KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

DIANA VASILIAUSKAITĖ

REIKALAVIMŲ SISTEMAI SPECIFIKACIJOS INTEGRALUMO SU TESTAVIMO PROCESU TYRIMAS

Magistro darbas

Darbo vadovas
Prof. Dr. R. Butleris

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

DIANA VASILIAUSKAITĖ

REIKALAVIMŲ SISTEMAI SPECIFIKACIJOS INTEGRALUMO SU TESTAVIMO PROCESU TYRIMAS

Magistro darbas

Darbo vadovas:
Prof. Dr. R. Butleris
2013-05-24

Recenzentas:
doc. dr. E. Karčiauskas
2013-05-24

Atliko:
IFM-0/4 gr. studentė
Diana Vasiliauskaitė
2013-05-24

KAUNAS, 2013

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLĖLIŲ SĄRAŠAS

Summary

Research of System Requirements Specification Integrity with the Testing Process

The aim of this work is to create a requirements specification integrity with testing process method. For this purpose a process model was developed, which consists of five main tasks: requirements identification, specification, testing plan, traceability links creation and research. The first two tasks helps to decrease the dispersal of crosscutting requirements in requirements specification, establishes the required requirements and dependencies. The goal of composition task is to compose the requirements to give the developer a view of the whole system and to identify and manage conflicts between requirements. Integrity of requirements are ensured keeping all requirements in a single form.

The paper analyzes the identification of crosscutting requirements and gathering strategies. Templates of requirements specification, requirements of standards classification and requirements management tools were analyzed. Methods and techniques for the integrity were analyzed.

The unique template for requirements specification was created, which includes functional and non-functional requirements, functional and non-functional crosscutting requirements. The template for requirements management was made in RequisitePro environment.

Keywords: requirements, requirement specification, requirements specification template, RequisitePro tool, test cases, test plan.

TERMINŲ IR SANTRAUPŲ ŽODYNAS

NFR – nefunkcinis reikalavimas

FR – funkcinis reikalavimas

IT – informacinės technologijos

UML (angl. Unified Modeling language) – unifikuota modeliavimo kalba

SRS (angl. Software Requirements Specification) – Programinės įrangos reikalavimų Specifikacija

URS (angl. User Requirements Specification) – vartotojo reikalavimų specifikacija

DOORS (angl. Dynamic Object Oriented Requirements System) – dinaminis objektais orientuota reikalavimų sistema.

VGRA – verte grindžiamas reikalavimų atsekamumas

SORA – savybėmis orientuotas reikalavimų atsekamumas

RSISTPM - reikalavimų sistemai specifikacijos integralumą su testavimo procesu užtikrinantis modelis.

Įvadas

Šiuo metu pasaulyje yra vykdoma milijonai informacinių technologijų projektų. Visų šių projektų tikslas – laiku ir neviršijant nustatyto biudžeto sukurti vartotojo reikmes atitinkančią kompiuterizuotą sistemą. Tačiau pasaulio IT projektų statistika rodo, kad dauguma tokių projektų turi rimtų problemų: viršija biudžetą, vėluoja, dažnai nepasiekia užsibrėžtų tikslų. Tik mažiau nei ketvirtadalis visų IT projektų pasiekia užsibrėžtus tikslus laiku ir neviršydami biudžeto [1]. Žinoma, kad esant tokiai statistikai, būtina ieškoti nesėkmių priežasčių, o po to ir būdų, kaip su jomis kovoti.

Programinės įrangos sistemos tampa vis svarbesnės organizacijoms, tuo pačiu metu tapdamos vis didesnėmis ir sudėtingesnėmis. Programinės įrangos kokybės reikalavimai taip pat auga. Programinės įrangos klaidos sąlygoja didelių pinigų sumų ar netgi žmonių gyvybių praradimą. Jeigu kokybė netampa geresnė augant sistemos dydžiui, sudėtingumui ir svarbumui, šie praradimai tik didėja.

Kokybės užtikrinimas programinės įrangos kūrime prasideda nuo reikalavimų. Reikalavimai yra kiekvieno projekto pagrindas. Jie aprašo ko kiekviena suinteresuota šalis – naudotojai, klientai, tiekėjai, kūrėjai – tikisi iš potencialiai naujos sistemos, taip pat ką sistema privalo daryti, kad patenkintų jų visų poreikius. Nuo sėkmingo reikalavimų surinkimo, dokumentavimo didele dalimi priklauso viso projekto kūrimo sėkmė. Pagal reikalavimus planuojama projekto apimtis, biudžetas, vykdymo darbų išdėstymas laike.

Nepakankamas užsakovo įtraukimas, neišsamūs, nuolat besikeičiantys reikalavimai yra pagrindinė programinės įrangos kūrimo nesėkmių priežastis. Norint išspręsti šią problemą, į pagalbą galima pasitelkti testavimą.

Per daugelį metų daugybė skirtingų metodų buvo siūloma testavimo atvejų generavimui. Testavimo atvejai gali būti generuojami iš sistemos reikalavimų. Vienas iš privalumų generuojant testavimo atvejus iš specifikacijos yra tas, kad jie gali būti sukurti ankstyvame programinės įrangos kūrimo etape, ir gali būti parengti naudojimui dar prieš sukuriant programas. Be to, kai testavimo atvejai sugeneruojami anksti, galima aptikti neaiškumų ir prieštaravimų reikalavimų specifikacijoje, valdyti reikalavimų pasikeitimus ir įvertinti poveikį kitiems reikalavimams dar prieš pradėjus projektuoti sistemą. Tai neabejotinai sumažina programinės įrangos kūrimo kainą, nes sistemos klaidos yra pašalinamos sistemos projektavimo etapo pradžioje, o kuo toliau pažengęs projektas, tuo aukščiau kyla problemų taisymo kaina.

Šiame darbe siekima sudaryti reikalavimų specifikacijos integralumą su testavimo procesu užtikrinantį metodą, remiantis VGRA (*angl. Value Based Requirement Traceability*) ir Volere šablonu. Siekiant užtikrinti reikalavimų ir testavimo atvejų integralumą, būtina sudaryti sąsajas tarp reikalavimų ir jiems testuoti skirtų testavimo atvejų. Būtina identifikuoti besikeičiančius reikalavimus ir nustatyti jų daromą įtaką kitiems reikalavimams bei testavimo atvejams. Visiškas

reikalavimų ištestavimas negalimas, todėl būtina nustatyti reikalavimų prioritetus atitinkamai pagal jų pasikeitimo riziką bei vertę.

Reikalavimais pagrįstas testavimo metodas, užtikrinantis visišką reikalavimų specifikacijos ir testavimo proceso integralumą, įgyvendinimas gali padidinti programinės įrangos kūrimo veiksmingumą, sumažinti nesėkmės riziką ir pagerinti bendrą programinės įrangos kokybę. Reikalavimais grįstas testavimo procesas padeda ne tik aptikti defektus, bet ir jų išvengti.

Reikalavimais paremtas testavimo procesas apima du pagrindinius klausimus: pirmas, ar reikalavimai yra teisingi, išsamūs, nedviprasmiški ir logiškai nuoseklūs; ir antras, ar iš tų reikalavimų suprojektuoti būtini ir pakankami testavimo atvejai, kurie užtikrintų, kad dizainas ir kodas pilnai atitinka šiuos reikalavimus. Projektuojant testus šiems dviem klausimams, reikia įveikti šiuos dalykus: sumažinti nepaprastai didelį skaičių potencialių testų iki pagrįstai nustatyto dydžio komplekto ir užtikrinti, kad testai gaus tinkamus atsakymus dėl tinkamų priežasčių.

Tyrimo tikslas – sukurti reikalavimų specifikacijos integralumą su testavimo procesu užtikrinantį metodą, sudarant reikalavimų rinkinius reikalavimų specifikacijoje, taip užtikrinant reikalavimų ir testavimo proceso vientisumą. Metodas sudaromas remiantis VGRA (*angl. Value Based Requirement Traceability*) ir pasirinkto reikalavimų specifikavimo šablono pagrindu.

Tyrimo uždaviniai:

- Atlikti reikalavimų valdymo įrankių analizę ir pasirinkti tinkamą įrankį metodo realizavimui;
- Atlikti reikalavimų specifikavimo šablonų analizę;
- Papildyti pasirinktą šabloną, kuris išpildytų reikalavimų sistemai specifikacijos integralumo su testavimo procesu užtikrinimo metodą.
- Išbandyti metodo taikymą pasirinktai informacinei sistemai.

Šį darbą sudaro įvadas, keturi pagrindiniai skyriai, išvados, literatūros sąrašas.

Pirmame skyriuje nagrinėjama programinės įrangos kokybė, reikalavimų surinkimo problemos, reikalavimų atsekamumo nauda, reikalavimų specifikacija grindžiamas testavimas, reikalavimų surinkimui ir testavimui naudojami standartai.

Antrame darbo skyriuje nagrinėjami esami sprendimai. Išanalizuoti reikalavimų specifikavimo šablonai, bei reikalavimų valdymo įrankiai.

Trečiame skyriuje pateikta specifikavimo procesų modeliai. Integralumo modelis projektuojamas bei sudaromas remiantis analitinės dalies medžiaga. Modelis atvaizduotas hierarchine notacija, kurioje kiekviena veikla turi hierarchinę tvarką.

Ketvirtas darbo skyrius aprašo pasirinktų priemonių išplėtimą. Reikalavimų valdymui

sudarytas šablonas RequisitePro įrankiu. Sukurtas šablonas, atsižvelgiant į reikalavimų ir testavimo atvejus naudojamus metodikoje. Penktame skyriuje pateikiamos išvados. Šeštame skyriuje naudota literatūra.

1. REIKALAVIMŲ SISTEMAI SPECIFIKACIJOS INTEGRALUMO SU TESTAVIMO PROCESU ANALIZĖ

1.1.Analizės tikslas

Magistro darbo tikslas - išnagrinėti ir aprašyti reikalavimų sistemai specifikacijos integralumo su testavimo procesu tyrimą ir testavimo metodikas, išbandyti praktinį jų taikymą parodant, kaip pagal jas kuriami testavimo atvejai ir testuojamos kompiuterinės sistemos bei sprendžiamos jų, o kartu ir informacinių sistemų kūrimo, kokybės įvertinimo ir perdavimo užsakovui problemos.

1.2. Tyrimo sritis, objektas ir problema

Tyrimo objektas – sistemos reikalavimų specifikacijos integralumą su testavimo procesu užtikrinantys metodai ir priemonės.

Tyrimo sritis - apima reikalavimų valdymo įrankių, šablonų analizę, testo plano bei testavimo atvejų kūrimo metodus. Tiriami metodai ir jų pritaikymas praktikoje.

Problema – reikalavimų gausa, jų dviprasmiškumas, neišbaigtumas. Neišsamūs reikalavimai dažniausiai atsiranda dėl neišsamaus ir netikslaus reikalavimo specifikavimo, dėl tokių reikalavimų atsiranda grėsmė galutiniam projektui. Esant netiksliems reikalavimams, gaunamas produktas neatitinkantis netenkinantis klientų poreikių.

1.3. Programinės įrangos kūrimas

Programinės įrangos kūrimo procesas yra veiklų ir su jomis susijusių rezultatų rinkinys, kuris veda prie programinės įrangos produkto kūrimo. Tokios veiklos gali apimti programinės įrangos kūrimą nuo pat pradžių arba praplečiant ar modifikuojant egzistuojančias sistemas [1].

Šiandien rinkoje egzistuoja didelis skaičius programinės įrangos kūrimo proceso programų, tačiau nėra idealaus proceso. Nors yra naudojama daug skirtingų procesų, egzistuoja pagrindinės veiklos, kurios įprastos visiems programinės įrangos kūrimo procesams [1]:

1. *Programinės įrangos specifikavimas*: programinės įrangos funkcionalumas ir jos operacijų apribojimai, privalo būti apibrėžti;
2. *Programinės įrangos projektavimas ir kūrimas*: turi būti sukurta programinė įranga, atitinkanti reikalavimų specifikaciją;

3. *Programinės įrangos validavimas*: sukurta programinė įranga privalo būti tikrinama su reikalavimų specifikacija, kad būtų įsitikinta, jog ji atitinka savo paskirtį;
4. *Programinės įrangos raida*: daugeliu atvejų programinė įranga privalo plėtotis, kad atitiktų besikeičiančius kliento reikalavimus.

1.4. Programinės įrangos kokybė

Kokybės sąvoka yra gana sudėtinga, ir autoriai bei organizacijos per daugelį metų terminą „kokybė“ apibrėžia skirtingai. Phil Crosby [2] apibrėžia jį kaip „vartotojų reikalavimų atitikimas“. Watts Humphrey [3] apibrėžia jį kaip „puikaus lygio tinkamumo naudojimui pasiekimas“, tuo tarpu IBM [4] naudoja frazę „rinkos grindžiama kokybė“, kuria jie grindžia visišką kliento pasitenkinimo pasiekimą. Baldrige programa [5] kokybei naudoja panašią frazę „kliento grindžiama kokybė“, ji kliento pasitenkinimą laiko pagrindiniu rūpesčiu. Kokybė buvo apibrėžta standarte ISO 9001:2000 [6], kaip “laipsnis, kuriuo neatsiejamas savybių rinkinys tenkina reikalavimus“.

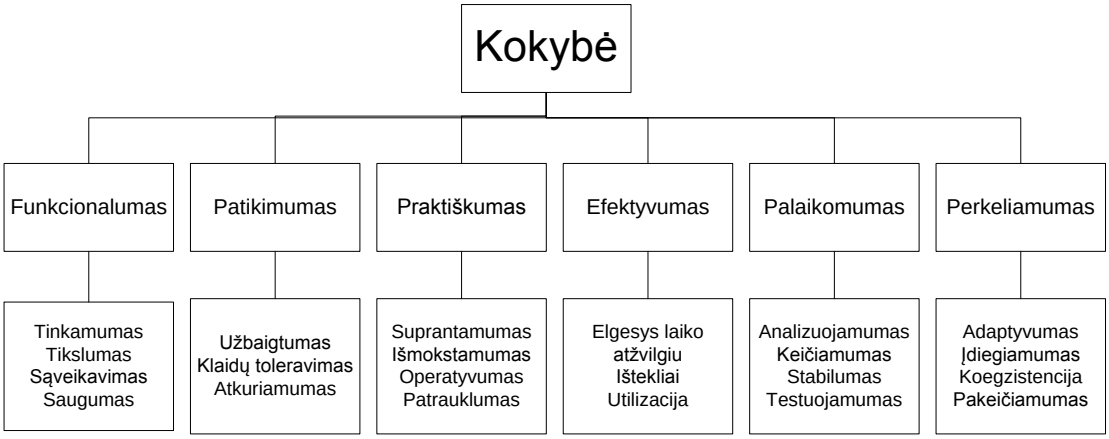
Programinės įrangos kokybė privalo būti sukurta remiantis sąvokomis:

- Programinės įrangos reikalavimai yra pagrindas ant kurio matuojama kokybė. Reikalavimų atitikimo trūkumas yra ir kokybės trūkumas;
- Specifiniai standartai apibūdina kūrimo kriterijus, kurių turi būti laikomasi kuriant programinę įrangą. Jeigu kriterijų nesilaikoma, beveik visada rezultate bus kokybės trūkumas [7].

ISO 9126 kokybės standartas yra tarptautinis standartas programinės įrangos kūrimui. Šiame standarte aprašomas kokybės modelis klasifikuoja programinės įrangos kokybę į rinkinį charakteristikų ir subcharakteristikų [8].

Kiekviena kokybės subcharakteristika yra toliau dalinama į atributus. Atributas yra esybė, kuri gali būti patvirtinta arba išmatuojama programinėje įrangoje. Šių atributų žinojimas gali būti naudingas reikalavimų identifikavimo fazėje, taip pat testavimo metu.

1 pav. vaizduoja kokybės aspektus nurodomus ISO 9126 standarte [8]:



1 pav. ISO 9126 standarto kokybės modelis [8]

Funkcionalumas – tai gebėjimas teikti funkcijas, kurios atitinka aiškius ir besąlygiškus programinės įrangos funkcinius reikalavimus. Susiję subaspektai:

- **Tinkamumas** - programos funkcijų, atliekančių reikiamas užduotis, pilnumas bei atitikimas reikalavimams;
- **Tikslumas** - programos veikimas pateikiant teisingus arba sutartus rezultatus;
- **Sąveikavimas** - programos bendradarbiavimo su kitomis sistemomis galimybės;
- **Saugumas** - programos galimybės uždrausti neautorizuotą priėjimą prie programos arba programos duomenų.

Patikimumas – gebėjimas laikytis sutarto elgesio sutartomis sąlygomis. Susiję subaspektai:

- **Užbaigtumas** – gebėjimas išvengti programinės įrangos sustabdymo, klaidų atveju;
- **Klaidų toleravimas** – gebėjimas vykdyti sutartus elgesio lygius, klaidų atveju;
- **Atkuriamumas –** gebėjimas atkurti nustatytus elgesio lygius ir duomenis po klaidų.

Praktiškumas – programinės įrangos gebėjimas būti suprastai ir naudojamai su vartotojo pasitenkinimu, sutartomis sąlygomis. Susiję subaspektai:

- **Suprantamumas** - vartojo pastangos, reikalingos programos loginio konteksto atpažinimui;
- **Išmokstamumas** - vartotojo pastangos, reikalingos siekiant išmokti dirbti su programa;
- **Operatyvumas** – funkcijų, skirtų naudotis programine įranga, prieinamumas;
- **Patrauklumas** – potraukis naudotis programa.

Efektyvumas – ryšys tarp elgesio ir išteklių, naudojamų normaliomis sąlygomis. Susiję subaspektai:

- **Elgesys laiko atžvilgiu** - programos atsako bei veikimo laikas;
- **Ištekliai** - programos naudojamų resursų apimtis bei jų panaudojimo trukmė.

Palaikomumas – tai programinės įrangos galimybė būti modifikuotai su mažai pastangų. Susiję subaspektai:

- **Analizuojamumas** - pastangų apimtis, reikalinga programų trūkumų arba defektų analizei, arba modifikuojamų programos dalių nustatymui;
- **Keičiamumas** - pastangų apimtis, reikalinga programos modifikacijoms, klaidų pašalinimui arba perėjimui prie kitos funkcionavimo aplinkos;
- **Stabilumas** - rizikos dydis susijęs su nenuspėjamu funkcionavimu po programos modifikacijų;
- **Testuojamumas** - pastangų apimtis, reikalinga atliekant programinės įrangos validavimą po modifikavimo.

Perkeliamumas – programinės įrangos gebėjimas būti perkeltai iš vienos operacinės sistemos į kitą. Susiję subaspektai:

- **Adaptyvumas** – programinės įrangos gebėjimas save adaptuoti naujoje operacinėje sistemoje, taikant atitinkamus veiksmus;
- **Įdiegiamumas** – pastangos reikalingos įdiegti programinei įrangai specifinėje aplinkoje;
- **Koegzistencija** – programinės įrangos gebėjimas koegzistuoti su kita programine įranga toje pačioje aplinkoje, dalinantis resursais;
- **Pakeičiamumas** - programinės įrangos gebėjimas būti naudojamai kitos programinės įrangos vietoje, ir atlikti tuos pačius veiksmus toje pačioje aplinkoje.

1.5. Reikalavimų inžinerija

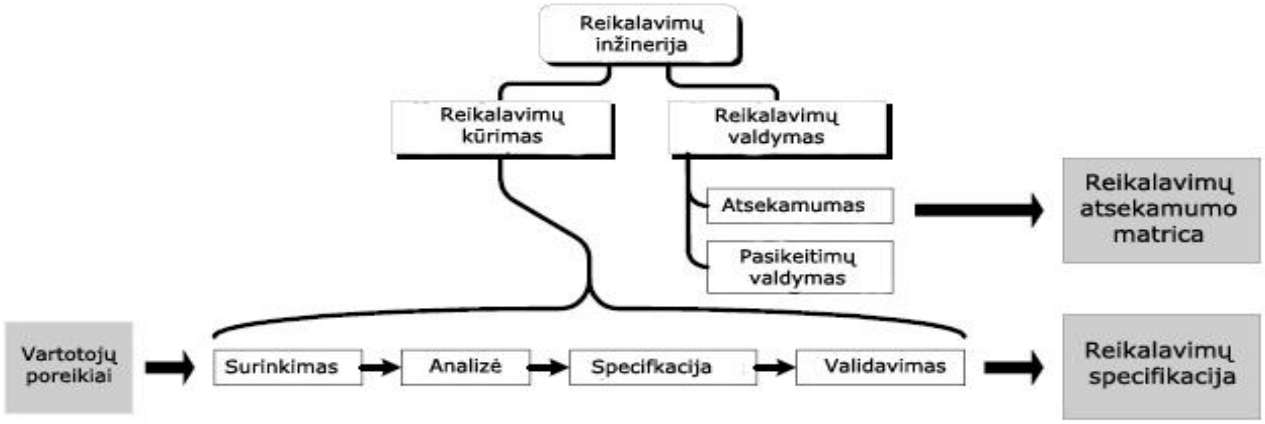
Reikalavimų inžinerija yra vienas svarbiausių žingsnių sistemos kūrimo procese. Reikalavimų išgavimo procesas leidžia išsiaiškinti ir surinkti vartotojo keliamus reikalavimus reikalingus sistemai. Pirmiausia, analizuojami sistemos dalyviai [9].

Yra trys svarbiausios dalyvių kategorijos: kūrėjas, vartotojas ir klientas. Į reikalavimų išgavimo procesas įtraukiami ir kiti dalyviai, pvz.: advokatai, standartai, organizacijos ir t.t.

Reikalavimų inžinerija susiduria su tokiomis problemomis:

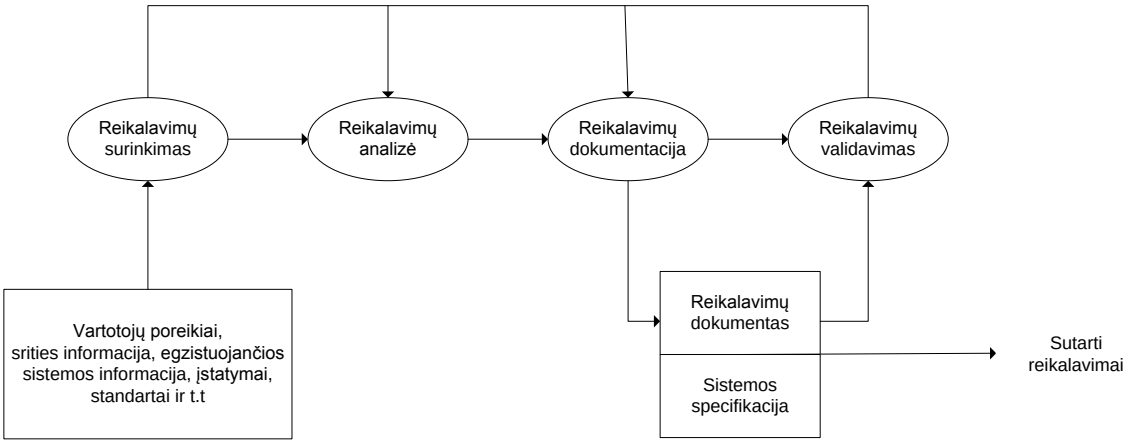
- Klientai nesupranta, kas įmanoma techniškai;
- Reikalavimų pakitimas, atsižvelgiant į sistemos galimybes;
- Klientas ne pilnai išdėsto, ko jis norėtų;
- Vėliau iškilę reikalavimai gali prieštarauti jau įgyvendintiems reikalavimais;
- Prieš projektuojant sistemą, sunku apibrėžti visus reikalavimus ir jų korektiškumą.

Programinės įrangos inžinerija susideda iš daug žinių sričių, apimant reikalavimų inžineriją. Reikalavimų inžinerijos procesas yra pirminis darbas, prasidedantis reikalavimų fazėje ir atliekamas per vėlesnes projekto fazes kai reikalavimai pasikeičia. Tai susideda iš dviejų pagrindinių veiklų tipų. Pirma yra reikalavimų kūrimas, kuris yra procesas generuojantis reikalavimus iš vartotojų įvesties ir analizės. Antras yra reikalavimų valdymas, kuris apima visas kitas reikalavimų veiklas už proceso kūrimo dalių. 2 pav. pavaizduotos pagrindinės reikalavimų inžinerijos veiklos. Procesas rodo vartotojų poreikius naudojamus sugeneruoti reikalavimų specifikaciją ir reikalavimų atsekamumo matricą (Requirements Traceability Matrix, RTM). Keturios reikalavimų kūrimo veiklos formuoja subprocesą, tuo metu, kai reikalavimų valdymo veiklos yra atskiros [10].



2 pav. Reikalavimų inžinerijos procesų apžvalga [10]

Reikalavimų inžinerijos modelis dažnai vaizduojamas linijiniu modeliu. Šiame modelyje, įprastos RI veiklos, tokios kaip reikalavimų surinkimas ir analizė, vyksta paprastais linijiniais perėjimais. Toks linijinis modelis buvo pasirinktas šiam darbui. Kotonya ir Sommerville [11] siūlo linijinį RI proceso modelį, kuris rodo pakartojimus tarp veiklų (3 pav.). Jie nurodo, kad veiklos modelyje sutampa ir dažnai vykdomos pakartotinai.



3 pav. Kotonya ir Sommerville linijinis reikalavimų inžinerijos proceso modelis [11]

1.6. Reikalavimai ir reikalavimų specifikacija

Vartotojo reikalavimų supratimas yra būtina informacinių sistemų projektavimo dalis, bet šių reikalavimų specifikavimas nėra lengva užduotis. Dabar jau suprantama, kad sėkmingos sistemos ir produktai prasideda nuo vartotojų poreikių ir reikalavimų supratimo. Kaip nurodyta IEEE STD 830-1998 [12], į vartotoją orientuotas projektavimas prasideda nuo vartotojo poreikių ir reikalavimų supratimo. Tai gali padėti padidinti produktyvumą, darbo kokybę, pasiektą vartotojo pasitenkinimą, sumažintas palaikymo bei apmokymų kainas. Reikalavimų analizė nėra lengvas procesas. Programinės įrangos inžinieriai ir analitikai dažnai turi spręsti sudėtingas problemas.

Pagrindinės problemos, su kuriomis susiduriama, yra:

- Sudėtingų organizacinių situacijų su daugeliu suinteresuotų šalių sprendimas;
- Vartotojai ir projektuotojai galvoja apie tradicines situacijas liečiančias būtent tą sistemą ir procesus, bet ne apie naujoviškumą;
- Vartotojai dažnai nežino ko jie nori iš būsimos sistemos;
- Spartūs projektavimo ciklai mažina laiką skiriamą vartotojų poreikių analizei;
- Vartotojų reikalavimų pristatymas tinkamoje formoje.

