Techninis pasiūlymas

DĖL INFORMACINĖS SISTEMOS, SKIRTOS VIEŠŲJŲ INTERNETO TINKLŲ ATVAIZDAVIMUI, STEBĖSENAI IR ANALIZEI, SUKŪRIMO, ĮDIEGIMO IR TESTAVIMO PASLAUGŲ

UAB "Netcode"
Žalgirio g. 92, Vilnius

info@netcode.lt
www.netcode.lt

Santrauka

Šis pasiūlymas teikiamas remiantis Lietuvos Respublikos ryšių reguliavimo tarnybos (toliau – Perkančioji organizacija) vykdomo **Informacinės sistemos, skirtos viešųjų interneto tinklų atvaizdavimui, stebėsenai ir analizei, sukūrimo, įdiegimo ir testavimo paslaugų** (toliau – Paslaugos) pirkimo atviro konkurso sąlygomis.

Šį pasiūlymą teikia įmonė UAB „Netcode“, kartu su subrangovu UAB „Santa Monica Networks“ (toliau kartu vadinami – Paslaugų teikėjas).

Atsižvelgiant į Projekto tikslus ir Projektui įgyvendinti reikalingų Paslaugų pirkimo techninėje specifikacijoje keliamus reikalavimus, šioje santraukoje pateikiami svarbiausi pasiūlymo aspektai, atskleidžiantys Paslaugų teikėjo kompetenciją, gebėjimus ir galimybes suteikti kokybiškas Paslaugas.

Atsižvelgiant į siūlomas Paslaugas, galima išskirti šias pagrindines Paslaugų teikėjo stipriąsias puses:

- *Sukaupta patirtis informacinių sistemų kūrimo projektuose, atliekant išsamias įvairių veiklos sričių poreikių analizes, projektuojant ir kuriant informacines sistemas, atliekant jų testavimą, rengiant projektinę dokumentaciją bei įgyvendinant kitas susijusias veiklas.*
- *Sukaupta patirtis kompiuterinių tinklų srities projektuose, teikiant patikimus duomenų perdavimo tinklų, IP komunikacijos ir informacinių technologijų saugos produktus ir paslaugas vidutiniam ir stambiam verslui bei valstybinėms įmonėms.*
- *Subalansuota Paslaugų teikėjo ekspertų komanda, turinti aukštą ir tarptautiniu mastu pripažintą kvalifikaciją veiklos analizės, specifikavimo, informacinių sistemų kūrimo, testavimo, kompiuterinių tinklų ir kitose susijusiose srityse bei šią kvalifikaciją patvirtinančius sertifikatus – ISEB, CompTIA Project+, Oracle certified programmer, CIW Database specialist, Cisco Certified Network Professional ir kt.*
- *Veikla yra sertifikuota ir atitinka aukščiausius tarptautinius standartus kokybės vadybos srityje (ISO 9001:2015 sertifikatas).*
- *Sėkmingai įgyvendinti informacinių sistemų kūrimo projektai, apie kuriuose plačiau rašoma 1.3 poskyryje.*

Šis techninis pasiūlymas parengtas vadovaujantis Paslaugų pirkimo dokumentuose pateikiamais pasiūlymų ekonominio naudingumo vertinimo kriterijais ir suskirstytas į šias pagrindines dalis:

- **2 skyriuje** įvairiais pjūviais aprašomas techninis sprendimas. Siūlomas sprendimas parengtas pagal Perkančiosios organizacijos suteiktą informaciją ir bus derinamas projekto analizės ir projektavimo etapų metu. Skyriuje detaliai aprašyta ir pagrįsta, kaip bus įgyvendintas duomenų surinkimas ir apjungimas iš skirtingų šaltinių į vieningą duomenų bazę. Aprašyti ir pagrįsti sprendimai efektyvumui užtikrinti surenkant ir apdorojant didelius duomenų kiekius. Detaliai aprašyta sistemos architektūra, o aprašymui naudojamos aiškios sąvokos, šiuolaikiškos programinės priemonės, atviri komponentai, gerosios praktikos, atviros technologijos, UML diagramos. Detaliai aprašytas sistemos plečiamumas ir pakartotinis panaudojimas. Numatytos pagrįstos ir efektyvios priemonės, kuriomis bus užtikrinama sistemos sauga. Detalizuoti duomenų praradimo ir atstatymo scenarijai. Numatytos probleminės situacijos ir nustatytos sistemos veiklos (veikimo) rizikos, aprašytos pagrįstos ir efektyvios sąnaudų prasme jų valdymo ir sprendimo procedūros.
- **3 skyriuje** pavaizduota ir aprašyta tinklo topologinio žemėlapio ir susijusių duomenų atvaizdavimo logika. Rengiant eskizus buvo siekiama, kad naudotojo sąsaja būtų paprasta ir aiški, leidžianti greitai išmokti visų pagrindinių sistemos funkcijų valdymo ir darbo eigos (angl. *Workflow*). Skyriuje taip pat aprašyti tarptautiniai

standartai, kuriais bus remiamasi siekiant užtikrinti, kad visos funkcijos būtų pasiekiamos greitai ir efektyviai, užtikrinamas aukščiausio produktyvumo mažiausiomis laiko ir pastangų sąnaudomis.

- **4 skyriuje** aprašomas Paslaugų teikimo išdėstymo racionalumas ir neprieštaringumas, jų sąsajų suvokimas bei laiko sąnaudų paslaugų teikimui optimalumas (P_3). Šis aprašymas apima Paslaugų teikimo projekto valdymo metodiką, bei principus, Paslaugų teikimo grafiko, žmogiškųjų išteklių bei pakeitimų valdymo metodikas bei strategijas.
- **5 skyriuje** aprašomas koordinavimo ir kokybės valdymo planas (P_4). Šis aprašymas apima koordinavimo planą, kokybės valdymo metodiką bei strategiją bei testavimo strategiją.
- **6 skyriuje** aprašomas rizikos vertinimo pagrįstumas (P_5). Šis aprašymas apima rizikų valdymo metodiką ir jos taikymo aspektus bei numatomas Projekto rizikas ir jų valdymo priemones.

Santrumpos ir sąvokos

Pasiūlyme naudojamos santrumpos ir sąvokos pateikiamos žemiau esančioje lentelėje.

Santrumpa / sąvoka	Paaiškinimas
AOP	Į aspektus (komponentus) orientuotas programavimas (angl. *Aspect-oriented programming*)
API	Programinė sąsaja
DB	Duomenų bazė
DBVS	Duomenų bazių valdymo sistema
Sistema, ITSIS	Informacinė sistema, skirta viešųjų interneto tinklų atvaizdavimui, stebėsenai ir analizei
HTTP	Žiniatinklio protokolas (angl. *Hypertext Transfer Protocol*)
IS	Informacinė sistema
ISO 31000	Standartas ISO 31000 *Risk management – Principles and guidelines*
ISO 9241-151	LST EN ISO 9241-151:2008 Žmogaus ir sistemos sąveikos ergonomika. 151 dalis. Žiniatinklio vartotojo sietuvų rekomendacijos (angl. *Ergonomics of human-system interaction -- Part 151: Guidance on World Wide Web user interfaces*)
ISO 9241-210	LST EN ISO 9241-210:2011 Žmogaus ir sistemos sąveikos ergonomika. 210 dalis. Į žmogų orientuotas sąveikiųjų sistemų projektavimas“ (angl. *Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems*)
IT	Informacinės technologijos
ITIL	Informacinių technologijų infrastruktūros biblioteka (angl. *Information Technology Infrastructure Library*)
MIDAS	Nacionalinis atviros prieigos mokslo informacijos duomenų archyvas
MVC	Modelio-vaizdo-kontrolerio (angl. *Model-View-Controller*) architektūra.
OpenUP	Programinės įrangos gyvavimo ciklo valdymo metodika *Open Unified Process*
ORM	Objektinis reliacinis susiejimas (angl. *Object-Relational Mapping*)
PA	Panaudos atvejis
Paslaugos	Informacinės sistemos, skirtos viešųjų interneto tinklų atvaizdavimui, stebėsenai ir analizei, sukūrimo, įdiegimo ir testavimo paslaugos
Paslaugų teikėjas	UAB „Netcode“ kartu su subtiekėju UAB „Santa Monica Networks“
Perkančioji organizacija, RRT	Lietuvos Respublikos ryšių reguliavimo tarnyba
PĮ	Programinė įranga
PM	Projekto valdymas
PMBOK	Projektų valdymo instituto (angl. *Project Management Institute*) projektų valdymo metodinės gairės (angl. *Project Management Body of Knowledge*)
PPK	Projekto priežiūros komitetas
Projektas	Informacinės sistemos sukūrimo, įdiegimo ir testavimo paslaugų projektas
PVG	Projekto valdymo grupė
SDLC	Programinės įrangos kūrimo gyvavimo ciklas (angl. *Software Development Life Cycle*)
SOA	Į paslaugas orientuota architektūra (angl. *Service-Oriented Architecture*)
TĮ	Techninė įranga
TS	Informacinės sistemos sukūrimo, įdiegimo ir testavimo paslaugų pirkimo techninė specifikacija
UML	Unifikuota modeliavimo kalba (angl. *Unified Modelling Language*)

Turinys

1 Įvadas

1.1 Bendra informacija

Lietuvos Respublikos ryšių reguliavimo tarnyba (toliau – Perkančioji organizacija, RRT) vykdo atvirą konkursą informacinės sistemos, skirtos viešųjų interneto tinklų atvaizdavimui, stebėsenai ir analizei, sukūrimo, įdiegimo ir testavimo paslaugoms įsigyti. Paslaugų teikimas toliau vadinamas Projektu.

Viena iš Perkančiosios organizacijos veiklų – vykdyti elektroninių ryšių tinklų ir informacijos saugumo būsenos stebėseną bei registruoti ir tirti incidentus, įvykusius viešuosiuose elektroninių ryšių tinkluose ir (ar) informacinėse sistemose Lietuvos Respublikoje.

Projekto tikslas – sukurti, įdiegti ir ištestuoti informacinę sistemą, kuri rinktų visą reikiamą informaciją iš viešai prieinamų šaltinių (RIPE, BGP ir pan.), sujungtų juos į vieną visumą ir atvaizduotų Lietuvos arba kitų šalių interneto infrastruktūros loginę struktūrą – jos komponentus ir ryšius tarp jų (toliau – Sistema, ITSIS). Sistema pagal poreikį taip pat rinktų duomenis iš neviešų informacijos šaltinių (incidentų dalinimosi platformų ir pan.) ir susietų šiuos duomenis su jau surinktais. Patogus topologijos atvaizdavimas suteiks galimybę išryškinti tinklo kritinius taškus, tarptautinį ir nacionalinį junglumą bei jų dinamiką. Sistema suteiks galimybę atlikti nuolatinę šių objektų būsenos stebėseną, matyti topologijos kitimo istoriją, galimybę juos koreliuoti, lyginti ir analizuoti istoriškai, nustatyti tinklo gedimus, užvaldytų sistemų aktyvumą, generuoti pranešimus, dalintis informacija su partneriais.

Pagrindinės Sistemos funkcijos:

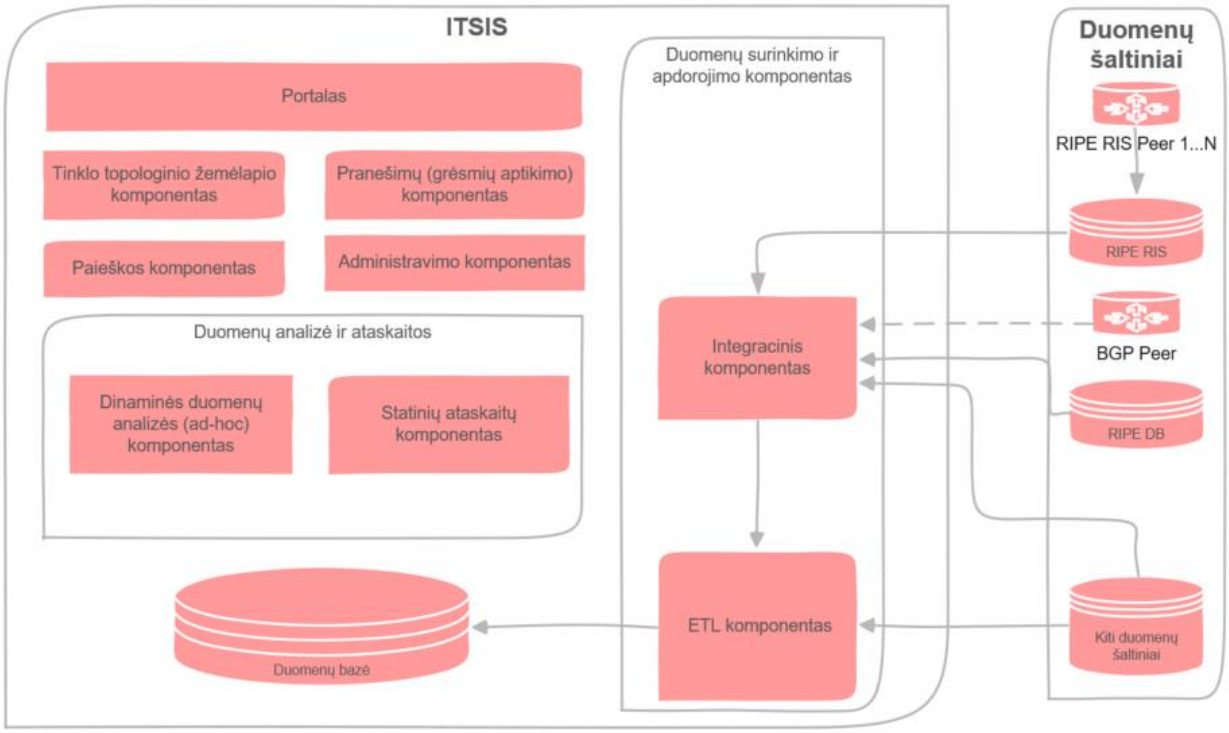
- Vizualinis tinklo topologijos analizavimas;
- Automatinis duomenų surinkimas;
- 24/7 monitoringas (remiantis BGP ir kitais protokolais);
- Grėsmių aptikimas ir automatiniai pranešimai apie jas;
- Duomenų analizavimas ir ataskaitų generavimas;
- Duomenų dalinimasis su partneriais.

Taip pat planuojama, kad Sistema bus naudojama ir kitose ES šalyse. Perkančioji organizacija galės naudotis Sistemos pilnu funkcionalumu. Kitoms ES šalių organizacijoms Perkančioji organizacija galės suteikti prieinamumą prie vizualinio tos šalies interneto tinklo topologijos analizavimo funkcijos (surinkusi atitinkamos šalies pagrindinius duomenis). Norėdamos naudotis pilnu Sistemos funkcionalumu, kitos ES šalių organizacijos turės įsidiegti Sistemą savo techninėje įrangoje

1.2 Informacinės sistemos koncepcija

Projekto metu numatyta sukurti informacinę sistemą, kuri rinktų visą reikiamą informaciją iš viešai prieinamų šaltinių (RIPE, BGP ir pan.), sujungtų juos į vieną visumą ir atvaizduotų Lietuvos arba kitų šalių interneto infrastruktūros loginę struktūrą – jos komponentus ir ryšius tarp jų (toliau – Sistema, ITSIS).

Atsižvelgiant į aukščiau įvardintus tikslus, bei kitus Paslaugų pirkimo techninėje specifikacijoje (toliau – TS) aprašytus reikalavimus buvo sudaryta Sistemos ir jos funkcinių koncepcinė schema, kuri vaizduojama žemiau esančiame paveiksle.



1 paveikslas. Sistemos koncepcinė schema

Duomenų surinkimas ir apdorojimas

Numatoma, kad pagrindiniai Sistemos duomenų šaltiniai bus RIPE registro duomenų bazė, RIPE RIS duomenų bazė bei Sistemos naudotojų įvesti interneto resursai (svetainės bei kitos sistemos). Siekiant užtikrinti pilniausią BGP lentelės duomenų gavimą planuojama, kad BGP duomenys bus surenkami pasinaudojant RIPE RIS vieša RIPEstat duomenų gavimo žiniatinklio paslauga (https://stat.ripe.net/docs/data_api). Jeigu Projekto analizės ir projektavimo etapų metu paaiškės, kad BGP duomenis tikslingiau rinkti tiesiogiai iš maršrutizatorių, bus svarstoma alternatyva panaudoti Paslaugų teikėjo (Santa Monica Networks) maršrutizatorių teikiamus BGP duomenis, kurie apima pilną BGP lentelę ir yra gaunami iš Telia Latvia bei Huricane Electrics. Duomenų surinkimo ir apdorojimo komponento paskirtis – užtikrinti duomenų paėmimą iš išorinių šaltinių, įgyvendinti reikiamą loginę kontrolę, vykdyti duomenų apdorojimą, susisteminimą ir išsaugojimą. Šis komponentas surenka duomenis iš išorinių duomenų šaltinių, juos apdoroja (transformuoja), paruošia analizei ir įkelia į Sistemos duomenų saugyklos struktūrą. Duomenų transformavimas yra vienas iš svarbiausių procesų duomenų surinkimo ir analizės sistemose, tai etapas kai surinkti duomenys yra pritaikomi veiklos analizės poreikiams. Projekto metu bus realizuotos integracinės sąsajos bei duomenų paėmimo, transformavimo ir pakrovimo procesai (angl. *Extract Tranform Load* - ETL), kurie iš išorinių sistemų gaunamus duomenis, transformuoja ir įkelia į Sistemos duomenų bazę. Kaip pagrindinį sprendimą duomenų surinkimui ir apdorojimui siūlome naudoti Pentaho Data Integration Community Edition - PDI programinę įrangą, kuri suteikia galimybę integruoti įvairius duomenų šaltinius, tokius kaip duomenų bazės, failai, išorinės sistemos bei iš šių šaltinių gautus duomenis transformuoti į vieningą formatą, tinkantį analizei (detaliau žr. skyriuje „Duomenų surinkimo ir apdorojimo komponentas - Pentaho data integration“).

Duomenų analizė ir ataskaitos

Atsižvelgiant į pirkimo TS reikalavimus, Sistemoje numatoma realizuoti kelių tipų duomenų analizės priemones. Pirmiausia tai dinaminės (angl. *ad-hoc*) duomenų analizės priemonės, kurių pagrindą sudarys OLAP (angl. *On-Line Analytical Processing*) programiniai komponentai: Pentacho Mondrian – atvirojo kodo OLAP serveris, kuris suteikia galimybę realiu laiku analizuoti didelius kiekius kompleksinių duomenų, bei Saiku CE –atvirojo kodo OLAP analizei skirta naudotojo sąsaja. Saiku CE veikia interneto naršyklėje ir palaiko įvairius OLAP serverius, tame tarpe ir Mondrian. Sistemoje taip pat numatoma realizuoti ir statines ataskaitas, t.y. ataskaitas formuojamas pagal iš anksto apibrėžtą šabloną. Statinių ataskaitų formavimui numatoma naudoti Pentaho Reporting programinį komponentą.

