaunamas neigiamas projekto vykdymas stabdomas, nes yra priežasčių dėl kurių nėra galimybių įgyvendinti projekto.

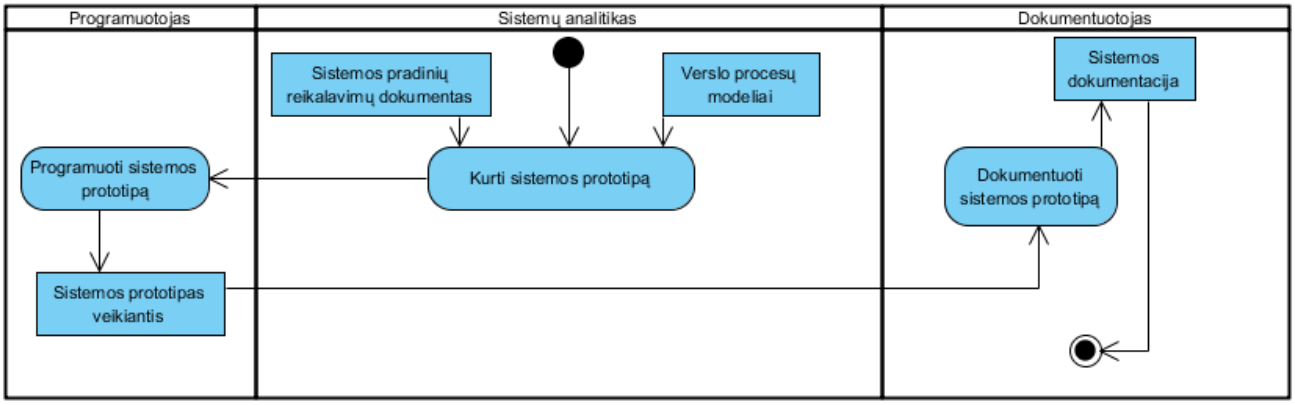


10.6 pav. Sistemos verslo poreikių analizės modelis

Po to kai yra gautas teigiamas analitikų atsakymas dėl projekto tolimesnio vykdymo galima lygiagrečiai pradėti vykdyti **verslo poreikių ir procesų analizę**, **sistemos prototipo kūrimą** ir pradėti pildyti **sistemos kūrimo įsipareigojimų dokumentą**. Analitikams atliekant **verslo poreikių ir procesų analizę**, kurios vykdymo procesas pavaizduotas 10.6 paveikslėlyje stengiamasi išsiaiškinti visus verslo aspektus projekte, kuriuos reiks realizuoti ir atsakyti į klausimus:

1) Ar šis sprendimas turės įtakos verslui?
2) Kokie yra verslo dalyviai, kurie naudosis sistema?
3) Kokie yra tinkamiausi verslo procesai?
4) Kuriuos verslo procesus reikia realizuoti?
5) Kaip galima patobulinti verslo procesus, kad verslui būtų didesnė nauda?

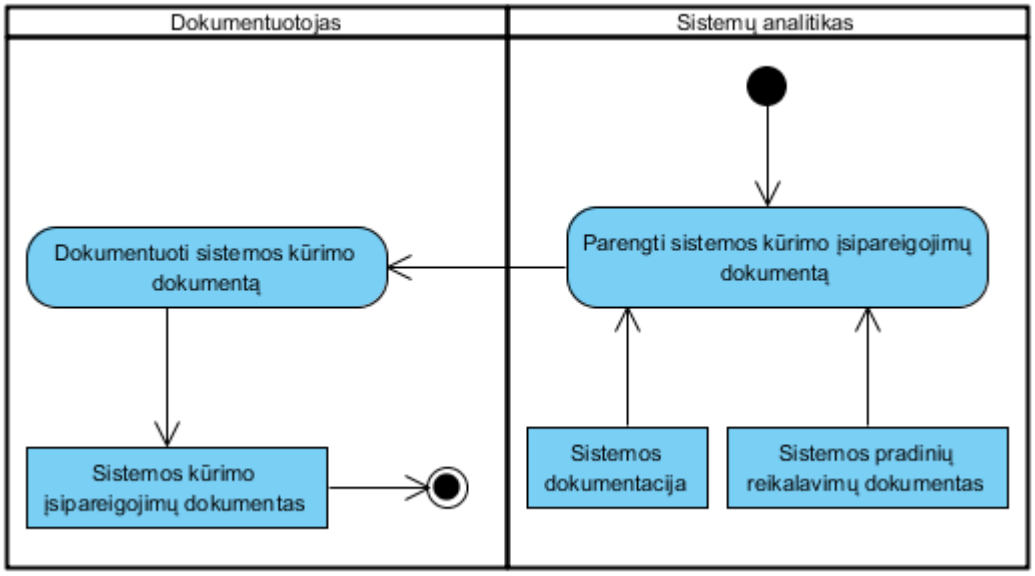
Verslo poreikius ir procesus analizuojant visą surinktą informaciją galima apsirašyti laisva forma tekstiniame dokumente ar aiškinantis procesus ir poreikius juos grafiškai modeliuoti. Pagal analizės metu pasidarytus užrašus ar grafinius modelius sistemos projektavimo fazėje bus galima naudotis modeliuojant grafiškai sistemos modulius.



10.7 pav. Sistemos prototipo kūrimo modelis

Pagal pradinės analizės metu surinktus reikalavimus galima pradėti daryti **sistemos paprastą prototipą**, kurio kūrimo procesas pateikiamas 10.7 paveikslėlyje, kuris bus skirtas galimo sistemos funkcionalumo aptarimui. Tai gera pagalba renkant sistemos reikalavimus, kadangi klientas mato, kaip buvo suprasti ir interpretuoti pradiniai reikalavimai. Geriausia prototipą padaryti su kiek galima daugiau pasiūlymų, iš kurių klientas galėtų pasirinkti ir pateikti savo reikalavimus matydamas

preliminarų vaizdą. Taip pat darant prototipą kyla daugybė klausimų susijusiu su sistemos funkcionalumu ir išvaizda, kurie gali inicijuoti reikalavimų surinkimą iš kliento.



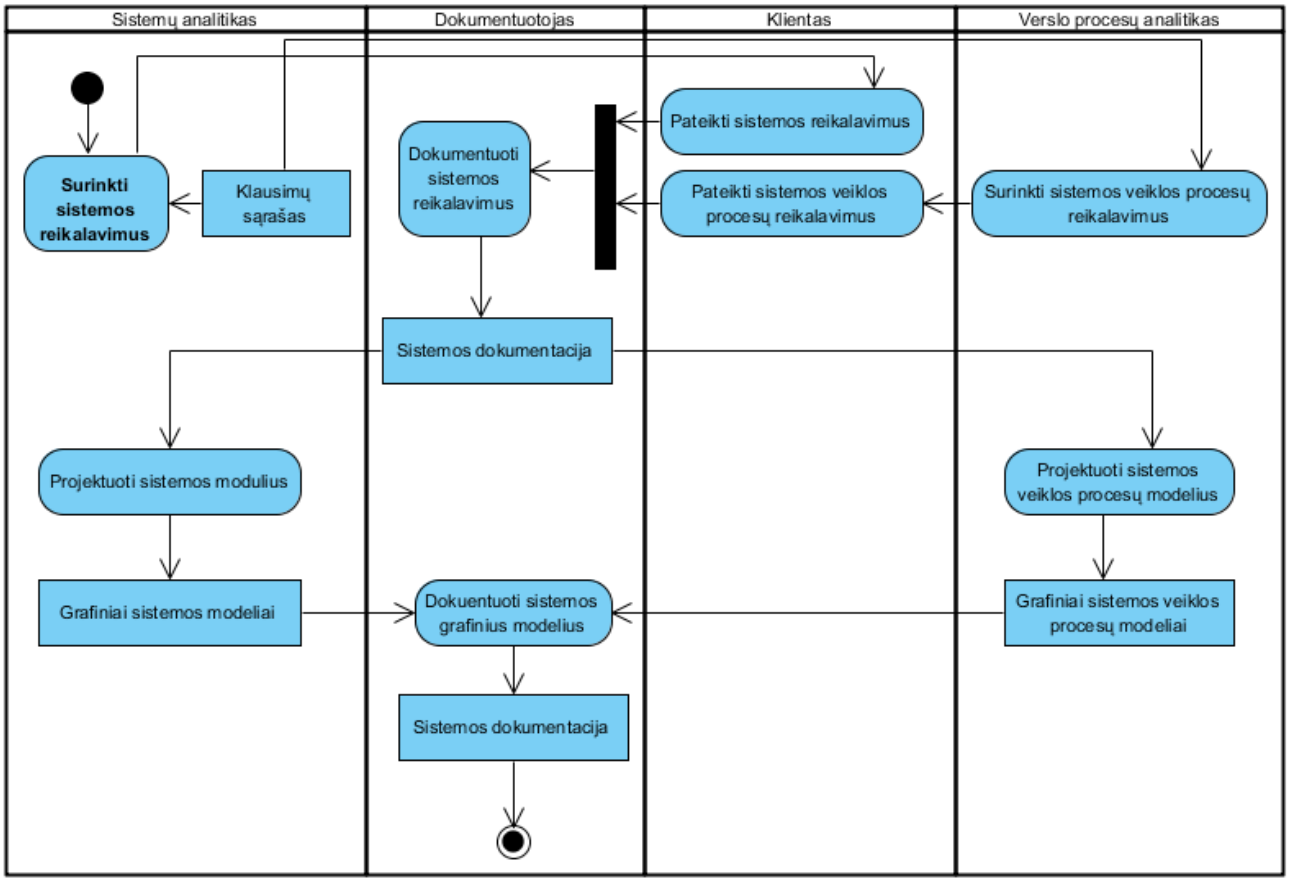
10.8 pav. Sistemos įsipareigojimo dokumento parengimo modelis

Kuriant prototipą jis yra dokumentuojamas. Dokumentacijoje aprašoma visa, kaip turėtų atrodyti sukurta sistema ir koks planuojamas jos funkcionalumas, įsipareigojimų dokumento pildymo procesas pateiktas 10.8 paveikslėlyje. Pildant dokumentaciją turėtų būti aprašomos visos sistemos busimos formos, kurias vartotojas galės atsidaryti, aprašant visus duomenų įvedimo ir išvedimo laukus esančius formose. Po to kai sistemos prototipas yra galutinai nuspęstas ir žinoma, kokios funkcijos turėtų būti realizuojamos galima atlikti sistemos įsipareigojimų dokumentacijos korekcijas. Kai padaromos korekcijos su įsipareigojimų dokumentu galima pasirašyti sistemos realizavimo sutartį.