Suprasti iš kur kilo problemos, gali būti labai sunku, ypač jeigu sistema yra nauja, ir todėl sunku įgyvendinti būtent tai, ką sistema turi daryti. Veiksmų ir apribojimų apibūdinimai yra reikalavimai sistemai [1].

Programinės įrangos reikalavimas privalo būti įgyvendintas programinės įrangos arba spręsti tam tikrą problemą. Reikalavimai konkrečiai programinei įrangai paprastai yra sudėtinga reikalavimų kombinacija iš skirtingų žmonių skirtinguose organizacijos lygiuose ir iš aplinkos, kurioje programinė įranga veiks [13].

Galima rasti daug gerų sistemos ir programinės įrangos reikalavimų specifikacijos apibrėžimų, kurie duos gerą pagrindą, kurio pagalba galima apibūdinti gerą specifikaciją ir identifikuoti trūkumus. Taip pat yra daug geros medžiagos apie geros specifikacijos rašymą. Bėda yra ne žinių trūkumas, kaip sukurti teisingai suformuluotą specifikaciją ir net ne kas turi įeiti į specifikaciją. Problema yra ta, kad mes apibrėžiame ne tai, ką turime apibrėžti [14].

Tikslas yra ne sukurti puikią specifikaciją, bet sukurti puikų produktą ir puikią programinę įrangą.

Specifikacija yra svarbiausias dokumentas, susijęs su sistemos kokybe. Specifikacijoje yra pateikiami ir kaupiami visi kliento kokybės požymiai – naudojimo tinkamumas, santykinis lengvumas vartotojui naudojantis programa, galimybė ir sugebėjimas paleisti sistemą nepriklausomai nuo techninės įrangos architektūros, taip pat daugkartinis naudojimas, t. y. galimybė perkelti programinės įrangos komponentus iš vienos programinės įrangos sistemos i kitą, galimybė naudotis produktu sename kompiuteryje, kompanijos, sukūrusios programinę įrangą, palaikymo telefono numeris, produkto kaina, sistemos tvirtumas, kaip dažnai sistema stringa ar paranda informaciją.

Vien tik kliento reikalavimų analizė yra primityvi informacija. Ji neatskleidžia produkto raidos, ji tik patikslina, kur produktas turėtų arba neturėtų būti naudojamas ir kokiu sistemos veikimo principų pageidauja klientas. Specifikacija susijusi su visa šia informacija apie sistemos funkcinius reikalavimus, vartotojo sąsajos reikalavimus ir bet kuriuos nepareikštus, bet privalomus reikalavimus ir tiksliai atskleidžia, koks bus produktas, kokias funkcijas atliks, kaip atrodys.

Specifikacijų formatas keičiasi labai dažnai. Bet esmė išlieka ta pati – viskas yra aprašoma ypač detaliai ir ištestavus yra „užšaldoma“, t. y. gali buti keičiama tik kraštutiniu atveju [15].

Daug skirtingų grupių žmonių turi poreikį suprasti sistemos funkcionalumą. Visu pirma, tai yra sistemos savininkai. Jie privalo suprasti sistemos funkcionalumą tada, kai ji bus sukurta, kad būtų užtikrinti, jog sukurtoji sistema bus tikrai tokia sistema, kurios jiems reikia. Labai svarbūs sistemos projektuotojai, kurie privalo suprasti, kokia sistemą jie ruošiasi kurti. Kita svarbi grupė yra testuotojai, kurie privalo tikrinti, kad tai, ką sistema siūlo, atitiktu tai, dėl ko buvo tariamasi. Kitas grupes, kurios turi suprasti sistemos veikimą, sudaro vadovai tos organizacijos, kurioje sistema bus naudojama, techniniai redaktoriai, kurie kuria naudojimosi vadovus, aptarnavimo personalas, kuris įdiegia ir palaiko sistemą, ir paskutiniai, bet ne mažiau svarbūs, patys vartotojai. Šis sąrašas gali buti didesnis priklausomai nuo sistemos. Svarbu yra užtikrinti, kad visi sistemos tarpininkai gautu tokį bendrą sistemos funkcionalumo aprašą, kuris būtų visiems suprantamas ir pakankamai detalus, kad galima būtų prasmingai ir nedviprasmiškai susitarti.

Pagal IEEE STD 830-1998 standartą [12]**,** kuris yra puikus programinės įrangos reikalavimų specifikavimo šablono dokumento apibrėžimų šaltinis, programinės įrangos specifikacijos dokumento kūrėjas turi detaliai perduoti šias charakteristikas:

1 lentelė Programinės įrangos kūrėjo perduodamos charakteristikos

Charakteristikos	**Savybės**
Funkcionalumas	Ką turi daryti programinė įranga?
Išorinės sąsajos	Kaip programa sąveikauja su vartotoju, sistemos technine įranga, kita programine įranga?
Eksploatacijos parametrai	Koks yra atskirų programinės įrangos funkcijų greitis, naudingumas, reakcijos laikas, atstatymo laikas ir t.t.
Atributai	Kokie yra portatyvumo, teisingumo, palaikomumo, saugos ir kiti sprendimai?
Projektavimo apribojimus, kuriant produktą	Ar yra nustatyti papildomi reikalavimai ir apribojimai atsižvelgiant į realizavimo programavimo kalbą, duomenų bazes, resursus ar aplinkos kintamuosius ir t.t.?

1.6.1. *Reikalavimų specifikacijos dokumento savybės*

Pagal IEEE STD 830-1998 standartą [12] programinės įrangos specifikacijos dokumentas turi būti:

***Korektiškas*.** Tai reškia, kad dokumentas turi būti visą laiką aktualus, su visais paskutiniais pakeitimais;

***Nedviprasmiškas*.** Reikalavimas yra nedviprasmiškas tik tada, kada įmanoma tik viena vienintelė jo interpretacija;

***Pilnas*.** Dokumentas turi turėti viską, kas reikalinga sistemos analitikams. Neturi trūkti nei vieno reikalavimo ar būtinos informacijos. Pilnumas taip pat yra reikalaujama individualių reikalavimų charakteristika. Sistemos reikalavimų specifikacijoje turi būti kuriama reikalavimų hierarchija, kuri padės jos skaitytojams suprasti aprašyto funkcionalumo struktūrą, taigi bus lengviau pamatyti, jei joje kažko trūksta.

Jeigu labiau susitelkiama ties vartotojo užduotimis negu ties sistemos funkcijomis per reikalavimų iškėlimą, bus mažesnė tikimybė, kad bus pražiūrėti reikalavimai arba, kad bus įtraukti reikalavimai, kurie nėra būtini. Panaudos atvejų diagrama puikiai tinka šiam tikslui. Grafinės analizės modeliai, kurie leidžia skirtingus požiūrius į reikalavimus, taip pat gali parodyti reikalavimų neišbaigtumą.

Nuoseklus. Programinės įrangos specifikacijos dokumentas turi būti nuoseklus pats ir nurodomų dokumentų atžvilgiu. Nuoseklūs reikalavimai nekonfliktuoja su kitais programinės įrangos reikalavimais ar aukštesnio lygio reikalavimais. Nesutarimai tarp reikalavimų privalo būti išspręsti prieš kuriant programą. Negalima sužinoti kuris (jeigu nors vienas) yra teisingas, kol nėra atliktas tam tikras tyrimas. Reikia atsargiai keisti reikalavimus, nes nenuoseklumas gali būti nepastebėtas, jeigu bus žiūrima tik į būtent tą pasikeitimą, o ne į bet kuriuos kitus su juo susijusius reikalavimus.

Išdėstytas pagal reikšmingumą. Reikalavimai turi būti išdėstyti pagal jų svarbą, nes dažnai neįmanoma ar nepakanka lėšų įgyvendinti visus reikalavimus. Tai turi būtų irgi nurodyta;

Griežtas ir pagrįstas. Negalimos tokios frazės ir paaiškinimai, kaip „Sistema turėtų dirbti patikimai“ arba „Sistema turi pateikti greitą atsakymą“. Teisingas pavyzdys būtų: „Sistema turi pateikti atsakymą per 10 milisekundžių“;

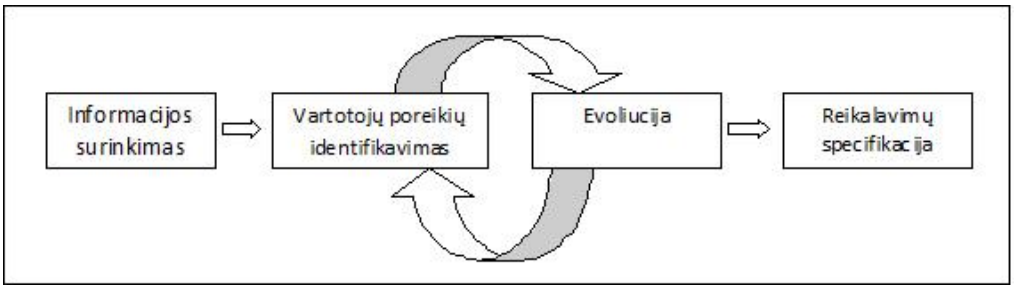
Koreguojamas. Turi būti galimybė peržiūrėti sistemos reikalavimų specifikaciją, greitai ir paprastai papildyti ar koreguoti reikalavimus, ir išsaugoti visus pakeitimus padarytus kiekvienam reikalavimui. Tai reiškia, kad kiekvienas reikalavimas turi būti unikaliai pažymėtas ir išreikštas atskirai nuo kitų reikalavimų, kad juo būtų galima remtis vienareikšmiškai. Sistemos reikalavimų specifikaciją gali būti padaryta labiau modifikuojama, jeigu jos susiję reikalavimai bus grupuojami kartu, ir sukuriant turinį, indeksą ir kryžminį nuorodų sąrašą.

Atsekamas. Dokumento struktūra turi būti grįžta ir kiekviena jo dalis turi būti lengvai atsekama. Turi būti galimybė reikalavimą susieti su jo šaltiniu, kuris gali būti aukštesnio lygio sistemos reikalavimas, panaudojimo atvejis, ar užsakovo įrašas. Taip pat turi būti kiekvieno programinės įrangos reikalavimo nuoroda su dizaino elementais, kodu, testavimo atvejais, kurie suprojektuoti verifikuoti reikalavimams. Atsekami reikalavimai yra unikaliai pažymėti ir parašyti struktūriniu, gerai suskaidytu būdu.

1.7. Reikalavimų analizė

Sistemos analitikas turi koordinuoti reikalavimų surinkimą ir panaudojimo atvejų modeliavimą, nubrėžiant bendrais bruožais sistemos funkcionalumą ir atribojimus sistemai. Pvz.: identifikuoti, kokie aktoriai egzistuoja ir kokių panaudojimo atvejų jiems reikia bendraujant su sistema. Sistemos analitikas taip pat atsakingas už bendro terminų žodyno apibrėžimą, kuris gali būti panaudotas visame tekstiniame sistemos apibūdinime. Kad būtų įgyvendinti šie tikslai, sistemos analitikas turi įvykdyti komplektą veiksmų, nutaikytų į sistemos konteksto nustatymą. Tikslas aiškiai identifikuoja sistemos sienas: nustatyti, kas turi būti sistemos viduje ir kas turi būti

už jos ribų. Sienos yra apribotos identifikavimo aktorių ir panaudojimo atvejų. Skirtingų vartotojų reikalavimų metodų pagrindas yra paprastas procesas parodytas 4 pav. kuris apima 4 žingsnius:



4 pav. Pagrindinis vartotojų reikalavimų analizės procesas[16]

Pirmas žingsnis. Vartotojų reikalavimų analizėje yra surinkti pagrindinę informaciją apie vartotojus, suinteresuotas šalis, procesus, kurie vyks [16].

Suinteresuotų šalių analizė identifikuoja visus vartotojus ir suinteresuotas šalis, kurios turės įtakos ar bus paveiktos sistemos. Tai padeda užtikrinti, kad visų jų poreikiai bus užfiksuoti.

Vartotojų grupės gali apimti galutinius vartotojus, prižiūrėtojus, diegėjus. Kitos suinteresuotos šalys apima sistemos išvesties gavėjus, marketingo personalą, pirkėjus ir pagalbinį personalą. Suinteresuotų šalių analizė kiekvienai vartotojų ir suinteresuotų šalių grupei identifikuoja jų pagrindinius vaidmenis, atsakomybes ir užsibrėžtus tikslus susijusius su sistema. Vienas iš pagrindinių klausimų yra kaip kartu susieti skirtingų suinteresuotų šalių konkuruojančius poreikius naujoje sistemoje [16].

Antras žingsnis. Rinkos analizė, kuri apima analizę spausdintų šaltinių, tokių kaip tyrimų pranešimai, surašymo duomenys, demografinė informacija, kuri parodytų galimų rinkos vartotojų kategorijas.

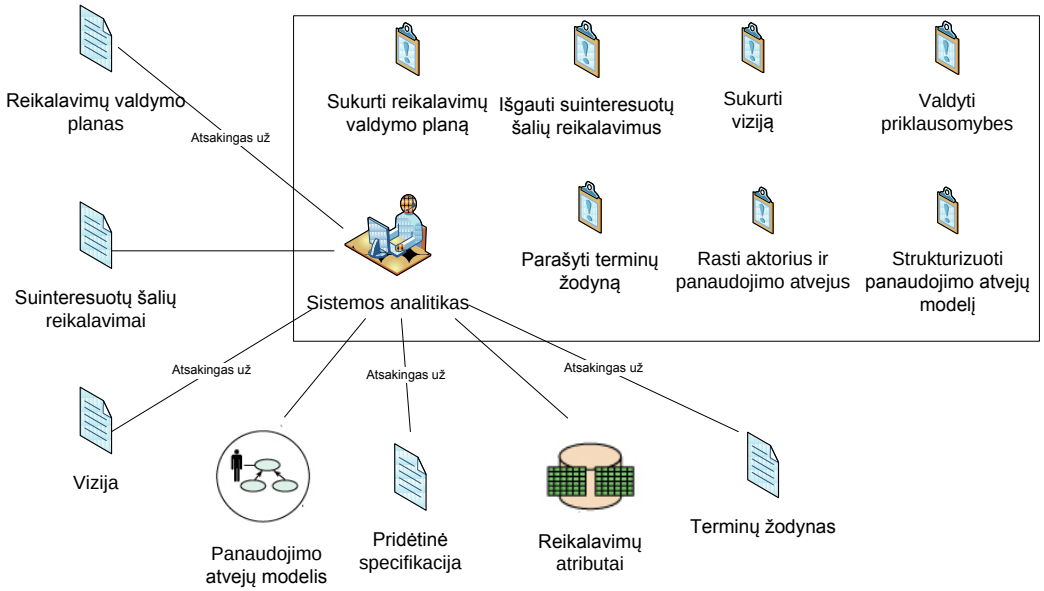
Vartotojų analizės kontekstas yra naudojamas, kai sistema ar produktas yra sukurtas. Sistemos kokybė, įskaitant praktiškumą, pasiekiamumą ir socialinius pasiekiamumo faktorius, priklauso nuo labai gero supratimo apie sistemos naudojimo kontekstą. Pavyzdžiui, bankomatą bus patogiau naudoti, jei jis bus suprojektuotas naudojimui naktį taip pat gerai, kaip ir dieną, ryškioje šviesoje taip pat kaip ir normalioje šviesoje, bus tinkamas naudoti žmonėms invalido vežimėlyje kaip ir tiems, kurie gali stovėti. Panašiai ir ofiso aplinkoje yra daug charakteristikų, kurios gali paveikti naujo programinės įrangos produkto praktiškumą, pvz.: vartotojo darbo krūvis, pasiekiama pagalba ar pertrūkiai. Kontekstinės informacijos fiksavimas yra svarbi pagalba specifikuojant vartotojų reikalavimus. Norint surinkti kontekstinę informaciją, suinteresuotos šalys lanko

susirinkimus, vadinamus konteksto susirinkimais. 2 lentelėje pateikiama anketa su svarbiausiais vartotojų, jų užduočių ir aplinkos komponentais [16].

2 lentelė Vartotojų faktorių kontekstas

Vartotojų grupė	• Įgūdžiai ir patirtis su sistema • Užduočių žinojimas • Mokymai • Kvalifikacija • Kalbos žinios • Amžius ir lytis • Fiziniai gebėjimai • Požiūris ir motyvacija
Užduotys	• Užduočių sąrašas • Tikslas • Išvestis • Žingsniai • Svarba • Dažnumas • Trukmė • Priklausomybės
Techninė aplinka	• Techninė įranga • Programinė įranga • Tinklas • Kita įranga
Fizinė aplinka	• Girdimoji aplinka • Šiluminė aplinka • Vizualioji aplinka • Vibracija • Erdvė ir baldai • Pavojai sveikatai • Apsaginiai rūbai ir įrankiai
Organizacijos aplinka	• Darbinė praktika • Pagalba • Pertraukimas • Valdymas ir bendravimo struktūra • Kompiuterių naudojimo politika • Organizaciniai tikslai • Pramoniniai ryšiai • Darbo charakteristikos
Vartotojų grupė	• Įgūdžiai ir patirtis su sistema • Užduočių žinojimas • Mokymai • Kvalifikacija • Kalbos žinios • Amžius ir lytis • Fiziniai gebėjimai • Požiūris ir motyvacija
Užduotys	• Užduočių sąrašas • Tikslas

	• Išvestis • Žingsniai • Svarba • Dažnumas • Trukmė • Priklausomybės
Techninė aplinka	• Techninė įranga • Programinė įranga • Tinklas • Kita įranga
Fizinė aplinka	• Girdimoji aplinka • Šiluminė aplinka • Vizualioji aplinka • Vibracija • Erdvė ir baldai • Pavojai sveikatai • Apsaginiai rūbai ir įrankiai
Organizacijos aplinka	• Darbinė praktika • Pagalba • Pertraukimas • Valdymas ir bendravimo struktūra • Kompiuterių naudojimo politika • Organizaciniai tikslai • Pramoniniai ryšiai • Darbo charakteristikos



5 pav. Sistemos analitiko per reikalavimų kūrimo fazę gaunamos veiklos ir produktai

1.8. Reikalavimų surinkimo problemos

Programinės įrangos kūrime, sukurta sistema gali neatitikti sistemos, kurios reikalauja klientas, daugeliu atvejų. Viena iš priežasčių yra sunkumai reikalavimų surinkimo darbe. Sunkumai reikalavimų surinkimo darbe gali būti padalinti į dvi pagrindines kategorijas: sunkumai priskiriami klientams ir kūrėjams [17]:

Sunkumai priskiriami klientams. Pagrindinė problema priskiriama klientams kyla dėl to, kad klientai dažnai nesupranta apie programų kūrimo technologijas. Dėl šios priežasties, klientai dažnai nežino kokia informacija turi būti duodama programinės įrangos kūrėjams. Šito rezultatas – klientų duodami reikalavimai programinės įrangos kūrėjams yra tik dalis būtinų sąlygų sistemos realizavimui. Be to, reikalavimų išreiškimas nėra konkretus, o abstraktus. Pavyzdžiui, reikalavimai sistemai, pateikti kliento, būtų tokie: “aš noriu sukurti inventoriaus kontrolės programinę įrangą; aš noriu, kad procesai vyktų taip lengvai ir skubiai kaip įmanoma; ir aš noriu sumažinti darbo išlaidas per pusę, mažindamas darbuotojus tiek kiek įmanoma.” Visi šitie prašymai yra dviprasmiški, ir reikalavimų lygmuo yra nenuoseklus [17].

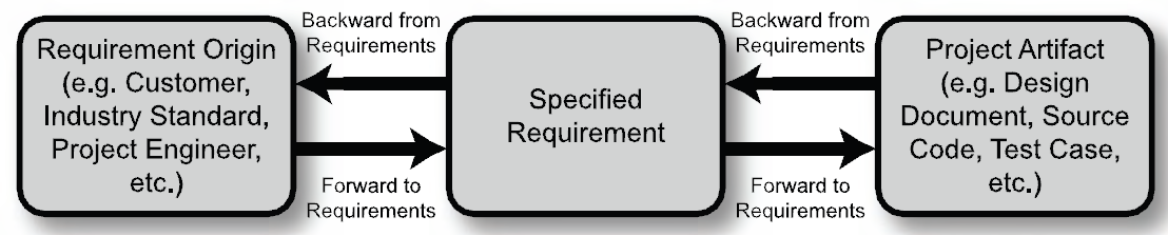
Sunkumai, priskiriami programinės įrangos kūrėjams. Pagrindinis sunkumas priskiriamas kūrėjams yra tas, kad naujokai kūrėjai neturi žinių verslo turiniui, su kuriuo klientai yra pažįstami. Dėl šios priežasties, sunku programinės įrangos kūrimo naujokui apsispręsti, kokios ir kiek funkcijų bus reikalingos programinei įrangai palaikyti kliento verslo sistemos kūrimui. Be to, jei klaidos yra atsitiktinai įtrauktos į reikalavimus, naujokas kūrėjas gali nesugebėti išspręsti klaidų. Šito rezultatas yra tas, kad naujokai kūrėjai nežino nuo ko pradėti ir kaip toli jie turi paprašyti reikalingų funkcijų. Naujokas kūrėjas nežino ar informacija iš surinktų reikalavimų yra teisinga ar klaidinga. Todėl, praleidimai ar klaidos pasirodo surinktuose reikalavimuose. Tačiau, tiktai nedaugelis patyrusių kūrėjų, net didelėse organizacijose, sugeba surinkti vartotojų reikalavimus programinei įrangai, kuri bus sukurta be praleidimų ar klaidų [17].

1.9. Atsekamumas

Visi artefaktai programinės įrangos sistemoje yra susiję. Kiekvienas artefakto tipas yra rezultatas kito ankstesnio žemesnio lygio artefakto. Šios nuorodos tarp artefaktų yra vadinamos atsekamumo nuorodomis arba pėdsakais: atsekamumo nuorodos apibūdina ryšius, kurie egzistuoja tarp įvairių programinės įrangos inžinerijos procesų artefaktų [18].

ISO 9001:2000 ir IEEE JSTTD-016 programinės įrangos kūrimo standartai siūlo naudoti reikalavimų atsekamumo praktiką programinės įrangos kūrimo procese. Reikalavimų

atsekamumas gali būti apibūdintas, kaip “galimybė apibrėžti ir sekti reikalavimo gyvavimą, priekine ir atgaline kryptimi“ [19]. Ši koncepcija pavaizduota 6 pav. [20].

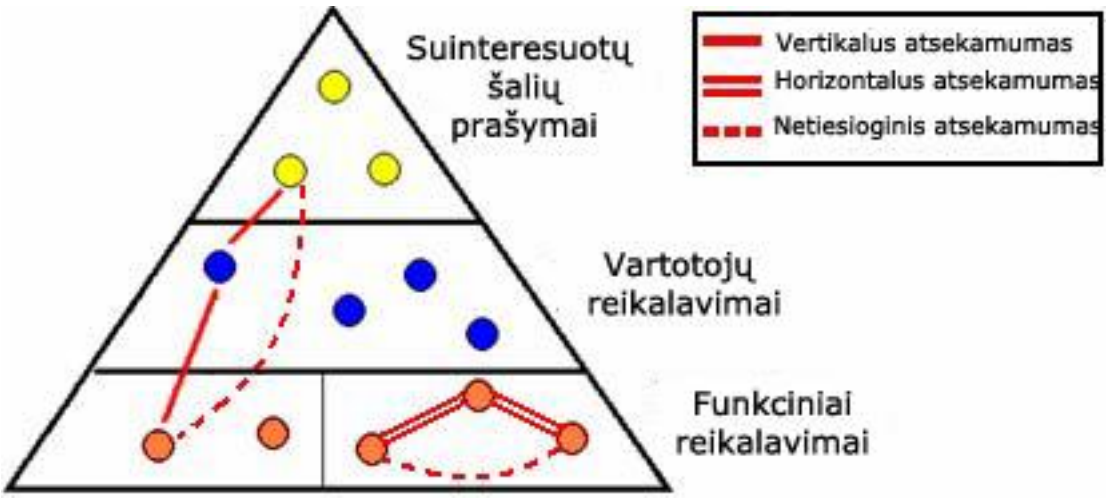


6 pav. Programinės įrangos reikalavimų atsekamumo vaizdas [20]

Vertikalus atsekamumas. Vertikalus atsekamumas identifikuoja dalykų kilmę (pavyzdžiui, kliento poreikių) ir seka šiuos tuos pačius dalykus kaip jie keliauja per hierarchiją pas projekto komandą ir galiausiai pas klientą [21]. Tai užtikrina, kad reikalavimas gali būti sekamas nuo jo pradžios per bet kurį ir visus pakeitimus, kol jis neįgauna savo galutinės formos. Vertikalių santykių pavyzdžiai: tarp reikalavimų lygių, tarp reikalavimų ir projekto planų, tarp planų ir darbo produktų, tarp darbo produktų ir produkto komponentų, ir t.t.

Horizontalus atsekamumas. Horizontalus atsekamumas identifikuoja santykius tarp susijusių dalykų tarp darbo grupių ar produkto komponentų su tikslu išvengti potencialių konfliktų [21]. Tai leidžia projektui numatyti, ir sušvelninti ar išspręsti galimas problemas prieš testavimą. Horizontalių santykių pavyzdžiai: tarp reikalavimų, per sąsajų, tarp produkto komponentų, nuo funkcijos iki funkcijos, ir t. t.

Netiesioginis atsekamumas. Atsekamumas turi kryptį. Pirmyn nukreiptas atsekamumas yra nukreiptas atsekamumas, kurio kilmė yra reikalavimas. Atgal nukreiptas atsekamumas yra tas, iš kurio tas maršrutas yra reikalavimas. Netiesioginis santykis gali atsirasti tarp dviejų loginių esybių per tranzityvumą. Pavyzdžiui, jei pasikeitimas esybėje A turi poveikį esybei B, ir pasikeitimas esybėje B turi poveikį esybei C; tada pakeitimas A turi poveikį C, netiesiogiai. Todėl, toks netiesioginis santykis turi būti susektas per reikalavimų valdymo procesą. 7 pav. rodo atsekamumo pavyzdžius [10].



7 pav. Vertikalus, horizontalus ir netiesioginis reikalavimų atsekamumas [10]

Reikalavimų atsekamumas gali padėti analizuojant poveikį ir nesuderinamumą tarp gautų reikalavimų ar produktų dėl reikalavimų pasikeitimų.