Portalas

Didžioji dalis sistemos funkcijų naudotojams bus pasiekiama per interneto naršyklėje veikiantį Sistemos portalą. Portalas į bendrą visumą apjungs įvairias Sistemos funkcines sritis bei komponentus. Technologiniu požiūriu portalas bus kuriamas pasitelkiant naudotojo sąsajos kūrimo karkasą AngularJS, detaliau žr. skyriuje „Naudotojo sąsajos kūrimo karkasas - AngularJs“.

Tinklo topologinis žemėlapis

Tinklo topologinis žemėlapis, tai vienas iš pagrindinių Sistemos naudotojo sąsajos elementų, kuriame bus galima interaktyviai peržiūrėti bei analizuoti Sistemoje sukauptus duomenis. Naudotojui bus sudaryta galimybė atlikti įvairius žemėlapio valdymo veiksmus pvz. keisti mastelį, filtruoti objektus, slankioti vaizdą ir pan. Detaliau apie žemėlapyje numatomas realizuoti funkcijas žr. skyriuje „Tinklo topologinio žemėlapio panaudos atvejai“.

Pranešimų (grėsmių aptikimo) komponentas

Pagrindinė šio komponento paskirtis - vykdyti nuolatinę duomenų stebėseną ir, susidarius sąlygoms, nurodytoms pranešimų taisyklėse, sugeneruoti ir išsiųsti pranešimą. Administratoriams taip pat bus sudaryta galimybė kurti bei redaguoti pranešimų siuntimo taisykles. Detaliau apie šio komponento funkcionalumus žr. skyriuje „Pranešimų (grėsmių aptikimo) panaudos atvejai“.

Administravimo komponentas

Administravimo komponente numatoma realizuoti įvairias Sistemos administravimui reikalingas funkcijas, tokias kaip naudotojų administravimas, klasifikatorių tvarkymas, duomenų archyvavimas, audito įvykių žurnalo peržiūra ir pan.

Paieškos komponentas

Šiame komponente realizuojamos funkcijos, reikalingos Sistemoje saugomų duomenų indeksavimui ir paieškai atlikti. Modulis leis vykdyti tiek greitą paiešką, įvedus tik paieškos frazę, tiek ir detalią paiešką pagal konkrečius paieškos kriterijus. Detaliau apie šio komponento funkcionalumus žr. skyriuje „Duomenų paieškos panaudos atvejai“.

1.3 Paslaugų teikėjo patirtis

UAB „Netcode“ specializuojasi kompleksinių informacinių sistemų kūrime. Teikiamos paslaugos apima reikalavimų analizę ir specifikavimą, programinės įrangos architektūros projektavimą, programavimą ir testavimą. Įmonėje dirba sertifikuoti technologijų specialistai, kurie turi sukaupę didelę patirtį kuriant kompleksinius programinės įrangos

sprendimus, paremtus moderniais programiniais karkasais ir technologijomis. Siekdami užtikrinti programinės įrangos kokybę, Netcode vadovaujasi tarptautiniu mastu pripažintomis metodikomis, standartais bei gerosiomis praktikomis. Įmonėje įdiegta kokybės vadybos sistema atitinkanti LST EN ISO 9001:2008 standarto reikalavimus.

„Santa Monica Networks“ teikia patikimus duomenų perdavimo tinklų, IP komunikacijos ir informacinių technologijų (IT) saugos produktus ir paslaugas vidutiniam ir stambiam verslui bei valstybinėms įmonėms. Klientai Santa Monica Networks paslaugų portfelyje visuomet suras platų pasirinkimą – nuo tinklo, saugumo įrangos įsigijimo su garantiniu (arba be garantinio) aptarnavimo iki visuminių paslaugų - įrangos priežiūros, įdiegtų sprendimų valdymo, apimančių visus pagrindinius elementus – kūrimas, diegimas, valdymas, optimizavimas bei profesionalių - konsultacijos, auditai, diegimo projektai, mokymai.

"Santa Monica Networks" Lietuvoje yra įmonių grupės veikiančios Lietuvoje ir Latvijoje dalis. Ši priklausomybė leidžia teikti kokybiškas paslaugas klientams Lietuvoje ir už jos ribų. "Santa Monica Networks" turi sukaupusi didelę patirtį duomenų perdavimo tinklų projektavimo ir diegimo srityse. Vienas didžiausių projektų yra UAB "Technologijų ir inovacijų centras" duomenų perdavimo tinklo atnaujinimas, kuris buvo užbaigtas šiais metais. Sutarties vertė – 5.431.804 Eur su PVM. Projekto metu buvo atnaujintas kamieninis tinklas, prieigos mazgai, VoIP Sistema, saugos sistema ir stebėsenos sistema.

Paslaugų teikėjas yra sėkmingai įgyvendinęs ir įgyvendina daug įvairių IT srities projektų, kurių metu buvo ir yra teikiamos IS kūrimo, techninių specifikacijų rengimo, techninės priežiūros ir kitos susijusios paslaugos. Projektų įgyvendinimo metu paslaugų teikėjas įgijo daug darbinės patirties ir specifinių IT srities žinių, kurias pritaikys ir šių Paslaugų teikimo metu. Iš visų Paslaugų teikėjo įvykdytų ir vykdomų projektų galima išskirti tris, kurių metų įgyta patirtis ir žinios bus itin naudingi teikiant Paslaugas. Šių projektų aprašymai pateikiami 1 lentelėje.

1 lentelė. Paslaugų teikėjo įgyvendinti projektai

Projektas (data)	Trumpas aprašymas	Paslaugų teikėjo vaidmuo projekte, vykdytos veiklos
„Nacionalinis atviros prieigos mokslo informacijos duomenų archyvas“ (2011-05 – 2015-05)	Projekto tikslas – sukurti vieningą nacionalinį mokslinių tyrimų duomenų skaitmeninį archyvą, leidžiantį teikti el. paslaugas, kaupti ir saugoti biomedicinos, fizinių, humanitarinių, socialinių, žemės ūkio ir technologijos mokslų tyrimų duomenis ir kitą su moksliniais tyrimais susijusią informaciją, sudarant galimybę visiems norintiesiems nemokamai, lengvai ir patogiai, nepažeidžiant autorių ir intelektinės nuosavybės teisių, juos pasiekti internetu. Projekto metu be pagrindinės archyvo informacinės sistemos (www.midas.lt) sukūrimo, taip pat buvo: ▪ įsigyta techninė įranga leidžianti archyve saugiai talpinti iki 3 PB duomenų; ▪ sukurtas atskiras biomedicinos komponentas (biomedicina.midas.lt), ypatingas savo duomenų specifiškumu ir duomenų surinkimo metodais bei standartais; ▪ bei sukurtas duomenų analizės įrankis (damis.midas.lt/), sudarantis tyrėjams galimybę atlikti pagrindinius duomenų	Projekto apimtyje buvo sukurtas nacionalinis atviros prieigos mokslo informacijos duomenų archyvas – MIDAS. UAB „Netcode“ vykdė šias projekto veiklas: ▪ Parengė ir suderino MIDAS analizės dokumentaciją. ▪ Parengė ir suderino MIDAS detalaus projektavimo dokumentaciją ir rezultatus. ▪ Parengė MIDAS prototipą ir atliko MIDAS techninės infrastruktūros, MIDAS sisteminės programinės įrangos ir MIDAS taikomosios programinės įrangos esminių komponentų integralumo įvertinimą. ▪ Parengė MIDAS kūrimo aplinką, sukūrė (suprogramavo) MIDAS programinę įrangą, atliko vidinį testavimą. ▪ Parengė MIDAS iteracinių testavimų aplinką bei ištaisė iteracinio testavimo metu nustatytas klaidas. ▪ Parengė MIDAS priėmimo testavimo aplinką bei ištaisė priėmimo testavimo metu nustatytas klaidas. ▪ Parengti MIDAS gamybinę aplinką ir parengė bandomosios eksploatacijos ataskaitą. ▪ Parengė MIDAS paslaugų gavėjų nuotolinio mokymosi bei tiesioginių mokymų medžiagą, naudotojo vadovą. ▪ Apmokė naudotojus ir paslaugų gavėjus dirbti su MIDAS funkcionalumais.

Projektas (data)	Trumpas aprašymas	Paslaugų teikėjo vaidmuo projekte, vykdytos veiklos
	analizės tyrimus, vizualios analizės priemonėmis tirti daugiamačių duomenų projekcijas į plokštumą, naudojant VU MII klasterio ir VU MIF superkompiuterio išteklius.	▪ Šiuo metu vykdomas MIDAS taikomosios programinės įrangos garantinis aptarnavimas.
„VĮ Ignalinos atominės elektrinės aukcionų vykdymo informacinės sistemos sukūrimas ir įdiegimas“ (2016-04 – dabar)	Pagrindinis projekto tikslas - sukurti aukcionų vykdymo informacinę sistemą, kuri leistų kompiuterizuotu būdu kaupti, tvarkyti, apdoroti, saugoti ir teikti duomenis apie nereikalingo arba netinkamo naudoti turto pardavimus, einamuosius pardavimus, pirkėjų pasiūlymus, pardavimų vykdymą ir rezultatus.	Projekto metu sukurta VĮ Ignalinos atominės elektrinės aukcionų vykdymo informacinė sistema. UAB „Netcode“ vykdė šias projekto veiklas: ▪ Parengė ir suderino sistemos kūrimo projekto vykdymo ir valdymo dokumentaciją. ▪ Parengė ir suderino sistemos reikalavimų analizės dokumentaciją. ▪ Parengė ir suderino sistemos projektavimo dokumentaciją. ▪ Sukūrė sistemos programinę įrangą, atliko vidinį testavimą. ▪ Parengė sistemos testavimo ir gamybinę aplinkas bei jose įdiegė sukurtą sistemos programinę įrangą. ▪ Organizavo priėmimo testavimą bei taisė priėmimo testavimo metu nustatytas klaidas. ▪ Organizavo bandomąją eksploataciją ir parengė bandomosios eksploatacijos ataskaitą. ▪ Apmokė Sistemos naudotojus ir administratorius. Šiuo metu vykdoma sistemos garantinė priežiūra.
RC „Archyvo“ programos modernizavimo techninės specifikacijos parengimas (2016-12 – 2017-01)	Pagrindinis projekto tikslas – atlikti reikalavimų analizę ir parengti detalią techninę specifikaciją, kuri aiškiai ir išsamiai aprašo taikomas technologijas bei kokie programavimo darbai turi būti atlikti, perkeliant „Archyvo“ programą į naują aplinką, išlaikant esamą funkcionalumą bei jį papildanti naujomis funkcijomis. **Pažymėtina, kad „Archyvo“ programa yra skirta nekilnojamo turto registro bylų administravimui.**	Projekto metu UAB „Netcode“ vykdė šias veiklas: ▪ Išanalizavo esamos programos „Archyvas“ versijos funkcionalumą. ▪ Naudojantis UML notacijos diagramomis aprašė esamus ir naujus funkcinius reikalavimus. ▪ Pagrindiniams ir sudėtingesniems programos panaudojimo atvejams parengė naudotojo sąsajos prototipus. ▪ Aprašė visus duomenis ir duomenų struktūras. ▪ Specifikavo nefunkcinius reikalavimus, užtikrinančius programos saugumą, duomenų patikimumą, veiksmų atsekamumą ir kt. ▪ Parengė darbų planą ir rekomendacijas techninės specifikacijos realizavimui.
Globalios tapatybės paslaugos sukūrimas (2016-06 – dabar)	Projekto tikslas – sukurti pilotinę globalios tapatybės (gID) paslaugą, kuri sudarytų galimybę identifikuoti asmenį fizinėje aptarnavimo vietoje (ne elektroninėje erdvėje), kuomet prisistatantis asmuo elektroninėmis priemonėmis patvirtina savo asmens duomenų (paso, tapatybės kortelės, kitų dokumentų bei duomenų) teikimą ir perduoda duomenis, o tikrinantis asmuo elektroninėmis priemonėmis gali įsitikinti šių duomenų autentiškumu ir korektiškumu. Projekto metu buvo įgyvendintos šios veiklos: ▪ gID paslaugos poreikio, naudojimosi ypatumų ir įgyvendinimo alternatyvų analizė; ▪ pilotinės gID sistemos projektavimas; ▪ piltinės gID sistemos realizavimas; ▪ pilotinės gID sistemos testavimas;	Projekto metu Paslaugų teikėjas teikė sprendimo analizės, projektavimo, testavimo ir kitas projekto apimtyje numatytas paslaugas: ▪ atlikome poreikių ir įgyvendinimo alternatyvų analizę; ▪ atlikome gID sistemos ir mobiliosios programėlės projektavimą, konstravimą ir diegimą; ▪ atlikome sukurtų sprendimų testavimą; ▪ parengėme projektinę dokumentaciją (projektavimo dokumentas, testavimo ir bandomosios eksploatacijos dokumentai ir kt.); ▪ išanalizavome tolimesnes gID paslaugos vystymo galimybes ir parengėme gID paslaugos vystymo planą; ▪ vykdėme projekto valdymo veiklas ir rengėme su tuo susijusias projektines ataskaitas bei kitus dokumentus. Šiuo metu yra derinami paskutiniai projektiniai dokumentai, todėl projektas jau yra beveik užbaigtas.

Projektas (data)	Trumpas aprašymas	Paslaugų teikėjo vaidmuo projekte, vykdytos veiklos
	▪ pilotinės gID sistemos įvedimas į eksploataciją; ▪ tolimesnio gID paslaugos vystymo plano parengimas.	

Teikiant Paslaugas labiausiai pasitarnaus 1 lentelėje aprašytų projektų metu įgyta didelių duomenų tvarkymo ir integracinių sąsajų kūrimo patirtis. Projekto „Nacionalinis atviros prieigos mokslo informacijos duomenų archyvas“ metu buvo sukurta programinė įranga, skirta tvarkyti ir saugoti itin didelius duomenų kiekius (iš viso PĮ saugykloje galima talpinti iki 3 PB duomenų). Šios PĮ kūrimo metu buvo panaudoti įvairūs duomenų apdorojimo efektyvumo, sistemos našumo, duomenų saugos užtikrinimo ir kiti susiję sprendimai ir įsitikinta jų efektyvumu, kokybiškumu, paprastumu. Pažymėtina ir tai, kad MIDAS projekto metu buvo sukurta dagiau nei 10 integracinių sąsajų. Šie sprendimai ir įgyta patirtis taip pat bus įgyvendinami kuriant ERDAIS (apie planuojamus naudoti techninius sprendimus detaliai rašoma 2 skyriuje).

Projekto „Globalios tapatybės paslaugos sukūrimas“ apimtyje buvo realizuotos šios integracinės sąsajos – su Lietuvos Respublikos gyventojų registru, su Juridinių asmenų registru, su Įgaliojimų registru, su mobiliojo parašo infrastruktūra bei su tapatybės nustatymo paslauga iPasas. Paslaugų teikėjas yra realizavęs integracines sąsajas ir kituose IT projektuose, kurių metu įgijo didelę patirtį bei žinias ir juos pritaikys derinant technines integracinių sąsajų detales ir kuriant reikalingus programinius sprendimus.

2 Duomenų surinkimo, apdorojimo ir saugojimo įgyvendinimas, sistemos architektūra, plečiamumas, pakartotinis panaudojimas bei sistemos sauga (P_1)

Remdamasis Perkančiosios organizacijos pateikta informacija Tiekėjas šiame skyriuje ir jo poskyriuose įvairiais pjūviais aprašo siūlomą techninį sprendimą. Siūlomas sprendimas parengtas pagal Perkančiosios organizacijos suteiktą informaciją ir bus derinamas projekto analizės ir projektavimo etapų metu.

Skyriuje detaliai aprašyta ir pagrįsta, kaip bus įgyvendintas duomenų surinkimas ir apjungimas iš skirtingų šaltinių į vieningą duomenų bazę. Aprašyti ir pagrįsti sprendimai efektyvumui užtikrinti surenkant ir apdorojant didelius duomenų kiekius. Detaliai aprašyta sistemos architektūra, o aprašymui naudojamos aiškios sąvokos, šiuolaikiškos programinės priemonės, atviri komponentai, gerosios praktikos, atviros technologijos, UML diagramos. Detaliai aprašytas sistemos plečiamumas ir pakartotinis panaudojimas. Numatytos pagrįstos ir efektyvios priemonės, kuriomis bus užtikrinama sistemos sauga. Detalizuoti duomenų praradimo ir atstatymo scenarijai. Numatytos probleminės situacijos ir nustatytos sistemos veiklos (veikimo) rizikos, aprašytos pagrįstos ir efektyvios sąnaudų prasme jų valdymo ir sprendimo procedūros.

2.1 Bendri architektūros principai

Programinės įrangos architektūra gali būti projektuojama laikantis tam tikro architektūrinio stiliaus. Architektūrinį stilių sudaro aibė architektūrinių principų, kurie sprendžia tam tikras architektūrines problemas. Šiame skyrelyje aprašomi pagrindiniai architektūriniai principai, kurių bus laikomasi projektuojant ir kuriant sistemą.

2.1.1 Mikroservisų architektūra

Šiuo metu plačiai paplitusi praktika – kurti gana vienalytes sistemas (angl. *monolithic systems*), kurios nors ir suskaidomos į atskirus architektūrinius sluoksnius, tačiau viename sluoksnyje veikiantys moduliai, nėra pilnai atskirti ir dažniausiai veikia kaip vienas operacinės sistemos procesas. Dėl šios priežasties sistema nėra lanksti, ją sunku plėsti bei palaikyti, pavyzdžiui, atlikus net minimalius pakeitimus, visą vienalytę sistemą reikia iš naujo perkompiliuoti ir perdiegti. Taip pat didėja tikimybė, kad atlikti pakeitimai įtakos kitų sistemos funkcijų veikimą, nors jos ir nebuvo keičiamos.