Paskutinė, bet viena iš svarbiausių veiklų pirmojoje fazėje yra pagal turimą **įsipareigojimų dokumentą** ir susirinktus reikalavimus įsivertinti **rizikas** ir **susidėlioti projekto planą**, kuriuo remiantis bus vykdomas visas projektas.

10.2 Informacinės sistemos reikalavimų analizė ir projektavimas

Šis etapas yra vienas iš svarbiausių sėkmingam sistemos kūrimo projektui užtikrinti. Šio etapo metu surenkami visi sistemos reikalavimai: funkciniai, nefunkciniai ir veiklos procesų. Tinkamai atlikus sistemos reikalavimų analizę ir surinkus visus sistemos reikalavimus galima išvengti nepilnų reikalavimų problemos, dėl kurios apie 34% IT projektų žlunga [14]. Kai dalis **sistemos reikalavimų** jau būną suspecifikuotų galima pradėti **sistemos modeliavimą.** Visas fazės vykdymo procesas pateiktas 10.9 paveikslėlyje.



10.9 pav. Sistemos reikalavimų analizės ir projektavimo fazės veiklos proceso modelis

Šioje fazėje analitikas turi bendraujant su klientu suprasti kliento poreikius ir norus, kuriuos jis kelia naujai kuriamai sistemai. Pokalbių metu surinkti reikalavimus ir juos dokumentuoti, prioretizuoti. Pagal surinktus reikalavimus reikės atlikti sistemos projektavimo darbus, kurie taip pat turėtų būti dokumentuojami. Surenkant reikalavimus galima remtis FURPS+ reikalavimų modeliu, kuris apibrėžia, ką vartotojas gali papasakoti apie naujai kuriama sistemą:

1. **Funkcionalumas** (Functionality)
2. **Vartojimo patogumas** (Usability)
3. **Patikimumas** (Reliability)
4. **Veikimas** (Performance)
5. **Palaikomumas** (Supportability)

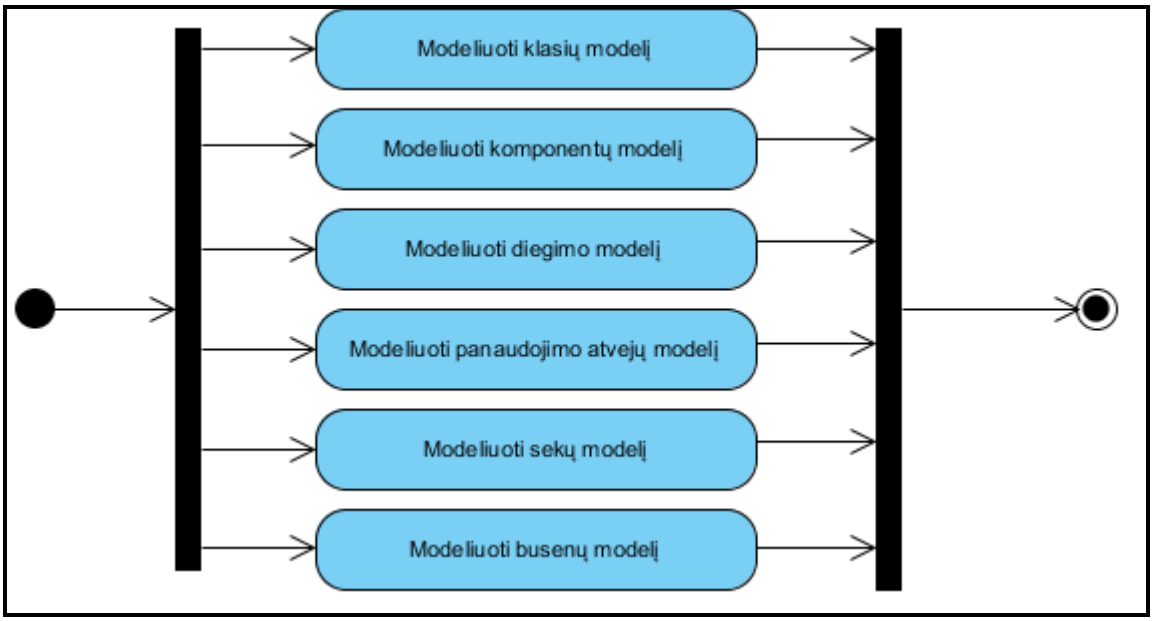
Dokumentuojant reikalavimus patogiausia juos aprašyti naudojant lentelę [27], kurioje nurodytumėte reikalavimo tipą, reikalavimo tekstą, kas inicijavo reikalavimą, išpildymo kriterijus, reikalavimo poreikis, sukūrimo ar pakeitimo data. Kiekvieną reikalavimą aprašant lentele išlaikomas dokumento tvarkingumas ir lengviau yra rasti informaciją, kai reikalavimai surašyti naudojant apibrėžtą struktūrą. Nurodant reikalavimų tipą reikia rinktis tarp funkcinio ir nefunkcinio reikalavimo, nes funkciniu reikalavimu nusakoma, ką sistema turi atlikti, o visi likę reikalavimai yra laikomi nefunkciniais reikalavimais. Aprašant reikalavimus patogu yra nusirodyti ir reikalavimo iniciatorių, nes ateityje, jeigu iškiltų klausimas susijęs su šiuo reikalavimų, kad būtų galima paprasčiau sužinoti, su kuo reikia susisiekti dėl konsultacijos. Dokumentuojant reikalavimą reikia nusirodyti reikalavimo įvykdymo kriterijų, kad būtų galima paprasčiau sekti, kada reikalavimas įvykdytas. Ir vienas iš svarbiausių kriterijų apsirašant reikalavimus yra jų dokumentavimo data. Tai padeda sekti reikalavimų pokyčius ir žinoti, kada paskutinį kartą keitėsi reikalavimas.

Dokumentuojant **kiekvienas reikalavimas** turėtų būti: **aiškus, teisingas, nesunkiai įgyvendinamas, reikalingas, pakankamai išsamus ir patikrinamas**. Kadangi reikalavimams

neatitinkant, kurio nors iš išvardintų punktų vėliau vykdant projektą gali prireikti vykti pas klientą tikslinti reikalavimų, o tai reiškia tik sugaišta laiką be reikalo, nes to buvo galima išvengti.

Kol renkami, analizuojami ir dokumentuojami sistemos reikalavimai svarbu susirinkti ir pasidaryti naujai kuriamos sistemos veiklos procesų analizę, kurios metu būtų aprašomi veikiantys **veiklos procesai**. Apsirašius veikiančius procesus galima atlikti jų analizę ir rasti vietų, kuriose pertvarkius veiklos proceso eigą galima jį paspartinti ir taip pagerinti kompiuterizuojamą procesą. Taip pat veiklos procesų analizės metu svarbu nustatyti, kurias veiklas **reikės** realizuoti, o kurių **nereikės**. Analizės metu išsiaiškinus visus veiklos procesus, kuriuos reikės realizuoti, taip pat išsiaiškinti ir kokie **funkciniai** reikalavimai turėtų būti realizuojami, kiekviename veiklos procese. **Veiklos procesus dokumentuojant** patogiausia apsirašinėti juos lentele, kurioje prie kiekvieno veiklos proceso pateikiama informacija: **veiklos proceso pavadinimas, įėjimo informacija, išėjimo informacija, kokios funkcijos realizuojamos, kokie nefunkciniai reikalavimai reikalingi**.

Kai jau būna dalis reikalavimų suspecifikuotų galima pradėti **sistemos projektavimą**, kurio vykdymo procesas pateiktas 10.9 paveikslėlyje, naudojant grafinius **UML** įrankius. Projektuojant sistemą grafiniais UML įrankiais patogu suprojektuotus modelius pateikti sistemos dokumentacijoje ir juos aprašyti. Realizacijos kūrimo metu programuotojams bus galima pateikti užduotis su nuorodomis į dokumentaciją, kurioje bus pateikiami sistemos grafiniai modeliai.