1.9.1. *Reikalavimų atsekamumo nauda*

Jeigu norime parodyti, kuris reikalavimas padengtas testavimo atvejo, kiekvieną reikalavimą reikia susieti mažiausiai su vienu testavimo atveju. Atsekamumo naudojimas užtikrina, kad reikalavimas bus susietas su kitais darbo produktais, tokiais kaip testavimo atvejai.

Atlikti tyrimai parodė, kad naudojamas netinkamas reikalavimų atsekamumas yra svarbus programinės įrangos projektų nesėkmių ir viršyto biudžeto faktorius [22]. Kaip atsakas į šį tyrimą, buvo atlikta daugiau tyrimų susijusių su reikalavimų atsekamumu ir išleista knygų šia tema, taip pat daugelis organizacijų stengėsi pagerinti savo atsekamumo praktiką. Šios pastangos nenuėjo veltui. 2006 metais *The Standish Group* atnaujino savo 1994 metų tyrimą [23], parodydama, kad tik 19% programinės įrangos kūrimo projektų buvo nesėkmingi ir 46% viršijo biudžetą. Aiškus pagerėjimas lyginant su 1994 metų tyrimu parodytas 3 lentelė The Standish Group atliktų tyrimų 1994 ir 2006 palyginimas.

3 lentelė The Standish Group atliktų tyrimų 1994 ir 2006 palyginimas

Metai	Nesėkmingi projektai	Projektai iššūkiai	Sėkmingi projektai
1994	53%	31%	16%
2006	19%	46%	35%

Nors atsekamumo svarba yra gerai suprantama programinės įrangos kūrime, tyrimai parodė, kad daug organizacijų vis dar nesupranta atsekamumo principų ir joms kyla problemų įgyvendinant atsekamumo praktiką programinės įrangos kūrimo gyvavimo cikle [24].

Reikalavimų atsekamumas gali duoti daug naudos organizacijai, jei tik naudojamos teisingos atsekamumo technikos. Dėl šios priežasties, reikalavimų atsekamumas yra svarbus komponentas daugelio programinės įrangos kūrimo standartų tokių, kaip CMMI ir ISO 9001:2000. Svarbią atsekamumo naudą galima pastebėti šiose srityse: projekto valdymo, proceso matomumo, verifikavimo ir validavimo [25].

Reikalavimų atsekamumo teikiama nauda kiekvienai sričiai:

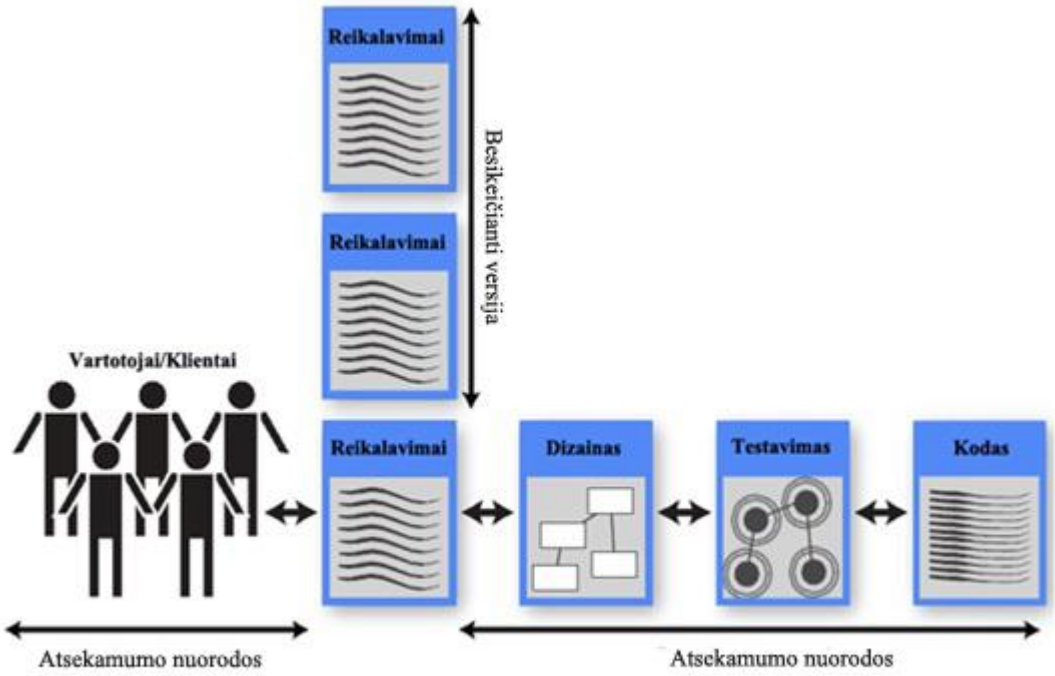
Projekto valdymas. Atsekamumas daro projekto valdymą lengvesnį. Sekant atsekamumo nuorodas , projekto vadovas gali greitai pamatyti, kaip daugelis artefaktų bus paveikti pasiūlyto pakeitimo ir gali priimti sprendimą dėl kainos ir rizikos, susijusios su tuo pakeitimu. Projekto vadovai taip pat gali pasitelkti atsekamumą į pagalbą, matuojant projekto progresą. Kadangi reikalavimai susiejami su testavimo atvejais, galima numatyti projekto statusą pagal tai, kiek artefaktų sukuriama vėliau kūrimo cikle. Ši informacija gali būti naudojama numatyti projekto trukmei ir įvertinti rizikai [20].

Projekto matomumas. Atsekamumas siūlo įgyvendintą projekto matomumą tiek projekto inžinieriams, tiek klientams. Per atsekamumą, kiekvienas projekto inžinierius gali prieiti prie kontekstinės informacijos, kuri gali padėti jiems nustatyti iš kur atsirado reikalavimas, jo svarbą, kaip jis buvo įgyvendintas ir kaip jis buvo testuotas. Atsekamumas taip pat gali būti naudojamas kliento padidina kliento pasitenkinimą, nes tai užtikrina klientą, jog jis gaus produktą, kurio prašė [20].

Verifikavima ir validavimas. Pati didžiausia atsekamumo nauda gali būti suprasta per verifikavimo ir validavimo stadiją. Atsekamumas siūlo galimybę susieti sistemos funkcionalumą atskirai su kiekvienu reikalavimu, nuo kilmės iki kiekvieno reikalavimo testavimo. Teisingai įgyvendintas, atsekamumas gali būti naudojamas įrodyti, kad sistema atitinka savo reikalavimus ir jie buvo įgyvendinti teisingai. Jeigu reikalavimas gali būti atsektas į priekį iki dizaino artefaktų, jis validuoja, kad reikalavimas buvo įgyvendintas sistemoje. Jeigu reikalavimas gali būti atsektas iki testavimo atvejo tai rodo, kad reikalavimo įgyvendinimas buvo verifikuotas per testavimą. Be atsekamumo, yra neįmanoma pademonstruoti, kad sistema buvo pilnai verifikuota ir validuota [20].

Sistemos valdymas. Surinkti reikalavimai dažnai keičiasi net po projekto pabaigimo, ir yra svarbu turėti galimybę nustatyti šių pasikeitimų įtaką. Atsekamumas lengvai leidžia nustatyti, koks reikalavimas, dizainas, kodas, testavimo atvejai turi būti atnaujinti, kad patenkintų pasikeitimo poreikį. Tai leidžia įvertinti laiką ir kainą reikalingą šiam pasikeitimui.

Atsekamumas yra apibrėžtas kaip pastebima asociacija tarp dviejų ar daugiau loginių esybių tokių, kaip reikalavimai, sistemos elementai, verifikacija, ar užduotys [26]. Atsekamumas gali apimti tokius santykių tipus: vertikalius, horizontalius, netiesioginius.



8 pav. Alternatyvus reikalavimų atsekamumo vaizdas [20]

1.9.2. *Reikalavimų atsekamumo iššūkiai*

Nepaisant privalumų, kuriuos siūlo atsekamumas, programinės įrangos kūrimo industrija susiduria su daugybe iššūkių. Šie iššūkiai gali būti identifikuoti kainos, laiko, pastangų, sudėtingumo valdant atsekamumą per pasikeitimus terminais, taip pat skirtingas daugelio suinteresuotų šalių požiūris į atsekamumą, organizacinės problemos, prastas įrankių palaikymas.

Kaina. Vienas pagrindinių iššūkių, su kuriuo susiduriama atsekamumo įgyvendinime, yra kaina. Sistema auga dydžiu ir sudėtingumu, todėl reikalavimų atsekamumas greitai tampa sudėtingas ir brangus [27]. Todėl atsekamumas reikalauja didesnio biudžeto. Kaip bebūtų, projekte įgyvendintas atsekamumas padeda išvengti nenumatyto biudžeto viršijimo, nes atsekamumas gali nustatyti projekto problemas anksti kūrimo procese, kai ta problemas taisyti yra pigiau ir lengviau.

Būdas išvengti aukštos atsekamumo kainos yra verte pagrįsto reikalavimų atsekamumo praktika vietoj pilno atsekamumo. Verte pagrįstas atsekamumas prioritetizuoja visus sistemos reikalavimus, su laiko kiekiu ir pastangomis skiriamomis sekant kiekvieną reikalavimą priklausomai nuo to reikalavimo svarbos [27]. Tai gali sutaupyti didelį kiekį pastangų, susifokusuojant atsekamumo veiklas į svarbiausius reikalavimus.

Pasikeitimų valdymas. Atsekamumo valdymas per pasikeitimus sistemai yra kitas svarbus iššūkis. Studijos parodė, kad pasikeitimų gyvavimo cikle galima tikėtis beveik kiekviename programinės įrangos kūrimo projekte [28]. Kai tokie pasikeitimai atsiranda, svarbu atnaujinti atsekamumo duomenis, kurie yra veikiami šio pasikeitimo. Tai reikalauja disciplinos tų, kurie daro pasikeitimus atsekamumo duomenų atnaujinime, o tai gali kainuoti daug laiko ir pastangų, kai pasikeitimai yra dideli.

1.9.3. *Reikalavimų atsekamumas su testais*

Atsekdami reikalavimus iki testavimo atvejų ir atgal, turime būda validuoti, kad reikalavimai yra įgyvendinti ir ištestuoti. Tai taip pat padės testuotojams pamatyti, ką jie turi pakeisti po to, kai buvo pakeisti ar pašalinti reikalavimai ir kuriuos reikalavimus jiems reikia testuoti. Tai taip pat naudinga, kaip sistemos validavimas, kad sistema yra įgyvendinta teisingai.

1.10. Testavimas

Testavimas – tai procesas, kurio metu siekiama rasti programinės įrangos klaidas ir įrodyti, kad programinė įranga tenkina reikalaujamą kokybę. Testavimo metu siekiama rasti klaidas testuojamame objekte. Tai veikla vykdoma produkto kokybės įvertinimui, ir siekianti jį pagerinti, identifikuojant defektus ir problemas. Testavimo objektu gali būti – modulis, sistema, dokumentacija. Testavimas turi dvi dalis – verifikavimą ir validavimą [29].

Programinės įrangos testavimas yra praktika dažnai naudojama parodyti programinės įrangos kokybei. Testavimas yra daug laiko užimantis ir brangus procesas, kuris dažnai kainuoja nuo 40 iki 80% galutinės programinės įrangos produkto kainos. Nesisteminguose kūrimo procesuose, testavimas tęsiasi tol, kol baigiasi jam skirtas laikas. Kai kūrėjai sistemiškesni, organizacija stengiasi išmatuoti testavimo išbaigtumą ir gerumą, sukurdami testų užbaigimo kriterijus [29].

Testavimu nesiekiama parodyti, kad programinė įranga veikia teisingai. Priešingai, testuotojas turi pradėti testavimą laikantis prielaidos, kad programa turi klaidų ir stengtis surasti jų kiek galima daugiau.

Tikimybė, kad programoje egzistuoja daugiau klaidų yra proporcinga jau surastų klaidų skaičiui, t.y. modulyje, kuriame buvo surasta sąlyginai daugiau klaidų, galima tikėtis rasti daugiau klaidų nei tame, kuriame rasta mažiau klaidų. Surastas didesnis klaidų kiekis rodo, kad programuotojas neteisingai ar ne išsamiai išsiaiškino reikalavimus, arba neteisingai panaudojo

programavimo galimybes ir pan. Be to, kiekvienos klaidos taisymas gali sąlygoti papildomų klaidų atsiradimą, t.y. kuo daugiau klaidų taisoma, tuo didesnė tikimybė surasti naujas klaidas.

Programinės įrangos testavimas yra komplektas veiklų, kurios apima planavimą ir pasiruošimą kaip testuoti ir ką testuoti programinėje įrangoje, ir vykdyti sistemą su ketinimu rasti defektus ir validuoti programinę įrangą prieš reikalavimus. Dideliuose projektuose, rankiniu būdu atliekamas testavimas yra daug pastangų ir laiko užimanti veikla. Dėl to, testo automatizavimas yra procesas, sumažinantis testavimo kainą, sukurdamas testo kodą, kuris gali būti vykdomas automatiškai tiek kartų, kiek reikia [30].

Sistemos programinės įrangos testavimą atlieka ne jos kūrėjas. Programuotojai neturėtų testuoti savo programų. Programuotojui yra sunku pakeisti mastymą, kad jo programoje gali būti klaidų. Taip pat programa gali turėti klaidų dėl programuotojo blogai suprastų reikalavimų. Bandant testuoti, jis bus tokios pačios nuomonės apie reikalavimus. Be to, kritikuoti savo paties darbą yra labai sunku. Todėl programuotojai savo sukurtų programų negali efektyviai testuoti.

Testai turi būti rengiami taip, kad juos galima būtų kartoti. Testų pakartojimas ypač reikalingas programuotojams, kad galėtų pakartoti (suprojektuoti) testavimo atvejį, kuriame buvo rasta klaida. Taip pat testų pakartojimo galimybė yra viena iš sąlygų vykdyti regresinį testavimą, kai pakartotinai testuojamas išleistas naujas programos leidimas.

Testavimas turi būti planuojamas. Kiekvienam testavimo etapui turi būti sudaromas testavimo planas tam, kad pakankamai laiko ir resursų būtų skiriama testavimo užduočių vykdymui, kad testavimo procesas būtų tinkamai kontroliuojamas ir valdomas. Testavimo planavimas turi būti derinamas su projekto valdymo planu. Testavimo grupės vadovas ir projekto vadovas turi bendradarbiauti, kad suderinti darbą. Sudarant testavimo planą turi būti įvertintos testavimo rizikos, t.y. kur gali būti uždelsimas testuojant. Būtina įvertinti, kokio sudėtingumo funkcijas-operacijas atlieka programos, kuris komponentas reikalauja sudėtingesnio testavimo arba kad testuojant komponentą testuotojas turės papildomai išmokti naudotis nauju įrankiu (programine priemone).

Testavimo plano negalima sudarinėti remiantis prielaida, kad klaidų nebus rasta. Testavimas yra procesas, kurio tikslas yra surasti klaidas. Negalima sudarinėti testavimo plano, atsižvelgiant vien tik į tai, kokio sudėtingumo funkcijas-operacijas atlieka programos. Papildomai turi būti skiriamas dėmesys tokiems aspektams – ar tai visiškai nauja programa, jeigu tai nėra nauja – kiek klaidų prieš tai vykusio testavimo metu buvo rasta, koks testuotojas jį atliko ir atliks ir t.t. Be to, daugiau laimi tie, kurie pasilieka galimybę koreguoti testavimo planą testavimo eigoje.

Testavimas - daug kūrybiškumo reikalaujantis darbas. Testavimui siūlomi įvairūs metodai. Tačiau nei vienas jų negarantuoja, kad programa bus ištestuota pilnai – t.y. joje nebeliks klaidų. Kiek klaidų bus rasta ir kiek jų liks nepastebėta, priklauso vien tik nuo testuotojo kūrybiškumo:

kokius metodus jis pasirinks, kaip juos suderins tarpusavyje, kokius testavimo atvejus parašys ir pan.

4 lentelė Testavimo principai [31]:

Principas	**Savybė**
Testai turi būti atsekami su kliento reikalavimais	Sistemos testavimo tikslas yra atrasti defektus. Didžiausi defektai yra tie, dėl kurių programa neatitinka savo reikalavimų.
Testai turi būti suplanuoti dar prieš prasidedant testavimui	Testo planavimas gali prasidėti iš karto, kai užbaigiamas reikalavimų modelis. Detalus testavimo atvejų aprašymas gali būti pradedamas, kai sukuriamas projekto modelis, o visi testai gali būti suplanuoti ir sukurti dar prieš sugeneruojant kodą.
Išsamus testavimas neįmanomas	Skaičius įvykio derinių net mažai sistemai yra ypatingai didelis. Dėl šios priežasties, yra neįmanoma įvykdyti kiekvieną kombinaciją per testavimą.
Kad būtų efektyviausias, testavimas turi būti atliekamas nepriklausomos trečios šalies	„Efektyviausias“ reiškia, kad turi didžiausią tikimybę rasti defektus. Dėl šios priežasties, programinės įrangos inžinieriai, kurie sukūrė sistemą, nėra tinkamiausi asmenys kurti testams.
Geras testavimo atvejis yra tas, kuris turi didelę galimybę rasti dar nerastas klaidas	
Sėkmingas testas yra tas, kuris atskleidžia dar neatskleistas klaidas.	
Testavimas negali parodyti defektų nebuvimo, jis tik gali parodyti, kad programinės įrangos esamas klaidas	

1.10.1. *Grįžtamasis testavimas*

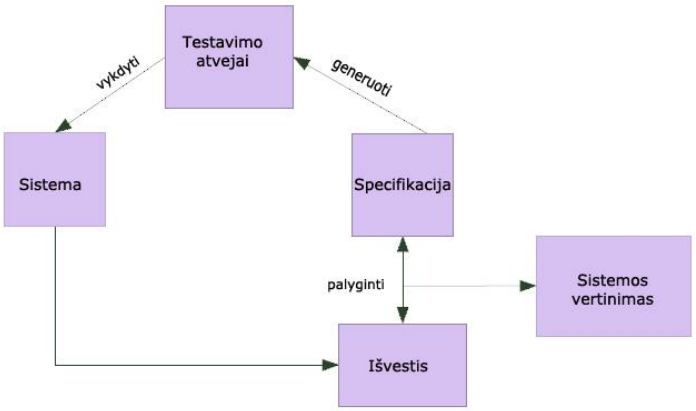
Grįžtamasis testavimas (angl. Regression testing) yra procesas, kurio metu validuojama modifikuota programinė įranga, kad būtų užtikrinta, jog pakeistos programinės įrangos dalys elgiasi taip, kaip numatyta, ir nepakeistos programinės įrangos dalys nebuvo neigiamai paveiktos modifikacijos [32]. Studijos parodė, kad grįžtamasis testavimas kainuoja 80% visos testavimo kainos [33].

Grįžtamasis testavimas yra bet kokio tipo programinės įrangos testavimas, kuris ieško programinės įrangos defektų pakartotinai testuodamas modifikuotą programą.

Grįžtamasis testavimas yra svarbi ir būtina veikla, kuri gali išlaikyti modifikuotos programinės įrangos sistemos kokybę [34]. Yra daug pasiūlytų grįžtamojo testavimo technikų. Pavyzdžiui, grįžtamojo testavimo pasirinkimo technikos [35], sumažina testavimo kainą parinkdamos testavimo atvejus, kurie yra būtini testuoti modifikuotai programai. Testavimo atvejų prioritetų technikos [36] pertvarko testavimo atvejus, suplanuodamos testavimo atvejus su didžiausiu prioritetu pagal kažkokius kriterijus nustatytus anksčiau testavimo procese, kad duotų naudą tokią, kaip aprūpinimas ankstyvesniais atsiliepimais testuotojams ir ankstyvesnis klaidos radimas.

1.11. Testavimo atvejai

Programinės įrangos testavimas yra vis dar plačiai naudojama technika programinės įrangos kokybės užtikrinimui. Testavimo procedūra pavaizduota 1.6 pav.. Testavimo Atvejis yra įvesties duomenys paduodami į testuojamą sistemą. Kai testavimo atvejis parenkamas, tada sistema gali būti vykdoma su tuo testavimo atveju. Tada testuotojas nusprendžia ar vykdymo rezultatas yra toks, kokio tikėtasi ar ne (t.y. lygina rezultatą su sistemos specifikacija). Kai įvykdomas visas testavimo atvejų rinkinys, identifikuojamos klaidos (kai sistema neatitinka savo specifikacijos). Tačiau, kai nerandama klaidų, negalima teigti, kad sistema atitinka savo specifikaciją. Todėl, kad sistemos turi nepaprastai daug potencialių testavimo atvejų, o testuotojas gali vykdyti tik ribotą jų skaičių. Testavimo atvejai parenkami prieš vykdant testavimo atvejus ir klaidos gali būti identifikuojamos po to, kai testavimo atvejai įvykdyti. Tai tampa dideliu iššūkiu programinės įrangos testavime. Kaip turi būti testavimo atvejai parinkti? [37]



9 pav. Testavimo procedūra

Testavimo atvejai paprastai generuojami priklausomai nuo duotų testo duomenų pakankamumo kriterijų, kurie sieja pakankamumo laipsnį su testų komplektu (t.y. sugeneruotų testavimo atvejų rinkiniu), kad identifikuoti kaip testo rinkinys padengia sistemos specifikaciją.

Testavimo atvejis yra sudarytas iš tam tikrų įvesties duomenų, veiksmų ir numatomų rezultatų. Testavimo atveju siekiama patikrinti sistemos atitikimą specifikacijoje keliamiems reikalavimams. Remiantis testavimo apibrėžimu, sėkmingas testavimo atvejis yra tas, kuris surado klaidas. Taigi, geras testavimo atvejis yra tas, kuris turi didelę tikimybę surasti dar nerastas klaidas. Kiekvienam testavimo atvejui nurodomi tikslūs numatomi rezultatai. Jeigu testavimo atvejis neturi jų, yra tikimybė, kad klaidingas rezultatas bus interpretuotas kaip teisingas. Todėl būtina tiksliai išsiaiškinti visus galimus programos išvesties duomenų rinkinius. Kiekvieną testavimo atvejį turi sudaryti dvi dalys – įvesties duomenų aprašymas ir tuos duomenis atitinkančių teisingų numatomų rezultatų nurodymas. Lyginant numatomą ir gautą rezultatus lengviau identifikuojamos klaidos.

Testavimo atvejai turi būti aprašomi tiek galimiems įvesties duomenims, tiek ir negalimiems įvesties duomenims. Pastebima, kad testuojant susikoncentruojama tik ties galimais ir numatomais įvesties duomenimis, t.y. stengiamasi patikrinti, ar programa teisingai interpretuoja galimus duomenis, ar gaunami rezultatai atitinka reikalavimus. Tačiau, nemažai klaidų pastebima, kai su programomis dirbama skirtingomis nei buvo tikėtasi sąlygomis. Todėl papildomai turi būti rašomi testavimo atvejai negalimiems įvesties duomenims – sąlygoms, kurios turi didesnę klaidų aptikimo tikimybę tam, kad nustatyti kaip programa veiks su nenumatytais įvedimo duomenimis ar nenumatytomis sąlygomis.

Testavimo atvejus specifikuojantis šablonas skiriasi kiekvienoje organizacijoje. Sunku sujungti kiekvieną elementą daugybės testavimo atvejų šablonų naudojamų skirtingose organizacijose. Todėl testavimo atvejų dokumentavimui naudosime šabloną, kuris grindžiamas IEEE 829-1998 testų specifikavimo standartu [38].

Vienas testavimo dažniausiai negali patenkinti visų reikalavimų, todėl paprastai reikalaujama naudoti testavimo atvejų rinkinį, kad būtų patenkinta kuo įmanoma daugiau reikalavimų. Paprastai kuo daugiau testavimo atvejų naudojama, tuo daugiau galimų reikalavimų yra patenkinama. Praktiškai, testo rinkinys paprastai atlieka išplėtimo procesą, nes nauji testavimo atvejai įtraukiami į testo rinkinį, kad užtikrinti, jog reikalavimai bus patenkinti. Šito rezultatas yra tas, kad testo rinkinys gali turėti daugiau negu pakankamai testavimo atvejų reikalavimų patenkinimui. Galima sakyti, kad kai kurie testavimo atvejai pašalinami iš testo rinkinio, testo rinkinys vis dar gali patenkinti visus reikalavimus, kurie gali būti patenkinti originaliu testo rinkiniu. Rasti egzistuojančio testo rinkinio subrinkinį, kuris gali patenkinti tuos pačius reikalavimus, kaip ir originalus testo rinkinys, yra problema. Ši problema paprastai vadinama testo rinkinio sumažinimu (angl. Test suite reduction), o testavimo atvejų subrinkinys paprastai vadinamas atstovaujančiu komplektu (angl. Representative set). Testo rinkinio sumažinimas gali sumažinti laiką skiriamą testo rinkinio vykdymui, o tai sumažina testavimo kainą [39].

Testų rinkinio suprantamumas yra svarbus faktorius, kuris veikia testo vykdymo kainą. Testo automatizavimas pasiekiamas koduojant testavimo atvejus programavimo kalba. Šių testavimo atvejų suprantamumas išauga, jeigu jie būna parašyti aiškiai, su teisingais pavadinimais, kurie nurodo jų paskirtį [30].

Dauguma testavimo atvejų generavimo metodų yra arba padengiamumo kriterijais arba klaidom grindžiami. Padengiamumo kriterijais grindžiami metodai, tokie kaip sąlygų/sprendimų ar būsenų perėjimų padengiamumas, reikalauja, kad visi specifikuoti kodo ar sistemos modelio elementai būtų vykdomi [40]. Klaidomis grindžiamas testų rinkinio generavimas turi privalumą, kad bet kokia produktas, per kurį praėjo testo rinkinys, garantuoja, kad jis neturės nustatytų klaidų. Padengiamumu grindžiamas testo rinkinys paprastai neturi šios savybės.

Testavimo atvejų prioritetų nustatymas

Grįžtamasis testavimas yra brangus testavimo procesas naudojamas validuoti modifikuotai programinei įrangai ir surasti, jei naujos klaidos buvo įtrauktos į anksčiau testuotą kodą. Gali būti brangu vykdyti pilnus grįžtamojo testo rinkinius, todėl testų kūrėjai gali kurti grįžtamųjų testų prioritetus taip, kad tie, kurie yra svarbiausi tam tikrais matavimais, yra vykdomi anksčiau grįžtamojo testavimo procese.