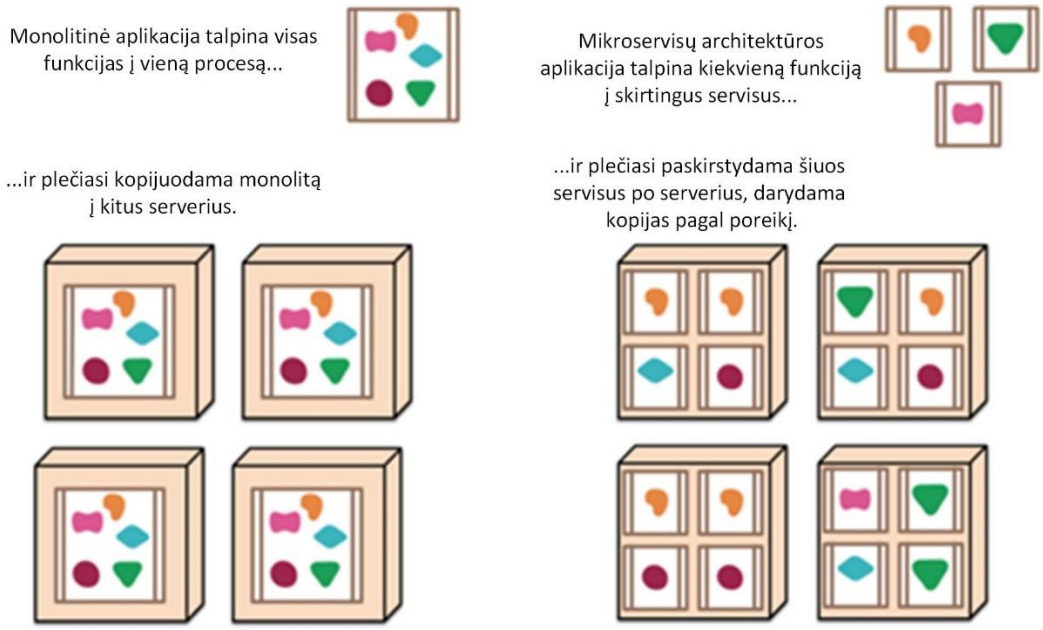
Šiuo metu plačiai kalbama apie mikroservisų naudojimą ir jų pranašumą prieš vienalytes sistemas. Taip pat dabartinės sistemų kūrimo tendencijos rodo vis dažnesnį perėjimą nuo vienalyčių sistemų prie mikroservisų architektūra paremtų sistemų. Dalis didžiųjų kompanijų, tokių kaip *Netflix, Amazon, eBay* ir kt., jau gana seniai tobulina savo vienalytes sistemas ir pereina prie mikroservisų architektūros. Pavyzdžiui, kompanija *Amazon* savo sistemą suskaidė į mikroservisus, atsakingus už tokias sistemos funkcijas kaip prekių krepšelis, rekomendacijos, sąskaitų išrašymas, sandėlio valdymas ir t.t. Toks sprendimas buvo pasirinktas todėl, kad vis labiau augo sistemos funkcijų kiekis, naudotojų skaičius, vykdomų operacijų skaičius, darėsi sudėtinga plėsti, diegti ir palaikyti vienalytes sistemas.

Mikroservisai – tai nedidelės apimties programos, turinčios savo programinę sąsają (API) ir atliekančios apibrėžtą ir gana siaurą funkcijų aibę. Šios programos gali būti diegiamos, atnaujinamos ir plečiamos nepriklausomai viena nuo kitos.

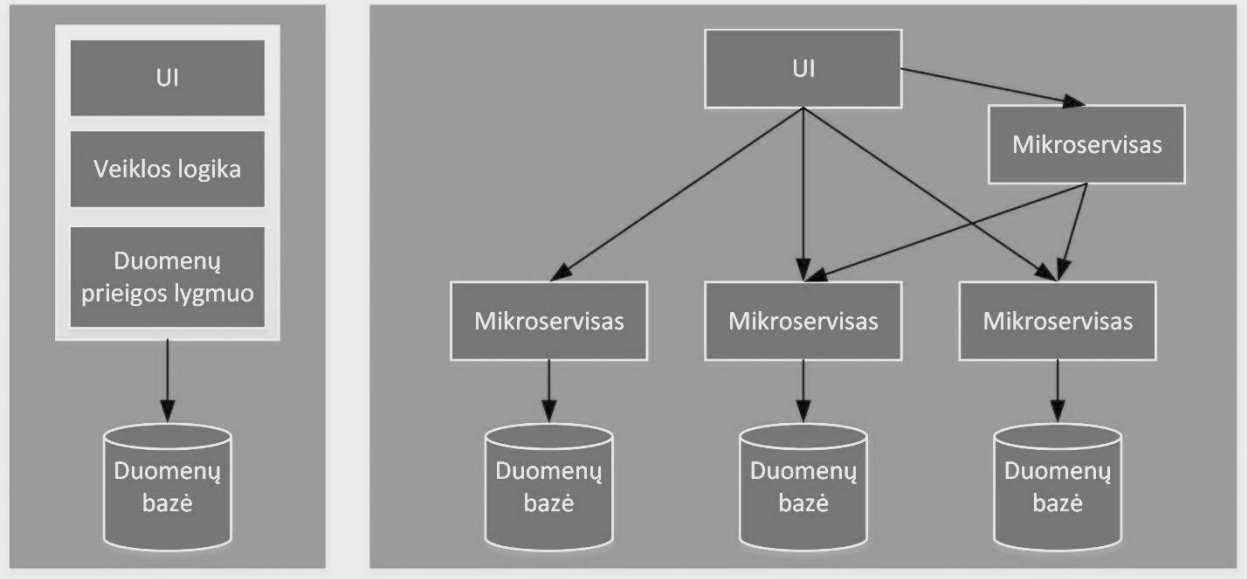
Vienalytės sistemos atveju, visos sistemos funkcijos dažniausiai yra vykdomos viename operacinės sistemos servise, t.y. komunikacija vyksta tarp operatyviojoje atmintyje inicializuotų objektų (angl. *In-memory calls*). Siekiant sistemą padaryti lankstesnę, lengviau diegiamą ir lengviau plečiamą, funkcijos gali būti realizuojamos kaip atskiri programiniai moduliai, kurie veikia atskiruose procesuose ir tarpusavyje bendrauja naudodami standartinius mechanizmus (pvz. per SOAP, RESTful API). Mikroservisų architektūrinis principas taip pat užtikrina aiškias ribas tarp modulių, kurie vienalytėse sistemose dažniausiai nebūna taip gerai išskaidyti. Mikroservisai yra ganėtinai nepriklausomi vienas nuo kito, esant poreikiui

toje pačioje sistemoje veikiantys mikroservisai netgi gali būti realizuojami skirtingomis programavimo kalbomis. Visa tai kuriamą sistemą padaro lankstesne, lengvai plečiama ir diegiama, kadangi atlikus vieno modulio pakeitimą nebūtina perkompiliuoti ir perdiegti visos sistemos.

Žemiau esančiuose paveiksluose pateikiami pavyzdžiai, išryškinantis esminius monolitinės ir mikroservisais paremtos aplikacijos skirtumus.



2 paveikslas. Esminiai skirtumai tarp monolitinės ir mikroservisais paremtos sistemos



3 paveikslas. Vienalytės sistemos išskaidymas į mikroservisus

Žinoma, tam tikrais atvejais monolitinės architektūros aplikacijos gali būti tinkamesnis pasirinkimas, tačiau kuo toliau, tuo labiau pastebimi tokių aplikacijų trūkumai, o ypač kai aplikacija dažnai tobulinama, reikia atlikti dažną atnaujinimų diegimą arba padidinti tam tikros funkcijos našumą.

Vienas iš esminių mikroservisų architektūros reikalavimų yra tai, kad kiekvieną programinės įrangos komponentą būtų galima nepriklausomai diegti ir atnaujinti. Atsižvelgiant į šį reikalavimą, mikroservisų architektūroje komponentai yra išskiriami ne kaip atskiros programinio kodo bibliotekos, bet kaip atskiri servisai, veikiantys atskiruose operacinės sistemos procesuose, todėl jie gali būti nepriklausomai diegiami ir atnaujinami. Žinoma, tam tikrais atvejais atliekant komponento atnaujinimą gali keistis ir paties serviso sąsaja (angl. *Interface*), tuomet neišvengiamai teks atnaujinti ir susijusius komponentus.

Būtina pažymėti ir tai, kad mikroservisų architektūra puikiai dera su debesų kompiuterijos principais (angl. cloud native) bei šiuo metu sparčiai populiarėjančiais programinės įrangos konteinerių valdymo sprendimais (pvz. Docker, Rocket, LXD), kurie suteikia galimybę kiekvieną mikroservisą paleisti kaip atskirą konteinerį.

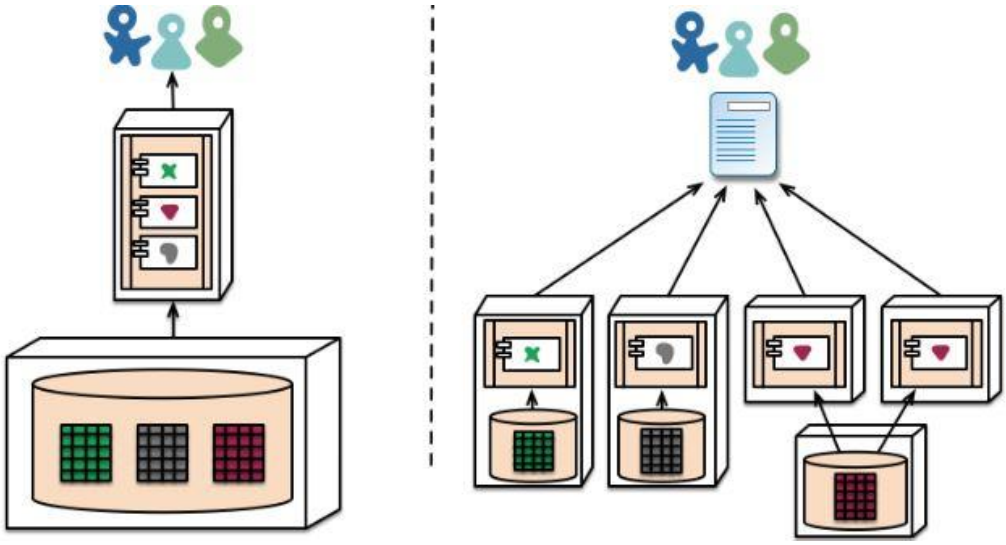
Šiame projekte kuriamai programinei įrangai mes siūlome naudoti *Docker* konteinerių formatą ir jų valdymo infrastruktūrą. *Docker* – tai *de facto* konteinerių formatas, kurį galima diegti tiek savo infrastruktūroje, tiek ir išorinių paslaugų teikėjų infrastruktūrose: Google Container Engine, Amazon EC2 Container Service. Daugiau apie *Docker* technologiją skaitykite skyriuje „Konteinerių valdymas - Docker“.

Mikroservisų architektūrą, lyginant su vienalytės sistemos architektūra, galima išskirti šiuos esminius privalumus:

- **Mikroservisai yra nepriklausomi vienas nuo kito**. Lyginant su vienalyte sistema mikroservisai yra nepriklausomi vienas nuo kito t.y., kiekvienas mikroservisas ar keli apjungti mikroservisai yra kuriami atskirai tik tam tikram sistemos funkcionalumo vykdymui. Mikroservisai į bendrą visumą apjungiami naudotojo sąsajos

lygmenyje, todėl naudotojui atrodo, kad jis dirba su vientisa sistema. Kiekvienas mikroservisas turi savo programinę sąsają (API) ir bendrauja naudodamas standartinių protokolų užklausas (pvz. RPC, SOAP, REST ir pan.). Dažniausiai užklausų atsakymai yra pateikia HTML arba JSON/XML formatais.

- **Sumažinama klaidų įtaka sistemai.** Neveikiant vienam mikroservisui, kiti mikroservisai gali sėkmingai veikti. Lyginant su vienalyte sistema, vienos funkcijos neveikimas gali įtakoti visos sistemos neveikimą.
- **Greitesnis ir patogesnis sistemos diegimas.** Kuriant sistemą iš atskirų mikroservisų yra palengvinamas sistemos atnaujinimas ir diegimas tiek į testinę, tiek į gamybinę aplinką. Lyginant su vienalyte sistema, atlikus programinio kodo pakeitimus, nereikia perdiegti visos sistemos, užtenka sudiegti tik atnaujintus mikroservisus. Šis privalumas ypač išryškėja, kai diegimas yra vykdomas laikantis nuolatinio integravimo (angl. *Continuous integration*) principų bei vykdant automatinius testus.
- **Padidėja sistemos plečiamumo galimybės**. Klasikiniu atveju, norint padidinti tam tikros funkcijos našumą, sistema yra plečiama horizontaliai pvz. į klasterį apjungiami du aplikacijų serveriai, turintys identišką visos aplikacijos kopiją. Mikroservisų architektūros atveju našumo plėtimas įmanomas konkrečios funkcijos (mikroserviso) lygmenyje, t.y. įdiegiant papildomus konkretaus serviso egzempliorius ir tarp jų paskirstant užklausas. Kalbant apie funkcionalumo plėtimą, esminis privalumas yra tai, kad atnaujinant sistemos funkcionalumą užtenka pakeisti tam tikrą mikroservisą ar kelis mikroservisus. Tai leidžia lengvai atlikti sistemos pakeitimus nekeičiant kitų sistemos komponentų. Šiuo atveju užtenka sistemoje sudiegti tik pakeisto mikroserviso atnaujinimus. Lyginant su vienalyte sistema keičiant funkciją reiktų perdiegti visą sistemą. **Pastaba:** *Jei keičiant komponentą yra pakeičiama jo sąsaja (*angl. *Interface) tuomet turi būti atnaujinami ir perdiegiami susiję komponentai.*
- **Didesnės technologinės platformos pasirinkimo galimybės**. Vienalytės sistemos yra kuriamos pasirinkus vieną tinkamiausią technologiją (php, JAVA, C/C++ ir pan.), kuri leistų realizuoti visas kuriamos sistemos funkcijas. Kadangi mikroservisai yra atskiri komponentai turintys savo API, todėl kiekvienas komponentas gali būti kuriamas skirtingomis technologijomis. Tai ypač aktualu, kai sistemą kuria keli sistemos kūrėjai ar vystytojai, kurie gali pasirinkti tinkamiausią technologinį sprendimą mikroserviso realizacijai.
- **Lengvesnis kelių programuotojų komandų valdymas**. Kiekviena komanda gali būti atsakinga tik už savo komponentų kūrimą, diegimą bei palaikymą.
- **Lengvesnis sistemos palaikymas.** Mikroservisai yra sąlyginai mažos apimties, apimantys vieną ar kelias funkcijas, todėl lengvai suprantami ne tik sistemos kūrėjui, bet ir jos vystytojams bei administratoriams.
- **Lankstesnis duomenų valdymas**. Naudojant mikroservisų architektūrą sistemos kūrimui galima decentralizuoti duomenų bazės valdymą, nes kiekvienas mikroservisas gali kreiptis į atskirą duomenų bazę ir naudoti tik jam reikiamus duomenis. Taip pat keli mikroservisai gali naudotis viena duomenų baze, žr. žemiau esantį paveikslą.



4 paveikslas. Lankstesnis duomenų valdymas – kiekvienas mikroservisas gali turėti atskirą prieigą prie duomenų bazės

2.1.2 Daugiapakopė (N-Tier) architektūra

Siūlomas techninis sprendimas yra paremtas daugiapakopės (angl. *N-Tier*) architektūros principais. Tai tokia architektūra, kai programinės įrangos sprendimą sudarančios dalys - duomenų valdymas, naudotojo sąsaja ir veiklos logikos apdorojimas – yra išskiriami į skirtingas fizines dalis. Nuo daugiasluoksnės architektūros skiriasi tuo, kad lygiai dažniausiai yra išskiriami į atskirus fizinius serverius, o daugiasluoksnės architektūros atveju išskyrimas yra loginis. Siūlome dviejų pakopų techninio sprendimo architektūrą, t.y. sistemą sudarantys programiniai komponentai būtų diegiami dviejų tipų tarnybinėse stotyse – naudotojo sąsajos ir veiklos logikos stotyje bei duomenų ir resursų stotyje. Daugiapakopės architektūros privalumai:

- Didesnis sistemos infrastruktūros lankstumas;
- Galimybė pridėti papildomus lygius, nekeičiant programinio kodo;
- Galimybė plėsti lygių techninį pajėgumą atskirai.

2.1.3 Daugiasluoksnė (N-Layer) architektūra

Daugiasluoksnė (angl. *N-Layer*) architektūra yra sistemos realizacija, kai naudotojo sąsaja, veiklos logika ir duomenų saugojimas bei prieiga yra skirtinguose loginiuose sluoksniuose. Naudotojo sąsaja, veiklos logika ir duomenų saugojimo sluoksniai realizuoti kaip atskiri sistemos komponentai. Skiriasi nuo daugiapakopės architektūros tuo, kad lygių atskyrimas yra loginis ir nebūtinai padalintas į atskirus fizinius serverius. Daugiasluoksnis sprendimas leidžia bet kurį iš sluoksnių plėtoti, atnaujinti ar pakeisti nepažeidžiant likusiųjų sluoksnių, be to, yra galimybė programinį sprendimą papildyti nauju funkciniu sluoksniu. Dėl to daugiasluoksnės architektūros naudojimas leidžia kurti lanksčią programinę įrangą.

Daugiasluoksnės architektūros privalumai:

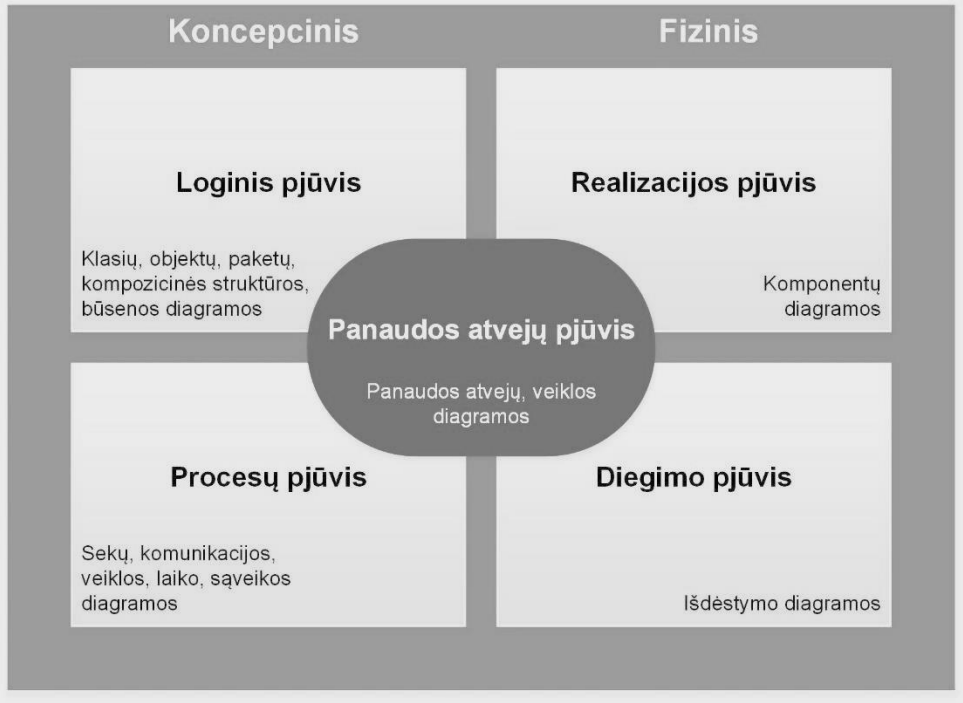
- Lankstesnė lengviau plečiama ir tobulinama programinė įranga;
- Loginis sistemos dalių atskyrimas supaprastina programinį kodą ir sumažina funkcionalumo įgyvendinimo sąnaudas ir reakcijos laiką į kintančius verslo reikalavimus;
- Leidžia efektyviau pakartotinai panaudoti programinį kodą.