10.10 pav. Sistemos modeliavimo metu modeliuojamų UML diagramų sąrašas

Remiantis internetiniu puslapiu [26], kuriame teigiama, kad sistemos projektui sumodeliuoti užtenka **7 pagrindinių UML diagramų**, kurios pateiktos 10.10 paveikslėlyje. Jomis remiantis, galima sumodeliuoti visą sistemos veikimą ir pagal padarytus modelius suprogramuoti sistemą. Kiekviena iš 7 diagramų gali būti naudojama priklausomai nuo situacijos ir poreikio kuriant sistemą. Pagrindinės 7 diagramos yra:

1) **Klasių diagrama** – naudojama sumodeliuoti statiniam sistemos vaizdui, nusakyti sistemos komponentų atsakomybes tarpusavio, taip pat gali būti naudojama, kaip pagrindas diegimo diagramoms;
2) **Komponentų diagrama** – naudojama atvaizduoti sistemos komponentams ir ryšiams tarp jų, nusakyti organizacijos ir sistemos komponentų ryšius;
3) **Diegimo diagrama** – naudojama sumodeliuoti sistemos diegimo modeliui ir sistemos veikimui reikalingų kompiuterinės įrangos išsidėstymą organizacijoje. Taip pat pateikti vaizdą, kokie įrenginiai bus naudojami sistemos diegimui;

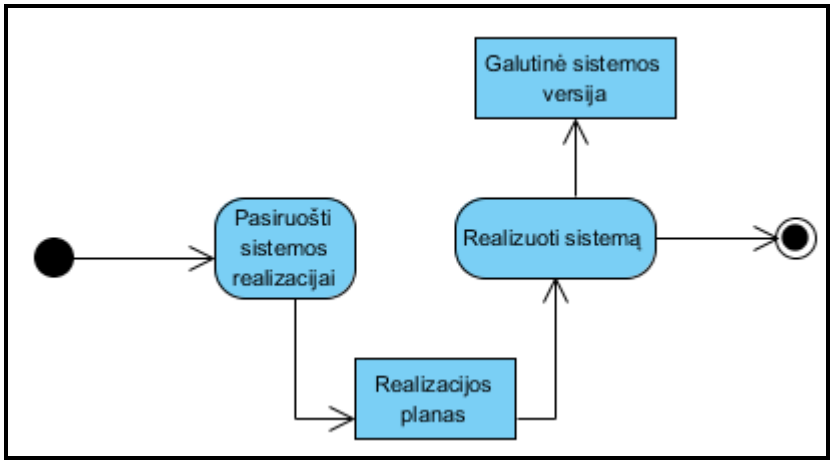
4) **Panaudojimo atvejų** – naudojama sumodeliuoti sistemos panaudojimo atvejams, kuriais remiantis yra paprasčiau surinkti sistemos visus reikalavimus. Taip pat padėti nustatyti visus vidinius ir išorinius faktorius, kurie veikia sistemą, pateikti vaizdą tarp sistemos reikalavimų ir busimų sistemos vartotojų;
5) **Sekų diagrama** – naudojama parodyti sistemos funkcijų veikimo seką ir sistemos komponentų tarpusavio ryšius;
6) **Būsenų diagrama** – naudojama pateikti grafiškai sistemos komponento įgaunamas būsenas sistemos veikimo metu;
7) **Veiklos procesų diagrama** – naudojama sumodeliuoti sistemos procesų veikimą ir jų eiliškumą sistemoje. Modeliuojant veiklos procesų diagramą galima pateikti ne tik veiklos procesų veikimo eilę, bet ir funkcijas, kurias kiekvieno proceso metu sistema vykdys.

Projektuojant sistemos veiklos procesų modelius reikia remtis **sistemos reikalavimų surinkimo metu surinktais reikalavimais** veiklos procesams. Grafiniuose modeliuose kuo tiksliau nurodyti proceso įėjimo ir išėjimo funkcijas ir duomenų rinkinius. Šis projektavimas turi būti kuo detaliau sumodeliuotas.

Atlikus reikalavimų analizės, surinkimo ir sistemos projektavimo darbus, tiek reikalavimus, tiek ir sistemos grafinius modelius aprašius sistemos dokumentacijoje remiantis RUP metodika turėtų jau būti paruošta apie **80 % dokumentacijos**, kuria remiantis galima atlikti sistemos realizaciją. Taip pat iki šio etapo pabaigos jau turėtų būti paruoštas pilnai veikiantis sistemos prototipas, kuris bus naudojamas, kaip vaizdinė medžiaga kuriant sistemą.

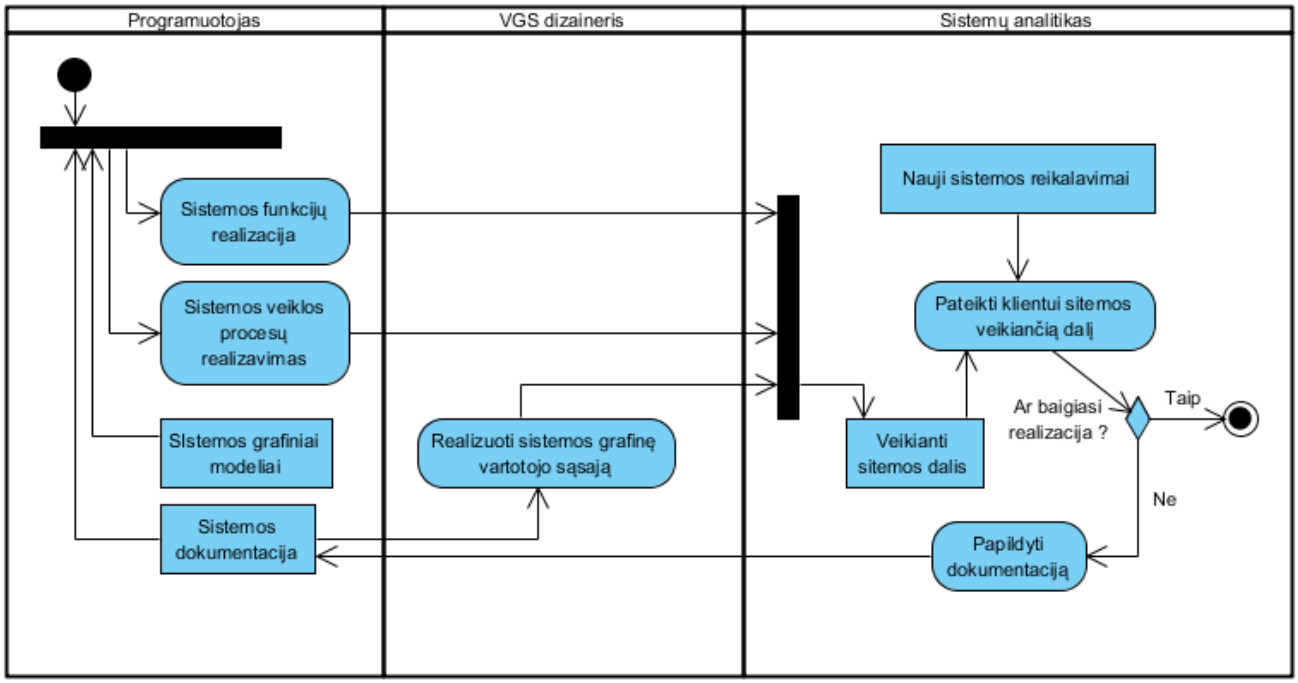
Kai baigiama rinkti ir analizuoti sistemos reikalavimus pilnai jau matosi visos sistemos vaizdas ir kliento poreikiai kuriant sistemą, todėl naudingą visas sistemos realizacijos proceso veiklas per naujo įvertinti rizikos ir prioritetų atžvilgiu, kad sudėtingiausių su didžiausiu prioritetu veiklų nepasilikti sistemos realizacijos pabaigai. Per naujo įsivertinus rizikas ir prioritetus reikia peržiūrėti anksčiau susidarytą projekto planą ir jį atnaujinti pagal naujausią įsivertinimą.

10.3 Informacinės sistemos realizacijos kūrimas



10.11 pav. Sistemos realizacijos kūrimo fazės proceso modelis

Sistemos realizavimo fazę galima vykdyti remiantis Agile Scrum metodikoje aprašomu vykdymo planu, kuris pateikiamas 10.11 paveikslėlyje. Tai yra visas veiklas, kurias reikia atlikti reikia surašyti į vieną ilgą sąrašą, kuris galės atstoti darbų sąrašą. Prieš kiekvieną naują iteracija iš ilgo darbų sąrašo išsirenkamos veiklos, kurios bus vykdomos iteracijos metu. Iteracijos procesas pateikiamas 10.12 paveikslėlyje. Iteracijų ilgis svyruotu tarp dviejų ir trijų savaičių ir kiekvienos iteracijos pabaigoje suprogramuotas produktas pristatomas klientui.