Programinės įrangos inžinieriai dažnai saugo testo rinkinius, kuriuos sukuria savo programinei įrangai, kad vėliau galėtu pakartotinai naudoti tuos testo rinkinius, kai programinė įranga vystysis. Vykdymas visų testavimo atvejų esančių testo rinkinyje gali pareikalauti daug pastangų. Pavyzdžiui, produktui susidedančiam iš 20 000 eilučių kodo, įvykdyti visam testo

rinkiniui reikia septynių savaičių. Daugeliu atvejų, naudoti visą rinkinį nėra praktiška, nes užima daug laiko įvykdyti visus testavimo atvejus esančius testo rinkinyje.

Vienas pagrindinis testavimo atvejų prioritetų kūrimo privalumas yra tas, kad išauga testo rinkinio klaidų radimo laipsnis (angl. rate of fault detection) – matas kaip greitai testo rinkinys suranda klaidas per testavimo procesą. Pagerintas klaidos radimo laipsnis gali teikti ankstyvesnius atsiliepimus apie testuojamą sistemą, leisti ankstyvesnį derinimą (angl. debugging), ir padidinti tikimybę, kad, jei testavimas yra pirma laiko sustabdytas, tie testavimo atvejai, kurie siūlo didžiausią klaidos radimo galimybę duotu testavimo laiku bus įvykdyti.

Reikalavimų inžinerijos fazė apima skirtingas fazes reikalavimų verifikavimui ir validavimui: reikalavimų surinkimas, analizė, dokumentacija ir validavimas. Per surinkimo fazę, aktoriai ir panaudojimo atvejai identifikuojami, analizuojami analizavimo fazėje, kad būtų identifikuoti trūkstami panaudojimo atvejai. Visi šia panaudojimo atvejai ir aktoriai yra dokumentuojami ir validavimo fazėje jie validuojami, kad būtų įsitikinta, jog buvo padaryta tai, ko reikalavo klientas. Panaudos atvejų modelių su paminėtomis fazėmis naudojimas padeda suinteresuotoms šalims geriau suprasti sistemos kontekstą ir sumažinti reikiamus išteklius bei sutaupyti laiko programinės įrangos kūrimo cikle. Kadangi reikalavimų inžinerijos fazė yra susijusi su visomis programinės įrangos kūrimo gyvavimo ciklo fazėmis, ji turi efektyviai fiksuoti visus pakeitimus vykdomus kitose fazėse. Panaudojimo atvejų naudojimas reikalavimų inžinerijoje ne tik padeda reikalavimų inžinieriams geriau susieti reikalavimus su skirtingomis kūrimo fazėmis, bet taip pat teikia geresnį būdą suprasti reikalavimus. Besikeičiančių reikalavimų labiausiai įtakojama ir dėl to reikalaujanti daugiausia išteklių yra testavimo fazė.

1.12. Testų padengiamumo kriterijai

Testavimas užima daug laiko ir išteklių, todėl būtų naudinga turėti būdus sumažinti testavimo kainą, ir įvertinti šią kainą. Laikas ir ištekliai, būtini testavimui, padidėja, kai padidėja testo atvejų skaičių. Reiškia, norint sumažinti testavimo kainą, skaičius testo atvejų, sukurtų patenkinti išrinktus testo padengiamumo kriterijus, turi būti kiek galima mažesnis. Be to, skaičius testų, kurie turi būti įvykdyti, kad patenkintų parinktą testavimo strategiją, gali būti panaudotas testuotojų, kad įvertintų pastangas reikiamas atlikti testus [41].

Testų padengiamumas kartais naudojamas išmatuoti, kaip gerai programinė įranga yra ištestuota, o kūrėjai ir pardavėjai kartais naudoja jį įvertinti savo pasitikėjimą savo sukurta programine įranga.

Įvairių kriterijų, ar testavimo strategijų, gali būti laikomasi, kad išrinkti tinkamą ir baigtinį poaibį testų iš potencialiai begalinės srities. Testų kriterijai praktikoje nurodo reikalavimų kolekciją,

kuri bus įvykdytas. Struktūriniams padengiamumo kriterijams šitie reikalavimai yra pažymimi nustatyti komplektą objektų programos srauto grafuose, kurie turi būti padengti, kai testai bus vykdomi. Šie objektai gali būti gaunami iš kontrolės srautų ar iš duomenų srautų struktūros. Nuo 1970-ųjų, ir net anksčiau, struktūriniai padengiamumo kriterijai buvo plačiai naudojami išmatuoti testo rinkinių visapusiškumą [42].

1.13. Reikalavimų specifikacija grindžiamas testavimas

Reikalavimų specifikaciją grindžiamas testavimas yra viena iš perspektyviausių testavimo sričių. Tai vienas iš testavimo metodų, pagrįstas programos specifikacija, apibūdinančia jos elgesį. Toks testavimo procesas apima dvi pagrindines veiklas: pirma, tai validavimas, kad reikalavimai yra teisingi, išbaigti, nedviprasmiški ir logiškai nuoseklūs; antra, tai būtinų ir pakankamų, iš juodosios dėžės perspektyvos, testavimo atvejų komplekto kūrimas iš tų reikalavimų, kad būtų užtikrinta, jog dizainas ir kodas pilnai atitinka šiuos reikalavimus. Reikalavimų testavimo strategija yra integruoti testavimą per kūrimo gyvavimo ciklą ir susikoncentruoti ties reikalavimų specifikacijos kokybe. Reikalavimų testavimo procesas taip pat susitelkia ties defektų išvengimu, ne tik defektų radimu. Reikalavimų testų padengiamumo demonstravimas yra svarbus reglamentuotose srityse, tokiose kaip sveikatos priežiūra, saugumui kritinė programinė įranga ir t.t. Reikalavimai dažnai yra įvertinami, kaip netinkami testavimui, nes jie yra neišbaigti ar per daug dviprasmiški [42].

Specifikacija pagrįsta testavimo technika testo atvejus ir testo sąlygas gauna iš sistemos ar programinės įrangos reikalavimų specifikacijos. Testavimo atvejai gali būti kuriami nesigilinant į programos realizaciją.

Specifikacija grindžiamas testavimas teikia daug privalumų programinės įrangos procesui. Specifikacija duoda tikslų programinės įrangos aspektų aprašymą. Tai leidžia testuotojui gauti produkto pagrindinį funkcionalumą. Kuriant testus iš programinės įrangos specifikacijos, testai gali būti sukurti prieš sukuriant programinę įrangą. Kadangi daugelis klaidų atsiranda projektavimo fazėje, ankstyvas jų radimas gali sumažinti programinės įrangos kūrimo laiką ir kainą. Testų kūrimas priverčia atidžiau pažiūrėti į pačią specifikaciją, kas gali parodyti neaiškumus ir prieštaravimus. Jie gali būti ištaisyti anksti kūrimo cikle minimalia kaina.

Specifikacija grindžiamas testavimas tikrina ar programinės įrangos sistemos įgyvendinimas atitinka tos pačios sistemos specifikaciją. Formalios specifikacijos gali būti naudojamos kaip įvestis generuoti testo atvejams, kurie įgyvendina duotus kriterijus, generuoti įvesties duomenis, ir kaip orakulas apskaičiuoti numatomus rezultatus. Jeigu reikalavimai keičiasi

su programinės įrangos projektu, specifikacija gali būti modifikuojama ir testo atvejai generuojami dar kartą.

Naudojamos specifikacijos kalbos charakteristikos turės įtakos technikai, kuri bus naudojama testo atvejų generavimui specifikacija grindžiamame testavimo procese. Formali specifikacija gali būti vykdoma arba ne. Kai tikslas yra automatizuoti testavimo procesą, vėliau tikslas gali virsti į sudėtingesnį pasiekti [43].

Nepaisant testavimo svarbos programinės įrangos patikimumui, testavimas yra intensyvus ir brangus darbas. Testavimas be geros strategijos nebus efektyvesnis už testavimą su atsitiktiniais duomenimis. Testavimo efektyvumas stipriai priklauso nuo testų rinkinio, kuris naudojamas. Taip yra todėl, kad testų rinkinio išsamumas veikia testavimo apimtį ir galimybę rasti programinės įrangos klaidas. Specifikacija grindžiamas testavimas generuoja testavimo atvejų rinkinį iš informacijos gautos iš specifikacijos, tai nereikalauja žinių apie vidinę programos struktūrą [9].

Programinės įrangos kūrime, reikalavimai turi būti sukurti prieš vykdymą, ir specifikacija turi egzistuoti prieš kodo kūrimą. Specifikacija grindžiamos priemonės testų rinkinio generavimui yra labai naudingos, nes testo atvejai gali būti generuojami prieš kodavimą. Tai palengvina programinės įrangos kūrimo fazę, nes ji gali būti atlieka paraleliai, o tai duoda daugiau laiko testo planų paruošimui ir sutrumpina viso proceso laiką.

Reikalavimais grindžiamas testavimo procesas remiasi dvejomis pagrindinėmis veiklomis: pirma, reikalavimų validavimu, kad jie yra teisingi, išbaigti, nedviprasmiški, logiškai nuoseklūs; antra, projektuojant būtiną ir pakankamą testavimo atvejų komplektą iš tų reikalavimų, kad dizainas ir kodas visiškai atitinka šiuos reikalavimus. Projektuojant testus, reikia įveikti du klausimus: sumažinti nepaprastai didelį potencialių testų skaičių iki pagrįsto dydžio komplekto ir užtikrinti, kad testai gaus tinkamą atsakymą dėl tinkamos priežasties.

Reikalavimais paremto testavimo strategija yra integruoti testavimą per programinės įrangos kūrimo gyvavimo ciklą ir susifokusuoti į reikalavimų specifikacijos kokybę. Tai leidžia anksti aptikti defektus, kas yra pigiau negu rasti defektus per integravimo testavimą ar dar vėliau. Reikalavimais grindžiamas testavimo procesas taip pat sufokusuotas į defektų išvengimą, ne tik defektų radimą.

Specifikacijomis paremto testavimo privalumai:

- Mažinama kaina defektų radimui ir ištaisymui;
- Testai gali būti sukurti anksti, dar projektavimo procese;
- Testai gali būti dažnai pakartotinai panaudoti, pasikeitus modeliui;
- Testavimo atvejai gali būti suprojektuoti, kai tik specifikacijos baigtos;
- Testai sukuriami iš vartotojo požiūrio, ne iš projektuotojo;
- Testuotojui nereikalingos žinios apie jokias specifines programavimo kalbas;

- Testai yra objektyvūs, nes projektuotojas ir testuotojas yra nepriklausomi vienas nuo kito.

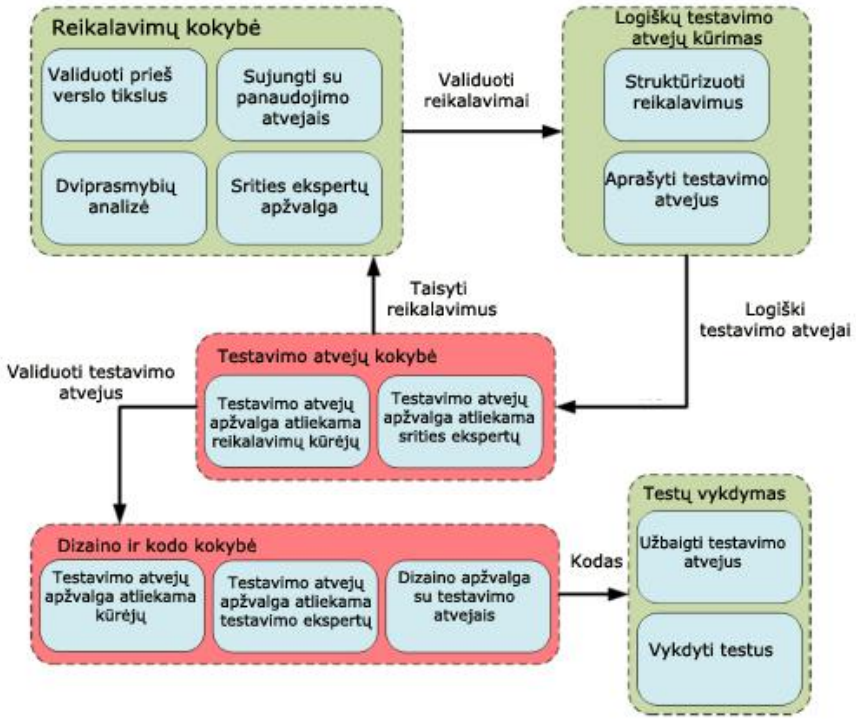
Reikalavimais grindžiamas testavimas gali būti padalintas į aštuonias veiklas [44]:

1. **Nustatyti testų išbaigimo kriterijus.** Testavimo pastangos turi specifinius, kiekybinius ir kokybinius tikslus. Testavimas yra išbaigtas tik tada, kai tikslai yra pasiekti (t.y. testavimas yra baigtas, kai visi funkciniai pokyčiai yra pilnai suaktyvinti defektų radimui, ir 100% visų pareiškimų bei atsišakojimų buvo įvykdyti sėkmingai vienu paleidimu arba paleidimų serija be kodo keitimo tarp jų);
2. **Testavimo atvejų projektavimas.** Logiški testavimo atvejai yra apibrėžiami penkiomis charakteristikomis: pradine sistemos būsena prieš testo vykdymą, duomenimis duomenų bazėje, įvestimi, numatoma išvestimi, ir galutine sistemos būsena;
3. **Testavimo atvejų kūrimas.** Yra dvi dalys, kurių reikia siekiant sukurti testo atvejus: būtinų duomenų kūrimas, ir kūrimas komponentų, kurie palaiko testavimą (t.y. sukurti navigaciją, kuri nurodytų, kurios programos dalys turi būti testuojamos);
4. **Testų vykdymas**. Vykdyti testavimo atvejų žingsnius sistemai ir dokumentuoti rezultatus;
5. **Testo rezultatų verifikavimas.** Verifkuoti, kad testo rezultatai yra tokie, kokie buvo numatyti;
6. **Testų padengiamumo verifikavimas.** Sekti funkcinio padengimo lygį ir kodo padengimą pasiektą testavimo serijos sėkmingo vykdymo metu;
7. **Defektų valdymas ir stebėjimas.** Bet kokie defektai aptikti per testavimo procesą yra stebimi. Stebimos defektų tendencijos ir statusas;
8. **Testų bibliotekos valdymas.** Testuotojas nustato ryšį tarp testavimo atvejų ir programos, kuri yra ištestuota. Testuotojas stebi ar testai buvo ar nebuvo vykdyti, ir ar vykdyti testai buvo išlaikyti ar ne.

Reikalavimais grindžiamų testų generavimas yra efektyvus kainos atžvilgiu ir padeda kurti testavimo atvejus ankstyvoje stadijoje. Testavimo veiklų pradėjimas ankstyvoje stadijoje gali sumažinti kainą ir pastangas. Be to, testavimo atvejai grindžiami reikalavimais gali geriau ištestuoti sistemą, nes jie yra sugeneruoti tiesiai iš kliento reikalavimų. Kaip bebūtų, ši tiriama sritis susiduria su daug problemų [45]:

- Testavimo atvejų generavimasis specifikacijos vis dar problema, nes reikalavimų specifikacija nepakankamai standartizuota, kad galėtų padėti gaunant kokybę ir reikiamą testavimo atvejų sistemai skaičių;
- Reikalavimų padengiamumas, t.y. kiek reikalavimų buvo patenkinta testavimo atveju, vis dar iššūkis;
- Atsekamumo nuorodų automatinis generavimas ir valdymas reikalauja daug pastangų.

1.13.1. *Reikalavimais grindžiamo testavimo metodika*



10 pav. Reikalavimais grindžiamo testavimo procesai [44]

1. **Validuoti reikalavimus prieš tikslus**. Optimizuoti projektą užtikrinant, kad kiekvienas reikalavimas tenkina mažiausiai vieną verslo tikslą. Jeigu nėra atitikimo tarp reikalavimų ir verslo tikslų (jeigu „kas“ neatitinka „kodėl“), tobulinimas yra būtinas.
2. **Sukurti panaudojimo atvejus reikalavimams.** Jei vienas ar daugiau panaudojimo atvejų negali būti adresuoti reikalavimams, tada reikalavimai neišbaigti.
3. **Vykdyti dviprasmybių apžvalgą**. Dviprasmybių apžvalga yra technika dviprasmiškų žodžių, frazių ir konstrukcijų identifikavimui. Ši apžvalga duoda aukštesnės kokybės reikalavimų komplektą.

4. **Vykdyti srities ekspertų apžvalgas**. Vartotojų ir srities ekspertų atsiliepimai turėtų būti naudojami reikalavimų patobulinimui.
5. **Struktūrizuoti ir formalizuoti reikalavimus**. Kad būtų pasiektas aukštas testų padengiamumas, reikalavimus reikia formalizuoti ir struktūrizuoti.

1.14. Reikalavimų specifikacijos integralumas su testavimo procesu

Integralumas – tai reikalavimų susietumas su testavimo procesu.

Motyvacija sekti reikalavimus atsiranda augant programinės įrangos sistemų sudėtingumui ir pastoviam greitos evoliucijos ir atnaujinimų poreikiui.

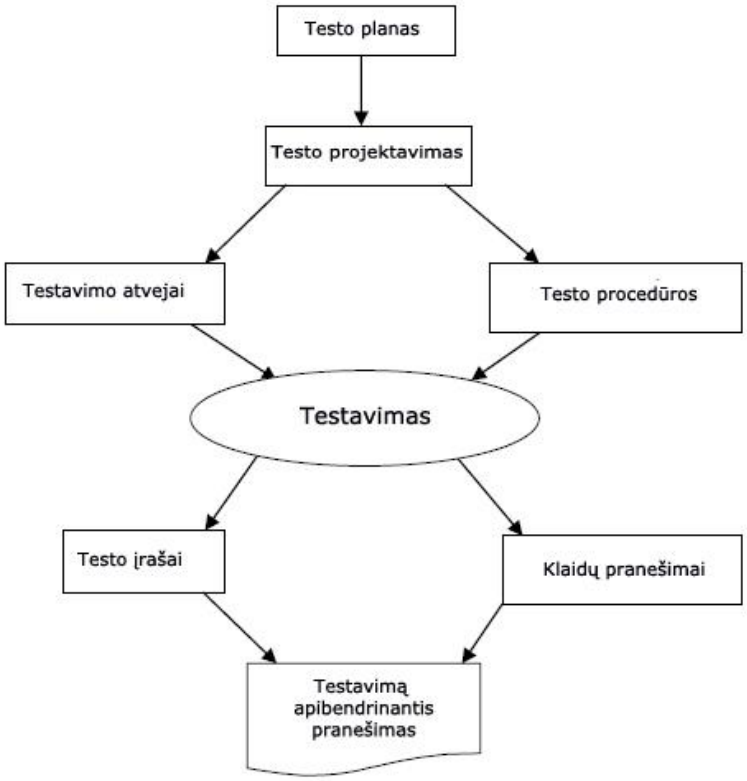
Pagrindinės reikalavimų specifikacijos integralumo su testavimo procesu naudojimo priežastys:

- Gilesnis kuriamos programinės įrangos supratimas;
- Galimybė efektyviai valdyti pasikeitimus;

1.15. IEEE829 standartas programinės įrangos testavimo dokumentavimui

IEEE829 standartas duoda tarptautiniu mastu pripažintą standartų, testo planavimo dokumentacijai, rinkinį. Šis standartas gali būti pritaikomas kiekvienoje testavimo gyvavimo ciklo stadijoje.

11 pav. vaizduoja visas testavimo veiklas pagal IEEE829 standartą [44]:



11 pav. Testavimo procesas pagal IEEE829 standartą [44]

IEEE829 standartas apima 8 dokumentų tipus [44]:

Testo specifikacija

1. **Testo planas**: aprašo, kaip testavimas bus valdomas ir vykdomas.
2. **Testo projektavimo specifikacija**: logiškai aprašo ką reikia testuoti egzaminuojant reikalavimus. Testo specifikacija aprašo testo sąlygas iš reikalavimų ir funkcinio dizaino dokumentaciją. Joje aprašomi reikalavimai reikšmėms, įvedamoms testavimui. Ši specifikacija yra nuoroda tarp testo reikalavimų ir testavimo atvejų.
3. **Testavimo atvejų specifikacija**: konvertuoja testo sąlygas į testavimo atvejus, pridedant esamus duomenis, išankstines sąlygas ir numatomus rezultatus.
4. **Testo procedūra**: praktiniais terminais aprašo, kaip vykdomi testai.
5. **Objektų naudojamų testavime ataskaita**: specifikuoja objektus sukurtus testavimui.

Testo vykdymas

1. **Testo įrašai**: testų detalės chronologine tvarka.
2. **Testo klaidų pranešimas**: netikėtų įvykių ir elgesio, kuris turi būti tiriamas, detalės.

Testo ataskaita

1. **Testavimą apibendrinantis pranešimas**: apibendrina ir įvertina testus.

Pasiruošimas testavimui yra pati svarbiausia bet kurio programinės įrangos projekto dalis. Per šia stadiją privaloma sukurti testus ir aprašyti reikalavimus testavimo aplinkai.

Testavimo atvejų specifikacija

Testavimo atvejai gali būti sukuriami, kai sukuriamas testo projektas. Testavimo atvejai gali padengti vieną ar daugiau testo sąlygų, išvestų iš testo projekto. Testavimo atvejis turi apimti:

- tikslius duomenis, kas reikalaujama testo vykdymui;
- Numatomus rezultatus ir išvestį;
- Pradines sąlygas, kurios aprašys pradinį tašką kiekvienam testui. Savybė (reikalavimas) iš testo projekto gali būti testuojama daugiau nei vienu testavimo atveju, ir testavimo atvejis gali testuoti daugiau nei vieną savybę. Tikslas yra, kad testavimo atvejų rinkinys ištestuotų kiekvieną savybę bent kartą.

Testo procedūrų specifikacija

Testo procedūros yra gaunamos iš testo projektavimo ir testavimo atvejų specifikacijų. Procedūrų dokumentas aprašo, kaip testuotojas fiziškai vykdys testus, reikalaujamus fizinius nustatymus, ir procedūros žingsnius, kuriuos reikia vykdyti.

Objektų naudojamų testavime ataskaita

Šis dokumentas yra perduodamas dokumentas ir aprašo prieš tai vykdytų testavimo stadijų detales. Jis aprašo, kas buvo sukurta testavimui ir parodo bet kokius pakeitimus bei naujus objektus. Dokumentas apima asmenų atsakomybes už kiekvieną objektą, kur jie fiziškai yra ir jų statusą.

Testo įrašai

Testo įrašai yra detalės, kurie testo atvejai buvo vykdomi, jų vykdymo tvarka, ir testo rezultatai. Testo rezultatai yra arba testas išlaikytas, reiškia, kad esami ir numatyti rezultatai buvo identiški, arba neišlaikytas, kai buvo neatitikimų. Jeigu yra neatitikimas, tada vienas arba daugiau testo klaidų pranešimų sukuriama arba atnaujinama, o jų identifikacija įrašoma testo įrašuose.

Testo įrašai yra svarbūs, nes leidžia stebėti testavimo progresą, taip pat teikia naudingos informacijos aiškinantis kas sukėlė incidentą. Jeigu incidentas yra kodo klaida, tai klaida galėjo

atsirasti ne testavimo atvejyje, kuris neišlaikė, bet tame, kuri buvo vykdytas anksčiau. Todėl testo sekų užrašymas padeda rasti klaidas.

Testo klaidos pranešimas

Šis pranešimo dokumentuoja bet kokį įvykį, kuris reikalauja vėlesnio tyrimo. Klaida turi būti dokumentuojama, kai testavimo metu atsiranda nenumatytas rezultatas. Pranešimas susideda iš visų detalių, tokių kaip esami ir numatyti rezultatai, kada jie neišlaikė, ir iš bet kokių įrodymų, kurie padės tolimesniam sprendimui.

2. EGZISTUOJANČIŲ SPRENDIMŲ ANALIZĖ

2.1. Reikalavimų specifikavimo šablonų analizė

Norint, kad reikalavimai būtų teisingai parengti, buvo nustatytos dvi svarbios prielaidos reikalavimų specifikavimo procesui: apimančios reikalavimus schemos ir reikalavimų saugyklos. Apimančios reikalavimus schemos turi būti pakankamai griežtos, kad galėtų išgauti svarbią informaciją, bet taip pat pakankamai lanksčios, kad užfiksuotų detales.

Sukurti kokybišką reikalavimų specifikaciją gali padėti šablonų naudojimas. Šablonai turi savo struktūrą ir grupavimą. Jų naudojimas reikalavimų specifikavimo procese padeda lengviau išsiaiškinti ir identifikuoti užsakovo poreikius, reikalavimus ir juos struktūrizuotai užfiksuoti.

Panagrinėsime labiausiai paplitusius reikalavimų specifikavimo šablonus, naudojamus reikalavimų inžinerijoje.