Daugiasluoksnės architektūros principai puikiai dera su daugiapakopės architektūros principais.

2.2 Sistemos architektūra

Sistemos architektūra dažniausiai aprašoma įvairiais pjūviais, kuriuos apjungus į visumą, gaunamas bendras sistemos architektūros vaizdas. Kiekvienas architektūrinis pjūvis skirtas specifiniams sistemos aspektams pateikti ir gali būti adresuotas specifinei projekto dalyvių grupei: naudotojams, programuotojams, sistemų inžinieriams, sistemos administratoriams ir pan. Architektūrinis pjūvis parodo kaip sistema yra suskaidyta į atskirus komponentus ir kaip tie komponentai sąveikauja tarpusavyje.

Iš esmės architektūriniai pjūviai – tai techninio sprendimo abstrakcijos, kuriose išryškinamos pagrindinės sprendimo charakteristikos, nuošalyje paliekant ne tokias svarbias sprendimo detales. Architektūrinių pjūvių skaičius priklauso nuo konkretaus projekto poreikių ir projekto komandos. Pasaulinėje informacinių sistemų kūrimo praktikose dažnai sutinkamas „4+1“ architektūrinių pjūvių modelis, kuris vaizduojamas 5 paveiksle.



5 paveikslas. „4+1" architektūrinių pjūvių modelis

- **Panaudos atvejų pjūvis** – apibūdina sistemos funkcionalumą, išorines sąsajas ir esmines naudotojų grupes. Šis pjūvis yra privalomas ir turi būti aprašomas pačioje projekto pradžioje, kadangi nuo jo priklauso visi kiti sistemos architektūros pjūviai. Panaudos atvejų vaizdavimui ir aprašymui dažniausiai naudojamos UML panaudos atvejų diagramos (angl. *Use case*) bei veiklų diagramos (angl. *activity*).
- **Procesų pjūvis** – apibūdina sistemoje vykstančius procesus ir kaip šių procesų metu tarpusavyje sąveikauja sistemos struktūriniai elementai (aprašyti loginiame pjūvyje). Procesų pjūvis gali būti vaizduojamas įvairiuose abstrakcijos lygmenyse, pavyzdžiui sąveika tarp sistemų, sąveika tarp posistemių bei sąveika tarp objektų. Procesų aprašymui gali būti naudojamos įvairios UML diagramos: sekų diagrama (angl. *sequence*), veiklos diagrama (angl. *activity*), komunikacijos diagrama (angl. *communication*) ir kt.
- **Loginis pjūvis** – apibūdina kaip sistema yra suskaidyta į loginius vienetus (paketai, klasės, sąsajos ir pan.). Loginis sistemos skaidymas gali būti vykdomas tiek vertikaliai, t.y. pagal sistemos funkcijas, tiek horizontaliai, t.y. pagal architektūrinius lygmenis (naudotojo sąsajos lygmuo, servisų lygmuo ir pan.). Jeigu duomenų saugojimas yra svarbus sistemos aspektas, loginiame pjūvyje taip pat gali būti pateikiamas ir sistemos duomenų modelis. Loginis pjūvis taip pat yra privalomas. Loginio architektūros pjūvio vaizdavimui ir aprašymui gali būti naudojamos įvairios UML diagramos: klasių diagrama (angl. *class*), paketų diagrama (angl. *package*), objektų diagrama (angl. *object*) ir kt.
- **Realizacijos pjūvis** – apibūdina kaip programinės įrangos kodas yra fiziškai struktūrizuojamas ir saugomas failų sistemoje. Realizacijos pjūvio elementai apima programinių paketų failus ir direktorijas. Realizacijos pjūviui pavaizduoti dažniausiai naudojamos UML komponentų diagramos (angl. *component*). Realizacijos pjūvis yra neprivalomas, todėl sistemų projektavime sutinkamas rečiau.
- **Diegimo pjūvis** – apibūdina kaip programinė įranga yra įdiegta techninėje infrastruktūroje, pavyzdžiui kiek ir kokių serverių reikia, kokia programinė įranga diegiama serveriuose. Diegimo pjūviui pavaizduoti dažniausiai naudojamos UML diegimo (angl. *Deployment*) diagramos.

Tiek šiame pasiūlyme, tiek projekto vykdymo metu techninio sprendimo aprašymui naudosime „4+1" architektūrinių pjūvių modelį bei visame pasaulyje pripažįstama UML notaciją (angl. *Unified Modeling Language*). Notacijos pagrindinis tikslas yra padėti veiklos analitikams, sistemų architektams, programinės įrangos inžinieriams ir programuotojams analizuojant, projektuojant, kuriant ir diegiant informacines sistemas. UML modeliavimo kalba gana paplitusi, lengvai išmokstama,

patogi bei turinti didelę diagramų įvairovę. Tolimesniuose poskyriuose įvairiais pjūviais aprašoma siūlomas sprendimas ir jo architektūra.

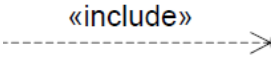
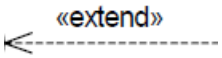
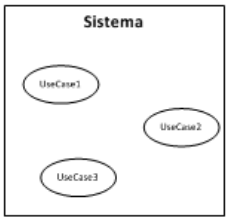
2.2.1 Panaudos atvejų pjūvis

Panaudojimo atvejų analizei naudojamos UML panaudos atvejų (angl. Use Case) diagramos, skirtos atvaizduoti sistemos funkcionalumą, nurodant sistemos aktorius bei jų atliekamus veiksmus. Projekto metu kiekvienas diagramoje pavaizduotas panaudos atvejis bus detalizuojamas, t.y. aprašomi pagrindiniai ir alternatyvūs vykdymo žingsniai, pradinės sąlygos, rezultatai bei specifiniai reikalavimai.

Projekto metu bus atliekama detali kompiuterizuojamos veiklos analizė, kurios metu šiame pasiūlyme pateiktas panaudos atvejų sąrašas bus pakartotinai peržiūrėtas ir, esant poreikiui, atnaujintas (pridėti nauji panaudos atvejai ir / ar patikslinti ar pašalinti esami panaudos atvejai) bei suderintas su Perkančiąja organizacija. Toliau kiekvienas panaudos atvejis bus detaliai išnagrinėtas bei aprašytas ir pagal tai bus kuriama programinė įranga.

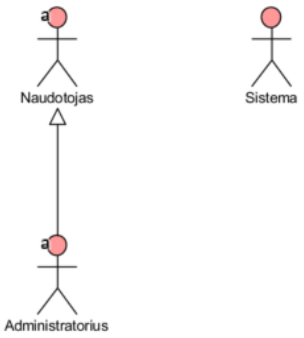
Žemiau esančioje lentelėje pateikiami panaudos atvejų diagramose naudojami elementai ir jų grafiniai žymėjimai.

2 lentelė. UML panaudos atvejų diagramose naudojami elementai ir jų grafiniai žymėjimai

Eil. Nr.	Elementas	Grafinis žymėjimas	Reikšmė
1.	Aktorius (angl. *Actor*)		Elementas vaizduoja sistemos naudotoją. Tai gali būti tiek žmogus tiek ir kita sistema. Kiekvienas aktorius aprašo skirtingą naudotojo rolę.
2.	Panaudos atvejis (angl. *Use Case*)		Veiksmų seka (įskaitant jų variantus), kurią sistema gali atlikti sąveikaudama su sistemos aktoriais.
3.	Asociacija		Aktoriaus dalyvavimas panaudos atvejyje, t. y., aktoriaus egzempliorius ir panaudos atvejis bendrauja tarpusavyje.
4.	Įtraukimas (angl. *Include*)		Naudojamas nurodyti, kad atliekant vieną panaudos atvejį, būtinai atliekamas ir kitas.
5.	Išplėtimas (angl. *Extend*)		Naudojamas nurodyti, kad atliekant vieną panaudos atvejį, esant tam tikrai sąlygai, atliekamas ir kitas.
6.	Sistemos riba		Nurodo sistemos ribas, kuriose bus vykdomi panaudos atvejai.

Toliau šio skyriaus poskyriuose pateikiami preliminarūs PA bei trumpi jų aprašymai. Paslaugų teikimo metu bus sudarytas galutinis PA sąrašas, kuris bus suderintas su Perkančiąja organizacija, bei parengti detalūs kiekvieno PA aprašymai.

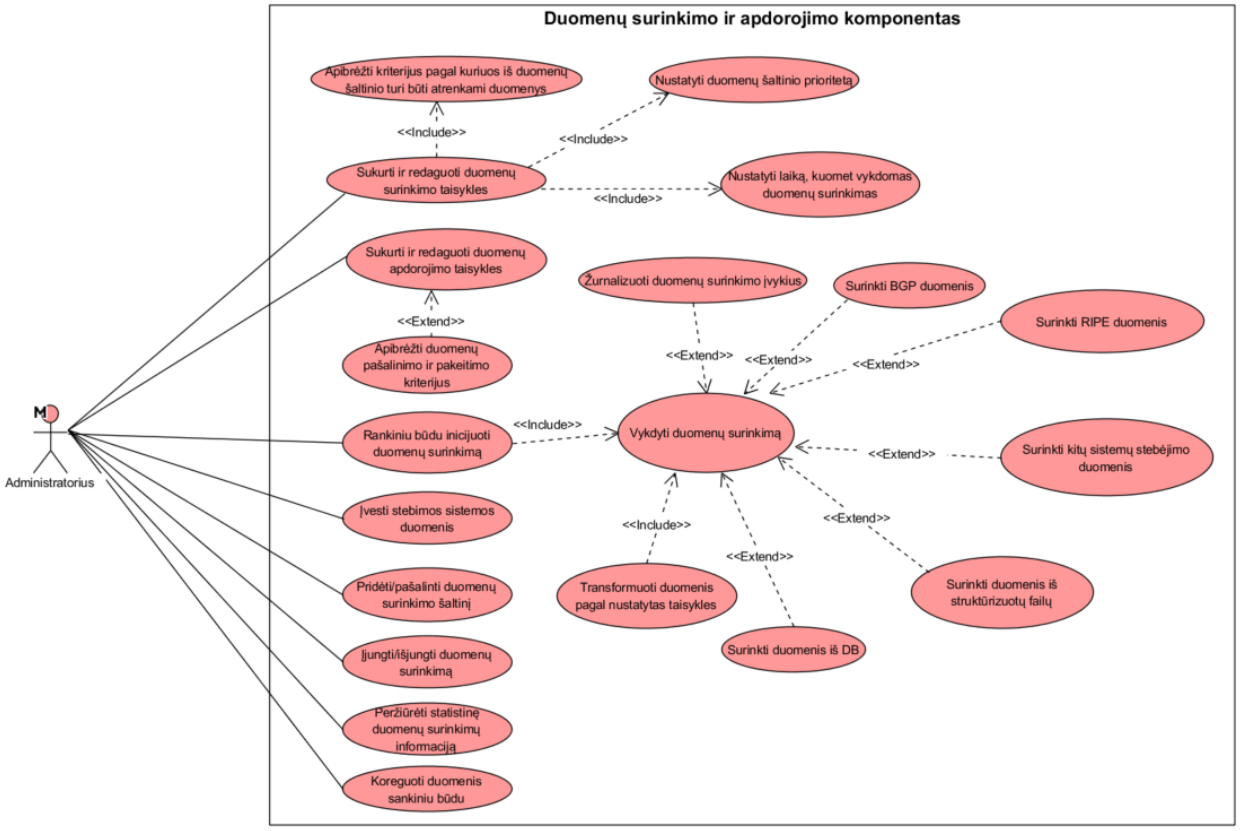
2.2.1.1 Sistemos naudotojų hierarchija



3 lentelė. Naudotojai

Aktoriai	Aprašymas
Naudotojas	Tai naudotojas, kuris Sistemoje gali vykdyti tinklo topologinio žemėlapio, duomenų paieškos, analizės, ataskaitų bei pranešimų funkcijas.
Administratorius	Tai Sistemos naudotojas, kuris gali administruoti kitus naudotojus, koreguoti duomenų surinkimo ir apdorojimo taisykles, archyvuoti senus duomenis, peržiūrėti sistemos audito įvykių žurnalą bei vykdyti kitas Sistemos administravimo funkcijas.
Sistema	Tai Sistemos automatinių procesų servisas, kuris automatiškai vykdo duomenų surinkimą, duomenų apdorojimą, grėsmių aptikimą bei kitas sistemines funkcijas.

2.2.1.2 *Duomenų surinkimo ir apdorojimo panaudos atvejai*



Žemiau esančioje lentelėje pateikiamas preliminarių duomenų surinkimo ir apdorojimo panaudos atvejų sąrašas, jų aprašymai bei įvardijami TS punktai, kurie bus įgyvendinami kiekvienu panaudos atveju.

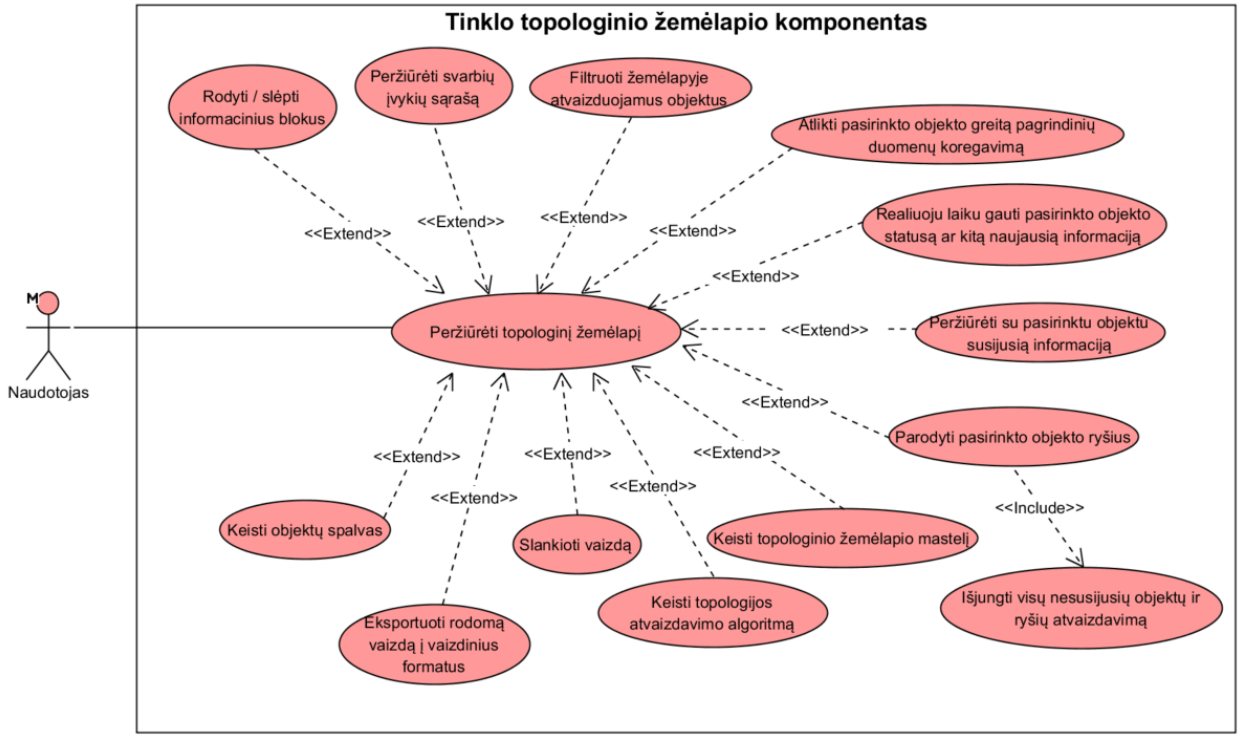
4 lentelė. Duomenų surinkimo ir apdorojimo panaudos atvejai

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
1.	Sukurti ir redaguoti duomenų surinkimo taisykles	PA apima duomenų surinkimo taisyklių kūrimo ir redagavimo funkcionalumą.	Administratorius	1.4, 1.12.1, 1.12.4, 1.12.5.
2.	Apibrėžti kriterijus pagal kuriuos iš duomenų šaltinio turi būti atrenkami duomenys	PA skirtas nustatyti, kokie konkrečiai duomenys turi būti paimami iš duomenų šaltinių. Nustatytų kriterijų netenkinantys duomenys bus nepaimami iš duomenų šaltinių duomenų surinkimo metu.	Administratorius	1.12.1.1, 1.12.1.2.
3.	Nustatyti duomenų šaltinio prioritetą	PA apima funkcionalumą, kuris sudaro galimybę nustatyti atitinkamo duomenų šaltinio prioritetą, jeigu skirtingi duomenų šaltiniai turi to paties atributo reikšmę, kuri skiriasi.	Administratorius	1.12.1.3, 1.12.1.4.
4.	Nustatyti laiką, kuomet vykdomas duomenų surinkimas	PA apima funkcionalumą, kuris leidžia nustatyti laiko intervalus arba konkretų paros laiką, kuomet vykdomas duomenų surinkimas.	Administratorius	1.12.1.5, 1.12.1.6, 1.12.1.7.
5.	Sukurti ir redaguoti duomenų apdorojimo taisykles	PA apima funkcionalumą, kuris leidžia sukurti ir redaguoti duomenų apdorojimo taisykles, t.y. nustatyti prie kokių sąlygų ir kokie automatiniai duomenų koregavimo/šalinimo veiksmai turi būti atlikti.	Administratorius	2.1, 2.2, 2.3.