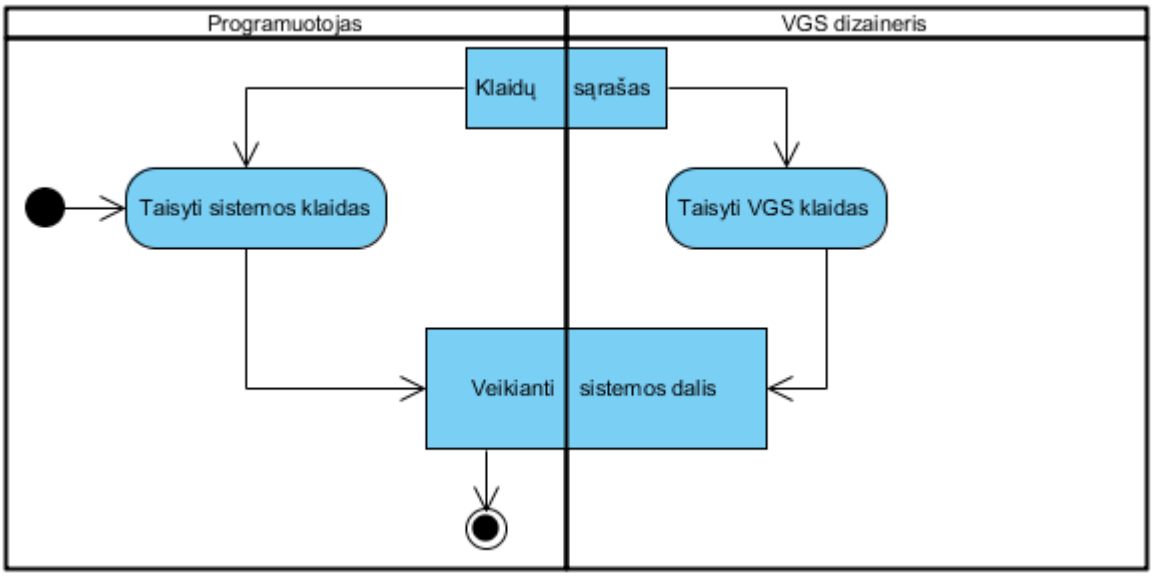


10.12 pav. Sistemos realizacijos kūrimo proceso modelis

Iteracijos pabaigoje pateikiant sistemą klientui reikia susirinkti likusius **20% reikalavimų**, nes kai klientas realiai mato veikiančią sistemos dalį jis žino, ko jam dar trūksta sistemoje ir tuo metu pateikia likusius reikalavimus. Surinkus reikalavimus juos **būtina sudokumentuoti**, kad būtų vykdoma sistemos reikalavimų pasikeitimų priežiūra. Surinkus naujus reikalavimus ir juos apsirašius dokumentacijoje, reikia kiekvienos naujos iteracijos metu per naujo visų veiklų prioritetus peržiūrėti ir taip į naujos iteracijos darbų sąrašą atsirinkti darbus, kuriuos reikia greičiausiai atlikti.

Kai realizacija vykdoma remiantis Agile Scrum [4] metodikoje aprašoma tvarka, tai reikia daryti ir komandos narių pasitarimus kas diena ar dvi, kad aptarti, ką padarė nuo paskutinio susitikimo, su kokiomis problemomis susidūrė. Susitikimai neturėtų trukti, daugiau kaip 10 minučių. Taip pat susitikimų metu neturi būti diskutuojama, kaip reiktų ištaisyti vieną ar kitą problemą iškilusią. Susitikimai skirti problemoms indikuoti, kad būtu žinomos iškilusios problemos vykdant projekto realizaciją. Pasibaigus susitikimams galima imtis problemų aptarimo ir diskusijų padėsiančių ištaisyti ir pašalinti problemą.

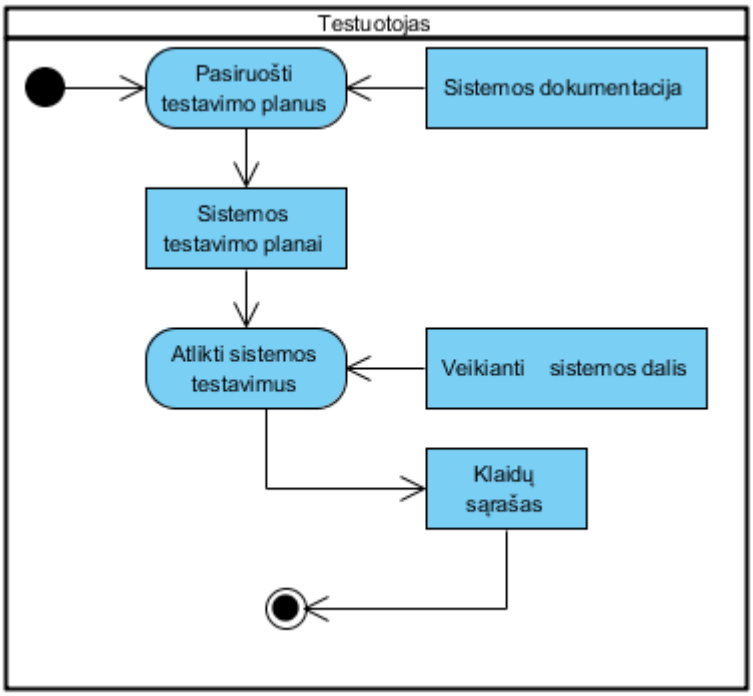
Sistemos realizacija kuriama dviem lygiais pirmasis ir pagrindinis yra tinkamai sukurti veiklos procesų vykdymo sistemos modulį, prie kurio vėliau būtų galima jungti kitas sistemos dalis. Veiklos procesų vykdymo modulis turi būti kruopščiai sukurtas remiantis sistemos veiklos procesų analizės dokumentacijos dalimi, kad atitiktu visus sistemai keliamus reikalavimus.



10.13 pav. Sistemos testavimo fazėje surinktų klaidų taisymo proceso modelis

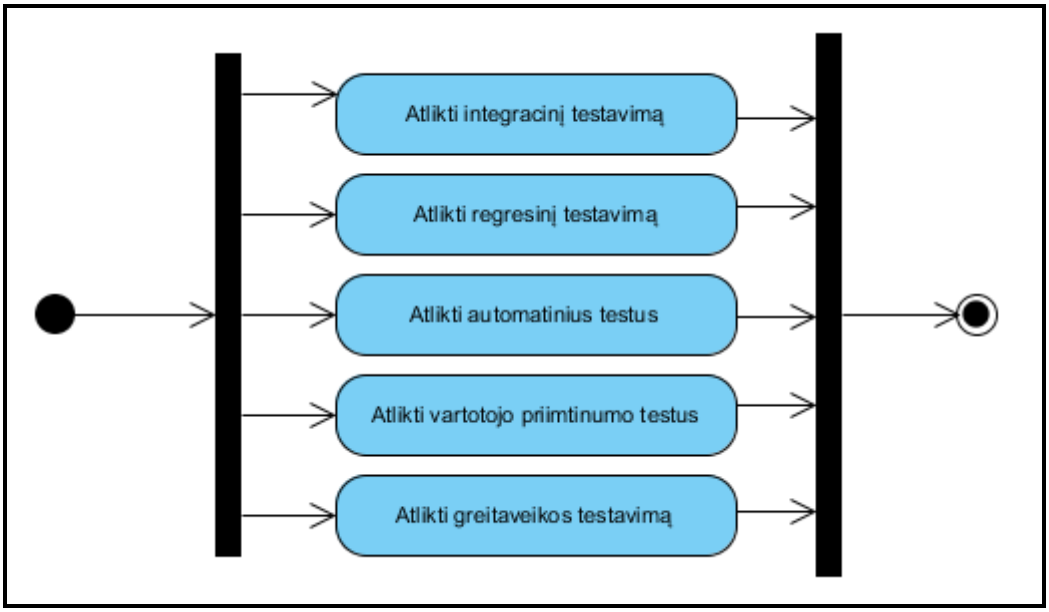
Sistemos realizavimas yra neatsiejamas nuo sistemos klaidų taisymo, kurias identifikavo testuotojai testavimo fazės metu. Sistemos klaidų taisymo procesas pateiktas 10.13 paveikslėlyje parodo, kokių resursu reikia, kad būtų galima atlikti sistemos klaidų taisymą.

10.4 Informacinės sistemos testavimas



10.14 pav. Sistemos testavimo fazės proceso modelis

Sistemos testavimo fazę pradedama vykdyti tuo pat metu, kaip ir realizacijos fazę. Pradžioje testuotojai remdamiesi analitikų padaryta dokumentacija pasiruošia sistemos testavimui, kaip pavaizduota 10.14 paveikslėlyje. Pasiruošimo metu yra susidaromi testavimo planai, pateikti 10.15 paveikslėlyje, kuriais remiantis yra testuojama sistema.



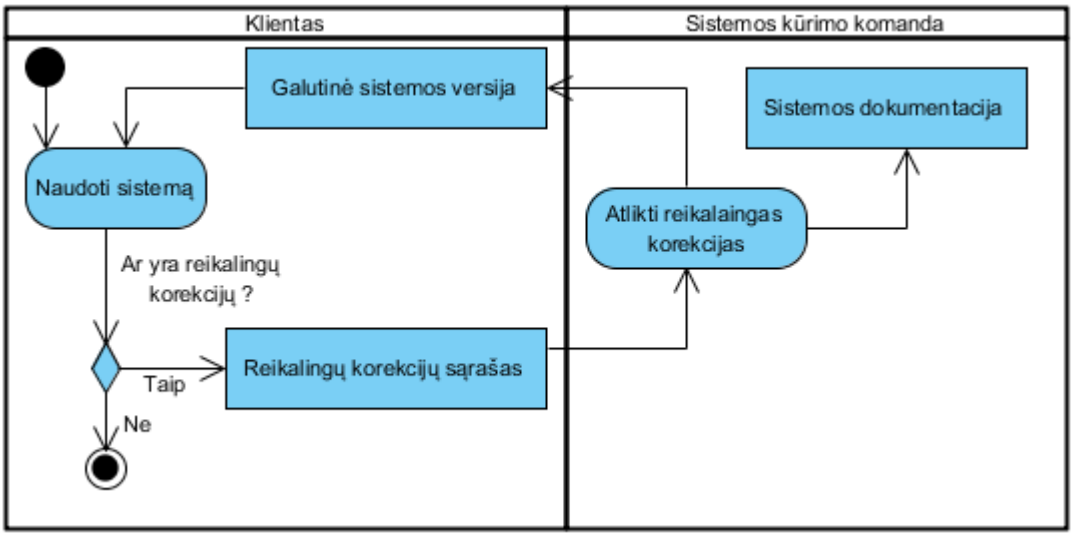
10.15 pav. Sistemos testavimo proceso modelis

Norint tinkamai ištestuoti sistemą reikia atlikti:

1. **Integracijos testavimą** – testavimas, kuris atliekamas po to kai sistemos moduliai apjungiami į grupes. Testuojant tikrinama ar sistema tinkamai veikia ir tinkamai atlikta sistemos modulių integracija;
2. **Regresinį testavimą** – šiuo testavimu siekiama atskleisti kitas sistemos klaidas. Testuojant siekiama išsiaiškinti, kaip padaryti pakeitimai vienoje sistemos dalyje įtakoja sistemos veikimą kitoje dalyje;
3. **Automatinius testus** – kai testavimui pasitelkiama programa, kuri tikrina ar sistemos gražinamos reikšmės yra tokios, kokių buvo tikimasi įvedant tam tikrų reikšmių rinkinį;
4. **Vartotojo priimtinumo testavimą** – testavimo metu tikrinama ar sistema atitinka visus reikalavimus keliamus sistemai, kurie buvo aptarti ir sutarti su klientu;
5. **Greitaveikos testavimą** – atliekamas norint išsiaiškinti sistemos veikimo greitaveiką. Taip tikrinamas sistemos stabilumas ir atsakomumas kai simuliuojamas tam tikras sistemos apkrautumas. Šis testavimas gali padėti atskleisti sistemos silpnąsias vietas, kurias reikia pataisyti prieš atiduodant klientui.