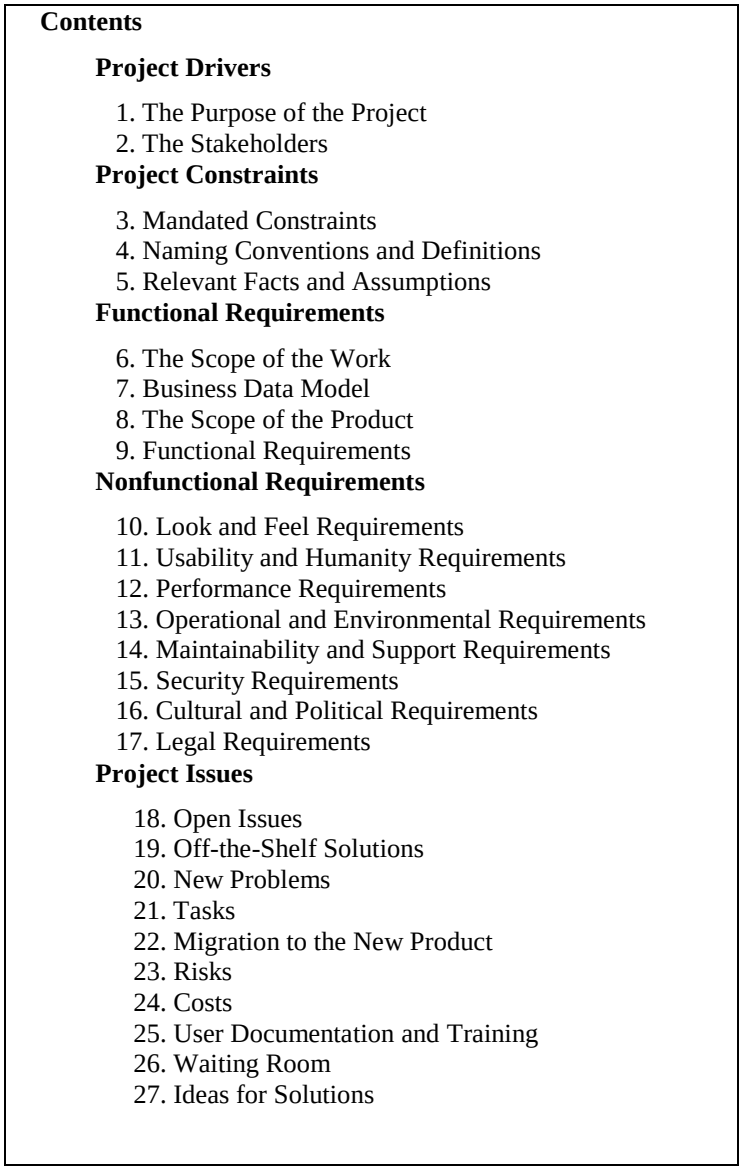
2.1.1. *Volere šablonas*

Volere šablonas (angl.Volere Requirements Specification Template) yra sukurtas Atlantic Systems Guild Limited Didžiojoje Britanijoje [43]. Paskutinioji versija (Edition 15) buvo išleista 2010 metais, todėl Volere galima laikyti naujausiu reikalavimų specifikavimo šablonu. Pirmasis šablonų variantas pasirodė 1995 metais. Kaip teigia autoriai Robertsonas ir Robertsonas, Volere yra sukaupta didelė reikalavimų specifikavimo konsultavimo ir tyrimo patirtis. Jų rekomenduojamas Volere procesas užtikrina, kad visi svarbūs reikalavimų aspektai yra atidžiai užfiksuoti.

Volere šablonas apima apribojimus veikiančius projektą, funkcinius ir nefunkcinius reikalavimus. Volere šablono kategorizavimas reikalauja apibūdinti tikimo kriterijus, kiekvieno reikalavimo priežastį ir kliento pasitenkinimą ir nepasitenkinimą, jeigu reikalavimas įvykdytas arba ne. Robertsono ir Robertsono filosofija atitinka ISO 13407 standartą, ir leidžia struktūrizuoti reikalavimus, užtikrinant, jog jie išliks tinkami ir testuojami.

Šablono dokumentą sudaro įvadinis skyrius ir pats šablonas. Įvadinis skyrius pateikia bendrą informaciją apie Volere, supažindina su reikalavimų tipais (funkciniais, nefunkciniais, projekto apribojimais ir t.t.). Tai pat aptariamas reikalavimų testavimas. Svarbus punktas yra patarimai ir reikalavimų specifikavimo ypatumų aprašymai. Įvadinis skyrius taip pat pateikia sutrumpinimų ir apibrėžimų sąrašą, kurie yra naudojami šablone. Antroji dalis susideda iš detalaus ir pilno reikalavimų specifikavimo dokumento šablono. Kiekvienas skyrius, poskyris ir pastraipa yra kruopščiai aprašyti ir pateikiami panaudojimo pavyzdžiai. Volere šablono turinys pateiktas 12 pav.

Volere šablonas naudojamas pradiniame sistemos kūrimo etape ir yra kaip pagrindas užregistruoti vartotojų reikalavimus. Šablonas suskirstytas į skyrius pagal reikalavimų tipą. Šablonas padeda sukaupti reikalavimus, kuriuos pateikia vartotojai per interviu arba kurie užregistruoti analizuojamo objekto veiklą reglamentuojančioje dokumentacijoje. Tai atviras šablonas, kurį galima pritaikyti konkrečiam atvejui. Šablono skyrių, kuris netinka nagrinėjamam objektui, galima išmesti arba sukurti naują skyrių, kuris leidžia specifikuoti specifines dalykinės srities charakteristikas.



12 pav. Volere šablono turinys [43]

Volere šablonas apima šiuos reikalavimų tipus [43]:

- funkciniai reikalavimai (angl. Functional requirements);
- nefunkciniai reikalavimai (angl. Nonfunctional requirements);
- projekto apribojimai (angl. Project constraints);
- projekto vykdytojai (angl. Project drivers);

- projekto išeiga (angl. Project issues);
- testavimo reikalavimai (angl. Testing requirements).

Visi reikalavimų tipai yra suskaidyti į smulkesnius ir nepersidengiančius potipius. Tokiu būdu užtikrinamas reikalavimų konkretumas, pilnumas, nepersidengiamumas ir vienareikšmiškumas. Volere reikalavimų specifikavimo šablone reikalavimai yra susieti su panaudojimo atvejais. Čia kiekvienas panaudojimo atvejis parodo tai, ką produktas atliks, todėl jis turi daug susijusių funkcinių reikalavimų. Be to, panaudojimo atvejis taip pat gali turėti ir nefunkcinių reikalavimų bei apribojimų. Kiekvienas surinktas reikalavimas, nepriklausomai nuo jo tipo (funkcinis, nefunkcinis, projekto apribojimas), turi daug atributų. Vienas atributas yra unikalus reikalavimo identifikatorius, o kitas – ryšys su visais panaudojimo atvejais, kuriuose dalyvauja šis reikalavimas. Aprašymas, pagrindimas bei tikimo kriterijus kartu nusako reikalavimo reikšmę ir tuo pačiu padaro jį matuojamu bei testuojamu.

Atributų naudojimo nauda:

- informacijos apie reikalavimą išskyrimas leidžia skirtingų peržiūrų filtravimą, rūšiavimą pagal tam tikrus atributus;
- leidžia reikalavimams būti specifikuotiems paprasta maniera keleto žmonių, nes atributai nukreipia, kas turi būti specifikuota nepriklausomai nuo to kas specifikavo reikalavimą;
- atributai sujungti su visu procesu ir reikalavimų būsenos leidžia kontrolę ir atsekamumą.

Naudojant Volere šabloną reikalavimams aprašyti, vadovaujamasi susisteminto proceso apribojimais, nusakančiais, kokio etapo metu ir kokie reikalavimai turi būti aprašyti. Kai kuriuose proceso etapuose naudojamas specializuotas apklausos lapas, padedantis surinkti reikalavimus, juos analizuoti ir apibendrinti.

Reikalavimas turi būti stebimas per visą sistemos kūrimo laiką, todėl logiška, kad numeravimas yra unikalus. Įvykių numeriai pritaikomi panaudojimo atvejams numeruoti. Reikalavimai siejami su panaudos atvejais, nurodant jų numerius.

• Funkciniai reikalavimai ir reikalavimai duomenims. Specialioje formoje aprašomi visi funkciniai reikalavimai 13 pav..

Volere reikalavimų šablonas		
Reikalavimo nr.	Reikalavimo tipas:	Panaudojimo atvejis:
Aprašas:		
Pagrindimas:		
Šaltinis:		
Tinkamumo kriterijus:		

Užsakovo pasitenkinimas:	Užsakovo nepasitenkinimas:
Priklausomybės:	Konfliktai:
Istorija:	Papildoma medžiaga:

13 pav. Volere šablonas funkciniams reikalavimam aprašyti [43]

2.1.2. *SRS šablono analizė*

SRS (Software Requirements Specification) yra samprata apie kliento reikalavimus. Tai abipusė apsidraudimo politika, garantuojanti kliento ir organizacijos tarpusavio reikalavimų suvokimą [46].

SRS šablone aprašomas sistemos elgesys, specifikuojami panaudojimo atvejai, aprašant vartotojų sąveikas su sistema. Šiame šablone visi panaudojimo atvejai įvardijami funkciniais reikalavimais. SRS šablone aprašomi ir nefunkciniai reikalavimai [46].

SRS šablono naudojimo privalumai [46]:

- Užfiksuoja susitarimą tarp kliento ir programinės įrangos kūrėjų;
- Pateikia apytikrį išlaidų planą ir tvarkaraštį;
- Padeda vartotojams nustatyti, ar programinė įranga atitinka nustatytus savo reikalavimus;
- Padeda sumažina kūrimui skiriamas pastangas.

Šablonas apima šias reikalavimų kategorijas [10]:

- Funkciniai reikalavimai;
- Patikimumo ir palaikomumo reikalavimai;
- Veikimo reikalavimai;
- Sąsajos reikalavimai;
- Tikrinimo reikalavimai;
- Dokumentacijos reikalavimai;
- Kokybės reikalavimai;
- Saugumo reikalavimai;
- Našumo reikalavimai;
- Išteklių reikalavimai.

2.1.3. *URS šablono analizė*

URS (User Requirements Specification) šablonas aiškiai apibrėžia, vartotojo norus ką sistema turi daryti. Dokumentas skirtas užrašyti vartotojų poreikiams [47].

URS šablono privalumai:

- Formalizuoja reikalavimus iš visų vartotojų;
- Atvaizduoja reikalavimus;
- Pateikia bendrą informaciją apie sistemos aplinką, funkcinius poreikius ir kokybės atributus.

URS parengimas susideda iš dviejų žingsnių [47]:

1. Šaltinių identifikavimo;
2. Reikalavimų organizavimo.

Dokumento šaltiniai - produkto vizija, vartotojų procedūriniai dokumentai (gidai, procedūrų aprašymas, mokymosi medžiaga), kita esamos sistemos dokumentacija ir kt. [47].

Kiekvienas reikalavimas turi unikalų identifikatorių, siekiant išvengti reikalavimų pasikartojimų šablone. Reikalavimų pilnumas URS šablone užtikrinamas sutelkiant dėmesį ne į sistemos funkcijas, o į visus galimus scenarijus ir vartotojo tikslus. Reikalavimų nuoseklumas URS šablone užtikrinamas, sprendžiant visus nesuderinamumus tarp reikalavimų, tai išvengiama specifikuojant reikalavimus prisilaikant struktūros. Kiekvienas reikalavimo pakeitimas turi būti dokumentuotas, tai reiškia, kad kiekvieno reikalavimo istorija turi būti fiksuojama. Taip pat URS šablone reikalavimas turi būti aprašomas tik vieną kartą. Šablone aprašytus reikalavimus galima atsekti, nes kiekvienas reikalavimas turi unikalų identifikatorių ir šaltinį.

Reikalavimai išdėstyti URS testuojami našumu. Vartotojų reikalavimų specifikacija nėra skirta techniniam dokumentavimui, skaitytojas bendromis žiniomis apie sistemą turi sugebėti suprasti reikalavimus, išdėstytus URS [47].

2.1.4. *Reikalavimų specifikavimo šablonų palyginimas*

Toliau pateikiama apžvelgtų reikalavimų specifikavimo šablonų palyginimo lentelė. Šablonai tarpusavyje lyginami atsižvelgiant į svarbius reikalavimų specifikavimo kriterijus: specifikavimo procesas, reikalavimų tipai bei detalumas, susietumas, panaudojimo atvejai, suderinamumas su įrankiu, modifikavimas ir išlaidos. Kriterijų paaiškinimai:

- specifikavimo procesas – reikalavimų surinkimo bei specifikavimo proceso pateikimas;
- reikalavimų tipai - šablono išskirstymas į skyrius pagal reikalavimų tipus;

- reikalavimų detalumas – reikalavimų detalizavimo ir tikslumo lygis;
- reikalavimų susietumas – specifikuojamų reikalavimų susietumas tarpusavyje;
- panaudojimo atvejai – panaudojimo atvejų naudojimas bei reikalavimų surišimas su pastaraisiais;
- suderinamumas su įrankiu – šablono panaudojimas reikalavimų valdymo įrankiuose;
- modifikavimas – šablono struktūros, turinio keitimas bei pritaikymas asmeniniams poreikiams;
- kaštai – šablono apmokestinimo nebuvimas;

Išanalizuotų reikalavimų specifikavimo šablonų privalumų bei trūkumų palyginimas pagal išvardintus kriterijus pateiktas 5 lentelėje.

5 lentelė Reikalavimo specifikavimo šablonų palyginimas

Savybė	Šablonas			
	Volere	SRS	URD	CPE4004
Specifikavimo procesas	+/-	-	-	-
Reikalavimų tipai	+	+	-	+
Reikalavimų detalumas	+	+/-	-	+/-
Reikalavimų susietumas	+	+/-	+/-	+/-
Panaudojimo atvejai	+	+	-	+
Suderinamumas su įrankiais	+/-	-	-	-
Modifikavimas	+	+	+	+
Kaštai (nemokamas)	+/-	+	+	+
Specifikavimo procesas	+/-	-	-	-

Iš 5 lentelės matome, jog tinkamiausias sprendimas reikalavimų specifikavimui yra Volere šablonas. Reikalavimų tipų pasirinkimo galimybė, detalumas bei specifikavimo procesas yra tikrai svarbūs specifikuojant reikalavimus. Be to, reikia atkreipti dėmesį ir į vartotojus, dirbsiančius su šablonais. Volere šablono dalinis suderinamumas su reikalavimų valdymo įrankiais yra itin svarbus. Pagrindinis jo minusas yra tame, jog šablonas, naudojamas nemokymo tikslams, yra mokamas.

2.2. Reikalavimų atsekamumo įrankių analizė

Įrankių analizei buvo pasirinkti keletas įrankių padedančių užtikrinti reikalavimų integralumą su testavimo procesu. Šie įrankiai kuria, atnaujina ir pašalina reikalavimus bei susieja juos su jų šaltiniu. Egzistuoja daugybė reikalavimų valdymo ir atsekamumo įrankių, kurie pasiekiami literatūroje ir internete. Buvo pasirinkti tik gerai žinomi ir plačiai naudojami įrankiai.

Tai įrankių, su kuriais buvo atlikta analizė sąrašas:

- DOORS;
- CaliberRM;
- TopTeam Analyst;
- Rational RequisitePro.

2.2.1. *DOORS įrankis*

DOORS *(Dynamic Object Oriented Requirements System)* yra reikalavimų valdymo įrankis sukurtas Telelogic, 2008 metais įsigytas IBM Rational. Reikalavimų valdymas procesas, kurio metu fiksuojami ir sekami vartotojo reikalavimai, kurie keičiasi per kūrimo gyvavimo ciklą. Šis įrankis skirtas užfiksuoti, susieti, atsekti, analizuoti ir valdyti pasikeitimus informacijoje, kad būtų užtikrintas projekto atitikimas su specifikuotais reikalavimais ir standartais. Reikalavimų atsekamumui DOORS teikia lengvą būda kurti nuorodas tarp skirtingų reikalavimų. IBM® Rational® DOORS® padeda sumažinti kainą, padidinti efektyvumą ir pagerinti kokybę optimizuojant reikalavimų bendravimą, bendradarbiavimą ir verifikavimą [48]. DOORS taip pat naudingas pasikeitimų valdyme, jis iš karto pažymi pasikeitimus, kurie gali paveikti kitus reikalavimus. Įrankis palaiko Unix Solaris ir Windows operacines sistemas.

DOORS teikia išsamią reikalavimų valdymo aplinką. Įrankis gali būti naudojamas organizacijose, kur daug žmonių dirba su reikalavimais tuo pačiu metu. Visos suinteresuotos šalys gali bendradarbiauti reikalavimų valdymo procese naudojant internetinę naršyklę. Vartotojai gali prisijungti prie duomenų bazės naudodamiesi kliento prisijungimą. Priklausomai nuo vartotojui suteiktų teisių, jis gali žiūrėti ar pakeisti informaciją duomenų bazėje.

Viena iš pagrindinių DOORS savybių yra galimybė valdyt nuorodas. Pagrindinė idėja yra ta, kad pasikeitimas tampa labiau atsekamas, jeigu žinoma koks reikalavimas priklauso nuo kokio kito reikalavimo.

14 pav. Nuorodų DOORS įrankyje langas [48]

Nuorodos automatiškai pažymimos, kaip abejotinos, kai nuorodos šaltinis pasikeičia. Abejotinos nuorodos gali būti lengvai randamos filtruojant ar ieškant jų, ir gali butį ištaisytos atlikus patikrinus pasikeitusį šaltinį.

Kiekvienam moduliui DOORS laiko vartotojo veiksmų istorijos įrašus, įskaitant vartotojo prisijungimus ir sesijas, taip pat pasikeitimus duomenims. Modulio istorija gali būti filtruojama ir randama pagal vartotojo vardą ar padaryto pasikeitimo tipą.

DOORS įrankio savybės:

- Sąsaja leidžianti reikalavimų valdymo pasirinkimą;
- Projekto išplėčiamumas;
- Lanksti, moderni, lengvai naudojama reikalavimų atsekamumo matrica;
- Išsami pagalba rašant, struktūrizuojant, valdant ir analizuojant reikalavimus ir jų atsekamumą;
- Integracija su kitais Telelogic sprendimais ir trečiųjų šalių įrankiais.

2.2.2. *CaliberRM įrankis*

CaliberRM yra reikalavimų valdymo įrankis sukurtas Borland. Jis skirtas užfiksuoti ir valdyti verslo, techninius, funkcinius ir sistemos reikalavimus. Įrankis leidžia suinteresuotoms šalims efektyviai bendradarbiauti, kad projektas būtų pristatytas laiku, neviršijant biudžeto ir atitiktų savo specifikaciją [49]. Įrankis valdo atsekamumo informaciją tarp skirtingų reikalavimų, taip pat tarp artefaktų ir kitų įrankių. Įrankis taip pat suteikia keletą būdų peržiūrėti visus reikalavimus ir jis sujungia šias savybes pasikeitimų įtakos analizei. CaliberRM saugo įrašus apie visus pasikeitimus artefaktams ir nuorodoms tarp jų, taigi yra įmanoma grįžti laiku ir pamatyti, kokie pasikeitimai reikalavimams anksčiau buvo padaryti.

CaliberRM turi du įrankius, kurie leidžia vartotojams valdyti jų reikalavimus naudojant interneto prieigą – CaliberRM Web ir CaliberRM WebView. Kiekvienas šių įrankių leidžia projekto komandos nariams prisijungti prie CaliberRM serverio per standartinę interneto naršyklę [50].

CaliberRM teikia du pagrindinius būdus peržiūrėti reikalavimų atsekamumo ryšiui. Pirmas yra atsekamumo diagrama, kuri pateikia informaciją grafiškai. Antras būdas yra filtruojama atsekamumo lentelė [50].

CaliberRM techninėje charakteristikoje parašyta, kad įrankis palaiko tik Windows operacinę sistemą, taigi nepalaiko daugialypių platformų [50].

CaliberRM įrankyje kiekvienas reikalavimas turi du skirtingus skaičius asocijuojamus su juo. Vienas yra hierarchijos numeris, kuris nurodomas pagal reikalavimo vietą projekto medyje ar hierarchijoje. Hierarchijos numeris keičiasi, kai reikalavimai pridedami, redaguojami ar ištrinami. Kitas numeris asocijuojamas su kiekvienu reikalavimu yra unikalus serijinis numeris (ID). ID numeris nesikeičia ir jis pakartotinai nenaudojamas net ištrynus reikalavimą [51].

2.2.3. *TopTeam Analyst įrankis*

TopTeam Analyst yra sprendimas reikalavimų surinkimui ir valdymui [50]. Jis palaiko panaudojimo atvejus ir testavimo atvejus, kad būtų palaikytas reikalavimų specifikavimo procesas. Įrankis turi galingą saugyklą visiems reikalavimų artefaktams tokiems, kaip prototipai, dokumentai ir t.t. Įrankio pagalba gali kurti ir vizualizuoti atsekamumą į sistemos elementus naudojant keturias skirtingas sąsajas. TopTeam palaiko keletą skirtingų būdų reikalavimų importavimui į įrankį.

TopTeam Analyst įrankio savybės:

- Hierarchinė reikalavimų struktūra;
- Reikalavimų medis;
- Reikalavimų importavimas iš MS Word dokumento;
- Galimybė peržiūrėti filtruotus reikalavimus;
- Atsekamų testavimo atvejų kūrimas;
- Atsekamumo matricos redagavimas;
- Atsekamumas tarp bet kurių sistemos elementų;
- Testavimo atvejų medis.

2.2.4. *Rational RequisitePro įrankis*

Retional RequisitePro taip pat yra reikalavimų valdymo įrankis sukurtas IBM. Įrankis leidžia išsaugoti reikalavimų specifikacijos dokumentą, susieti reikalavimus su panaudojimo atvejų diagramomis ir testavimo atvejais. Jis padeda projekto komandoms valdyti jų sukurtus reikalavimus, parašyti gerus testavimo atvejus, pagerinti atsekamumą, sustiprinti bendradarbiavimą,

sumažinti projekto pakartotinį darbą ir padidinti kokybę [49]. Kai atsiranda pasikeitimai reikalavimams, Rational RequisitePro identifikuoja atsakančius programinės įrangos artefaktus, kurie yra paveikti pasikeitimo.

2.2.5. *Kriterijai reikalavimų atsekamumo įrankiams*

Per įrankių analizės fazę nustatyti konkretūs kriterijai įrankių vertinimui. **Error! Reference source not found.** lentelėje pateikti kriterijai reikalavimų atsekamumo įrankiams.

6 lentelė Kriterijai reikalavimų atsekamumo įrankiams

Kriterijai	Aprašymas
Reikalavimų identifikacija	Įrankis turi palaikyti reikalavimų identifikaciją. Pvz.: ID numeris kiekvienam individualiam reikalavimui yra būtinas.
Reikalavimų klasifikavimas ir peržiūra	Įrankis turi turėti galimybę klasifikuoti reikalavimus į logines vartotojo nustatytas grupes. Taip pat įrankis turi palaikyti tų pačių duomenų įvairias peržiūras.
Formatai	Įrankis turi galėti specifikuoti reikalavimus tekstiniu, grafiniu ir modeliu paremtu aprašymu.
Pasikeitimų valdymas	Įrankis turi turėti galimybę valdyti pasikeitimus. Visi pasikeitimai su reikalavimais turi būti sekami ir saugomi duomenų bazėje.
Atsekamumas	Įrankis turi leisti vykdyti atsekamumą per nuorodas tarp reikalavimų. Taip pat įrankis turi palaikyti atsekamumą nuo reikalavimų į kitus artefaktus.
Grafinis vaizdavimas	Tai patogus atsekamumo vaizdavimas grafine forma. Atsekamumo nuorodos gali būti vaizduojamos naudojant matricas, medžius ar grafus, taigi informacija visada gali būti parodyta tinkama, labiau suprantama forma.
Dokumentacijos palaikymas	Įrankis turi turėti galimybę skaityti ir importuoti reikalavimus iš reikalavimų specifikacijos dokumentų sukurtų ne įrankyje, išlaikydamas dokumento turinio struktūrą. Taip pat jis turi

	turėti galimybę generuoti oficialius ir vidinius dokumentus. Įrankis naudoja iš anksto apibrėžtus dokumento apibūdinimus sugeneruoti dokumentui su tam tikrais duomenimis iš duomenų bazės.
Padengiamumo analizės palaikymas	Įrankis turi turėti galimybę teikti peržvalgą artefaktų, kurie neturi atsekamumo su kitais artefaktais.
Įrankių integracija	Įrankis turi turėti galimybę sąveikauti su kitais įrankiais, kad informacija saugoma juose būtų matoma ir nurodoma.

Skirtingi įrankiai turi skirtingas funkcijas ir savybes. **Error! Reference source not found.** lentelėje apibūdintas įrankių funkcionalumas aprašytas plačiau žemiau.

Reikalavimų identifikacija reiškia galimybę identifikuoti kiekvieną atskirą reikalavimą, kad būtų galima lengvai atskirti juos vieną nuo kito. Tai gali būti padaryta su reikalavimų identifikacijos numeriais ir reikalavimų atributų pagalba. Be to, įrankis turi palaikyti reikalavimų prioritezavimą, nes kai kurie reikalavimai yra svarbesni už kitus, ir savybės, kurios yra svarbiausios, turi būti įgyvendintos pirmiausiai.

Reikalavimų klasifikacija ir peržiūra yra galimybė klasifikuoti reikalavimus į logines vartotojo nustatytas grupes, taip pat galimybė peržiūrėti tuos pačius duomenis skirtingiems vartotojams skirtingomis peržiūromis. Peržiūra siūlo galimybę peržiūrėti ir keisti laisvai pasirenkamus projekto duomenis lengvai konfigūruojamoje reprezentacijoje.

Formatai yra galimybė specifikuoti reikalavimus tekstiniais, grafiniais ir modeliu paremtais aprašymais. Standartinės sistemos modeliavimo technikos ir notacijos turi būti palaikomos (UML, panaudojimo atvejai ir t.t.).

Pasikeitimų valdymas yra svarbiausia savybė reikalavimų atsekamumo įrankyje. Įrankis turi teikti galimybę sekti visus pasikeitimus reikalavimuose ir saugoti juos duomenų bazėje. Reikalavimų pasikeitimų istorija (kas, ką, kada, kur, kodėl, kaip?) turi būti registruojama. Pasikeitimų valdymas paprastai apima daugybę veiksmų tokių, kaip perspėjimai elektroniniu paštu ir įtakos analizė. Pasikeitimų valdymas yra ypatingai svarbus vykdant kūrimą bendradarbiaujant ir norint vykdyti kūrimo procesą taisyklingai.

Atsekamumas reiškia atsekamumą per nuorodas tarp reikalavimų. Tai taip pat yra galimybė aprašyti atsekamas asociacijas tarp reikalavimų. Atsekamumas gali būti iš reikalavimų į

savybes, produkto versijas, komponentus, dizaino dokumentus, testavimo atvejus ir t.t. Atsekamumas yra svarbi savybė pasikeitimų valdymo veiklose. Atsekamumu galima patikrinti reikalavimų ir kitų artefaktų nuoseklumą.

Grafinis vaizdavimas - tai patogus atsekamumo vaizdavimas grafine forma. Atsekamumo nuorodos gali būti vaizduojamos naudojant matricas, medžius ar grafus, taigi informacija visada gali būti parodyta tinkama, labiau suprantama forma vartotojui.

Dokumentacijos palaikymas padaro reikalavimų dokumento valdymą lengvesnį. Vartotojas gali sukurti reikalavimų dokumentą neprisijungęs prie reikalavimų atsekamumo įrankio. Vėliau reikalavimų dokumentas gali būti importuojamas į reikalavimų duomenų bazę su įrankio pagalbą. Reikalavimai iš dokumento gali būti perkelti į duomenų bazę pusiau automatiškai po vieną, ar net visas dokumentas gali būti patalpintas duomenų bazėje. Dokumentų generavimas reiškia, kad norimi reikalavimai yra išrenkami iš duomenų bazės į reikalavimų dokumentą. Nėra praktiška spausdinti visą duomenų bazės turinį dokumente, todėl turi būti galimybė pasirinkti tinkamus reikalavimus.