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
6.	Apibrėžti duomenų pašalinimo ir pakeitimo kriterijus	PA apima funkcionalumą, kuris leidžia sukurti ir redaguoti sąlygas, kurios turi būti tenkinamos prieš atliekant automatinį duomenų pašalinimą ar pakeitimą.	Administratorius	2.3.1, 2.3.2, 2.3.3.
7.	Rankiniu būdu inicijuoti duomenų surinkimą	PA apima funkcionalumą, kuris leidžia naudotojui rankiniu būdu inicijuoti duomenų surinkimą. Inicijavus duomenų surinkimą, įtraukiamas sisteminis PA „Vykdyti duomenų surinkimą“.	Administratorius	1.5
8.	Įvesti stebimos sistemos duomenis	PA apima funkcionalumą, kuris leidžia įvesti stebimos sistemos duomenis, pvz. aprašymas kokia tai sistema, URL adresas, sektorius, savininkas, svarba.	Administratorius	1.2.1.6, 1.2.1.7.
9.	Pridėti/pašalinti duomenų surinkimo šaltinį	PA apima funkcionalumą, kuris leidžia pridėti/pašalinti duomenų surinkimo šaltinius. Sistema palaikys įvairius duomenų šaltinius, pvz. duomenų bazių valdymo sistemos, failai, žiniatinklio paslaugos ir pan.	Administratorius	1.11, 1.12.3.
10.	Įjungti/išjungti duomenų surinkimą	PA apima funkcionalumą, kuris leidžia įjungti arba išjungti duomenų surinkimą iš esamų šaltinių.	Administratorius	1.12.2.
11.	Peržiūrėti statistinę duomenų surinkimų informaciją	PA apima funkcionalumą, kuris leidžia peržiūrėti statistinę duomenų surinkimų informaciją: iš kiekvieno šaltinio gautų įrašų skaičių, užkrautų rodiklių pilnumą, klaidų skaičių ir pan.	Administratorius	1.10.
12.	Koreguoti duomenis rankiniu būdu	PA apima funkcionalumą, kurio pagalba bus galima surinktus įrašus koreguoti rankiniu būdu. Pvz. jeigu dalies duomenų surinkimas buvo nesėkmingas. Klaidingi įrašai bus pažymėti ir lengvai randami specialistui, prižiūrinčiam duomenų surinkimo ir apdorojimo procesą.	Administratorius	1.9, 2.4, 2.5.
13.	Vykdyti duomenų surinkimą	Tai sisteminis PA, kuris gali būti inicijuojamas rankiniu būdu arba vykdomas nustatytu periodiškumu. PA pagal apibrėžtas duomenų surinkimo taisykles vykdo duomenų surinkimą iš nustatytų duomenų šaltinių. Šis PA taip pat inicijuoja ir duomenų apdorojimą (transformavimą) pagal nustatytas duomenų apdorojimo taisykles.	Sistema	1.3.
14.	Žurnalizuoti duomenų surinkimo įvykius	PA skirtas žurnalizuoti su duomenų surinkimu susijusią informaciją: duomenų šaltinis, duomenų surinkimo data ir laikas, klaidos įvykusios duomenų surinkimo metu ir pan.	Sistema	1.8
15.	Surinkti BGP duomenis	PA skirtas BGP feed duomenų (pasaulinė maršrutų lentelė) surinkimui. Preliminariai planuojama, kad Sistemoje bus naudojami du RIPE RIS žiniatinklio paslauga. Jeigu Projekto analizės ir projektavimo etapų metu paaiškės, kad BGP duomenis tikslingiau rinkti tiesiogiai iš maršrutizatorių, bus svarstoma alternatyva panaudoti Paslaugų teikėjo (Santa Monica Networks) maršrutizatorių teikiamus BGP duomenis, kurie apima pilną BGP lentelę ir yra gaunami iš Telia Latvia bei Huricane Electrics.	Sistema	1.6.2.1, 1.6.2.2
16.	Surinkti RIPE duomenis	PA skirtas RIPE duomenų (vietiniai interneto registrai – LIR) surinkimui. Preliminariai planuojama, kad šie duomenys bus gaunami per RIPE duomenų bazės žiniatinklio paslaugą (RIPE RESTful API) https://rest.db.ripe.net/ripe .	Sistema	1.6.1.
17.	Surinkti kitų sistemų stebėjimo duomenis	PA skirtas duomenų surinkimui iš nutolusio įrenginio, informacinės sistemos ar paslaugos (angl. *service*). Realizacijai naudojamas Pentacho Data Integration komponentas palaiko galimybę surinkti duomenis iš kitų sistemų per HTTP API.	Sistema	1.6.5, 1.7.1, 1.7.2, 1.7.3, 1.7.4.

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
		Taip pat bus naudojamos integruotos priemonės (PING, WGET, CURL, Traceroute), kurios grąžins informaciją, ar tikrinamas objektas tuo metu yra gyvybingas (live/dead), svetainės klaidos statuso kodą (404, 503 ar pan.), mazgo maršruto pasiekiamumą ir/arba kitą informaciją, kuri leistų identifikuoti užklausiamo objekto statusą.		
18.	Surinkti duomenis iš struktūrizuotų failų	PA skirtas duomenų surinkimui iš struktūrizuotų duomenų failų. Realizacijai naudojamas Pentacho Data Integration komponentas palaiko duomenų surinkimą iš įvairių failų formatų: XML, JSON, CSV.	Sistema	1.6.4, 1.6.4.1, 1.6.4.2, 1.6.4.3.
19.	Surinkti duomenis iš DB	PA skirtas duomenų surinkimui iš įvairių išorinių ir vidinių duomenų bazių. Realizacijai naudojamas Pentacho Data Integration komponentas palaiko duomenų surinkimą iš įvairių duomenų bazių valdymo sistemų pvz. MySQL, MariaDB, MS SQL Server, ElasticSearch ir pan.	Sistema	1.6.3, 1.6.3.1, 1.6.3.2, 1.6.3.3, 1.6.3.4.
20.	Transformuoti duomenis pagal nustatytas taisykles	Tai sisteminis PA, kuris vykdomas nustatytu periodiškumu pagal apibrėžtas duomenų apdorojimo taisykles. Šį panaudos atvejį inicijuoja PA „Vykdyti duomenų surinkimą“.	Sistema	2.1, 2.2, 2.3.

2.2.1.3 *Tinklo topologinio žemėlapio panaudos atvejai*



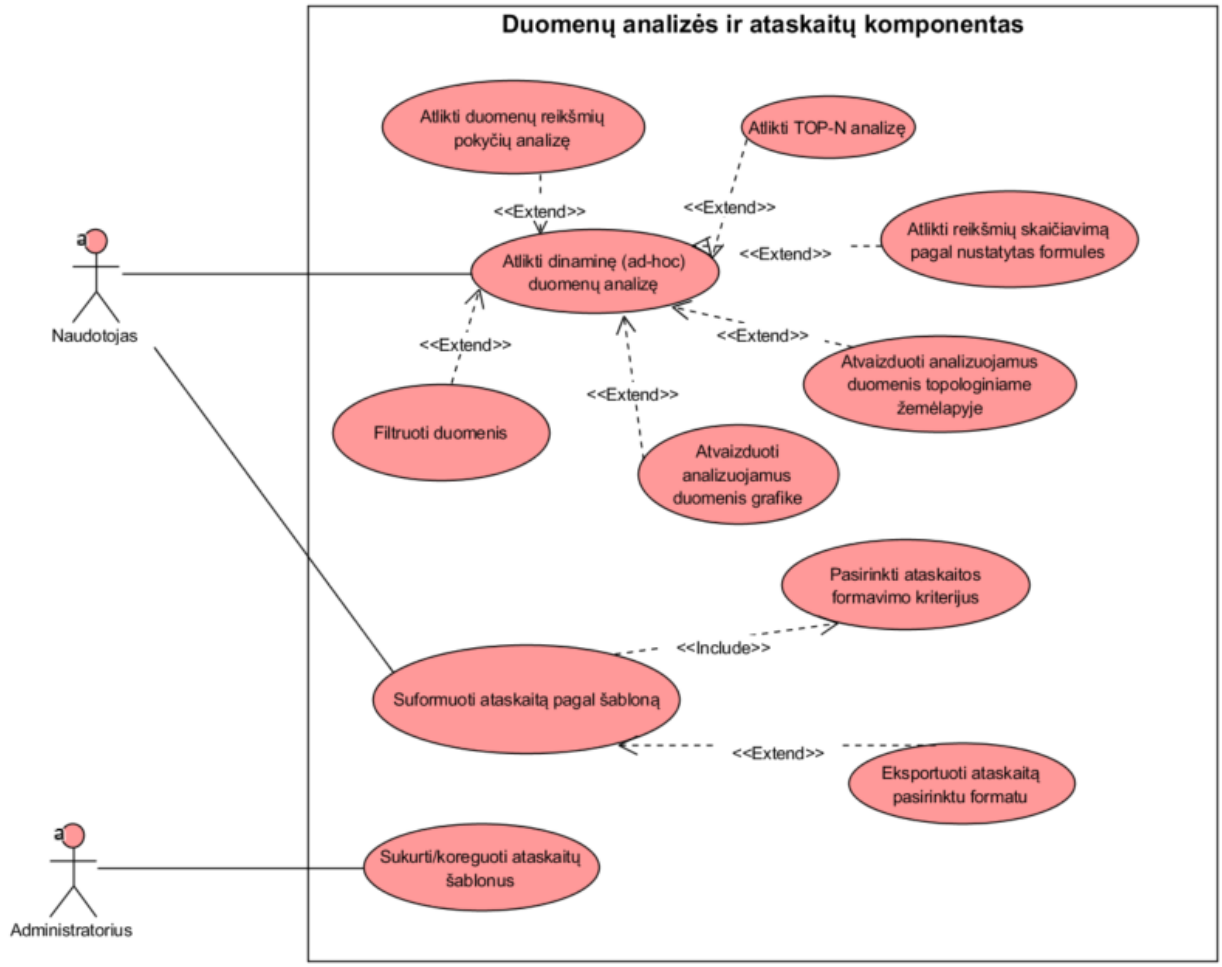
Žemiau esančioje lentelėje pateikiamas preliminarių tinklo topologinio žemėlapio panaudos atvejų sąrašas, jų aprašymai bei įvardijami TS punktai, kurie bus įgyvendinami kiekvienu panaudos atveju.

5 lentelė. Tinklo topologinio žemėlapio panaudos atvejai

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
1.	Peržiūrėti topologinį žemėlapį	Tai pagrindinis tinklo topologinio žemėlapio panaudos atvejis, t.y. naudotojas atsidaręs tinklo topologinį žemėlapį gali atlikti įvairius susijusius panaudos atvejus.	Naudotojas	3.2.1, 3.2.2, 3.2.3, 3.2.6, 3.2.7, 3.2.7.5, 3.2.7.6, 3.2.7.13, 3.2.7.14, 3.3,
2.	Rodyti/slėpti informacinius blokus	PA apima funkcionalumą, kuris suteikia galimybę rodyti/slėpti informacinius blokus bei redaguoti kriterijus, pagal kuriuos atrenkami duomenys šiems blokams. Taip pat sudaroma galimybė vienu pelės ir/arba klaviatūros klavišo paspaudimu paslėpti visus informacinius blokus, ekrane paliekant atvaizduotą tik topologinį žemėlapį.	Naudotojas	3.3.5, 3.4.
3.	Peržiūrėti svarbių įvykių sąrašą	PA skirtas peržiūrėti svarbių įvykių sąrašą, pvz.: BGP įspėjimai, stebimų svetainių pasiekiamumo sutrikimai ir pan.	Naudotojas	3.3.3, 3.3.3.1, 3.3.3.2.
4.	Filtruoti žemėlapyje atvaizduojamus objektus	PA skirtas atlikti topologiniame žemėlapyje pateikiamų objektų filtravimą pagal pasirinktus kriterijus: Turi būti galimybė rodyti (filtruoti) tik tam tikrus objektus ir ryšius su susijusiais objektais pagal pasirinktus kriterijus, pvz. pagal sektorių (pvz., finansų, energetikos, sveikatos priežiūros ir kt.), pagal ASN ir/arba ASN savininką, pagal ryšius su tam tikru objektu, pagal objekto atributus (pvz., tam tikro tipo incidentai).	Naudotojas	3.2.4, 3.2.5,
5.	Atlikti pasirinkto objekto greitą pagrindinių duomenų koregavimą	PA apima funkcionalumą, kurio pagalba bus galima pasirinkto objekto pagrindinius duomenis koreguoti rankiniu būdu.	Naudotojas	3.2.7.1.4.
6.	Realiuoju laiku gauti pasirinkto objekto statusą ar kitą naujausią informaciją	PA apima funkcionalumą, kurio pagalba galima realiuoju laiku gauti pasirinkto objekto statusą ar kitą naujausią informaciją, panaudojus vieną iš integruotų įrankių (pvz. ping, wget, traceroute ir pan.) arba BGP feed šaltinį (AS_PATH);	Naudotojas	3.2.7.1.3.
7.	Peržiūrėti su pasirinktu objektu susijusią informaciją	PA apima funkcionalumą, kurio pagalba galima greitai ir patogiai peržiūrėti su pasirinktu topologinio žemėlapio objektu susijusią informaciją: pagrindinę informaciją, išsamią informaciją.	Naudotojas	3.2.7.1.1, 3.2.7.1.2
8.	Parodyti pasirinkto objekto ryšius	PA skirtas parodyti pasirinkto objekto ryšius su kitais susijusiais objektais grafiškai atvaizduojant tik šią konkrečią tinko topologinio žemėlapio dalį tam skirtame informaciniame bloke arba atskirame lange.	Naudotojas	3.2.7.1.5.1, 3.2.7.1.5.2.
9.	Išjungti visų nesusijusių objektų ir ryšių atvaizdavimą	PA skirtas išjungti visų nesusijusių objektų ir ryšių atvaizdavimą aktyviame darbiniame žemėlapyje.	Naudotojas	3.2.7.1.5.3.
10.	Keisti topologinio žemėlapio mastelį	PA skirtas keisti atvaizduojamo tinklo topologinio žemėlapio mastelį – priartinti ir atitolinti vaizdą (angl. zoom in/out).	Naudotojas	3.2.7.2, 3.2.7.3, 3.2.7.4, 3.2.7.8.
11.	Keisti topologijos atvaizdavimo algoritmą	PA skirtas keisti topologinio žemėlapio atvaizdavimo algoritmą.	Naudotojas	3.2.7.7.
12.	Slankioti vaizdą	PA skirtas vaizdą slankioti (į viršų/apačią/kairę/dešinę) tiek naudojant klaviatūros klavišus, tiek naudojant pelės tempimą.	Naudotojas	3.2.7.9, 3.2.7.10.
13.	Eksportuoti rodomą vaizdą į vaizdinius formatus	PA skirtas eksportuoti rodomą žemėlapio vaizdą į vaizdinius formatus (JPG, PNG, PDF).	Naudotojas	3.2.7.11.
14.	Keisti objektų spalvas	PA apima funkcionalumą leidžiantį keisti visų žemėlapyje atvaizduojamų objektų spalvas pagal pasirinktus kriterijus (pvz. skirtingi sektoriai spalvinami skirtingai ir pan.).	Naudotojas	3.2.7.12.

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
15.	Analizuoti objektų ir ryšių pokyčius laike	PA skirtas analizuoti objektų ir jų ryšių pokyčius laike.	Naudotojas	5.5.

2.2.1.4 *Duomenų analizės ir ataskaitų panaudos atvejai*



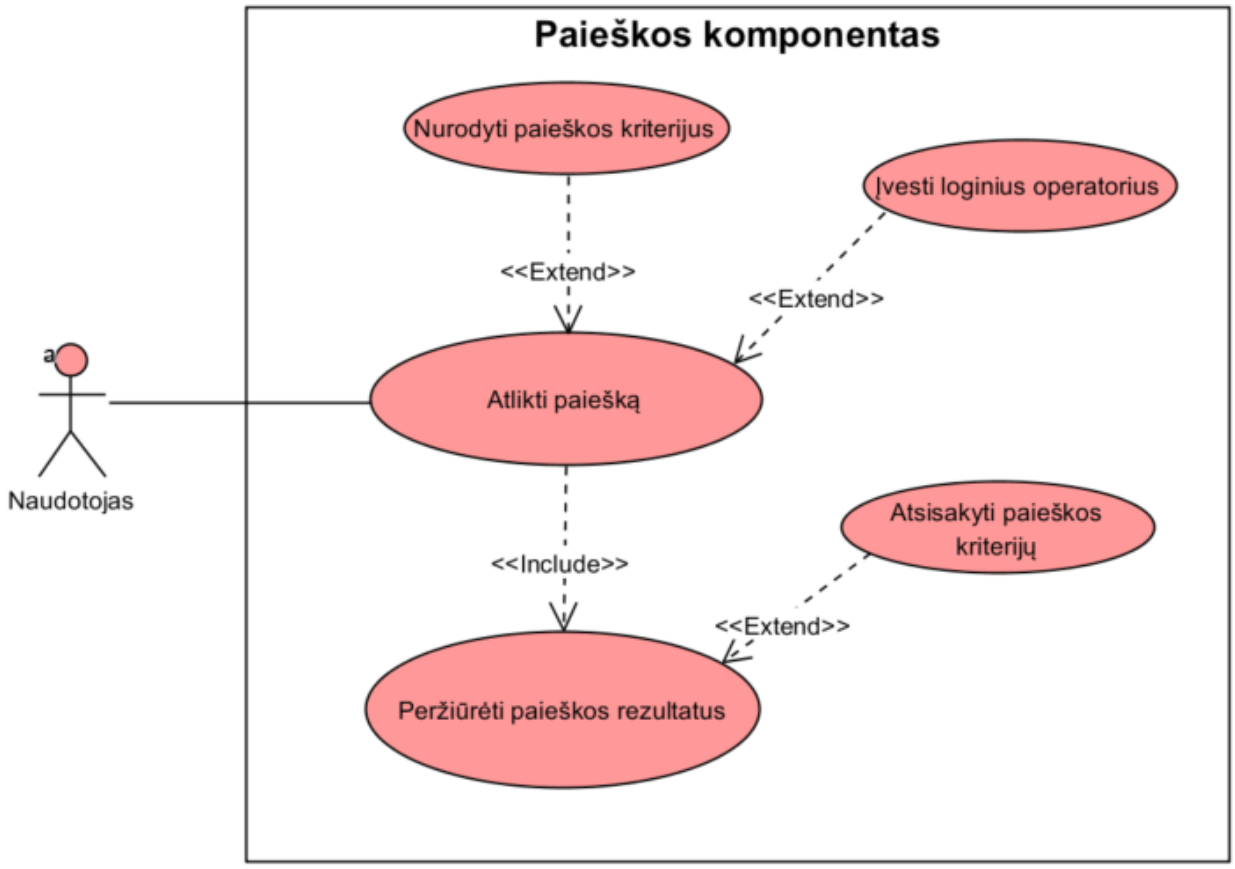
Žemiau esančioje lentelėje pateikiamas preliminarių duomenų analizės ir ataskaitų panaudos atvejų sąrašas, jų aprašymai bei įvardijami TS punktai, kurie bus įgyvendinami kiekvienu panaudos atveju.