Sistemos testavimo fazė trunka tiek pat laiko, kiek ir sistemos realizavimo fazė, nes nuolat yra testuojami sistemos moduliai ir bendra sistema atlikus pakeitimus sistemos realizacijos kode. Sistemos testavimo metu identifikuotos klaidos yra perduodamos programuotojams, kad jie ištaisytų identifikuotas sistemos klaidas.

10.5 Informacinės sistemos palaikymas



10.16 pav. Sistemos palaikymo fazės proceso modelis

Remiantis DSDM metodika, kurioje minima paskutinioji sistemos realizacijos fazė, pateikta 10.16 paveikslėlyje, sistemos palaikymas. Šios fazės metu sistemos naudojimas perduodamas klientui. Pagal pirmosios fazės metu sutartas palaikymo sąlygas sistemos veikimas yra stebimas jos kūrėjų.

Perdavus sistemos naudojimą užsakovo vartotojams prasideda sistemos beta testavimas, kai sistemos apkrova yra sudaroma realių sistemos vartotojų, kurie gali naudoti sistemą, ne taip kaip numatyta sistemos vartotojo vadove. Dėl tokio naudojimo sistema testuojama, kaip reaguoja į klaidingai pateikiamus duomenis ir atsiradus klaidoms jos greitai yra taisomos.

11 EKSPERIMENTINIS TYRIMAS

Pagal naujai sudarytą metodiką bandoma sukurti informacinę sistemą skirta **studijų modulio vedimo valdymui**. Naudojant metodiką bus stengiamasi sukurti informacinę sistemą, kurios veikimas bus grindžiamas darbų sekų vykdymu. Atliekant eksperimentinį tyrimą bus bandoma pažiūrėti, kaip sudaryta metodika leidžia palengvinti informacinės sistemos kūrimo procesą.

11.1 Pradinė projekto analizė

Kaip aprašant metodiką buvo minėtą, kad kiekvienos fazės metu yra vykdomas Scrum metodikoje aprašomas procesas. Remiantis pateikiama informacija [6] pradedant Scrum metodikos vykdymą pirmiausiai reikia susiplanuoti ir pasidaryti viso projekto darbų sąrašą ir jį papildyti prioritetais, kurie priskiriami atitinkamoms veikloms. Šiuo atveju reikia susidaryti fazės darbų sąrašą, kuriam po to priskirsime prioritetus (žr. 11.1 lent.). Bus galima kiekvienos iteracijos metu iš sudaryto darbų sąrašo pasirinkti veiklas pagal sudėliotus prioritetus ir juos atlikinėti.

11.1 lentelė Darbų sąrašo pavyzdys

Nr.	Veikla	Prioritetas	Iteracija
1.	Surinkti pradinius sistemos reikalavimus	1	1
2.	Aprašyti pradinius reikalavimus	2	1
3.	Atlikti galimybių analizę	3	2

Nr.	Veikla	Prioritetas	Iteracija
4.	Pateikti galimybių analizės sprendimą	4	2
5.	Atlikti verslo poreikių analizę	5	3
6.	Pateikti pradinę verslo poreikių analizę	6	3
7.	Sukurti sistemos prototipą	7	4
8.	Aprašyti sukurtą sistemos prototipą	7	4
9.	Paruošti įsipareigojimų dokumentą	8	5
10.	Papildyti reikalavimų surinkimo klausimų sąrašą	7	4
11.	Derinti sutarties detales	9	6
12.	Pasirašyti sistemos realizacijos sutartį	9	6
13.	Įvertinti projekto veiklų rizikas	10	7
14.	Susidaryti preliminarų projekto planą	11	7

Pagal fazės pradžioje susidarytą darbų sąrašą ir susidėliotus prioritetus pradedami fazės darbai. Pagal pateiktą metodiką **pirmojoje fazėje** pirmiausia reikia analitikams **surinkti pradinius sistemos reikalavimus** (žr. 11.2 lent.), kad būtų galima atlikti **galimybių analizę**, kuri po to nulems tolimesnę projekto eigą. Analitikams bendraujant su klientu ir bandant išsiaiškinti pradinius sistemos reikalavimus, kuriuos galima užrašyti lentelėje.

11.2 lentelė Pradiniai sistemos reikalavimai

Nr.	Reikalavimai
1.	Sistemos veikimas turėtų būti grindžiamas darbų sekų vykdymų
2.	Grafinė vartotojo sąsaja turėtų būti galima koreguoti panaudojant CSS
3.	Sistema gali naudotis tik registruotas vartotojas
4.	Visi sistemos vartotojai turėtų būti suskirstyti į vartotojų grupes, dėl paprastesnio vartotojų grupių valdymo
5.	Visa sistemos grafinė sąsaja turėtų būti bendra visoms sistemos formoms

11.1.1 Sistemos kūrimo galimybių analizė

Remiantis pateiktais pradiniais sistemos reikalavimais ir įsivertinant savo turimus resursus tiek technologinius, tiek žmogiškuosius, galutinis galimybių analizės įvertinimas pateiktas 11.3 lentelėje.

11.3 lentelė Galimybių analizė

Nr.	Reikalavimai	Įvertinimas
1.	Sistemos veikimas turėtų būti grindžiamas darbų sekų vykdymu	Tinka

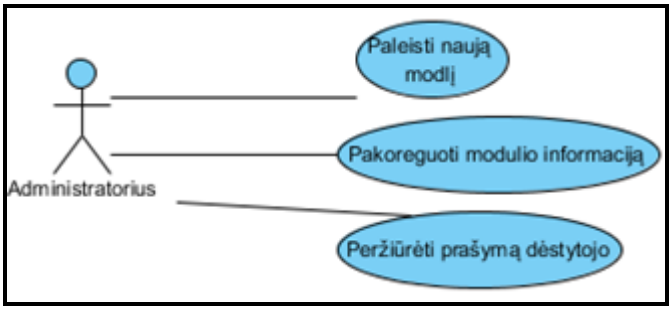
Nr.	Reikalavimai	Įvertinimas
2.	Grafinė vartotojo sąsaja turėtų būti galima koreguoti panaudojant CSS	Tinka
3.	Sistema gali naudotis tik registruotas vartotojas	Tinka
4.	Visi sistemos vartotojai turėtų būti suskirstyti į vartotojų grupes, dėl paprastesnio vartotojų grupių valdymo	Tinka
5.	Visa sistemos grafinė sąsaja turėtų būti bendra visoms sistemos formoms	Tinka
	Galutinis įvertinimas:	**Tinka**

Naudojant sistemos kūrimui Bonita Studio galima sistemas kurti, kurių veikimas grindžiamas **darbų sekų vykdymų**. Taip pat sistemos pagalba galima sugeneruoti sistemos grafinę vartotojo aplinką, kurią po to galima koreguoti pasinaudojant **CSS klasėmis**. Sistemos kūrimo metu realizuojami vartotojai suskirstomi į grupes pagal turimas teises, taip pat kiekvienas vartotojas norėdamas naudotis sistema turi turėti unikalų prisijungimą.