Padengiamumo analizės palaikymas - Įrankis turi turėti galimybę teikti peržvalgą artefaktų, kurie neturi atsekamumo su kitais artefaktais.

Įrankių integracija yra praktiškas, kai įmonė turi daug skirtingo dydžio projektų, ir daug skirtingų naudojama kartu su reikalavimų atsekamumo įrankiu. Įrankis turi būti lengvai adaptuojamas ir išplečiamas priklausomai nuo organizacijos ar projekto poreikių.

2.2.6. *Analizės rezultatai*

Visi šie įrankiai buvo analizuojami pagal 2.2.5. skyriuje aprašytus kriterijus. 7 lentelė Atliktos įrankių analizės rezultataipateikti analizės rezultatai.

7 lentelė Atliktos įrankių analizės rezultatai

Kriterijai	DOORS	CaliberRM	TopTeam Analyst	Rational RequisitePro
Reikalavimų identifikacija	✓	✓	✓	✓
Reikalavimų klasifikacija ir peržiūros		✓	✓	✓
Formatai	✓	✓	✓	✓

Pasikeitimų valdymas	✓	✓	✓	✓
Atsekamumas	✓	✓	✓	✓
Grafinis vaizdavimas	✓	✓	✓	✓
Dokumentacijos palaikymas	✓	✓	✓	✓
Padengiamumo analizės palaikymas	✓	✓	✓	✓
Įrankių integracija	✓	✓		✓

2.3. Technikos

Egzistuojančios reikalavimų atsekamumo technikos neaprašo jokio skirtumo tarp reikalavimų, kuriuos verta atsekti ir reikalavimų, kuriuos mažiau verta atsekti. Tai padidina pastangas susijusias su reikalavimų atsekamumu, todėl reikalavimų atsekamumo įgyvendinimas tampa brangesnis [27]. Atsekamumo vertė priklauso nuo keleto parametrų tokių, kaip svarbumas suinteresuotoms šalims, rizika ar kintamumas ir būtina atsekamumo kaina. Egzistuojami sprendimai išnagrinėti pagal šiuos parametrus.

2.3.1. *Verte grindžiamas reikalavimų atsekamumas (VGRA)*

B. Ramesh [48] identifikuoja su reikalavimų atsekamumo vartotojų tipus: žemos pabaigos atsekamumo naudotojai ir aukštos pabaigos atsekamumo vartotojai. Žemos pabaigos atsekamumo naudotojai fiksuoja atsekamumo informaciją tolygiai visiems reikalavimams. Toks atsekamumas yra brangus. Aukštos pabaigos atsekamumo vartotojai pripažįsta, kad visi reikalavimai nėra tolygus vienas kitam, atsižvelgdami į jų kritiškumą ar reikšmę. Be to, jie valdo atsekamumą. Taip pat jie naudoja atsekamumą kritiškiems projekto reikalavimams, kad išlaikytų kontroliuojamą kainą ir pasiektų atsekamumo naudos. Tikslais grindžiamas reikalavimų atsekamumo procesas.

VGRA tikslas yra identifikuoti, atsekamumo nuorodas paremtas prioritezuotais reikalavimais [48]. Atsekamumo nuorodų identifikavimas anksti projekto gyvavimo cikle yra lengviau nei vėlesnėse stadijose. VGRA sumažina atsekamumui reikiamas pastangas. VGRA procesas susideda iš penkių žingsnių:

1. **Reikalavimų apibūdinimo**. Per reikalavimų apibūdinimą projekto vadovas ar reikalavimų inžinierius analizuoja programinės įrangos reikalavimų specifikaciją identifikuoti

reikalavimams [48]. Unikalus identifikatorius yra priskiriamas kiekvienam reikalavimui. Reikalavimų apibūdinimo žingsnio rezultatas yra komplektas reikalavimų ir jų ID;

2. **Reikalavimų prioritezavimas.** Per šią fazę visos suinteresuotos šalys vertina reikalavimus pagal tris kriterijus: vertė, rizika ir pastangos kiekvienam reikalavimui [48]. Šio žingsnio rezultatas yra tvarkingas sąrašas prioritezuotų reikalavimų paremtų trimis svarbumo lygiais;
3. **Reikalavimų pakavimas.** Šis žingsnis yra svarbiausias ir jis leidžia identifikuoti reikalavimų telkinius [48]. Šie reikalavimų telkiniai padeda kurti ir patobulinti programinės įrangos architektūrą iš duoto reikalavimų komplekto;
4. **Nuorodų tarp artefaktų kūrimas**. Per šį žingsnį projekto komanda identifikuoja atsekamumo nuorodas tarp reikalavimų ir artefaktų [48]. Svarbūs reikalavimai yra atsekami detaliau nei kiti reikalavimai. Šie svarbūs reikalavimai gali būti identifikuoti iš prioritezuotų reikalavimų remiantis trejais lygiais sukurtais per reikalavimų prioritezavimo žingsnį;
5. **Evoliucija**. Projekto vadovas gali naudoti atsekamumą dėl skirtingų priežasčių tokių, kaip įvertinti pasikeitimo įtaką [48]. Paveikslėlis vaizduoja verte grindžiamo atsekamumo (VGRA) procesas.

Heindl ir Biffil [49] atliko tyrimą, kurio metu buvo pasirinkti 46 reikalavimai. Šių reikalavimų pilnas atsekamumas buvo lyginamas su VGRA. Šio tyrimo rezultatai:

- Tyrimo rezultatai parodė, kad susifokusavimas ties svarbiausiais reikalavimais sumažino skiriamas pastangas lyginant su skiriamas naudojant atsekamumą su visais reikalavimais. Didelės pastangos skiriamos reikalavimų atsekamumui yra faktorius dėl kurio projekto komanda neįgyvendina atsekamumo. Šios skiriamos pastangos gali būti sumažintos naudojant TGRA techniką, kuriai įgyvendinti reikia 35% mažiau pastangų lyginant su pilnu atsekamumu [49];
- Yra lengviau užfiksuoti atsekamumo informaciją ankstyvoje programinės įrangos kūrimo gyvavimo ciklo fazėje lyginant su atsekamumo fiksavimu vėlesnėse stadijose. Pvz.: jei nebuvo fiksuota atsekamumo informacija per projektą, tai, atsiradus pasikeitimui, atsiras poreikis atsekamumo informacijai, ir reikės daugiau pastangų valdyti atsekamumo nuorodoms;
- VGRA prioritetizavimo žingsnis identifikuoja vertingiausius reikalavimus, kuriuos reikia atsekti detaliau nei kitus. Šis žingsnis padeda sumažinti atsekamumui skiriamas pastangas. Visos suinteresuotos šalys paprastai dalyvauja reikalavimų prioritetizavime.

.

2.3.2. *Savybėmis orientuotas reikalavimų atsekamumas (SORA)*

Tam tikri artefaktai tokie, kaip programinės įrangos reikalavimų specifikacija, dizaino dokumentai, kodas ir testavimo atvejais yra sukuriami per programinės įrangos kūrimo procesą. Kai kurioje nors iš programinės įrangos kūrimo gyvavimo ciklo stadijoje atsiranda pasikeitimo poreikis, sunku atrasti, kurie programinės įrangos artifaktai bus paveikti šio pasikeitimo poreikio. Taip pat programinės įrangos inžinieriams sunku sukurti ir valdyti atsekamumo nuorodas. Savybėmis orientuoto reikalavimo atsekamumo technika sumažina atsekamumo nuorodų valdymo sudėtingumą identifikuojant jas per prioritetizuotus reikalavimus ir įvertinant kainą bei pastangas [50]. Tai keletas terminų geresniam SORA koncepcijos supratimui:

- **Savybės** yra pagrindinės produkto charakteristikos. Šios savybės gali būti klasifikuojamos pagrindu galimybių, srities technologijų, įgyvendinimo technologijų ir veikiančios aplinkos [50];
- **Savybių modeliavimas –** tai savybių identifikavimo procesas ir jų organizavimas į modelį, vadinamą savybių modeliu [50]. Veiklos įtrauktos į savybių modeliavimą yra aprašytos 8 lentelė Savybių modelio žingsniai ir veiklos

8 lentelė Savybių modelio žingsniai ir veiklos

Žingsniai	Veiklos
1. Srities planavimas	1.1. Srities pasirinkimas
	1.2. Ribų nustatymas
	1.3. Srities analizės organizavimas
	1.4. Srities direktorijos kūrimas
2. Savybių identifikavimas	2.1. Terminologijos analizavimas
	2.2. Produkto kategorijų identifikavimas
	2.3. Savybių patalpinimas kiekvienoje kategorijų
	2.4. Produktų ir savybių sąrašas
3. Savybių organizavimas	3.1. Organizuoti savybes į savybių diagramą
	3.2. Sumažinti savybių diagramos sudėtingumą
4. Savybių tobulinimas	4.1. Patobulinti savybių modelį
	4.2. Patobulinti abstrakcijų modelį

- **Galimybės** – tai vartotojo matomos charakteristikos, kurios gali būti identifikuotos, kaip operacijos, nefunkcinės charakteristikos ir nutolusios paslaugos [50];

- **Srities technologijos** atstovauja būdą įgyvendinti paslaugoms ar operacijoms [50];
- **Įgyvendinimo technikos –** tai bendros funkcijos, kurios naudojamos įgyvendinti srities funkcijoms, paslaugoms ir operacijoms [50];
- **Veikiančios aplinkos** – tai aplinkos, kuriose naudojama sistema [50];
- **Ryšiai** – savybių modeliavime egzistuoja trijų tipų ryšiai [50].“Sukomponuotas“ ryšys naudojamas, kai yra pilnas ryšys tarp savybės ir jos subsavybės. „Apibendrinimo/specializacijos“ ryšio savybės yra subsavybių apibendrinimai. Kai naudojamas „įgyvendinta per“ ryšys, tai viena savybė yra būtina kitos savybės įgyvendinimui.

Savybėmis orientuotas reikalavimų atsekamumo procesas susideda iš penkių fazių:

1. **Reikalavimų apibūdinimo**. Ši fazė susideda iš trijų veiklų: reikalavimų specifikacijos analizės, reikalavimų identifikavimo ir identifikatoriaus priskirimo kiekvienam reikalavimui [50]. Reikalavimų apibūdinimo tikslas yra normalizuoti vartotojų reikalavimus ir susieti juos su įvairiais artefaktais;
2. **Savybių modeliavimas**. Ši fazė susideda iš trijų veiklų, identifikuojančių kategorijas ir savybes, organizuojančių savybių diagramas ir priskiriančių susijusius reikalavimus prie savybių [50]. Šios fazės rezultatas yra savybių diagrama ir savybių sąrašas;
3. **Savybių prioritezavimas**. Ši fazė susideda iš dviejų veiklų: reikalavimų vertės vertinimo ir savybių sąrašo tvarkymo [50]. Šioje fazėje suinteresuotos šalys vertina reikalavimus pagal vertę, riziką ir skiriamas pastangas kiekvienam reikalavimui. Tada savybės prioritezuojamos. Savybėmis orientuotas reikalavimų atsekamumas naudoja skalę savybių prioritezavimui, kaip parodyta 9 lenetelė Prioritetų lygiai ir artefaktų klasifikacija

9 lenetelė Prioritetų lygiai ir artefaktų klasifikacija

Lygis	Detalumas	Artefaktų klasifikacija
1	Žemas	Komponentai
2	Vidutinis	Klasės
3	Aukštas	Metodai

4. **Nuorodų tarp reikalavimų kūrimas**. Ši fazė susideda iš trijų veiklų: artefaktų susiejimo su panašiomis savybėmis, įgyvendinami elementai pagal lygius ir sukuriamos reikalavimų atsekamumo nuorodos [50]. Naudojant SORA techniką, kuriant nuorodas tarp reikalavimų, atsekamumo nuorodos yra sugeneruojamos ir susiejami visi artefaktai su susijusiomis savybėmis. Šios atsekamumo nuorodos yra ryšiai tarp reikalavimų, savybių ir artefaktų. Panašaus svarbumo artefaktai reikalavimai yra atsekami detaliau, nei mažiau svarbūs reikalavimai. Šios fazės rezultatas yra atsekamumo nuorodų sąrašas;

5. **Atsekamumo nuorodų evoliucija**. Ši fazė susideda iš dviejų veiklų: atsekamumo nuorodų panaudojimo kūrimo procese ir atsekamumo nuorodų tobulinimo [50]. Per šią fazę atsekamumo nuorodos naudojamos konfliktų analizei, pasikeitimo įtakos analizei ir pilnumo kūrimo procese tikrinimui. Remiantis šia evoliucija gali būti keičiamos atsekamumo nuorodos.

2.4. Vartotojų analizė

Vartotojai yra tie asmenys, kuries dalyvauja reikalavimų sistemai integralumą su testavimo procesu užtikrinančiame procese. Išskiriamos katurios grupės:

- IS analitikas;
- IS užsakovas;
- Reikalavimų specifikavimo specialistas;
- Testuotojas.

Vartotojų tikslas - galimybė parengti specifikaciją taip, kad ja būtų galima lengvai naudotis, naudoti kuriant testavimo atvejus.

Panaudojant reikalavimų integralumą siekiama susisteminti reikalavimų gausą. Kadangi visų reikalavimų ištestavimas neįmanomas, siekiema atrinkti tik svarbiausius ir didžiausią riziką keistis turinčius reikalavimus.

Problema – reikalavimų ir jiems skirtų testavimo atvejų gausa.

Vartotojų siekiami tikslai ir problemos patiktos 6 lentelėje.

2.5. Analizės išvados

1. Atlikus reikalavimų analizę pastebėta, kad reikalavimų inžinerijoje vieno universalaus reikalavimo apibrėžimo nėra. Siekiant bent dalinai išspręsti šią problemą, priimta reikalavimus klasifikuoti į tipus;
2. Atlikus reikalavimų specifikavimo šablonų analizę, nustatyta, kad Volere šablonas pasižymi teigiamomis savybėmis ir geriausiai išpildo visas reikalavimų specifikavimo charakteristikas lyginant su kitų šablonų atžvilgiu;
3. Atlikus reikalavimų specifikavimo įrankių analizę, nustatyta, kad RequisitPro įrankiu galima sėkmingai valdyti pastovius ar nuolat besikeičiančius reikalavimus, pagerinti atsekamumą, sieti reikalavimus vienus su kitais ir su testavimo atvejais.
4. Besikertančių reikalavimų sekimas reikalauja daug laiko ir darbo, be to didelė klaidų tikimybė. Be to, kai kurie besikertantys reikalavimai negali būti pilnai suprasti, todėl jie

tvarkomi reikalavimų etape. Taigi, geriau valdyti besikertančius reikalavimus iš anksto kada jie yra išgaunami ir kaupiami, negu juos išgauti iš egzistuojančių reikalavimų dokumentų.

3. REIKALAVIMŲ INTEGRALUMĄ SU TESTAVIMO PROCESU UŽTIKRINANTIS METODAS

Šiame skyriuje aprašomas siūlymas, kaip būtų galima užtikrinti reikalavimų integralumą su testavimo procesu.

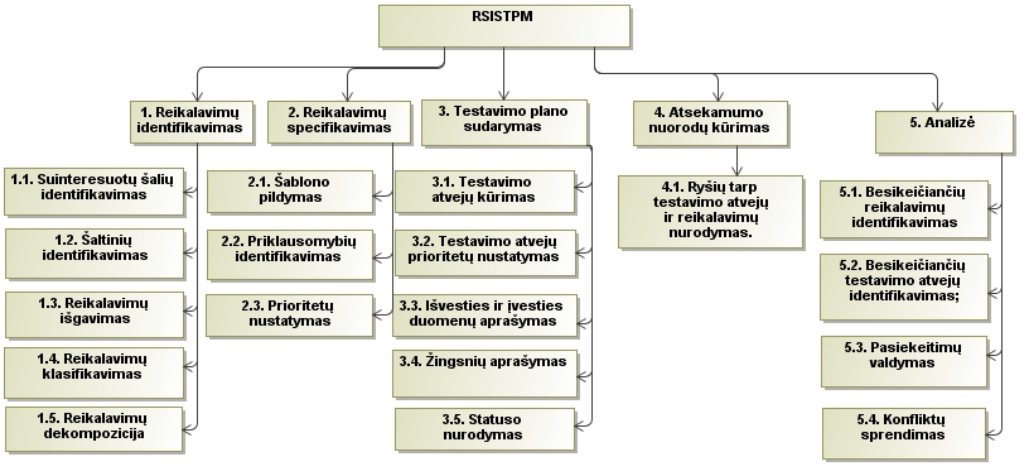
3.1. Reikalavimų specifikacijos integralumą su testavimo procesu užtikrinančio metodo procesų modelis

Reikalavimų sistemai specifikacijos integralumą su testavimo procesu (RSISTPM) užtikrinantis modelis pavaizduotas 15 pav. Modelyje kiekviena veikla turi hierarchinę tvarką.

Integralumą užtikrinantį modelį sudaro keturi pagrindiniai uždaviniai, kurie suskirstyti į smulkesnius:

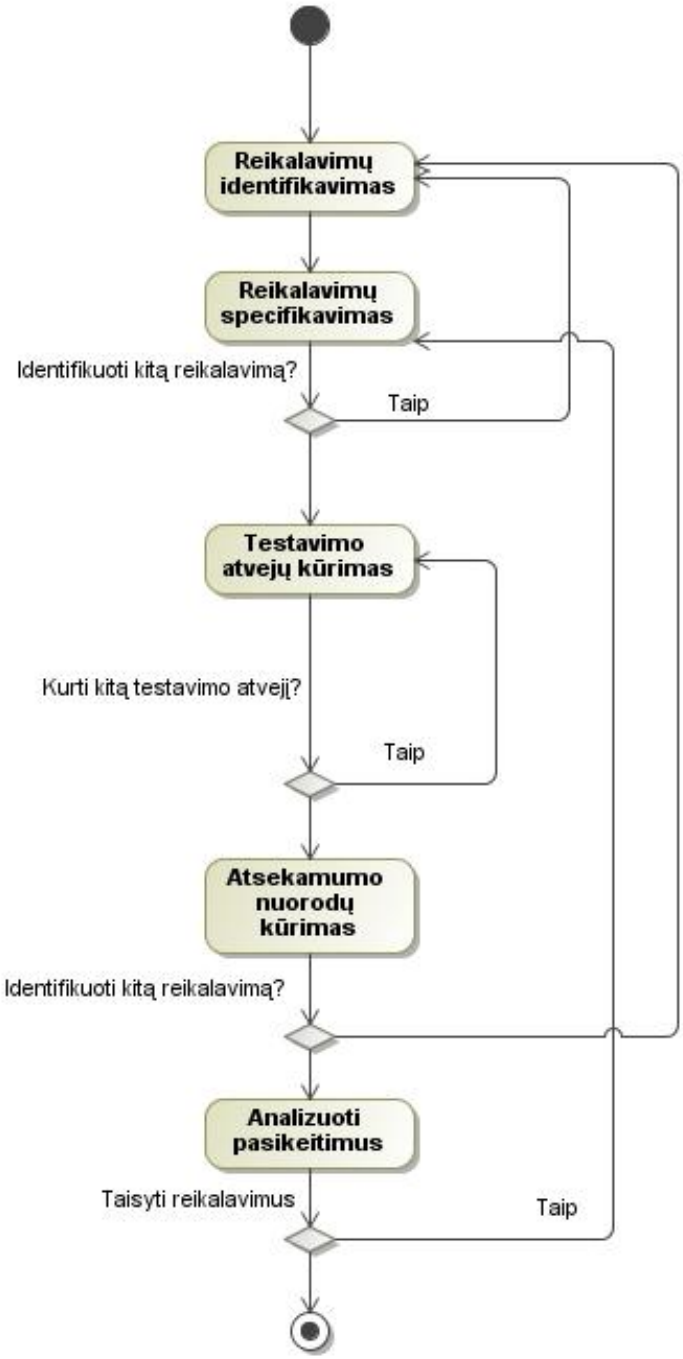
1. Reikalavimų identifikavimas;
   1.1. Suinteresuotų šalių identifikavimas;
   1.2. Šaltinių identifikavimas;
   1.3. Reikalavimų išgavimas;
   1.4. Reikalavimų klasifikavimas;
   1.5. Reikalavimų dekompozicija.
2. Reikalavimų specifikavimas;
   2.1. Šablono pildymas;
   2.2. Priklausomybių identifikavimas;
   2.3. Prioritetų nustatymas.
3. Testavimo plano sudarymas;
   3.1. Testavimo atvejų kūrimas;
   3.2. Testavimo atvejo prioriteto nustatymas;
   3.3. Išvesties ir įvesties duomenų aprašymas;
   3.4. Žingsnių aprašymas;
   3.5. Statuso nurodymas.
4. Atsekamumo nuorodų kūrimas;
   4.1. Ryšių tarp testavimo atvejų ir reikalavimų nurodymas;
5. Analizė;
   5.1.Besikeičiančių reikalavimų identifikavimas;
   5.2. Besikeičiančių testavimo atvejų identifikavimas;
   5.3. Pasikeitimų valdymas;

5.4. Konfliktų sprendimas.



15 pav. Reikalavimų specifikacijos integralumo su testavimo procesu modelis

Reikalavimų specifikacijos integralumą su testavimo procesu užtikrinantis procesas atvaizduotas UML veiklos diagramų pagalba. 16 pav. Pateikta veiklos diagrama vaizduojanti atliekamas užduotis. Kiekviena tokia užduotis modeliuojama kaip veiklos būsena, o sprendimo taškai naudojami, kai yra pasirinkimo galimybė. Kai identifikuojamas naujas reikalavimas, toliau kartojamas reikalavimų identifikavimo procesas arba kuriamos veiklos diagramos, o iš jų testavimo atvejai. Sukūrus testavimo atvejį, toliau kartojamas testavimo atvejo kūrimo procesas arba kuriamos nuorodos tarp reikalavimų ir testavimo atvejų. Procesas baigiamas, kai visi reikalavimai įrašomi į šabloną ir turi sukurtas nuorodas su testavimo atvejais.



16 pav. Pagrindinių užduočių veiklos diagrama

3.1.1. *Reikalavimų identifikavimo procesas*

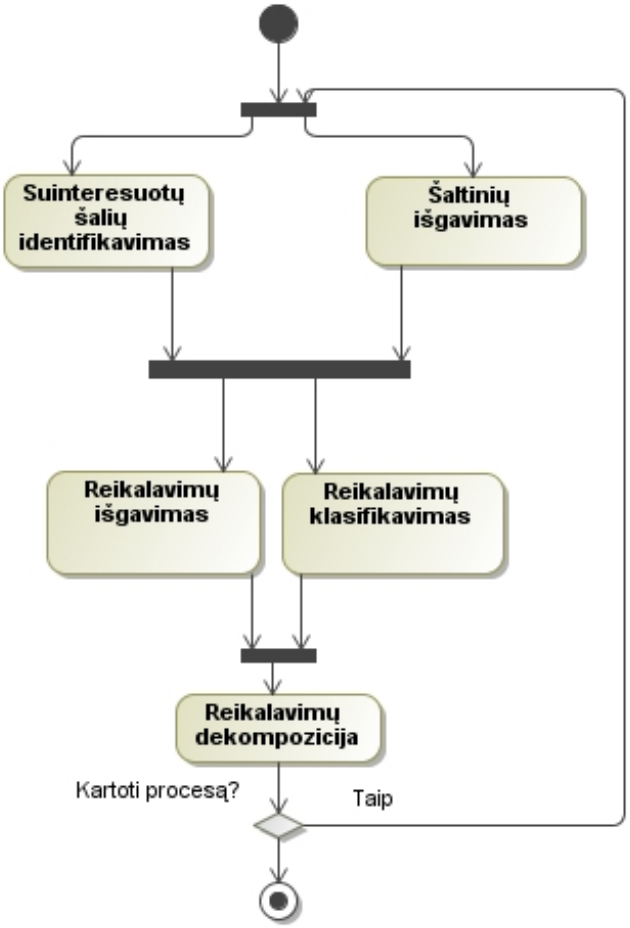
Reikalavimų identifikavimo proceso tikslas yra nustatyti kuriamos sistemos reikalavimus. Kiekvienas reikalavimas apibūdina sistemą, kokia ji turi būti, ką ji turi atlikti, ir nusako jai taikomus apribojimus.

Reikalavimų identifikavimo procesas susideda iš šių uždavinių:

1. Suinteresuotų šalių identifikavimas;
2. Šaltinių identifikavimas;

3. Reikalavimų išgavimas;
4. Reikalavimų klasifikavimas;
5. Reikalavimų dekompozicija.

17 pav. pavaizduotas procesas naudojamas reikalavimų identifikavimo proceso uždaviniams atlikti.



17 pav. Reikalavimų identifikavimo proceso veiklos diagrama

Kai kuriuos uždavinius galima atlikti sinchroniškai, t.y. tuo pačiu metu, kai kuriuos palaipsniui vieną po kito.

Reikalavimų surinkimas – sudėtingas procesas, nuo kurio priklauso viso projekto sėkmė. Jo metu būtina identifikuoti reikalavimų šaltinius ir iš jų išgauti reikalavimus.

Reikalavimui suteikiamas identifikacijos numeris. Jis yra unikalus ir suteikiamas automatiškai. Tada sukuriamas pavadinimas, kuris turi atspindėti specifikuojamo reikalavimo esmę ir kam jis skirtas. Kitus uždavinius galima atlikti sinchroniškai: reikalavimo aprašymas, reikalavimo pagrindimas, panaudojimo atvejo priskyrimas, reikalavimo šaltinio nurodymas. Kai užpildomi šie laukai, tada pereinama prie kirų dviejų svarbių uždavinių: tinkamumo kriterijaus ir reikalavimo prioriteto nustatymas.

Pirmas žingsnis – suinteresuotų šalių identifikavimas. Vartotojų reikalavimų identifikavime yra surinkti pagrindinę informaciją apie vartotojus, suinteresuotas šalis [16]. Suinteresuotų šalių analizė identifikuoja visus vartotojus ir suinteresuotas šalis, kurios turės įtakos ar bus paveiktos sistemos. Tai padeda užtikrinti, kad visų jų poreikiai bus užfiksuoti.