6 lentelė. Duomenų analizės ir ataskaitų panaudos atvejai

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
1.	Atlikti dinaminę (ad-hoc) duomenų analizę	Tai PA apimantis daugiamačių duomenų analizės (OLAP) funkcionalumą. Dinaminės duomenų analizės funkcionalumo realizavimui numatoma naudoti Saiku CE komponentą, kurio detalesnis aprašymas pateiktas skyriuje „Dinaminės duomenų analizės komponentas – Pentaho Mondrian ir Saiku CE“	Naudotojas	5.1, 6.4.
2.	Atlikti duomenų reikšmių pokyčių analizę	Tai standartinis OLAP duomenų analizės funkcionalumas, kuomet pasirenkama laiko dimensija bei pageidaujamas rodiklis.	Naudotojas	5.4, 5.5.

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
3.	Atlikti TOP-N analizę	Tai standartinis OLAP duomenų analizės funkcionalumas, kuomet duomenų kube yra aprašyta TOP-N atrinkimo užklausa.	Naudotojas	5.6.
4.	Atlikti reikšmių skaičiavimą pagal nustatytas formules	Tai standartinis OLAP duomenų analizės funkcionalumas. Rodiklių skaičiavimo formulės turi būti aprašytos OLAP duomenų kubuose (MDX formato užklausa).	Naudotojas	5.2, 5.3
5.	Atvaizduoti analizuojamus duomenis topologiniame žemėlapyje	PA aprašantis analizės metu atrinktų objektų atvaizdavimui topologiniame žemėlapyje.	Naudotojas	5.7.1.
6.	Atvaizduoti analizuojamus duomenis grafike	PA aprašantis suformuotos dinaminės ataskaitos duomenų atvaizdavimą grafiniu pavidalu. Šio funkcionalumo bus realizavimui numatoma naudoti Saiku CE komponentą, kuris palaiko įvairių tipų grafikus (pvz.: sritinis (angl. area), juostinis (angl. bar), stulpelinis (angl. column), linijinis (angl. line), skritulinis (angl. pie), taškinis (angl. scatter) ir pan.).	Naudotojas	5.7.2, 5.7.3
7.	Filtruoti duomenis	Tai standartinis OLAP duomenų analizės funkcionalumas. Filtravimo funkcionalumo realizavimui numatoma naudoti Saiku CE komponentą, kurio detalesnis aprašymas pateiktas skyriuje „Dinaminės duomenų analizės komponentas – Pentaho Mondrian ir Saiku CE“	Naudotojas	5.8.
8.	Suformuoti ataskaitą pagal šabloną	PA apima ataskaitos suformavimą, pagal iš anksto apibrėžtą šabloną. Prieš formuodamas ataskaitą, naudotojas pasirenka ataskaitos formavimo kriterijus.	Naudotojas	6.4, 6.9.
9.	Pasirinkti ataskaitos formavimo kriterijus	PA apima ataskaitos formavimo kriterijų pasirinkimo funkcionalumą pvz. pasirenkamas datos intervalas.	Naudotojas	6.4.
10.	Eksportuoti ataskaitą pasirinktu formatu	PA apima funkcionalumą, kuris leidžia suformuotą ataskaitą išeksportuoti pasirinktu formatu pvz. JPG, PNG, PDF, HTML, CSV	Naudotojas	6.2, 6.8, 6.10.
11.	Sukurti/koreguoti ataskaitų šablonus	PA apima ataskaitos šablono kūrimo/koregavimo funkcionalumą. Šio funkcionalumo įgyvendinimui numatoma naudoti Pentaho Report Designer ataskaitų kūrimo ir redagavimo įrankį.	Administratorius	6.1, 6.5, 6.6, 6.7

2.2.1.5 Duomenų paieškos panaudos atvejai

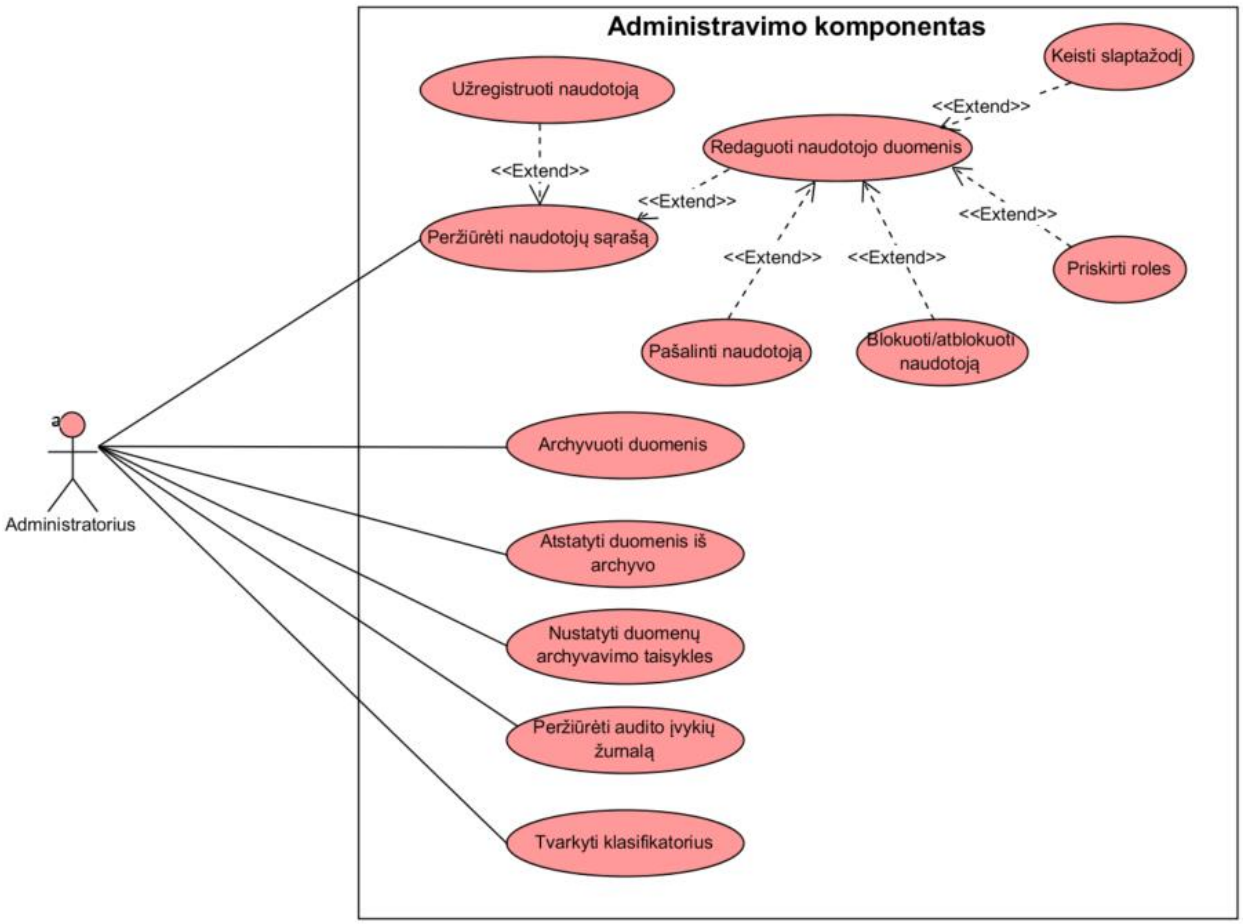


Žemiau esančioje lentelėje pateikiamas preliminarių paieškos panaudos atvejų sąrašas, jų aprašymai bei įvardijami TS punktai, kurie bus įgyvendinami kiekvienu panaudos atveju.

7 lentelė. Duomenų analizės ir ataskaitų panaudos atvejai

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
1.	Atlikti paiešką	PA apima tiek greitą paiešką (įvedant paieškos frazę), tiek ir detalią paiešką (nurodant paieškos kriterijus bei loginius operatorius).	Naudotojas	4.1, 4.1.4
2.	Nurodyti paieškos kriterijus	PA apima detalios paieškos funkcionalumą, kuomet naudotojas įveda pageidaujamus paieškos kriterijus pvz. pagal ID, ASN, Domeną, sektorių ir kt.	Naudotojas	4.1.1, 4.1.2, 4.1.6,
3.	Įvesti loginius operatorius	PA apima detalios paieškos funkcionalumą, kuris suteikia galimybę vykdyti paiešką naudojant sudėtinius loginius operatorius kaip kriterijus (pvz.: atitinkamas laukas turi būti daugiau už atitinkamą reikšmę ir kitas laukas turi būti ne tuščias). Numatoma naudoti bazinius loginius operatorius pvz. daugiau, mažiau, lygu ir kt.	Naudotojas	4.1.3.
4.	Peržiūrėti paieškos rezultatus	PA skirtas paieškos rezultatų peržiūrai tiek topologiniame žemėlapyje, tiek sąrašo pavidalu.	Naudotojas	4.2.
5.	Atsisakyti paieškos kriterijų	PA apima funkcionalumą, kuris leidžia atsisakyti paiešką siaurinančių kriterijų (pvz., spaudžiant "x" prie kriterijaus).	Naudotojas	4.1.5.

2.2.1.6 *Administravimo panaudos atvejai*



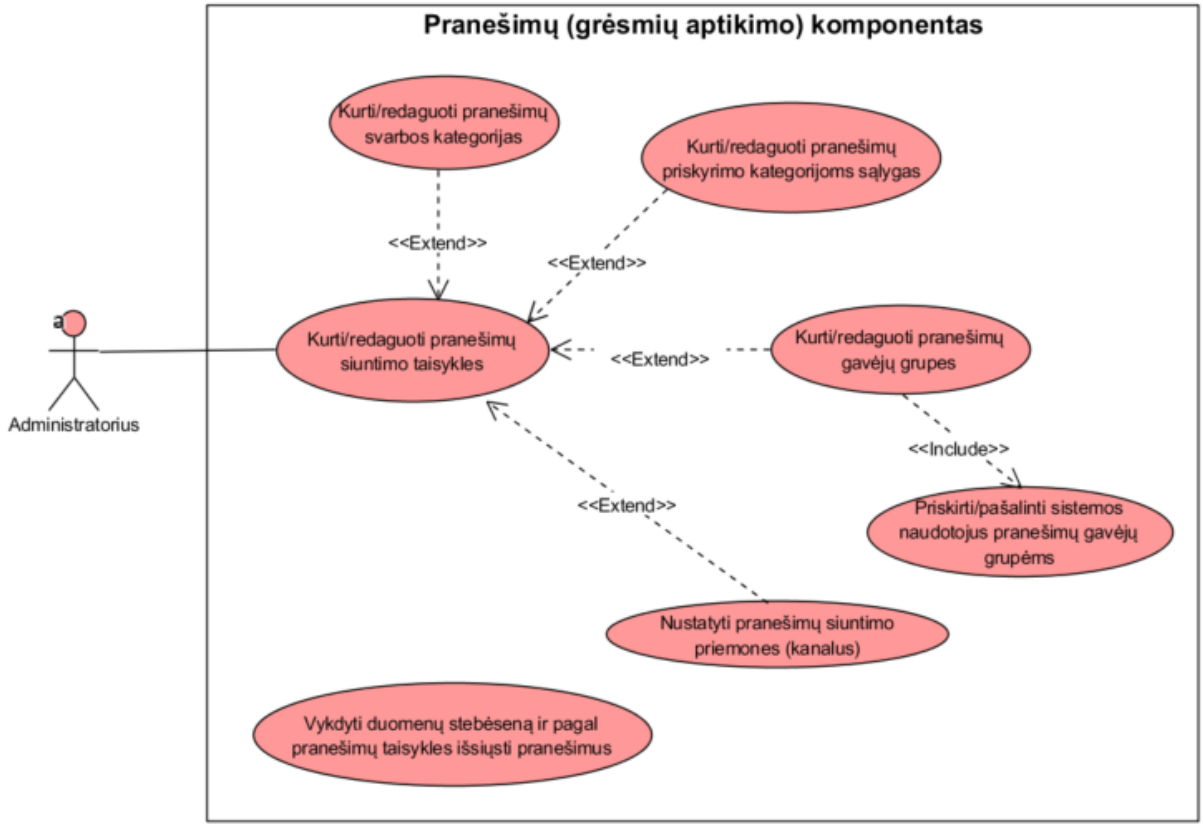
Žemiau esančioje lentelėje pateikiamas preliminarių administravimo panaudos atvejų sąrašas, jų aprašymai bei įvardijami TS punktai, kurie bus įgyvendinami kiekvienu panaudos atveju.

8 lentelė. Administravimo panaudos atvejai

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
1.	Peržiūrėti naudotojų sąrašą	PA skirtas naudotojų sąrašo peržiūros funkcionalumui.	Administratorius	8.4.
2.	Užregistruoti naudotoją	PA skirtas naujos naudotojo paskyros užregistravimui sistemoje.	Administratorius	8.4
3.	Redaguoti naudotojo duomenis	PA skirtas sistemos naudotojo duomenų redagavimui	Administratorius	8.4
4.	Keisti slaptažodį	PA skirtas slaptažodžio keitimo funkcionalumui. Numatoma, kad slaptažodį galės keisti tiek pats naudotojas, tiek ir administratorius.	Administratorius	8.4, 8.5
5.	Priskirti roles	PA apima funkcionalumą, kuris leidžia naudotojui priskirti roles (prieigos teisių rinkinius).	Administratorius	8.2, 8.3
6.	Blokuoti/atblokuoti naudotoją	PA skirtas naudotojo paskyros blokavimui/atblokavimui.	Administratorius	8.4
7.	Pašalinti naudotoją	PA skirtas naudotojo paskyros ištrynimui.	Administratorius	8.4
8.	Archyvuoti duomenis	PA skirtas tam tikro senumo duomenų archyvavimui. PA gali būti inicijuojamas tiek rankiniu būdu, tiek ir automatiškai, pagal nustatytas archyvavimo taisykles.	Administratorius Sistema	7.1

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
9.	Atstatyti duomenis iš archyvo	PA skirtas duomenų atstatymui iš archyvo.	Administratorius	7.2
10.	Nustatyti duomenų archyvavimo taisykles	PA skirtas archyvavimo taisyklių nustatymui pvz., po kiek laiko duomenys perkeliami į archyvą, kiek laiko saugomi archyve.	Administratorius	7.3
11.	Peržiūrėti audito įvykių žurnalą	PA skirtas audito įvykių peržiūrai ir paieškai. Audito įvykių žurnale bus kaupiami tiek sisteminiai įvykiai, tiek ir naudotojų atliekami veiksmai (prisijungimai, funkcijų vykdymai ir pan.).	Administratorius	8.8.
12.	Tvarkyti klasifikatorius	PA skirtas sistemoje naudojamų klasifikatorių tvarkymui, t.y. administratorius galės keisti arba šalinti esamas klasifikatorių reikšmes bei kurti naujas reikšmes.	Administratorius	-

2.2.1.7 *Pranešimų (grėsmių aptikimo) panaudos atvejai*



Žemiau esančioje lentelėje pateikiamas preliminarių pranešimų (grėsmių aptikimo) panaudos atvejų sąrašas, jų aprašymai bei įvardijami TS punktai, kurie bus įgyvendinami kiekvienu panaudos atveju.

9 lentelė. Pranešimų panaudos atvejai

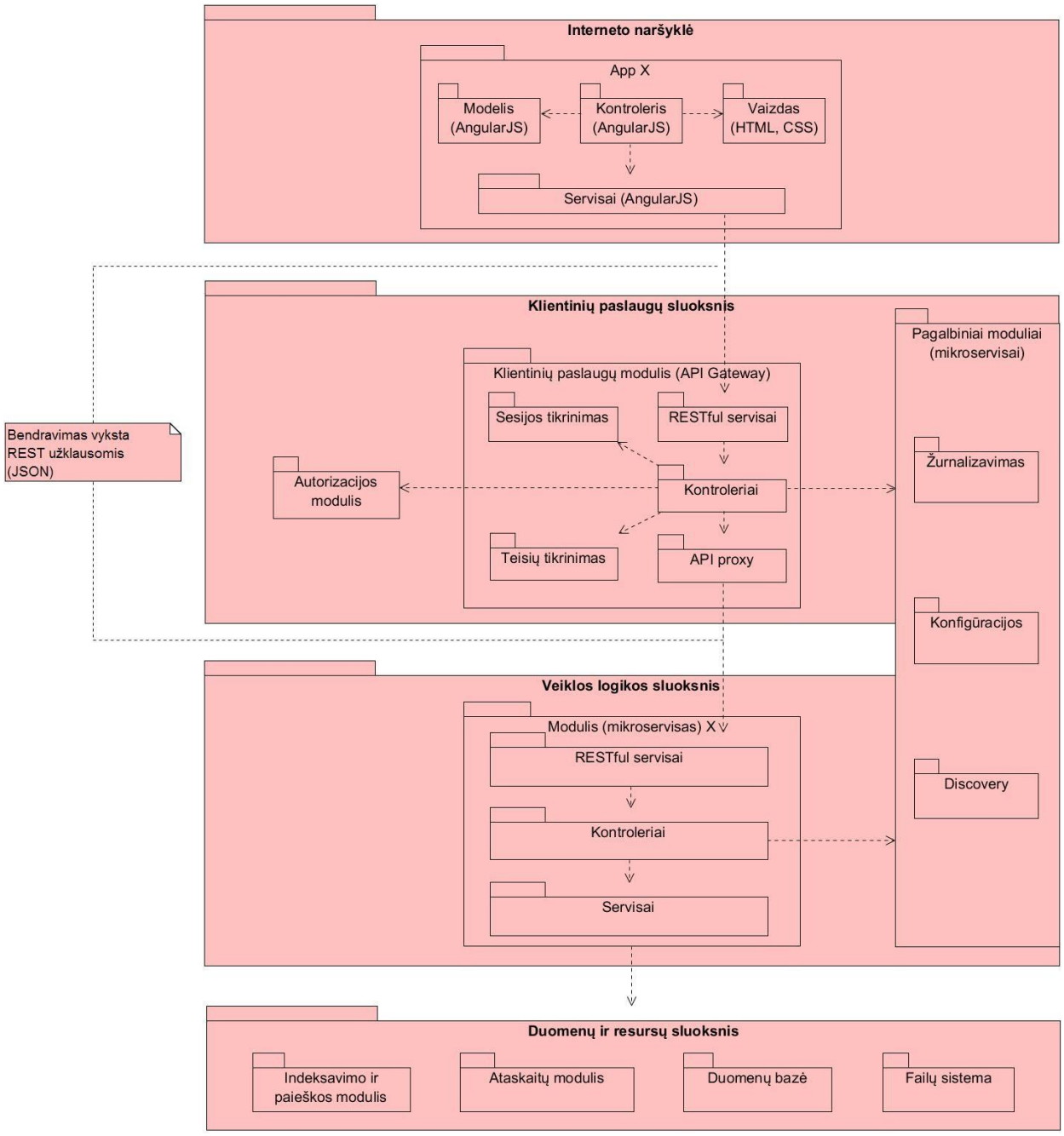
Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
1.	Kurti/redaguoti pranešimų siuntimo taisykles	Tai bendrinis PA, apimantis pranešimų siuntimo taisyklių kūrimo ir redagavimo funkcionalumą. Vykdant šį PA, galima atlikti žemiau aprašytus PA.	Administratorius	9.2, 9.2.3, 9.2.7