Sistemoje numatomos 3 skirtingos vartotojų grupės, tai yra:

11.4 lentelė Sistemos vartotojų grupės

Nr.	Vartotojų grupės	Aprašymas
1.	Administratorius	Sistemos administratorius, kuris gali paleisti dėstytojo nurodytą modulį ir pakoreguoti jį pagal pateiktus kriterijus. Taip pat gali koreguoti vartotojų grupes
2.	Dėstytojas	Dėstytojas sistemoje gali pateigti prašymą administratoriui dėl norimo modulio paleidimo ir jo korekcijų. Taip pat dėstytojas gali pateikti užduotis studentui ir jas įvertinti
3.	Studentas	Studentas sistemoje turi labiausiai apribotas teises galėdamas tik perskaityti ir atsisiųsti dėstytojo įkeltą medžiagą ir atlikus užduotis jas įkelti, o vėliau pamatyti savo įvertinimą

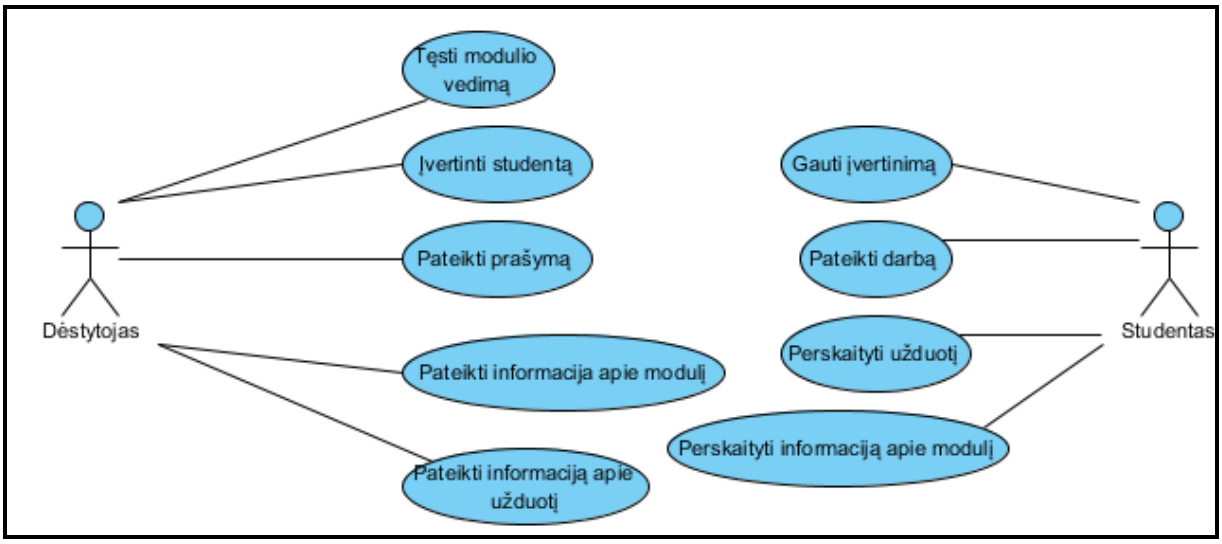


11.1 pav. Sistemos administratoriaus panaudojimo atvejų modelis

11.5 lentelė Administratoriaus panaudojimo atvejų paaiškinimai

Nr.	Panaudojimo atvejis	Aprašymas
1.	Paleisti naują modulį	Sistemos administratorius gavęs dėstytojo prašymą, kad būtų paleistas pasirinktas modulis gali jį paleisti būdamas to modulio

Nr.	Panaudojimo atvejis	Aprašymas
		iniciatoriumi
2.	Pakoreguoti modulio informaciją	Administratorius turi taip pat teises, kuriomis leidžiama koreguoti modulio veiklos procesą ir funkcijas pagal dėstytojo pateiktą prašymą
3.	Peržiūrėti prašymą dėstytojo	Administratorius gali peržiūrėti dėstytojo siųstą pranešimą

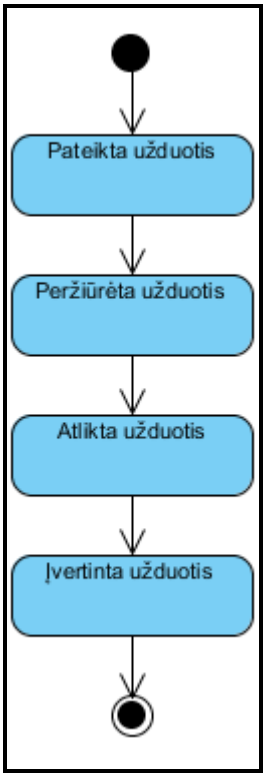


11.2 pav. Dėstytojo ir studento panaudojimo atvejų modelis

11.6 lentelė Dėstytojo ir studento panaudojimų atvejų paaiškinimas

Nr.	Panaudojimo atvejis	Aprašymas
1.	Tęsti modulio vedimą	Dėstytojas gali daryti sprendimą ar nori dar tęsti modulio vedimą
2.	Įvertinti studentą	Dėstytojas gali pagal studento atliktą užduotį jį įvertinti
3.	Pateikti prašymą	Dėstytojas gali pateikti prašymą administratoriui, kad jis paleistų pasirinktą modulį
4.	Pateikti informaciją apie modulį	Dėstytojas pradėdamas modulio vedimą gali pateikti informaciją apie vedama modulį
5.	Pateikti informaciją apie užduotis	Dėstytojas pateikdamas užduotis gali pateikti prisekdamas failus ir juos patalpindamas serveryje. Pateikdamas užduotį gali prisekti tiek teoriją susijusią su pateikiama užduotimi tiek ir pačią užduotį
6.	Gauti įvertinimą	Studentas už pateiktą vertinimui užduoties sprendimą gauna pažymį, kurį gali pamatyti, kai dėstytojas jį įvertina
7.	Pateikti darbą	Studentas gali pateikti užduoties sprendimą vertinimui prisekdamas ir nusiųsdamas dokumentą dėstytojui
8.	Perskaityti užduotis	Po to kai dėstytojas pateikia informaciją apie užduotį studentas ją gali peržiūrėti ir prisegtus failus atsisiųsti į savo kompiuterį

Nr.	Panaudojimo atvejis	Aprašymas
9.	Perskaityti informaciją apie modulį	Studentas gali perskaityti informaciją apie modulį, kurią yra patalpinęs dėstytojas

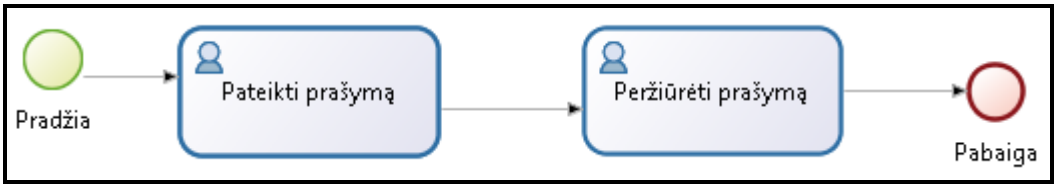


11.3 pav. Užduoties būsenų kitimo modelis

Visos sistemos veikimo metu dėstytojo pateikiamos užduotys gali įgauti 11.3 paveikslėlyje pateiktas būsenas. Dėstytojui sistemos ciklo veikimo pabaigoje pasirinkus, kad norima toliau tęsti modulio vedimą yra per naujo sukuriamos užduotys.

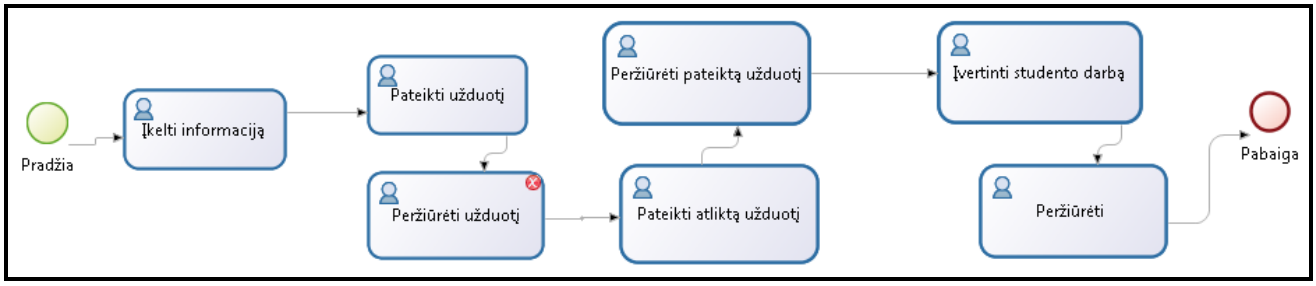
11.1.2 Sistemos veiklos procesų analizė

Atlikus sistemos veiklos procesų analizę buvo nustatyta, kad sistemoje reikia realizuoti studijų valdymo veiklos procesą, kuris būtų inicijuojamas, kai dėstytojas išreiškia prašymą. Pagal pirminius išreikštus reikalavimus veiklos procesai galėtų atrodyti:



11.4 pav. Dėstytojo prašymo pateikimo veiklos proceso modelis

Pagal 11.4 paveikslėlyje pateiktą modelį matoma, kaip dėstytojas pateikia prašymą administratoriui, kad jis paleistų norimą modulį. Administratorius gavęs dėstytojo prašymą su nurodymais korekcijoms modulio, jas atlikęs paleidžia nurodytą modulį, kurio veiklos procesų modelis pavaizduotas 11.5 paveikslėlyje.



11.5 pav. Pasirinkto modulio veikimo proceso modelis

Pateiktame paveikslėlyje aukščiau matomas procesas, kuriame dėstytojas pateikęs modulio informaciją ir patalpinęs užduoties informaciją pateikia viską studentui. Studentas gavęs užduoties aprašymą ir reikalingą teorinę dalį, kurios yra prisegtuose failuose. Atlikęs studentas pateiktą užduotį prisekdamas ir nusiųsdamas ją dėstytojui vertinimui. O po to kai dėstytojas įvertina studento atliktą darbą studentas gali pasižiūrėti savo įvertinimą.

11.1.3 Sistemos įsipareigojimų dokumentas

Tai dokumentas, kuris dažniausiai yra pradinė sistemos dokumentacija, kurioje aprašyti pradiniai reikalavimai, numatomos funkcijos ir numatomi sistemos vartotojai. Numatomos sistemos funkcijos gaunamos iš panaudojimo atvejų.