Vartotojų grupės gali apimti galutinius vartotojus, prižiūrėtojus, diegėjus. Kitos suinteresuotos šalys apima užsakovus, sistemos naudotojus, marketingo personalą, pirkėjus ir pagalbinį personalą. Suinteresuotų šalių analizė kiekvienai vartotojų ir suinteresuotų šalių grupei identifikuoja jų pagrindinius vaidmenis, atsakomybes ir užsibrėžtus tikslus susijusius su sistema. Vienas iš pagrindinių klausimų yra kaip kartu susieti skirtingų suinteresuotų šalių konkuruojančius poreikius naujoje sistemoje [16].

Iš identifikuotų suinteresuotų šalių sąrašo, pasirenkamas atstovaujantis asmuo ar grupė su kuriais galima išsiaiškinti reikalavimus, spręsti konfliktus.

Antras žingsnis - šaltinių išgavimas. Šios užduoties tikslas yra surinkti visus esančius dokumentus, kurie gali padėti identifikuoti reikalavimus. Tokie dokumentai gali būti standartai, organizacijos diagramos, proceso modeliais, katalonai ir t.t.

Siekiant išsiaiškinti reikalavimus sistemai, analizuojami dokumentai, kurie surinkti šaltinių identifikavimo etape, įtraukiant suinteresuotų šalių interviu užrašus, sistemos srities supratimui.

Ieškomi žodžiai ir sakiniai, kurie išreiškia savybes ir užduotis, kurias programa turi vykdyti.

Trečias žingsnis - reikalavimų išgavimas. Reikalavimų išgavimo technika priklauso nuo laiko ir turimų resursų ir, žinoma, nuo informacijos rūšies. Reikalavimų išgavimo būdai aptarti (1.5.1 skyriuje).

Siekiant išgauti reikalavimus, analizuojami dokumentai, kurie buvo surinkti šaltinių identifikavimo procese, įtraukiant ir suinteresuotų šalių interviu įrašus.

Ketvirtas žingsnis - reikalavimų klasifikavimas. Reikalavimų klasifikavimas yra svarbus reikalavimų specifikavimo procese. Reikalavimų klasifikavimą pasiūlė Sommerville [1]. Atsižvelgiant į tai, metode apibrėžiami du reikalavimo tipai:

- Funkciniai reikalavimai;
- Nefunkciniai reikalavimai.

Penktas žingsnis – reikalavimų dekompozicija. Kad būtų galima užtikrinti reikalavimų integralumą su testavimo procesu, būtina reikalavimus išskaidyti į detalesnius reikalavimus.

10 lenetelė Reikalavimų specifikavimo šablonas

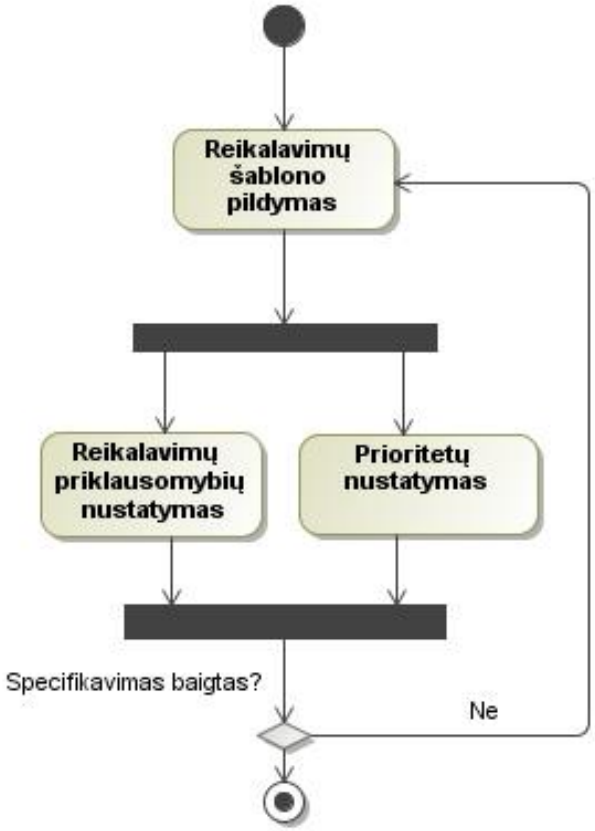
Reikalavimų specifikavimo šablonas		
Reikalavimo nr.	Pavadinimas:	Statusas:
Reikalavimo klasifikavimas:		Panaudojimo atvejis:
Aprašas:		
Pagrindimas:		
Šaltinis:		
Tinkamumo kriterijus:		
Prioritetas:		
Priklausomybės:		Konfliktai:
Istorija:		Papildoma medžiaga:

3.1.2. *Reikalavimų specifikavimo procesas*

Reikalavimų specifikavimo procesas susideda iš:

- Reikalavimų šablono pildymo;
- Priklausomybių nustatymo;
- Prioritetų nustatymo.

18 pav. pavaizduotas procesas naudojamas reikalavimų specifikavimo proceso uždaviniams atlikti.



18 pav. Reikalavimų specifikavimo procesas

Pirmas žingsnis – reikalavimų šablono pildymas. Reikalavimų specifikavimui buvo sudarytas reikalavimų specifikavimo šablonas. Atlikus šablonų analizę, buvo nuspręsta naudoti Volere šabloną, prie jo pridedant papildomus laukus. Šablonas sudarytas atsižvelgiant į reikalavimų sistemai integralumą su testavimo procesu užtikrinantį modelį.

11 lenetelė Specifikavimo šablono laukų paaiškinimas

Šablono laukas	**Aprašymas**
Reikalavimo nr.	Reikalavimo numeris turi būti unikalus. Jis priskiriamas automatiškai.
Pavadinimas	Pavadinimas duoda trumpą aprašą, kuris yra naudingas greitai visų reikalavimų peržiūrai
Statusas	Statuso nurodo reikalavimo išbaigimo stadiją.
Reikalavimo klasifikavimas	Funkciniai, nefunkciniai reikalavimai.
Panaudojimo atvejis	Reikalavimas susiejamas su panaudojimo atvejis, nurodant jų numerius.
Aprašas	Skirtas reikalavimo paskirčiai apibrėžti. Tai tekstinis reikalavimo apibrėžimas. Jame turi atsispindėti užsakovo arba vartotojo

	pageidavimai.
Pagrindimas	Detalesnis reikalavimo aprašymas
Šaltinis	Informacijos šaltinis, pavyzdžiui suinteresuota šalis, dokumentas, sistemos informacija, katalogas, verslo procesai ar standartai.
Tinkamumo kriterijus	Tikslas, kurį turi tenkinti sistema. Nors reikalavimas aprašomas vartotojo sąvokomis, tačiau tinkamumo kriterijus rašomas tiksliai nusakant jo išmatuotą kriterijų, kad priimtus sprendimus būtų galima testuoti reikalavimo tenkinimo aspektu.
Prioritetas	Reikalavimo svarbumas. Kiekvienam reikalavimui priskiriamas įvertinimas priklausomai nuo jo svarbumo. Pirmiausia įgyvendinami didžiausią prioritetą turintys reikalavimai. Prioriteto vertinimas atspindi, kiek bus patenkintas užsakovas, jei reikalavimas bus sėkmingai įvertintas. Vertinimui galima naudoti skalę nuo 1 iki 10. 1 reiškia minimalų prioritetą, jei reikalavimas bus įgyvendintas, o 10 reiškia maksimalų vartotojo patenkinimą reikalavimo įvertinimu.
Priklausomybės	Priklausomybės tai kiti reikalavimai, turintys įtaką nagrinėjamam reikalavimui. Pavyzdžiui, jei vienas reikalavimas keičiasi, tai turi keistis ir kitas. Arba vieno reikalavimo duomenys betarpiškai siejasi su kito reikalavimo duomenimis. Gali būti, kad vienas reikalavimas negali egzistuoti be kito reikalavimo įvertinimo.
Konfliktai	Tai reikalavimai, kurie prieštarauja nagrinėjamam reikalavimui.
Istorija	Čia užregistruojama, kada reikalavimas pirmą kartą buvo iškeltas, kada pakeistas

	pašalintas ar patikrintas.
Papildoma medžiaga	Jei yra kas nors papildomai aprašančio reikalavimą, tuomet gali būti panaudota nuoroda į šią medžiagą.

Antras žingsnis – priklausomybių nustatymas. Priklausomybės tai ryšiai tarp reikalavimų, parodantys, kad vienas reikalavimas yra priklausomas nuo kito. Jei reikalavimas neturi jokios priklausomybės, reiškia jis buvo neteisingai identifikuotas ir neturi jokios prasmės, todėl būtina pakartoti reikalavimų identifikavimo proceso žingsnius, kad būtų tiksliau nustatytas reikalavimas.

Priklausomybės įrašomos naudojamo šablono priklausomybių eilutėje.

Trečias žingsnis – prioritetų nustatymas. Reikalavimams turi būti suteikiamas prioriteto įvertinimas priklausomai nuo jų svarbumo. Skirtingos suinteresuotos šalys turi skirtingus poreikius ir supratimą, kas yra svarbiausia. Pvz.: vieni nori, kad visos sistemos funkcijos būtų įgyvendintos, kitiems svarbiau projekto kaina ir kad tai kuo mažiau kainuotų. Šie poreikiai dažnai konfliktuoja vienas su kitu. Todėl būtina efektyviai prioritetizuoti suinteresuotų šalių poreikius.

Reikalavimo prioriteto suteikimo fazėje suinteresuotos šalys vertina reikalavimus pagal tris kriterijus. Per reikalavimų prioritezavimą visi funkciniai reikalavimai identifikuoti reikalavimų identifikavimo fazėje yra kategorizuojami remiantis trimis parametrais. Šie parametrai yra: vertė, rizika ir pastangos.

Reikalavimo vertė nusako reikalavimo svarbumą suinteresuotoms šalims. Ši vertė matuojama trijų matų skalėje (aukšta „+“, vidutinė „0“, žema „-“). Šio kategorizavimo priežastis yra ta, kad aukštos vertės reikalavimai turėtu būti atsekami detaliau negu vidutinės ar žemos vertės reikalavimai, nes jie pristato sistemos funkcionalumą [55].

Rizikos įvertinimas pristato reikalavimo galimą kintamumą išmatuotą trijų matų skalėje (aukštas „+“,vidutinis „0“, žemas „-“). Rizikingesni reikalavimai yra linkę keistis ir per procesą jiems gali reikėti ne vieno tikslinimo. Svarbu atpažinti pasikeitimo įtaką sistemos dizainui ir kitiems sistemos artefaktams. Verta sekti aukštos rizikos reikalavimus, nes pasikeitimo įtakos analizėje, šie ryšiai tarp aukštos vertės reikalavimų yra reikalingi dažniau nei ryšiai su stabiliais reikalavimais [55].

Pastangų įvertinimas pristato numatomą apytikslį laiką ir žmogiškuosius išteklius skiriamus išpildyti reikalavimą. Pvz. yra „n“ skaičius artefaktų. Norint valdyti visus atsekamumą tarp visų „n“ artefaktų, reikalavimų atsekamumo sudėtingumas yra „n^2“. Pastangų kiekis didėja, kai didėja reikalavimų skaičius. Tai yra viena pagrindinių problemų susijusių su reikalavimų

atsekamumu. Todėl būtina prioritezuoti skiriamas pastangas naudojant trijų matų skalę (aukštas „+“,vidutinis „0“, žemas „-“).

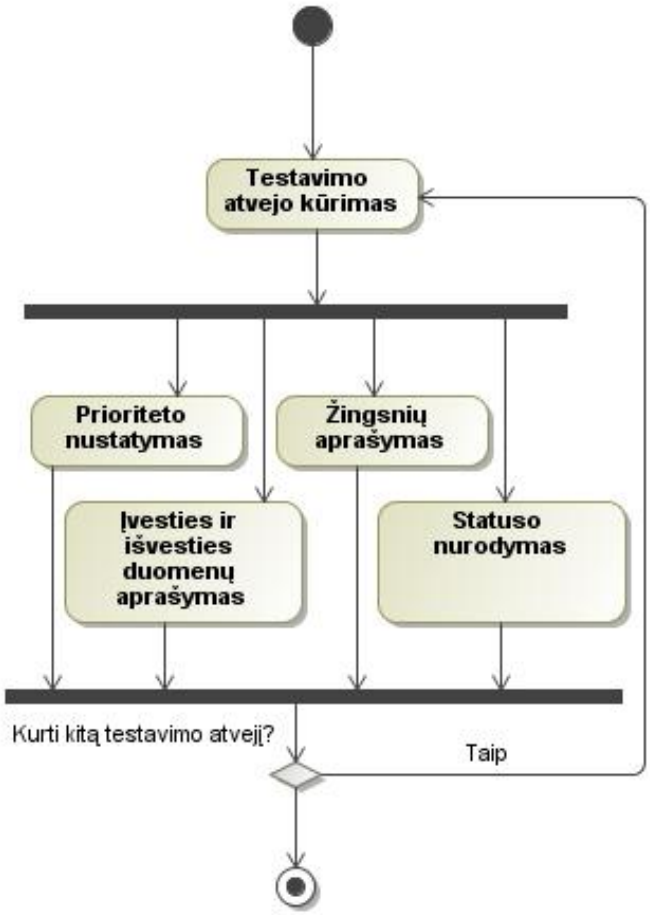
Tokie prioritezavimo žingsniai yra labai svarbūs žingsniai, nes jie padeda nustatyti reikalavimų svarbumo lygį. Šių žingsnių išvestis yra trys svarbumo lygiai, kaip parodyta 12 lentelė Reikalavimų svarbumo vertinimo lentelėReikalavimai, kurių svarbumo lygis yra 1, gali būti atsekami detaliau nei reikalavimai, kurių svarbumo lygis 2 ir svarbumo lygis 3. Šie žingsniai yra skirti identifikuoti, kurie reikalavimai yra vertingesni už kitus. Šių prioritezavimo žingsniu pagalba pastangos, susijusios su reikalavimų atsekamumu, sumažinamos. Pastangos sumažinamos, nes reikalavimus, kurių lygis 1, galima atsekti detaliau nei kitus.

12 lentelė Reikalavimų svarbumo vertinimo lentelė

Reikalavimo ID	Reikalavimas	Vertė (V)	Rizika (R)	Pastangos (P)
1				
2				
...				

3.1.3. *Testavimo plano sudarymo veiklos procesas*

Testavimo atvejų specifikavimui naudojame šabloną. Šablonas susideda iš rinkinio elementų, kurie užpildomi kiekvienam testavimo atvejui. Testavimo atvejų šablonas pasirinktas šiam metodui yra IEEE 829-1998 testavimo atvejų specifikacijos standartas [9]. Jis buvo pritaikytas siūlomam metodui, papildžius jį trūkstamais laukais.



19 pav. Testavimo atvejo kūrimo veiklos diagrama

13 lenetelė Testavimo atvejo šablonas

Testavimo atvejo šablonas	
Testavimo atvejo ID	Testavimo atvejo unikalus identifikatorius
Testavimo atvejo pavadinimas	Aiškus, trumpas testavimo atvejo pavadinimas
Testavimo atvejo paskirtis	Testavimo atvejo paskirties aprašas
Kliento reikalavimas	Sistemos pasirinktos testuoti šio testavimo atvejo pagalba elementų ir savybių aprašas
Išankstinė sąlyga	Prielaida ar sąlygos, kurias reikia atitiktį prieš testo vykdymą.
Įvestis	Sąrašas įvesties duomenų reikalingų testavimo atvejų vykdymui. Jis gali apimti kintamas reikšmes, failus ir t.t.
Žingsniai	Sąrašas žingsnių, kuriuos reikia atlikti norint įvykdyti testą.
Rezultatai, kurių tikimasi	Rezultatai, kurių tikimasi iš testavimo atvejų vykdymo.
Atsekamumo nuoroda	Nuoroda į reikalavimą ar reikalavimus, kuriems testuoti skirtas šis testavimo atvejis
Statusas	Testavimo atvejo statusas

Identifikacija, testavimo atvejo tikslas ir testuojamas elementas yra pagrindinės testavimo atvejo dalys. Šie laukai turi būti užpildyti kiekvienam testavimo atvejui. Tai padės valdant testavimo atvejų saugyklą. Testavimo atvejo tikslas yra būtinas, nes vėliau jis padeda sužinoti buvo testo tikslai pasiekti ar ne. Be to, labai svarbu identifikuoti testavimo atvejo išankstines sąlygas, nes jos apibūdina prielaidas ir sąlygas reikalaujamas prieš pradedant testavimo atvejo vykdymą. Įvestis yra būtina testo dalis, nes ji nurodo įvestį reikalingą testavimo atvejo vykdymui. Kad būtų vykdomas testavimo atvejis, būtina nurodyti žingsnius reikalaujamus testavimo atvejo vykdymui. Testas „išlaikytas“ arba „neišlaikytas“, kai esami rezultatai palyginami su numatytais rezultatais. Testų tikslas yra validuoti ar sistema atitinka savo specifikaciją, ar ne, tai analizuojama su esamų ir numatytų rezultatų pagalba.

3.1.4. *Atsekamumo nuorodų kūrimas*

Nuorodų tarp reikalavimų kūrime atsekamumo nuorodos yra kuriamos tarp reikalavimų ir kitų programinės įrangos artefaktų. Šie programinės įrangos artefaktai gali būti dizaino dokumentai, kiti reikalavimai ar testavimo atvejai. Aukšto svarbumo lygio reikalavimai gali būti atsekami detaliau lyginant su vidutinio svarbumo lygio ar žemo svarbumo lygio reikalavimais.

3.1.5. *Analizė*

Evoliucijos fazėje visas reikalavimų proceso atsekamumas gali būti analizuojamas. Šios analizės pagalba galima priimti įvairius sprendimus. Pvz.: sukurtų atsekamumo nuorodų pagalba galima matyti padaryto pasikeitimo įtaką tam tikram reikalavimui. Jei dėl kokios nors priežasties reikalavimas pasikeičia, tada su atsekamumo pagalba galima identifikuoti kitus elementus, kurie yra paveikti šio pasikeitimo.

Atsekamumo analizę sudaro:

- Besikeičiančių reikalavimų identifikavimas;
- Besikeičiančių testavimo atvejų identifikavimas;
- Pasikeitimų valdymas;
- Konfliktų sprendimas.

20 pav. Analizės kūrimo veiklos diagrama

4. METODO REALIZACIJA PASIRINKTOMIS PRIEMONĖMIS

Šiame skyriuje pateikiamas metodo realizavimas pasirinktomis ir sukurtomis priemonėmis. Reikalavimų valdymui ir specifikavimui naudojamas sukurtas šablonas RequisitePro aplinkoje. Ryšiams tarp reikalavimų atvaizduoti naudojamos atributų ir susietumo matricos, susietumo medžiai. Filtrų pagalba filtruojame informaciją pagal reikalavimų charakteristikas.

4.1. Dalykinės srities aprašymas

Paskaitų sistema bus naudojama universitete, kur reikia teikti informaciją apie paskaitas, jų tvarkaraščius, paskaitų naujienas, paskaitų medžiagą, ir paskirstyti dokumentus bei kitus failus. Sistema taip pat turi leisti su paskaitų dalyviais susijusį valdymą. Funkcijos, kurias produktas turi turėti yra: teikti informaciją apie paskaitas, teikti informaciją apie paskaitų naujienas, teikti paskaitų archyvo dokumentus, leisti valdyti asmeninį profilį ir visus sistemos dalyvius. Sistema turi leisti tik autorizuotus prisijungimus.

4.2. Reikalavimų identifikavimo etapas

14 lenetelė Pasirinkti reikalavimai

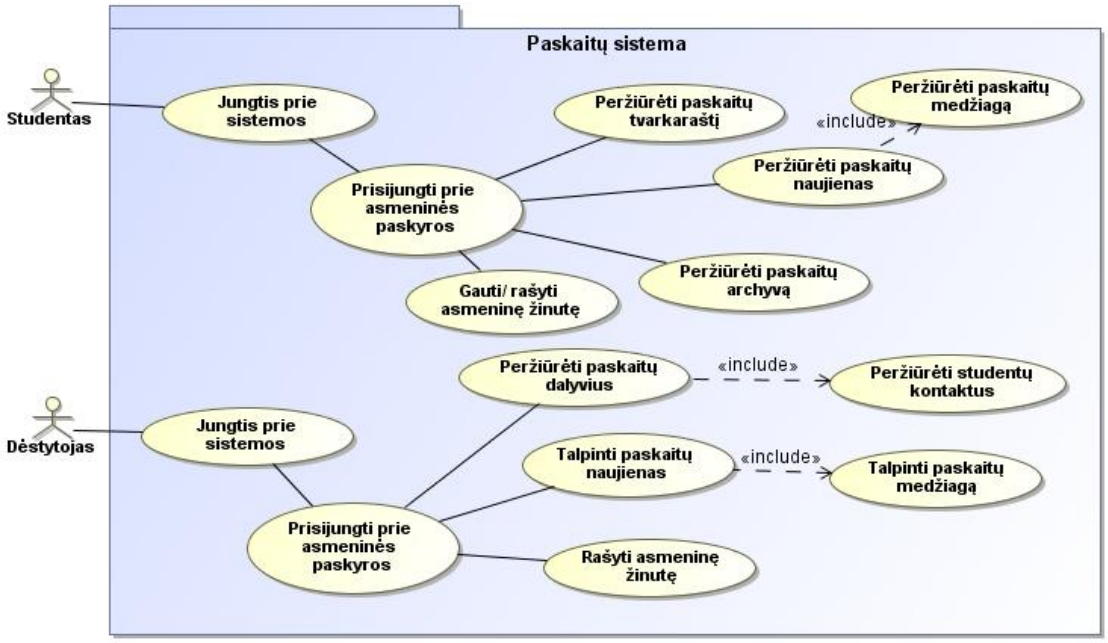
Reikalavimo nr.	Pavadinimas	Aprašas	Paaiškinimas	Apribojimas
1	Teikti informaciją apie paskaitas	Produktas turi teikti informaciją apie paskaitas prisijungusiems vartotojams		
2	Saugoti produktą	Produktas turi išvengti neautorizuotų naudojimų.	Vartotojai gali atlikti tik numatytas funkcionalumą. Vartotojas turi būti prisiregistravęs, prieš gaudamas priėjimą prie sistemos.	
3	Produkto vartotojo sąsaja	Visi prisijungimai prie sistemos turi vykti per paties vartotojo sąsają.		

4	Paskaitų medžiagos archyvas	Kursas turi turėti paskaitų medžiagos archyvą	Galimybę teikti medžiagą kurso dalyviams	Failai gali būti dideli, o tai reikalauja išteklių, kurie gali būti nepakankami
5	Paskaitų naujienos	Turi būti galimybė pasiekti paskaitų naujienas	Studentus informuoti apie paskaitas	
6	Asmeninis profilis	Vartotojas sistemoje turi turėti asmeninį profilį	Leisti kitiems vartotojams rasti kontaktinę informaciją. Palengvinti sąveiką tarp vartotojų	
7	Prisijungimas	Vartotojas privalo prisijungti prieš produkto naudojimą	Neautorizuoti vartotojai neturi turėti priėjimo prie produkto funkcionalumo	Vartotojas gali pamiršti slaptažodį
8	Paskaitų pradinis langas	Paskaitų pradinis puslapis turi turėti: paskaitų naujienas, nuorodą į paskaitų dalyvius, nuorodą į paskaitų medžiagą		
9	Pridėti ir ištrinti paskaitų dalyvius	Tik sistemos administratorius turi turėti galimybę pridėti ar pašalinti dalyvius		
10	Neteisingas slaptažodis	Jeigu vartotojas suveda neteisingą prisijungimo vardą ir/arba slaptažodį, prisijungimo langas turi parodyti pranešimą apie	Atgalinė nuoroda su vartotoju	Vartotojas gali nesuprasti informacijos

		klaidingai suvestus duomenis.		
11	Atsakymas į žinutę	Kai peržiūrima žinutė, turi būti įmanoma atsakyti tiesiai į žinutę. Tai prideda naują žinutę su tuo pačiu pavadinimu		
12	Sėkmingas prisijungimas	Kai vartotojas sėkmingai prisijungia, produktas turi rodyti vartotojo asmeninį pradinį puslapį		

Remiantis pateiktu reikalavimų specifikacijos integralumą su testavimo procesu užtikrinančiu metodu, pirmas žingsnis yra reikalavimų identifikavimo etapas. Reikalavimų identifikavimą galima išskirstyti į šiuos žingsnius:

1. Remiantis dalykinės srities aprašu identifikuojame suinteresuotas šalys, kurie padės reikalavimų išgavimo procese. Paskaitų sistemos suinteresuotos šalys:
- Dėstytojai;
- Studentai.
2. Kitas žingsnis yra šaltinių identifikavimas. Tai gali būti proceso modeliai, diagramos, standartai, kurie padėtų identifikuoti reikalavimus.
3. Šiame žingsnyje stengiamasi išgauti reikalavimus būdais, aprašytomis 1.4 skyriuje, iš jau išsiaiškintų suinteresuotų šalių. Kad būtų susidarytas bendras vaizdas, sudarome panaudojimo atvejų diagramą. 21 pav. pateikta paskaitų sistemos panaudojimo atvejų diagrama.



21 pav. Sistemos vartotojų panaudojimo atvejai

4. Ketvirtame žingsnyje įvedami reikalavimų atributai ir jų reikšmės. Atributų reikšmės neprivalomos.
5. Įvedamos reikalavimo priklausomybės su kitais reikalavimais.

4.3. Testavimo atvejų kūrimas

Testavimo atvejai bus kuriami iš reikalavimų duotų *lentelėje 7.1*. Testavimo atvejų kūrimasis reikalavimų pagrindu susideda iš dviejų žingsnių. Pirmiausia, informacijos testavimo atvejams išgavimas iš duotų reikalavimų aprašo. Jei testavimo atvejams trūksta informacijos, tada analizuojami visi susiję reikalavimai reikalavimų dokumente, kad būtų gauta daugiau informacijos ir testavimo atvejis būtų detalesnis. Tokiu būdu testavimo atvejai bus sukuriami iš kiekvieno pasirinkto reikalavimo. Šie testavimo atvejai bus aptarti ir jiems suteikiamas tinkamas statusas.

Testavimo atvejo ID	TA1
Testavimo atvejo pavadinimas	Ar pasiekiama paskaitos informacija?
Testavimo atvejo statusas	1
Testavimo atvejo paskirtis	Testuoja ar prisiregistravęs vartotojas gali pasiekti kurso informaciją.
Testuojamas elementas	R1
Išankstinė sąlyga	Vartotojas prisijungęs
Įvestis	Pakaita (id arba pavadinimas)

Žingsniai	-
Numatomi rezultatai	-

Lentelė vaizduoja testavimo atvejį sukurtą iš produkto lygio reikalavimų, iš reikalavimo 1. Šis testavimo atvejis validuoja, kad registruotas vartotojas gali peržiūrėti informaciją apie paskaitą. Testavimo atvejo šablonas susideda iš informacijos gautos iš reikalavimų aprašo duoto šablone.