Nr.	Panaudos atvejis	Aprašymas	Vykdo	TS punkto Nr.
2.	Kurti/redaguoti pranešimų svarbos kategorijas	PA skirtas pranešimų svarbos kategorijų kūrimui ir redagavimui.	Administratorius	9.2.1
3.	Kurti/redaguoti pranešimų priskyrimo kategorijoms sąlygas	PA skirtas kurti/redaguoti pranešimų priskyrimo kategorijoms sąlygas.	Administratorius	9.2.2
4.	Kurti/redaguoti pranešimų gavėjų grupes	PA skirtas kurti/redaguoti panešimų gavėjų grupes.	Administratorius	9.2.4
5.	Priskirti/pašalinti sistemos naudotojus pranešimų gavėjų grupėms	PA apima funkcionalumą, kuris leidžia pranešimų gavėjų grupei priskirti/pašalinti pageidaujamus naudotojus.	Administratorius	9.2.5
6.	Nustatyti pranešimų siuntimo priemones (kanalus)	PA apima funkcionalumą, kuris leidžia nustatyti pranešimų siuntimo priemones: atvaizduoti pranešimą sistemoje, tam numatytoje vietoje, išsiųsti pranešimą el. laišku, išsiųsti pranešimą SMS žinute.	Administratorius	9.2.6.1, 9.2.6.2, 9.2.6.3.
7.	Vykdyti duomenų stebėseną ir pagal pranešimų taisykles išsiųsti pranešimus	Tai sisteminis PA, t.y. sistemos procesas vykdo nuolatinę duomenų stebėseną ir, susidarius sąlygoms, nurodytoms pranešimų taisyklėse, sugeneruoja ir išsiunčia pranešimą.	Sistema	9.1

2.2.2 Loginis architektūros pjūvis

2.2.2.1 Loginiai sluoksniai

Planuojama, kad sistema bus realizuojama laikantis daugiasluoksnės (angl. *N-Layer*) architektūros principų, t.y. sistemos realizacija, kai naudotojo sąsaja, veiklos logika ir duomenų saugojimas bei prieiga yra skirtinguose loginiuose sluoksniuose. Atkreipiame dėmesį į tai, kad lygių atskyrimas yra loginis ir nebūtinai padalintas į atskirus fizinius serverius. Žemiau esančiame paveiksle pateikiama apibendrinta sistemos loginių sluoksnių schema.



6 paveikslas. Loginių sluoksnių diagrama

10 lentelė. Loginiai sistemos sluoksniai bei komponentai ir jų aprašymai

Sluoksnis / komponentas	Aprašymas
Interneto naršyklė	Sluoksnis apima sistemos komponentus veikiančius naudotojo interneto naršyklėje. Šio sluoksnio pagrindą sudaro atvirojo kodo JavaScript karkasas *AngularJS*, kuris suteikia galimybę realizuoti MVC (angl. *Model-View-Controller*) architektūros principus kliento pusėje.
App X	Aplikacija veikianti interneto naršyklėje, kurioje su JavaScript karkaso AngularJS pagalba realizuoti MVC architektūros principai. Aplikacija skirta komunikuoti su klientu ir perduoti kliento užklausas į klientinių paslaugų sluoksnį.

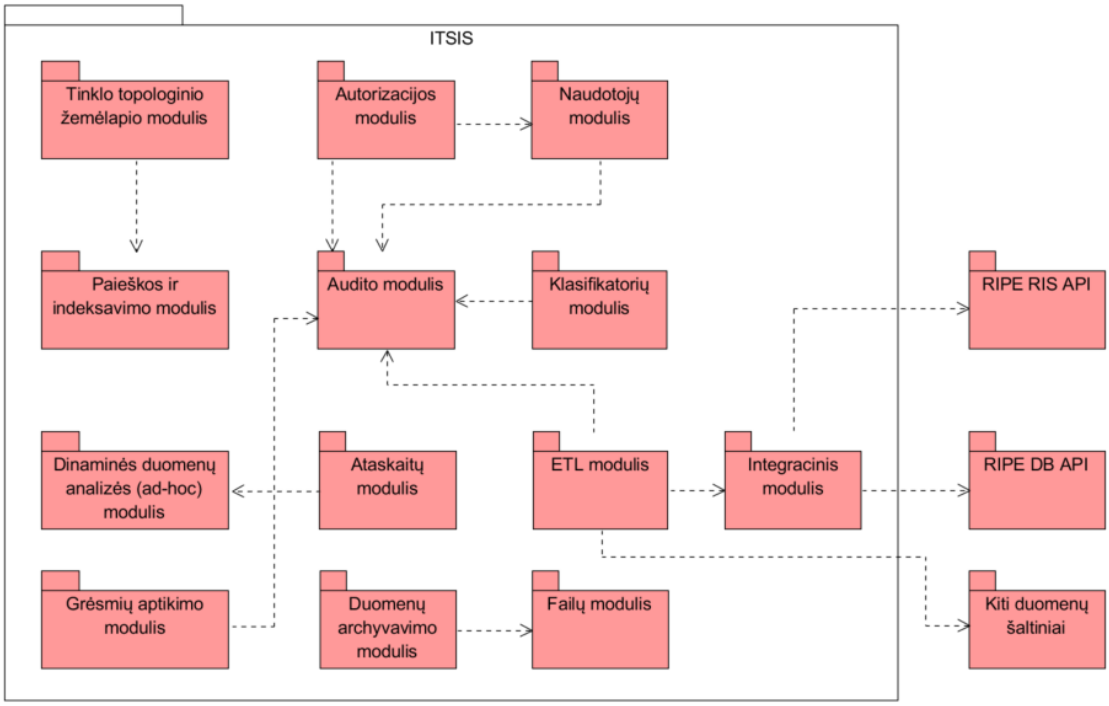
Sluoksnis / komponentas	Aprašymas
Modelis (AngularJS)	Modelyje esantys objektai apima su aplikacija susijusius duomenis bei duomenų apdorojimo logiką. Naudotojo veiksmai iš vaizdo komponento yra perduodami kontroleriui, kuris savo ruožtu atnaujina modelio objektą. Pasikeitus modelio objektui, vaizdas taip pat yra atnaujinamas.
Kontroleris (AngularJS)	Kontrolerio objektas veikia kaip tarpininkas tarp vaizdo ir modelio objektų. Kontroleriai reaguoja į naudotojo veiksmus naršyklėje veikiančioje aplikacijoje, asinchroniškai kviečia RESTful web servisus ir valdo duomenų modelį. Duomenų modelio pasikeitimai yra automatiškai pateikiami kliento naršyklėje.
Vaizdas (HTML, CSS)	Vaizdas yra HTML šablonai, į kuriuos AngularJS integruoja modelį ir sugeneruoja vaizdą pateikiamą kliento naršyklėje. Vaizdo objektas žino kaip atvaizduoti modelyje esančius duomenis ir kaip reaguoti į naudotojo veiksmus. Vaizdo komponento pagrindinė užduotis yra tvarkingai pavaizduoti modelyje esančius duomenis, pritaikyti formatus datos, skaičiaus, valiutos ir kt. laukams. Vaizdo komponentas neatlieka veiklos logikos funkcijų, visi naudotojo veiksmai iš karto yra perduodami kontroleriams.
Servisai (AngularJS)	Servisai realizuojami kliento naršyklėje veikiančioje aplikacijoje užtikrina komunikaciją su serveryje esančiais RESTful servisais. Pagrindinė serviso komponento užduotis yra iškviesti serveryje esančius RESTful servisus ir naudojantis asinchroniniais *Promise* objektais grąžinti rezultatus kontroleriams. Taip pat servisai atlieka kitas pagalbines funkcijas, kaip globalių duomenų (autentifikacijos, kalbos ir pan.) valdymą.
Klientinių paslaugų sluoksnis	Tai sistemos sluoksnis veikiantis web aplikacijų serveryje. Klientinių paslaugų lygmenyje yra apdorojamos iš kliento gautos užklausos, kviečiami servisai duomenų gavimui ir išsaugojimui, atliekamos autentikavimo, autorizavimo ir kitos bendros funkcijos.
Klientinių paslaugų modulis (API Gateway)	Šiame modulyje yra realizuoti pagrindiniai aplikacijos veikimo logikos komponentai (kontroleriai).
RESTful servisai	Klientinių paslaugų modulio RESTful servisai skirti komunikacijai tarp kliento naršyklėje veikiančioje aplikacijoje esančių servisų ir serverio pusėje veiklos logiką realizuojančių kontrolerių.
Kontroleriai	Kontroleriai - tai Java klasės realizuojančios aplikacijos veikimo logiką. Kiekvienas kontroleris suteikia vieną ar daugiau metodų, skirtų konkrečiai funkcijai atlikti. Dažnu atveju klientinių paslaugų lygmenyje esančių kontrolerių sąsaja atkartoja veiklos logikos lygmenyje esančių kontrolerių sąsają.
API proxy	API proxy - tai tarpinis komponentas, kuris kontrolerių užklausas perduoda į veiklos logikos lygmenį. Perdavimui naudojamos REST užklausos.
Sesijos tikrinimas	Tai modulis, kurio paskirtis yra identifikuoti naudotoją, kuris atsiuntė į aplikaciją užklausą. Naudotojas gali būti tiek autentifikuotas registruotas naudotojas, tiek anoniminis.
Teisių tikrinimas	Modulio paskirtis yra patikrinti, ar naudotojas turi prieigos teises prie tinklapio, bei kviesti užklausoje nurodytą RESTful servisą. Tolimesni tikrinimai (pvz: ar naudotojas turi teisę prieiti prie administratoriaus duomenų) nėra išskirti į atskirą komponentą ir yra atliekami veiklos logikos sluoksnyje.
Autorizacijos modulis	Autorizacijos modulis yra atsakingas už naudotojo prisijungimą prie sistemos.
Pagalbiniai moduliai (mikroservisai)	Pagalbiniai moduliai nepriklauso tik vienam loginiam sluoksniui. Į šiuos mikroservisus kreipiamasi ir iš klientinių paslaugų sluoksnio, ir iš veiklos logikos sluoksnio. Pagalbiniai moduliai leidžia pakartotinai panaudoti tam tikras sistemos funkcijas.
Žurnalizavimas	Šio bendrai naudojamo modulio pagalba žurnalizuojami sistemoje atliekami veiksmai. Komponentas žurnalizuoja, koks naudotojas, kokią funkciją iškvietė, bei koks buvo funkcijos rezultatas (sėkmingas, nesėkmingas).
Konfigūracijos	Konfigūracijų mikroservisas yra atsakingas už sistemos konfigūracijų saugojimą, apdorojimą ir taikymą.
Discovery	Discovery mikroservisas yra atsakingas už servisų registraciją ir užklausų paskirstymą į prieinamus servisus.
Veiklos logikos sluoksnis	Veiklos logikos sluoksnis veikia aplikacijų serveryje. Šiame lygmenyje esančiuose moduliuose (mikroservisuose) realizuotos pagrindines sistemos veiklos funkcijas įgyvendinančios klasės (kontroleriai). Kontroleriai įvairioms funkcijoms atlikti kviečia servisus, kurie užtikrina komunikaciją su duomenų ir resursų sluoksnyje esančiais komponentais (duomenų bazė, failų sistema ir pan.) bei trečiųjų šalių programiniais komponentais (ataskaitos, paieškos ir pan.).
Modulis (mikroservisas) X	Moduliuose (mikroservisuose) yra realizuoti sistemos veiklos funkcijas įgyvendinantys kontroleriai. Vienas mikroservisas yra atsakingas tik už tam tikrą sistemos veiklos sritį ir įgyvendina tik su ta sritimi susijusias funkcijas.
RESTful servisai	Veiklos logikos lygmenyje realizuotuose mikroservisuose esantys RESTful servisai skirti išskirti esamą sistemos funkcionalumą tarp klientinių paslaugų ir veiklos logikos lygmenyse esančių

Sluoksnis / komponentas	Aprašymas
	kontrolerių. Duomenų perdavimui tarp klientinių paslaugų ir veiklos logikos lygmenų naudojamos REST užklausos.
Kontroleriai	Kontroleriai - tai Java klasės realizuojančios aplikacijos veikimo logiką. Kiekvienas kontroleris suteikia vieną ar daugiau metodų, skirtų konkrečiai funkcijai atlikti.
Servisai	Servisai atlieka veiklos logikos veiksmus. Jie atlieka nurodytus naudotojų veiksmus, komunikuoja su duomenų baze ir atlieka kitus veiksmus.
Duomenų ir resursų sluoksnis	Duomenų ir resursų sluoksnyje yra trečiųjų šalių programiniai komponentai (ataskaitos, paieškos ir pan.) bei saugomi visi sistemos duomenys. Šis lygmuo apima tiek failų sistemą, tiek ir reliacines duomenų bazes.
Indeksavimo ir paieškos modulis	Indeksavimą ir paieškos užklausų apdorojimą atliekantis atvirojo kodo komponentas.
Ataskaitų modulis	Ataskaitų generavimui skirtas trečiųjų šalių programinis komponentas.
Duomenų bazė	Duomenų bazėje saugomi su failais susieti metaduomenys, naudotojų ir organizacijų duomenys, audito žurnalai ir kita sisteminė informacija. Sistemos realizacijai bus naudojama Postgre SQL duomenų bazių valdymo sistema.
Failų sistema	Failų sistema užtikrina duomenų objektų (failų, katalogų, jų versijų) saugojimą

2.2.2.2 *Loginiai moduliai*

Šiame skyriuje pateikiamas sistemos funkcijas realizuojančių programinių komponentų modelis. Kaip minėta ankstesniame skyriuje, visos pagrindinės sistemos funkcijos ir veiklos logika yra realizuojamos naudojant kontrolerius, todėl pagrindiniai vienetai charakterizuojantys loginio modulio vykdomas funkcijas yra jį sudarančios kontrolerių klasės.

Siūlomą sprendimą sudaro tarpusavyje integruoti moduliai, atitinkantys įvairias funkcines veiklos sritis. Tai leidžia suskirstyti programinį kodą į logines dalis, kurias lengva plėsti ir tobulinti neįtakojant kitų dalių veikimo. Žemiau esančiame paveiksle pavaizduoti sistemos loginiai moduliai ir jų tarpusavio sąryšiai.