11.2 Sistemos reikalavimų analizė ir projektavimas

11.7 lentelė Darbų sąrašo pavyzdys

Nr.	Veikla	Prioritetas	Iteracija
1.	Pasinaudojant klausimų sąrašu papildytu surinkti visus sistemos reikalavimus	1	1
2.	Aprašyti surinktus sistemos reikalavimus	2	1
3.	Surinkti veiklos procesų reikalavimus	1	1
4.	Sudokumentuoti veiklos procesų reikalavimus	2	1
5.	Sistemos modeliavimas laikantis sistemos reikalavimų	3	2
6.	Sumodeliuoti sistemos veiklos procesų diagramas	3	2
7.	Aprašyti grafinius sistemos modelius	4	2

11.2.1 Funkciniai sistemos reikalavimai

11.8 lentelė Funkcinis reikalavimas nr. 1

Reikalavimo tipas	Funkcinis reikalavimas
Reikalavimas	Tikrina ar tinkami duomenys įvesti į įvedimo laukelius
Iniciatorius	Dėstytojas
Išpildymo kriterijus	Tikrinami įvedimo laukeliai ar į laukelį kur reikia įvesti skaičius nevedamos raidės
Reikalavimo poreikis	Kad nebūtų pateikinėjami tekstai, kurie tik apkrauna sistemos darbą
Paskutinį kartą koreguotas/ sukurtas	2013-04-15

11.9 lentelė Funkcinis reikalavimas nr.2

Reikalavimo tipas	Funkcinis reikalavimas
Reikalavimas	Prisegamo failo dydis negali būti didesnis nei 15 mb
Iniciatorius	Dėstytojas
Išpildymo kriterijus	Tikrinti prisegamo failo dydžius ir neleisti prisegti didesnio nei 15 mb failo
Reikalavimo poreikis	Kad būtų apsisaugojama nuo kenkėjiško didelių failų siuntimų
Paskutinį kartą koreguotas/ sukurtas	2013-04-15

11.10 lentelė Funkcinis reikalavimas nr.3

Reikalavimo tipas	Funkcinis reikalavimas
Reikalavimas	Turi leisti prisegti failus, kuriuos norima persiųsti
Iniciatorius	Dėstytojas
Išpildymo kriterijus	Padaryti, kad leistu norimus failus siųsti leistų pasirinkus iš sąrašo prisegti
Reikalavimo poreikis	Dėl paprastesnio sistemos naudojimo
Paskutinį kartą koreguotas/ sukurtas	2013-04-15

11.2.2 Nefunkciniai sistemos reikalavimai

11.11 lentelė Nefunkcinis reikalavimas nr.1

Reikalavimo tipas	Nefunkcinis reikalavimas
Reikalavimas	Aiškūs ir lengvai suprantami mygtukų ir laukų pavadinimai
Iniciatorius	Dėstytojas
Išpildymo kriterijus	Sistemoje naudojamų laukelių pavadinimai ir jų prasmės lengvai suprantamos
Reikalavimo poreikis	Dėl paprastesnio sistemos naudojimosi
Paskutinį kartą koreguotas/ sukurtas	2013-04-15

11.12 lentelė Nefunkcinis reikalavimas nr.2

Reikalavimo tipas	Nefunkcinis reikalavimas
Reikalavimas	Ateityje esant poreikiui turi leisti įterpti naujas funkcijas.
Iniciatorius	Dėstytojas
Išpildymo kriterijus	Padaryti, kad ateityje esant poreikiui būtų galima pridėti naujų funkcijų
Reikalavimo poreikis	Dėl sistemos platesnio panaudojimo
Paskutinį kartą koreguotas/ sukurtas	2013-04-15

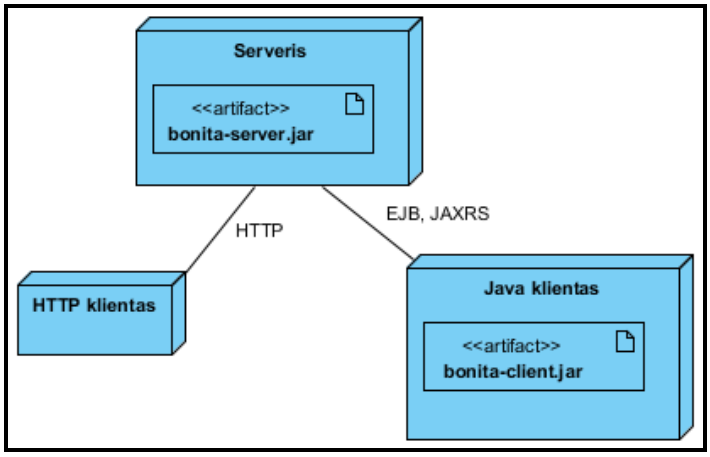
11.13 lentelė Nefunkcinis reikalavimas nr.3

Reikalavimo tipas	Nefunkcinis reikalavimas
Reikalavimas	Saugumas: kitiems skyrių darbuotojams nesuteikti kitų skyrių slaptažodžių
Iniciatorius	Dėstytojas
Išpildymo kriterijus	Padaryti, kad leistu norimus failus siųsti leistų pasirinkus iš sąrašo prisegti
Reikalavimo poreikis	Dėl paprastesnio sistemos naudojimo
Paskutinį kartą koreguotas/ sukurtas	2013-04-15

11.14 lentelė Nefunkcinis reikalavimas nr.4

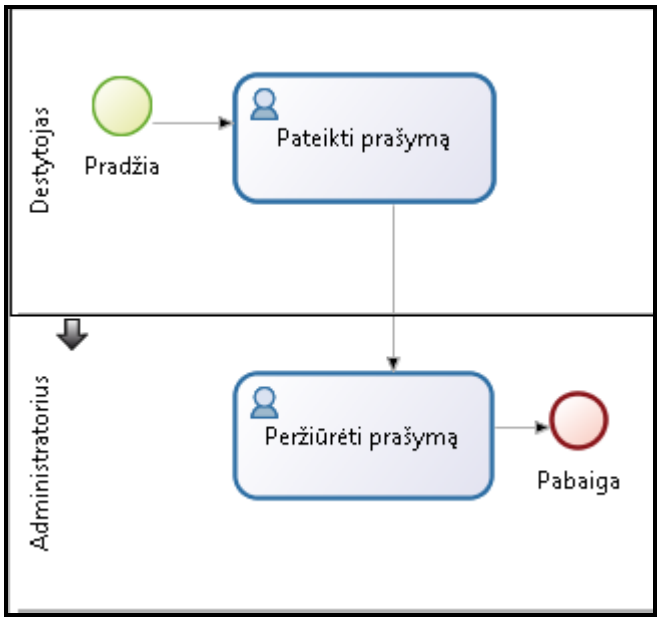
Reikalavimo tipas	Nefunkcinis reikalavimas
Reikalavimas	Sistemos grafinės vartotojo sąsajos spalvos neturėtų varginti akių
Iniciatorius	Dėstytojas
Išpildymo kriterijus	Padaryti sistemos grafinės vartotojo sąsajos spalvas neryškias ir paprastas
Reikalavimo poreikis	Dėl sveikatos sumetimų
Paskutinį kartą koreguotas/ sukurtas	2013-04-15

11.2.3 Sistemos grafinių modelių modeliavimas



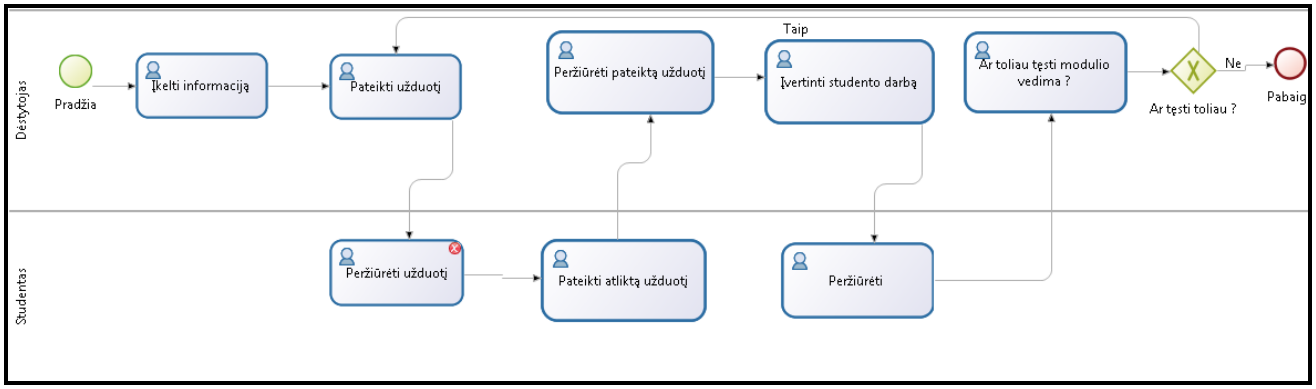
11.6 pav. Sistemos diegimo modelis

Pavaizduotame modelyje 11.6 paveiksle matoma, kaip sistema turėtų būti diegiama ir kokie ryšiai tarp atskirų sistemos dalių būtų.



11.7 pav. Dėstytojo ir administratoriaus bendravimo veiklos procesas

Pateiktame 11.7 paveikslėlyje pavaizduota, kaip dėstytojas pateikia prašymą administratoriui, kad jis paleistų jo norimą modulį ir jeigu reikia jį pataiso pagal pateiktus kriterijus. Tai administratorius gali padaryti pakoreguodamas ir po to paleisdamas 11.8 paveikslėlyje pateiktą veiklos procesą.



11.8 pav. Dėstytojo ir studento veiklos proceso modelis

Šiame veiklos proceso modelyje, pavaizduotame 11.8 paveiksle, pateiktas visas modulio vedimo veiklos procesas. Šiame procese pavaizduota, kaip dėstytojas pateikia modulio aprašą, įkelia užduotis ir jų teorinę dalį bei perduoda užduotis vykdyti studentams, kurie atlikę užduotis siunčia atliktas užduotis dėstytojui, kad jis įvertintu. Peržiūrėjęs dėstytojas pateikia studento darbo įvertinimą ir persiunčia jį studentui..