Turint tik tiek informacijos testavimo atvejui, testui trūksta kritinės informacijos. Testavimo atvejis nenurodo kokia informacija apie paskaitą bus testuojama. Tai dėl informacijos, duotos iš šio reikalavimo, trūkumo. Reikalavimo aprašas nenurodo, kokia informacija bus pasiekiama vartotojams per sistemą. Dėl informacijos trūkumo neįmanoma identifikuoti numanomų rezultatų. Be to, sunku identifikuoti žingsnius reikalaujamus šio testavimo atvejo vykdymui, nes nėra aišku kokią informaciją vartotojas gali pasiekti. Iš aprašo galime suprasti, kad vartotojai turi būti prisiregistravę prie sistemos ir vartotojai gali rasti informaciją apie paskaitą. Bet su tiek informacijos neįmanoma vykdyti testo, nes jam trūksta numanomų rezultatų ir galimų žingsnių reikalaujamų šio testavimo atvejo vykdymui.

Neįmanoma gauti daugiau informacijos iš kitų reikalavimų. Šis testavimo atvejis negali būti laikomas, kaip užbaigtas testavimo atvejis, bet jis turi testavimo tikslą ir truputi kitokios informacijos, todėl šiam testavimo atvejui suteikiamas statusas 2.

Testavimo atvejo ID	TA2
Testavimo atvejo pavadinimas	Testuoti, kad neprisijungę vartotojai gali prisijungti prie sistemos
Testavimo atvejo statusas	2
Testavimo atvejo paskirtis	Testuoti, kad neprisiregistravęs vartotojas negali pasiekti sistemos funkcionalumo
Testuojamas elementas	R2
Išankstinė sąlyga	Vartotojas neregistruotas
Įvestis	Sistemos URL
Žingsniai	1. Atidaryti interneto naršyklę; 2. Suvesti sistemos URL; 3. Paspausti Enter.
Numatomi rezultatai	-

Testavimo atvejis buvo sukurtas iš reikalavimo, kurio ID - R2. Informacija gauta iš šio reikalavimo yra labai abstrakti. Iš aprašo ir paskirties tik testo tikslas ir viena išankstinė sąlyga

žinomi. Iš paaiškinimo lauko galima sužinoti, kad vartotojas turi būti neprisiregistravęs, kad atitiktų tikslą. Testavimo atvejo paskirtis ir viena išankstinė sąlyga neduoda jokios informacijos kitiems testavimo atvejo laukams. Nebus įmanoma vykdyti šio testavimo atvejo su tiek mažai informacijos. Numanomi rezultatai neprieinami šiam testavimo atvejui. Nežinant šios informacijos, bus sunku nuspręst testas išlaikė ar ne. Kadangi šis testavimo atvejis turi šiokios tokios informacijos, jo statusas 2.

Testavimo atvejo ID	TA3
Testavimo atvejo pavadinimas	Testuoti ar sistema gali būti pasiekta ne per vartotojo sąsają
Testavimo atvejo statusas	1
Testavimo atvejo paskirtis	Testuoja ar sistema gali būti pasiekta ne per vartotojo sąsają
Testuojamas elementas	R3
Išankstinė sąlyga	-
Įvestis	-
Žingsniai	-
Numatomi rezultatai	-

TA3 iš vartotojo sąsajos reikalavimo. Šis testavimo atvejis turi užpildytą tik tikslo laukelį. Duoto reikalavimo aprašas labai dviprasmiškas, jis neduoda aiškaus supratimo per kokius kitus produktus galima bandyti pasiekti sistemą. Su tokia neišbaigta informacija TA3 statusas yra 1.

Testavimo atvejo ID	TA10
Testavimo atvejo pavadinimas	Neteisingas slaptažodis
Testavimo atvejo statusas	5
Testavimo atvejo paskirtis	Suveda neteisingą prisijungimo vardą ir/arba slaptažodį, prisijungimo langas turi parodyti pranešimą apie klaidingai suvestus duomenis.
Testuojamas elementas	R10
Išankstinė sąlyga	Vartotojas neprisijungęs
Įvestis	Vartotojo ID = neegzistuojantis vartotojas; Slaptažodis = neteisingas slaptažodis.
Žingsniai	1. Atidaryti sistemos prisijungimą; 2. Įvesti vartotojo prisijungimą; 3. Įvesti slaptažodį; 4. Spausti mygtuką „prisijungti“.

Numatomi rezultatai	Vartotojas turi matyti žinutę rodančią, kad „jūs suvedėte neteisingus prisijungimo duomenis ir/arba slaptažodį“.

Testavimo atvejis TA10 gautas iš reikalavimo R10 duoto lentelėje. Šis testavimo atvejis turi geras detales visuose laukuose, ir beveik visa reikiama informacija buvo gauta tiesiai iš reikalavimo R10. Kadangi testavimo atvejis turi aiškius tikslus, išankstinę sąlygą, žingsnius ir aiškius numatomus rezultatus, todėl jis gali būti laikomas vykdomu testu su beveik visa reikiama informacija gauta iš vieno reikalavimo. Su tokiu detalumo lygiu, jo statusas 5.

Testavimo atvejo ID	TA11
Testavimo atvejo pavadinimas	Atsakymas į žinutę
Testavimo atvejo statusas	3
Testavimo atvejo paskirtis	Kai peržiūrima žinutė, turi būti įmanoma atsakyti tiesiai į žinutę. Tai prideda naują žinutę su tuo pačiu pavadinimu
Testuojamas elementas	R11
Išankstinė sąlyga	Turi egzistuoti asmeninė pašto dėžutė, egzistuoti žinutė, vartotojas prisiregistravęs, vartotojas pradiniame kurso puslapyje
Įvestis	Žinutė
Žingsniai	1. Atidaryti asmenines žinutes; 2. Atidaryti žinutę; 3. Spausti „atsakyti“.
Numatomi rezultatai	Vartotojas mato asmeninį pirmą puslapį.

Testavimo atvejis TA10 sukurtas iš reikalavimo R10.I6. Iš duoto aprašo įmanoma gauti testo tikslą, išankstinę sąlygą, įvesties duomenis. Reikalavimas neduoda jokios informacijos apie kitus laukus. Žiūrint į kitus reikalavimus, įmanoma gauti truputi kitos informacijos. Reikalavimai „Kurso pradinis puslapis“ gali padėti identifikuoti veiksmus reikalaujamus norint atsakyti į žinutę. Kadangi šis testavimo atvejis gauna informacijos iš kitų reikalavimų, jis yra vykdomas, todėl jo skalė gali būti įvertinta 3.

Testavimo atvejo ID	TA12
Testavimo atvejo pavadinimas	Sėkmingas prisijungimas
Testavimo atvejo statusas	5
Testavimo atvejo paskirtis	Kai vartotojas sėkmingai prisijungia, produktas turi rodyti

	vartotojo asmeninį pradinį puslapį
Testuojamas elementas	R12
Išankstinė sąlyga	Vartojas egzistuoja, vartotojas neprisijungęs
Įvestis	Vartotojo ID = egzistuojantis vartotojas; Slaptažodis = teisingas slaptažodis.
Žingsniai	4. Atidaryti sistemos prisijungimą; 5. Įvesti vartotojo prisijungimą; 6. Įvesti slaptažodį; 7. Spausti mygtuką „prisijungti“.
Numatomi rezultatai	Vartotojas mato asmeninį pirmą puslapį.

Testavimo atvejis TA12 sukurtas iš reikalavimo R12. Reikalavimo aprašas yra pakankamai aiškus, kad būtu galima gauti informaciją beveik visiems testavimo atvejo laukams. Kadangi šis testavimo atvejis gauna informacija tiesiai iš reikalavimo, taigi šis testavimo atvejis gali būti įvertintas statusu 5.

4.4. Testo atvejų statuso skalė

Testavimo atvejai buvo sukurti priklausomai iš pasirinktų reikalavimų. Po to, kai buvo padaryta analizė, graduojami 1-5 skalėje. Lentelė susideda iš testavimo atvejų ir jų statusų. Padarius analizę pagal parinktus kriterijus, šie testavimo atvejai buvo sugraduoti skalėje 1-5. Lentelėje 7.2. aprašyti testavimo atvejai ir jų statusai.

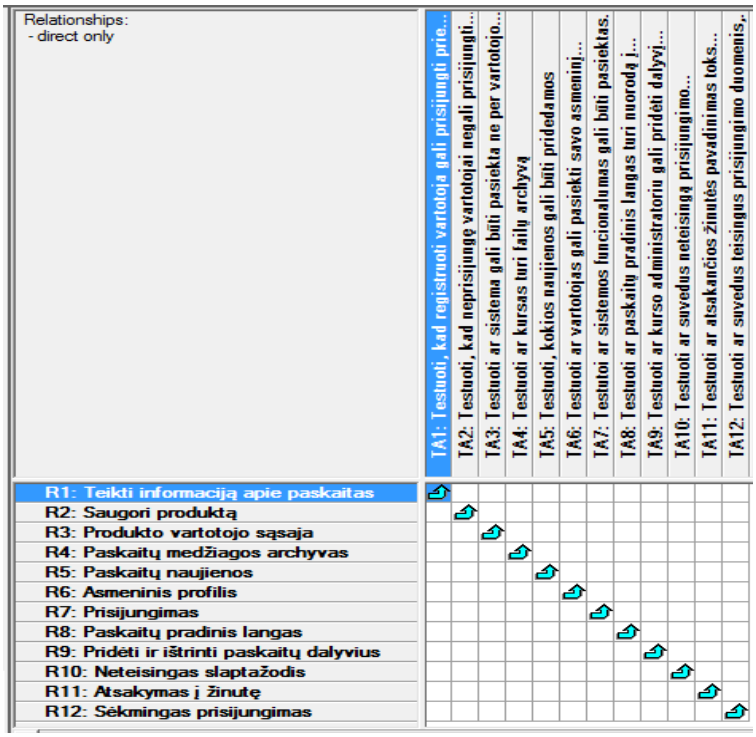
15 lentelė Testavimo atvejai ir jų statusai

Reikalavimo nr.	Testavimo atvejo nr.	Statusas
1	1	2
2	2	2
3	3	1
4	4	2
5	5	2
6	6	3
7	7	3
8	8	4

9	9	3
10	10	5
11	11	5
12	12	5

4.5. Atsekamumo nuorodų kūrimas

Reikalavimus susiejame su jiems testuoti sukurtais testavimo atvejais ir atvaizduojame susietumo matricą. Susietumo matrica pavaizduota 22 pav.



22 pav. Testavimo atvejų su reikalavimais atsekamumo nuorodos

Reikalavimų atsekamumas tikrina, kad kiekvienas reikalavimo atributas būtų padengtas mažiausiai vieno testavimo atvejo. Sukurta atsekamumo grandinė tarp reikalavimų atributų ir testavimo atvejų, sujungia reikalavimų atributus su testavimo atvejais, ir teikia būdą reikalavimų atsekamumui. Taip pat galima tikrinti ar kiekvienas testavimo atvejis padengia bent vieną reikalavimų atributą.

Reikalavimams sukurtos atributų matricos dėka galima keisti reikalavimo atributų savybes, pavyzdžiui, prioritetą, reikalavimo savybes. Atributų matrica pateikia visą išsamią informaciją apie reikalavimą (kada reikalavimas registruotas, kokiame pakete saugomas, kokius iš ir į ryšius turi su kitais reikalavimais, fiksuojamas reikalavimų pokytis, jei jis buvo ištrintas ar koreguotas). Panaudos Reikalavimų matrica pateikta 23 pav.

Requirements:	Unique ID	Location	Package	Author	Revision	Date	Reason	Traced-from	Traced-to	RootTag#
R1: Teikti informaciją apie paskaitas	1	Database	Sablono band	Diana	1.0012	2012.05.26 07	Cleared trace r		TA1	1
R2: Saugori produktą	3	Database	Sablono band	Diana	1.0006	2012.05.26 07	Deleted trace		TA2	2
R3: Produkto vartotojo sąsaja	4	Database	Sablono band	Diana	1.0004	2012.05.26 07	Deleted trace		TA3	3
R4: Paskaitų medžiagos archyvas	5	Database	Sablono band	Diana	1.0003	2012.05.26 07	Created trace		TA4	4
R5: Paskaitų naujienos	6	Database	Sablono band	Diana	1.0002	2012.05.26 07	Created trace		TA5	5
R6: Asmeninis profilis	7	Database	Sablono band	Diana	1.0001	2012.05.26 07	Created trace		TA6	6
R7: Prisijungimas	8	Database	Sablono band	Diana	1.0001	2012.05.26 07	Created trace		TA7	7
R8: Paskaitų pradinis langas	9	Database	Sablono band	Diana	1.0003	2012.05.26 07	Deleted trace		TA8	8
R9: Pridėti ir ištrinti paskaitų dalyvius	10	Database	Sablono band	Diana	1.0003	2012.05.26 07	Deleted trace		TA9	9
R10: Neteisingas slaptažodis	13	Database	Sablono band	Diana	1.0003	2012.05.26 07	Deleted trace		TA10	10
R11: Atsakymas į žinutę	14	Database	Sablono band	Diana	1.0003	2012.05.26 07	Deleted trace		TA11	11
R12: Sėkmingas prisijungimas	15	Database	Sablono band	Diana	1.0003	2012.05.26 07	Deleted trace		TA12	12
<Click here to create a requirement>	empty	Database	None	Diana	1.0000	2012.05.26 07				pending

23 pav. Atributų matrica

5. IŠVADOS

1. Atlikus reikalavimų integralumo su testavimo procesu metodų ir būdų analizę, pastebėta, kad vieningo ir nusistovėjusio metodo nėra. Pastebėta, kad moksliniuose darbuose naudojamos vis kitos priemonės ir principai, siekiant užtikrint funkcinių ir nefunkcinių reikalavimų integralumą su testavimo procesu.
2. Atlikus reikalavimų specifikavimo šablonų analizę, nustatyta, kad Volere šablonas pasižymi teigiamomis savybėmis, tokiomis kaip reikalavimų sekamumas ir susietumas geriausiai išpildo visas reikalavimų specifikavimo charakteristikas lyginant su kitais šablonais, todėl šis šablonas pasirinktas kaip pagrindas integralumą užtikrinančiam metodui. Tačiau, siekiant užtikrinti detalesnį reikalavimų integralumą, būtinas šablono išplėtimas.
3. Darbe pateiktas sudarytas šablonas reikalavimų specifikavimui. Specifikavimo etape aprašomi identifikuoti reikalavimai, naudojant sudarytą šabloną, reikalavimai specifikuojami vienodai, nepriklausomai nuo jiems būdingų savybių. Specializuotas šablonas renka atitinkamą informaciją konfliktų identifikavimui ir jų sprendimui.
4. Atlikus reikalavimų specifikavimo įrankių analizę nustatyta, kad Requisite Pro įrankis yra tinkamas sukurto metodo taikymui. Šiuo įrankiu galima sėkmingai valdyti pastovius ar nuolat besikeičiančius reikalavimus, pagerinti atsekamumą tarp reikalavimų ir su testavimo atvejais, sieti reikalavimus vienus su kitais, sumažinti projekto riziką ir pagerinti jo kokybę.
5. Darbe pateiktas metodas yra sudarytas iš šių etapų: reikalavimų identifikavimo, reikalavimų specifikavimo, testavimo plano iš reikalavimų kūrimo, atsekamumo nuorodų kūrimo, analizės. Identifikavimo etape identifikuojami reikalavimai ir renkama informacija reikalavimų pagrindimui, tokių kaip, suinteresuotų šalių ir šaltinių identifikavimas. Tai pasiekiama, analizuojant turimus dokumentus, įtraukiant esamus suinteresuotųjų šalių interviu užrašus, siekiant suprasti sistemos sritį.
6. Pasiūlytas metodas leidžia sekti reikalavimų atsekamumą, ir tikrinti, kad kiekvienas reikalavimas būtų padengtas mažiausiai vieno testavimo atvejo, arba atvirkščiai – tikrinti, kad kiekvienas testavimo atvejis padengia bent vieną reikalavimą.
7. Pasiūlytas metodas leidžia sekti svarbiausių reikalavimų atsekamumą, nes visiškas atsekamumas yra neįmanomas. Taip pat leidžia detaliau atsekti tuos reikalavimus, kurie labiausiai linkę keistis.
8. Pasiūlytas metodas leidžia valdyti reikalavimų pasikeitimus ir įvertinti su tuo susijusią riziką.

6. LITERATŪRA

[1] I. Sommerville, „Software Engineering,“ Addison Wesley, 2006.

[2] P. B. Crosby, „Quality Is Free,“ Mentor Books , 1992.

[3] W. Humphrey, „Managing the Software Process,“ Addison-Wesley, 1989, pp. ch. 8, 10, 16.

[4] R. A. Bauer, E. Collar, V. Tang, J. Wind ir P. Houston, „The Silverlake Project: Transformation at IBM. Product Details,“ Oxford University Press, USA, 1992.

[5] N. I. o. S. a. Technology, „Baldrige National Quality Program,“ [Tinkle]. Available: http://www.quality.nist.gov.

[6] „International Standard ISO 9001. Quality management systems – Requirements,“ 2000.

[7] T. D. a. A. C. f. Software, „Software Quality,“ 2010. [Tinkle]. Available: https://www.thedacs.com. [Kreiptasi 2012.01.15].

[8] A. [Tinkle]. Available: http://webstore.iec.ch/preview/info_isoiec9126-1%7Bed1.0%7Den.pdf.

[9] R. Krištapaitis, „Reikalavimų inžinerija,“ [Tinkle]. Available: http://www.slideshare.net/rollis/pi6paskaita. [Kreiptasi 2012.02.14].

[10] S. Soonsongtanee ir Y. Limpiyakorn, „Enhancement of Requirements Traceability with State Diagrams,“ įtraukta *2nd International Conference on Computer Engineering and Technology*, 2010.

[11] G. Kotonya ir I. Sommerville, „Requirements Engineering - Processes and Techniques,“ John Wiley & Sons Ltd, 1998.

[12] I. S. 830-1998, „IEEE STD 830-1998. IEEE Recommended Practice for Software Requirements Specification,“ *IEEE Computer Society*, pp. 1-39, 1998.

[13] A. Alain ir M. W. James., „Software Engineering Body of Knowledge (SWEBOK),“ *IEEE Computer Society*, pp. 31-38, 2004.

[14] R. Japenga, „How to write a software reequirements specification,“ [Tinkle]. Available: http://www.microtoolsinc.com/Howsrs.php. [Kreiptasi 2011.08.15].

[15] D. C. Phil Stocks, „A framework for specification-based testing.,“ *IEEE Transactions on Software Engineering*, t. 22, nr. 11, pp. 777-793, 1996.

[16] M. Maguire ir N. Bevan, „User requirements analysis. A review of supporting methods,“ įtraukta *IFIP 17th World Computer Congress*, Montreal, Kanada, 2002.

[17] T. Yamanaka ir S. Komiya, „A Method for Navigating Interview-driven Software Requirements Elicitation Work: Effectiveness Evaluation of the Method from the Viewpoint of Efficiency," *International Journal of Systems Applicaions, Engineering & Development,* t. 5, nr. 2, p. 161, 2011.

[18] C. Chang ir M. Christensen, „Event-based traceability for managing evolutionary change," *IEEE Transactions on Software Engineering,* t. 29, nr. 9, pp. 796-810, 2003.

[19] A. Finkelstein, „An Analysis of the Requirements Traceability Problem," įtraukta *First International Conference on Requirements Engineering*, Colorado Sprins, 1994.

[20] A. Kannenberg ir D. H. Saiedian, „Why Software Requirements Traceability Remains a Challenge," *The Journal of Defense Software Engineering,* nr. July/August, pp. 14-19, 2009.

[21] „Requirements Management," Software Engineering Institute, Carnegie Mellon University, 2010.

[22] Dömges, Ralf ir K. Pohl, „Adapting Traceability Environments to Project Specific Needs," *Communications of the ACM 41.12,* pp. 55-62, 2008.

[23] T. S. Group, „The Chaos Report," 2006.

[24] Ramesh, Balasubramaniam ir M. Jarke, „Toward Reference Models for Requirements Traceability," *IEEE Transactions on Software,* t. 1, nr. 27, pp. 58-93, 2001.

[25] J. Palmer, „Traceability," įtraukta *Software Requirements Engineering*, New York, IEEE Computer Society Press, 1997.

[26] M. B. Chrissis, M. Konrad ir S. Shrum, CMMI®: Guidelines for Process Integration and Product Improvement, second edition, Boston: Addison–Wesley, 2007.

[27] Heindl, Matthias ir Biffl, „A Case Study on Value-Based Requirements Tracing," įtraukta *10th European Software Engineering Conference*, Lisbon, Portugal, 2005.

[28] B. Boehm, „Value Based Software Engineering," *ACM SIGSOFT Software Engineering Notes,* t. 2, nr. 28, 2003.

[29] Q. Yang, J. J. Li ir D. M. Weiss, „A Survey of Coverage-Based Testing Tools," *The Computer Journal,* t. 52, nr. 5, pp. 589-597, 2009.

[30] N. Koochakzadeh ir R. Alhajj, „ Social Network Analysis in Software Testing to Categorize JUnit Test Case based on Coverage Information," įtraukta *IEEE International Conference on High Performance Computing and Communications,* 2011.

[31] R. S. Pressman, „Software Engineering: A Practioner's Approach, 3rd Edition," New York, McGraw Hill, 1992, p. 559.

[32] M. J. Harold, J. A. Jone, T. Li ir D. Liang, „Regression test selection for java software," įtraukta *ACM Conference*, 2001.

[33] E. Engstrm ir P. Runeson, „A qualitative survey of regression testing practices," įtraukta *International Conferenceon Product-Focused Software Process Improvement*, 2010.

[34] M. J. Arafeen ir H. Do, „Adaptive Regression Testing Strategy: An Empirical Study," įtraukta *22nd IEEE International Symposium on Software Reliability Engineering*, 2011.

[35] A. Orso, N. Shi ir M. J. Harrold, „Scaling regression testing to large software systems," įtraukta *International Symposium on Foundations of Software Engineering*, 2004.

[36] S. Elbaum, A. Malishevsky ir G. Rothermel, „Prioritizing test cases for regression testing„" įtraukta *International Symposium on Software Testing and Analysis*, 2000.

[37] L. Yang, Z. Dang ir T. R. Fischer, „Information gain of black-box testing. Formal Aspects of Computing," 2011, p. 513–539.

[38] „IEEE-STD 829-1998: IEEE standard for software test documentation," [Tinkle]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=5976. [Kreiptasi 2011.10.01].

[39] H. Zhong, L. Zhang ir H. Mei, „An Experimental Comparison of Four Test Suite Reduction Techniques," įtraukta *ICSE'06*, Shanghai, China, 2006.

[40] B. Aichernig, F. Arbab, L. As_tefa˘noaei, F. deBoer, S. Meng ir J. Rutten, „Fault based test case generation for component connectors," įtraukta *International Symposium on Theoretical Aspects of Software Engineering*, 2009.

[41] B. Beizer, „Software Testing Techniques, Second Edition," New York, Van Nostrand Reinhold, 1990.

[42] M. Marre ir A. Bertolinot, „Reducing and Estimating the Cost of Test Coverage Criteria," įtraukta *18th international conference on Software engineering*, 1996.

[43] „Volere Requirements Specification Template," [Tinkle]. Available: http://www.volere.co.uk. [Kreiptasi 2011.08.15].

[44] P. Skoković ir M. Rakić-Skoković, „Requirements-Based Testing Process in Practice," *International Journal of Industrial Engineering and Management (IJIEM)* , t. 1, nr. 4, pp. 155 - 161, 2010.

[45] P. Hsia, A. Davis ir D. Kung, „Status report: requirements engineering," *IEEE Software*, t. 10, p. 75 – 79, 1993.

[46] D. L. V. Jr, „Writing Software Requirements Specifications," 2011.04.12. [Tinkle]. Available: http://www.techwrl.com/techwhirl/magazine/writing/softwarerequirementspecs.htm.

[47] „User Requirements Specification,“ [Tinkle]. Available: http://www.canzz.com/2009/08/concept-phase-templates/.

[48] B. Ramesh, „Factors Influencing Requirements Traceability Practice,“ *Communication of the ACM,* t. 41, nr. 12, pp. 37-44, 1998.

[49] M. Heindl ir S. Biffl, „A Case Study on Value-based Requirements Tracing,“ įtraukta *10th European software engineering conference,* 2005.

[50] K. Chong ir S. Ahn, „A Feature-Oriented Requirements Tracing Method: A Study of Cost-benefit Analysis,“ įtraukta *2006 international conference on Hybrid Information Technology ICHIT‘06,* 2006.

[51] A. V. Knethen, „Change-Oriented Requirements Traceability. Support for Evolution of Embedded Systems,“ įtraukta *International Conference on Software Maintenance (ICSM'02),* 2002.

[52] R. F. Roggio, „Use Cases and Traceability: a Marriage for Improved Software Quality,“ įtraukta *16 th Annual NACCQ,* Palmerston North, New Zealand, 2003.

[53] B. Ramesh ir M. Edwards, „Issues in the Development of a Requirements Traceability Model,“ įtraukta *1st Intl. Symposium on Requirements Engineering*, San Diego, 1993.

[54] S. Supakkul ir L. Chung, „Integrating FRs and NFRs: A Use Case and Goal Driven Approach,“ įtraukta *International Conference on SERA'04*, 2004.

[55] P. Arkley ir S. Riddle, „Tailoring Traceability Information to Business Needs,“ įtraukta *14th IEEE International Requirements Engineering Conference (RE'06)*, 2006.

[56] R. B. Grady, „Practical Software Metrics for Project Management and Process Improvement,“ New Jersey, Prentice-Hall Inc., 1992.

[57] M. Fowler ir K. Scott, „UML Distilled Second Edition: a Brief Guide to the Standard Object Modeling Language,“ Addison Wesley, 2000.

[58] A. Gupta ir R. Bhatia, „Testing Functional Requirements using B Model Specifications,“ *ACM SIGSOFT Software Engineering Notes,* t. 35, nr. 2, pp. 1-7, 2010.

[59] L. Westfall, „Bidirectional Requirements Traceability,“ 2007. [Tinkle]. Available: http: //www.compaid.com/caiinternet/ezine/westfall-bidirectional.pdf .