11.3 Sistemos realizacijos kūrimas

11.15 lentelė Darbų sąrašo pavyzdys

Nr.	Veikla	Prioritetas	Iteracija
1.	Susidaryti veiklų sąrašą, kurias reikia realizuoti	1	1
2.	Realizuoti sistemos darbų sekų vykdymo modulį	2	2
3.	Realizuoti sistemos funkcinius reikalavimus	2	3
4.	Realizuoti sistemos nefunkcinius reikalavimus	3	3
5.	Sistemos klaidų taisymas	4	4

Pateikiamos realizuotos sistemos langų nuotraukos. Visa sistemos realizacija atlikta remiantis aprašyta metodika ir apsirašytus funkcinius ir nefunkcinius reikalavimus.

11.9 pav. Sistemos vykdymo lango vaizdas

Šiame paveikslėlyje (žr. 11.9 pav.) pateikiama sistemos formos vaizdas, kuriame matome kaip dėstytojas pateikia prašymą administratoriui, kad paleistu norimą modulį su padarytais atitinkamais pakeitimais.

11.10 pav. Sistemos vartotojo sąskaitos vaizdas

Sistemos vartotojo sąskaitos vaizdas pavaizduotas 11.10 paveikslėlyje matomas, kai prie sistemos prisijungia vartotojas. Iš sąskaitos kairiame šone esančių filtrų gali pasitikrinti kiek naujų darbų yra gavęs ir kokių darbų dar nėra atlikęs.

11.11 pav. Prisijungimo prie sistemos langas

Šiame 11.11 paveikslėlyje pateikiama sistemos prisijungimo forma. Prie sistemos gali prisijungti tik autorizuoti sistemos vartotojai. Padarius šią formą įgyvendintas vienas iš nefunkcinių reikalavimų, kad neatskleisti sistemos vartotojų slaptažodžių kitiems. Slaptažodžio įvedimo laukelis šifruojamas kiekvieną įvedama simbolį atvaizduojant rutuliuku.

11.4 Sistemos testavimas

11.16 lentelė Darbų sąrašo pavyzdys

Nr.	Veikla	Prioritetas	Iteracija
1.	Pasiruošti testavimo planus	1	1
2.	Sistemos testavimas remiantis testų planais	2	2

Nr.	Veikla	Prioritetas	Iteracija
3.	Rastų klaidų registravimas	2	2

Dėl to, kad kuriama sistemos apimtis yra labai nedidelė, tai nėra daromi dideli testavimo darbai. Testuojant sistemą patikrinami laukai ar teisingai atvaizduoja informaciją, kuri buvo įvesta ir ta, kurią turėjo atvaizduoti.

11.4.1 Sistemos testavimo atvejai

11.17 lentelė Sistemos testavimo suvestinė

Testavimo atvejis	Įvesta reikšmė	Tikėtasi reikšmės	Gauta reikšmė
Vidurkio skaičiavimas	4 ir 8	6	4
Vidurkio skaičiavimas (po klaidos taisymo)	4 ir 8	6	6
Pažymio įvedimas	Abcd	Klaida	Klaida
Pažymio skaičiaus formato tikrinimas	9,5	Klaida	9,5
Pažymio skaičiaus formato tikrinimas (po klaidos taisymo)	9,5	Klaida	Klaida
Bandoma neįvedus pažymio toliau vykdyti programą	Vykdyti neįvedus pažymio	Klaida	Klaida

11.5 Sistemos palaikymas

11.18 lentelė Darbų sąrašo pavyzdys

Nr.	Veikla	Prioritetas	Iteracija
1.	Perduoti sistemą klientui	1	1
2.	Sekti kliento pateikiamas klaidas	2	2
3.	Taisyti kliento nurodytas klaidas	3	3

Sukurtos sistemos palaikymas vykdomas ir stengiamasi kuo greičiau taisyti sistemoje iškilusias problemas susijusias su sistemos veikimu. Taip pat sistemos palaikymo metu konsultuojami sistemos vartotojai naudojimosi klausimais ir surinkus grupelę žmonių atliekami vidiniai mokymai, kaip naudotis sistema.

IŠVADOS

1. Sudarius metodiką ji buvo išbandyta pritaikant ją kuriant eksperimentinę informacinę sistemą, pagal, kurios realizacijos rezultatus galima daryti išvadas, kad sudaryta metodika padeda lengvai sukurti informacinę sistemą grindžiamą veiklos procesų veikimu.
2. Pagal atlikta esamų metodikų analizę prieita išvados, kad geriausiai tinkama yra RUP, o Agile Scrum ir DSDM puikiai tinka, kaip papildančios metodikos.
3. Sudarant naują metodiką skirtą darbų sekų vykdymų skirtų informacinių sistemų kūrimui panaudotos Agile Scrum, DSDM ir RUP metodikų geriausios savybės, kurios padės užtikrinti tinkamą sistemos kūrimo ciklą.
4. Atliekant eksperimentinės informacinės sistemos kūrimą pastebėtą, kad naudojant sudarytą naują metodiką jos fazes galima vykdyti lygiagrečiai taip sumažinant projekto trukmę ir pilnai išnaudojant turimus resursus.

LITERATŪRA

1. BPEL (Business Process Execution Language) [žiūrėta 2011-10-17] Prieiga per internetą: http://searchsoa.techtarget.com/definition/BPEL

2. BPMN (Business Process Model and Notation) [žiūrėta 2011-10-17] Prieiga per internetą: http://www.omg.org/spec/BPMN/2.0/

3. Wil van der Aalst ir Kees van Hee Workflow managament models, methods and systems. – K.: The MIT Press, 2004. – 369 p.

4. Scrum Is an Innovative Approach to Getting Work Done [žiūrėta 2011-11-15] Prieiga per internetą: http://www.scrumalliance.org/learn_about_scrum

5. The Agile Unified Process (AUP) [žiūrėta 2011-10-17] Prieiga per internetą: http://www.ambysoft.com/unifiedprocess/agileUP.html

6. Scrum (development) [žiūrėta 2011-12-17] Prieiga per internetą: http://en.wikipedia.org/wiki/Scrum_(development)

7. RUP (Rational Unified Process) [žiūrėta 2011-12-17] Prieiga per internetą: http://medlem.spray.se/perlin27/rup.html

8. Failure Causes Statistics [žiūrėta 2011-12-17] Prieiga per internetą: http://www.it-cortex.com/Stat_Failure_Cause.htm

9. Agile-Scrum [žiūrėta 2011-12-17] Prieiga per internetą: http://www.slideshare.net/rajivmisra/agile-scrum-methodology

10. Documents Associated With Business Process Model And Notation (BPMN) Version 2.0 [2011-12-17] Prieiga per internetą: http://www.omg.org/spec/BPMN/2.0

11. Reikalavimų analizė [2013-05-13] Prieiga per internetą: http://vaidila.vdu.lt/~i5dasi/se2/paskaitos/03_reikalavimai.pdf

12. Rational Unified Process Best Practices for Software Development Teams [2013-05-13] Prieiga per internetą: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

13. The Rational Unified Process For Dummies [2013-05-13] Prieiga per internetą: http://www.perftestplus.com/resources/rupfordummies_ppt.pdf

14. Failure Causes [2013-05-13] Prieiga per internetą: http://www.it-cortex.com/Stat_Failure_Cause.htm

15. What is the difference between RUP and SCRUM methodologies? [2013-05-13] Prieiga per internetą: http://www.chiron-solutions.com/chiron-professional-journal/2010/12/20/what-is-the-difference-between-rup-and-scrum-methodologies/

16. RUP – Rational Unified Process [2013-05-13] Prieiga per internetą: http://blog.otssolutions.com/rup-rational-unified-process/

17. The Crystal Methods, or How to make a methodology fit [2013-05-13] Prieiga per internetą: http://agile2007.agilealliance.org/downloads/handouts/Cockburn_818.pdf

18. Crystal Methods [2013-05-13] Prieiga per internetą: http://en.wikiversity.org/wiki/Crystal_Methods

19. Agile Methodologies: Crystal methods [2013-05-13] Prieiga per internetą: http://22andinvincible.wordpress.com/2012/04/17/agile-methodologies-crystal-methods/

20. The Crystal Methodologies [2013-05-13] Prieiga per internetą: http://www.itu.dk/courses/SASU/F2010/files/0666_001.pdf

21. RUP: Is an iterative development process, consisting of four pahses and nine descipline. [2013-05-13] Prieiga per internetą: http://www.hytechpro.com/our-approach/rup

22. Workflow [2013-05-13] Prieiga per internetą: http://encyclopedia2.thefreedictionary.com/Workflow+management+system

23. What is DSDM? [2013-05-13] Prieiga per internetą: http://www.codeproject.com/Articles/5097/What-Is-DSDM

24. The Top Ten Reasons Projects Fail (Part 7) [2013-05-13] Prieiga per internetą: http://www.projectmanagement.com/articles/187449/The-Top-Ten-Reasons-Projects-Fail--Part-7-

25. Project Management: Why Projects Fail [2013-05-13] Prieiga per internetą: http://www.projectsmart.co.uk/project-management-why-projects-fail.html

26. UML Standard Diagrams [2013-05-13] Prieiga per internetą: http://www.tutorialspoint.com/uml/uml_standard_diagrams.htm

27. Volere Requirements Specification Template [2013-05-13] Prieiga per internetą: http://www.volere.co.uk/template.htm