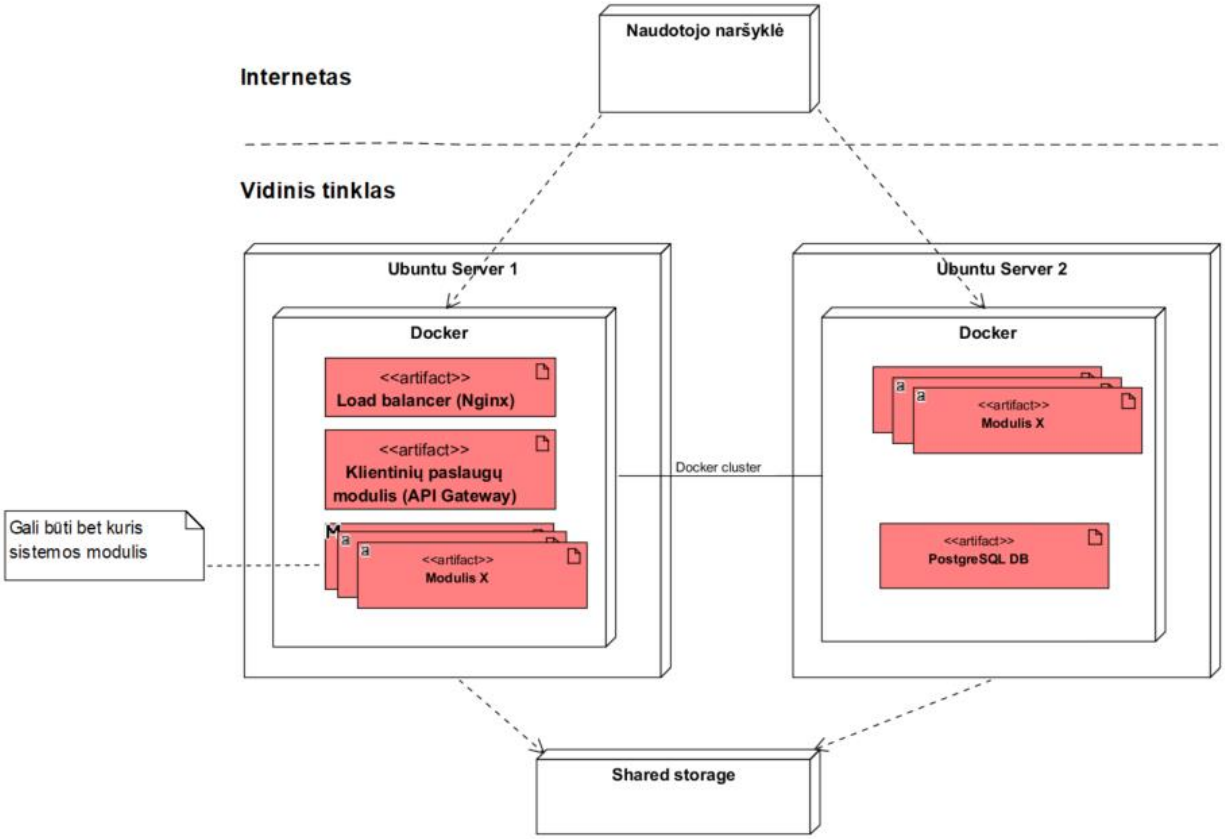
7 paveikslas. Loginių modulių diagrama

11 lentelė. Loginiai sistemos moduliai ir jų aprašymai

Eil. Nr.	Modulio pavadinimas	Aprašymas
1.	Naudotojų modulis	Naudotojų modulyje realizuojamos funkcijos, reikalingos naudotojų administravimui. Modulyje bus sukurtos kontrolerių klasės atsakingos už naudotojų teisių ir vaidmenų priskyrimą, sukūrimą, ištrynimą. Šis modulis palaikys vaidmenimis paremtą naudotojų teisų administravimo modelį. Kiekvienam vartotojui bus priskirtas vaidmuo. Kiekvienas vaidmuo turės jam priskirtas teises ar apribojimus.
2.	Autorizacijos modulis	Autorizacijos modulyje realizuojamos funkcijos, reikalingos naudotojų autorizacijai. Šis modulis atsakingas už naudotojo prisijungimą prie sistemos.
3.	Tinklo topologinio žemėlapio modulis	Modulis atsakingas už tinklo topologinio žemėlapio atvaizdavimui reikiamų duomenų pateikimą.
4.	Klasifikatorių modulis	Klasifikatorių modulyje realizuojamos funkcijos, reikalingos klasifikatorių administravimui. Šiame modulyje bus realizuotos funkcijos skirtos klasifikatorių reikšmių redagavimui, ištrynimui ir peržiūrai.
5.	Audito modulis	Audito modulyje realizuojamos funkcijos, reikalingos sistemos įvykių žurnalizavimui. Šio modulio funkcionalumu naudosis visi kiti moduliai. Audito modulyje bus žurnalizuojami registracijos bei autentifikavimo veiksmai, įvykusios sistemos klaidos ar sutrikimai, duomenų įkėlimo, keitimo, naikinimo ir panašūs faktai. Žurnaluose užregistruota informacija bus saugoma nustatytą laiko tarpą taip kad nebūtų galima jos ištrinti ar pakeisti, neturint tam suteiktų teisių.
6.	Integracinis modulis	Integracinis modulis atsakingas už sistemos bendravimą su RIPE DB API ir RIPE RIS API bei esant poreikiui su kitais duomenų šaltiniais. Integracinis modulis turės suformuoti tinkamas užklausas, kurias nusiuntęs į išorinę sistemą, gautų duomenis, ir apdoroti gautus duomenis, kad jie būtų tinkami išsaugoti.
7.	ETL modulis	Šis modulis surenka duomenis iš išorinių duomenų šaltinių, juos apdoroja (transformuoja), paruošia analizei ir įkelia į Sistemos duomenų saugyklos struktūrą.
8.	Dinaminės duomenų analizės (ad-hoc) modulis	Modulio pagrindą sudarys OLAP (angl. *On-Line Analytical Processing*) programiniai komponentai.
9.	Ataskaitų modulis	Ataskaitų modulyje realizuojamos funkcijos, reikalingos ataskaitų administravimui ir generavimui. Ataskaitų modulis leis generuoti ataskaitas pagal pasirinktus kriterijus. Administratoriui bus suteikta galimybė pasirinkti, kokia informacija pateikiama ataskaitoje ir kaip atvaizduojama. Ataskaitų modulis leis išsaugoti sugeneruotas ataskaitas pasirinktu formatu.
10.	Grėsmių aptikimo modulis	Modulyje realizuojami automatiniai procesai, kurie vykdys nuolatinę duomenų stebėseną ir, susidarius sąlygoms, nurodytoms pranešimų taisyklėse, sugeneruos ir išsiųs pranešimus. Administratoriams taip pat bus sudaryta galimybė kurti bei redaguoti pranešimų siuntimo taisykles.
11.	Paieškos ir indeksavimo modulis	Modulyje realizuojamos funkcijos, reikalingos Sistemoje saugomų duomenų indeksavimui ir paieškai atlikti. Modulis leis vykdyti tiek greitą paiešką, įvedus tik paieškos frazę, tiek ir detalią paiešką pagal konkrečius paieškos kriterijus.
12.	Duomenų archyvavimo modulis	Modulyje realizuojamos funkcijos, kurios leis atrinkti nebeaktualius Sistemos duomenis ir juos suarchyvuoti. Administratoriui taip pat bus sudaryta galimybė apibrėžti duomenų archyvavimo taisykles, pagal kurias duomenys būtų archyvuojami automatiškai.
13.	Failų modulis	Failų modulyje realizuojamos funkcijos, reikalingos failų administravimui. Failų modulio esminės funkcijos yra failų saugojimas ir ištrynimas.

2.2.3 Diegimo architektūros pjūvis

Žemiau esančiame paveiksle pavaizduota programinės įrangos diegimo (angl. *deployment*) schema.



8 paveikslas. Sistemos diegimo schema

Pagrindiniai siūlomos techninės architektūros aspektai:

- Programinę įrangą rekomenduojame diegti dviejuose serveriuose. Abiejuose serveriuose būtų įdiegta Ubuntu Server operacinė sistema bei konteinerių valdymo programinė įranga Docker, kuri suteiks galimybę patogiau valdyti mikroservisų infrastruktūrą bei prireikus virtualiai atskirti tinklo zonas.
- Abu serveriai būtų apjungti į Docker klasterį, kuris apsaugotų nuo techninės įrangos gedimų, t.y. fizinės tarnybinės stoties gedimo atveju Docker programinė įranga užtikrins automatinį jame veikusių konteinerių paleidimą kitoje fizinėje tarnybinėje stotyje.
- Siekiant užtikrinti sukurtos programinės įrangos veikimą, kūrimui bus naudojamos naujausios programinių paketų versijos, kurios yra palaikomos ir jas palaikantis gamintojas nėra anonsavęs palaikymo pabaigos.
- Naudojama programinė įranga ir architektūriniai sprendimai, kurie leidžia sistemos pajėgumų plėtimą, prijungiant papildomą techninę įrangą (angl. *scaling*). Nebus ribojamos galimybės didinti sistemos našumą, t.y. sistemos našumui padidinti užteks aparatinės įrangos papildymo.
- Komunikacija tarp kliento naršyklės ir sistemos vykdoma HTTPS protokolu.

2.3 Sistemos realizacijai siūlomos programinės priemonės ir technologiniai sprendimai

Paslaugų teikėjo pasirinkti ir šiame pasiūlyme aprašyti pagrindiniai programinės įrangos komponentai (bibliotekos, karkasai, aplikacijų serveriai ir pan.) yra gerai žinomi rinkoje ir turi dideles palaikymo bendruomenes. Sistema bus įgyvendinama panaudojant atviro kodo sprendimus (tiek programinio kodo, tiek programinio kodo kūrimo priemonėms). Toks Paslaugų teikėjo pasirinkimas užtikrina, kad:

- *nebus apribojimų programinę įranga norint panaudoti kitose ES šalyse;*
- *naudotojų skaičius nebus ribojamas licencijomis;*
- *apdorojamos informacijos apimtys nebus ribojamos licencijomis;*
- *programinės įrangos licencijos yra neriboto galiojimo laikotarpio ir turi kitus būtinus leidimus naudoti programinę įrangą;*
- *licencijos yra nuolatinio galiojimo (angl. Perpetual), o ne pateikiamos nuomos ar panašiu teisiniu pagrindu, ar su kitokiu jų galiojimo apribojimu laike.*

2.3.1 Duomenų surinkimo ir apdorojimo komponentas - Pentaho data integration

Duomenų surinkimo ir apdorojimo komponentas šio pasiūlymo kontekste suprantamas, kaip informacinių technologijų sprendimų rinkinys, kurio paskirtis – užtikrinti duomenų paėmimą iš išorinių šaltinių, įgyvendinti reikiamą loginę kontrolę, vykdyti duomenų apdorojimą, susisteminimą ir išsaugojimą. Šis komponentas surenka duomenis iš išorinių duomenų šaltinių, juos apdoroja (transformuoja), paruošia analizei ir įkelia į Sistemos duomenų saugyklos struktūrą. Duomenų transformavimas yra vienas iš svarbiausių procesų duomenų surinkimo ir analizės sistemose, tai etapas kai surinkti duomenys yra pritaikomi veiklos analizės poreikiams. Pagrindiniai Sistemos duomenų šaltiniai bus BGB feeds bei RIPE NCC duomenų bazė, todėl projekto metu bus realizuotos integracinės sąsajos bei duomenų paėmimo, transformavimo ir pakrovimo procesai (angl. *Extract Tranform Load - ETL*), kurie iš išorinių sistemų gaunamus duomenis, transformuoja ir įkelia į Sistemos duomenų bazę.

Kaip pagrindinį komponentą duomenų mainų posistemio realizacijai siūlome naudoti *Pentaho Data Integration Community Edition - PDI* programinę įrangą, kuri suteikia galimybę integruoti įvairius duomenų šaltinius, tokius kaip duomenų bazės, failai, išorinės sistemos bei iš šių šaltinių gautus duomenis transformuoti į vieningą formatą, tinkantį analizei. PDI programinė įranga turi duomenų surinkimo, transformavimo ir pakrovimo mechanizmus (angl. *Extraction Transformation Loading - ETL*), kurių pagalba galima realizuoti sudėtingus integracinius procesus. PDI taip pat turi grafinę naudotojo sąsają, skirtą ETL procesų projektavimui.

PDI palaiko įvairius duomenų saugyklos užkrovimo principus, tokius kaip lėtai besikeičiančios dimensijos (angl. *Slowly Changing Dimensions - SCD*), dirbtiniai raktai (angl. *surogate key*) bei yra pakankamai pajėgus apdorojant didžiulius duomenų kiekius. PDI suteikia galimybę duomenų apdorojimui naudoti tiek labai paprastas, tiek ir sudėtingas, kompleksines transformacijas.

PDI palaiko plačiai paplitusius integracinius standartus ir technologijas, pavyzdžiui transporto lygmenyje palaikomi protokolai ir technologijos: HTTP/S, SOAP, FTP, WSDL, TCP ir pan., duomenų formatų lygmenyje palaikomi standartai ir protokolai: CSV, JSON, XML ir pan. Gebėjimas apdoroti skirtingais formatais ir protokolais siunčiamas žinutes yra vienas iš pagrindinių PDI privalumų, leidžiančių tarpusavyje integruoti skirtingose technologinėse platformose veikiančias sistemos dalis ar išorines informacines sistemas.

Dažniausi PDI programinės įrangos panaudojimo atvejai:

- duomenų migravimas tarp skirtingų duomenų bazių ir sistemų;
- didžiulių duomenų kiekių pakrovimas į duomenų bazę, pasitelkiant paralelinio duomenų apdorojimo galimybes;
- duomenų išvalymas ir transformavimas;
- integravimas su išoriniais duomenų šaltiniais;
- duomenų sandėlio užkrovimas.

2.3.1.1 PDI architektūra

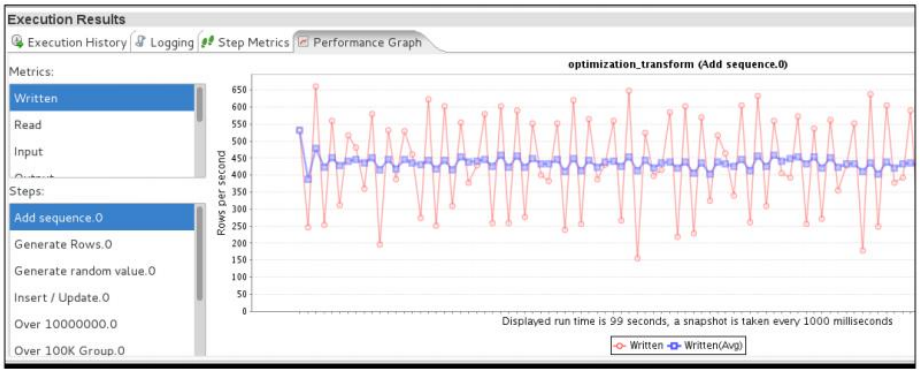
Pentaho Data Integration platforma susideda iš šių pagrindinių komponentų:

- **Spoon –** tai instaliuojama aplikacija, kuri suteikia galimybę grafinės naudotojo sąsajos pagalba kurti ir konfigūruoti ETL procesus. Šis įrankis suteikia galimybę nerašant programinio kodo, „drag and drop“ principu, sukurti gana sudėtingas ETL užduotis. *Spoon* įrankis naudojamas kiekvieną karta norint sukurti, redaguoti ar ištestuoti ETL procesą. Su Spoon aprašytos transformacijos ir užduotys gali būti vykdomos lokaliai pačiame įrankyje arba serveryje.
- **Pan –** tai komponentas (procesas), kuris vykdo su *Spoon* įrankiu sukurtas ETL transformacijas. Šis komponentas gali nuskaityti bei įrašyti duomenis į įvairius šaltinius.
- **Kitchen –** tai komponentas (procesas), kuris vykdo su *Spoon* įrankiu sukurtas ETL užduočių sekas. Užduočių sekų aprašai gali būti saugomi XML failuose arba duomenų bazėje. Dažniausiu atveju sukonfigūruojamas periodinis užduočių vykdymas**.**
- **Carte –** tai serveryje diegiamas WEB konteineris, kuriame gali būti vykdomos ETL užduočių sekos ir transformacijos**.**

Pagrindiniai PDI platformos privalumai:

- atvirojo kodo programinė įranga;
- patogi integracinių procesų kūrimo naudotojo sąsaja, turinti virš 100 paruoštų integracinių objektų, skirtų duomenų paėmimui, patikrinimui, transformavimui bei išsaugojimui;
- nesudėtingas programinės įrangos atnaujinimas;
- palaiko įskiepius (angl. *Plug-in*), kuriuos galima parsisiųsti iš centralizuotos saugyklos arba kurti patiems;
- programinę įrangą kuria ir testuoja didelė tarptautinė bendruomenė;
- palaiko į paslaugas orientuotos architektūros principus (SOA), kas suteikia galimybę PDI nesunkiai integruoti į kuriamas informacines sistemas;
- palaiko tokius standartus ir technologijas kaip AJAX, JDBC, MDX, SQL ir pan.;
- tai Java kalba parašyta programinė įranga, todėl pilnai gali veikti įvairiose operacinėse sistemose - Windows, Linux bei Macintosh;
- suteikia galimybę vienos transakcijos ribose vykdyti kelias ETL užduotis ir transformacijas;
- užtikrina didelę ETL procesų greitaveiką bei plečiamumo galimybes.

Spoon įrankis taip pat turi integruotas priemones, skirtas užduočių ir transformacijų greitaveikos analizei ir optimizavimui. Žemiau esančiame paveiksle pateikiamas proceso greitaveikos grafiko pavyzdys:

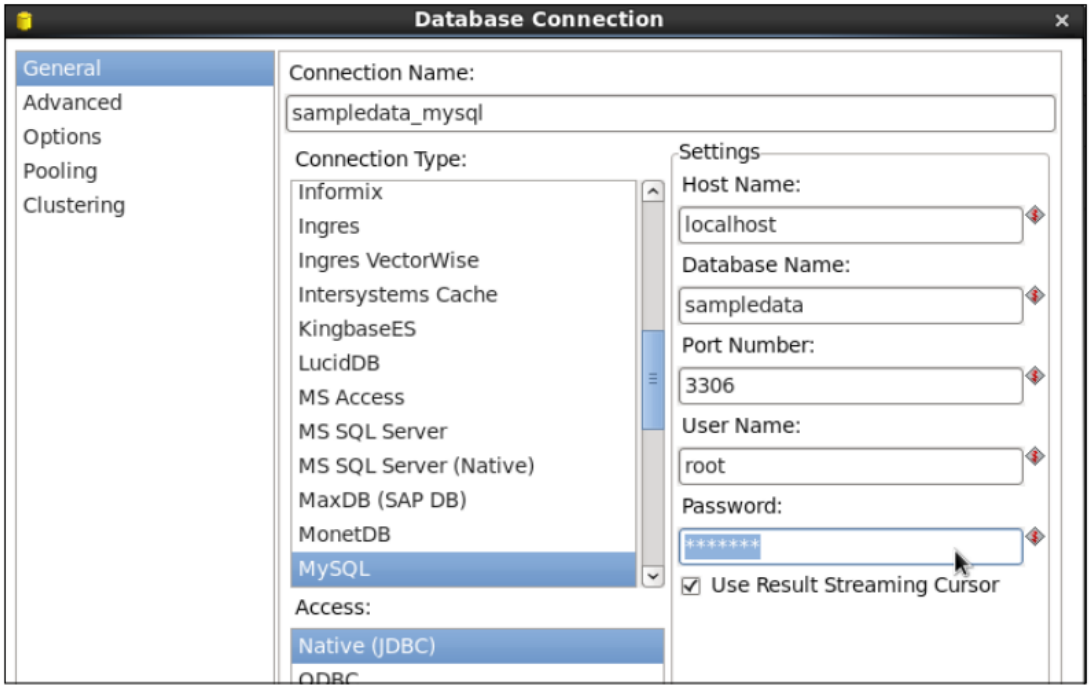


9 paveikslas. ETL proceso greitaveikos grafikas

2.3.1.2 Duomenų šaltiniai

PDI platforma palaiko įvarius duomenų šaltinius, įskaitant tekstinius, XML, Excel failus, komercines DBVS sistemas (pvz. *Oracle*), atvirojo kodo DBVS (pvz. *PostgreSQL*), tradicines eilutėmis paremtas (angl. *row-based*) DBVS, naujos kartos stulpeliais paremtas (angl. *column-based*) DBVS (pvz. *Infobright*), *in-memory* tipo DBVS (pvz. *HyperSQL*), NoSQL tipo DBVS (pvz. *MongoDB*) ir t.t. Verta pažymėti ir tai, kad *Pentaho* buvo vienas iš pirmųjų ETL įrankių, palaikančių didelių duomenų kiekių apdorojimo platformą *Hadoop*.

Prisijungimas prie duomenų šaltinio yra aprašomas *Spoon* įrankyje. Žemiau esančiame paveiksle matomas prisijungimo prie reliacinės duomenų bazės konfigūracijos langas:



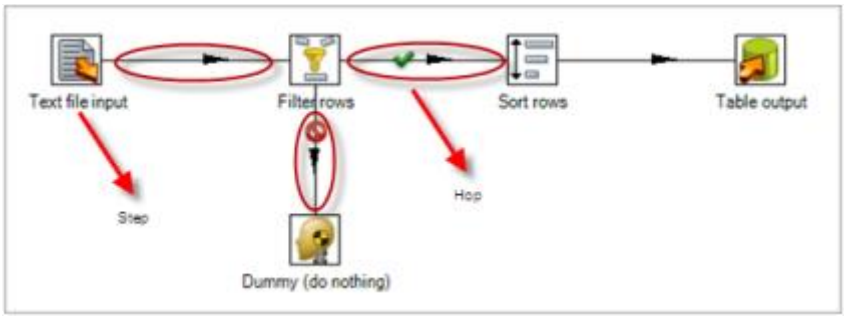
10 paveikslas. Prisijungimas prie SQL duomenų bazės

Sėkmingai prisijungus prie duomenų šaltinio galima atlikti įvairias operacijas su duomenimis, pvz. jeigu tai reliacinė duomenų bazė, galima vykdyti SQL komandas – kurti lenteles, atrinkti ir keisti duomenis ir t.t. Pažymėtina ir tai, kad tas pats prisijungimas prie duomenų šaltinio gali būti naudojamas daugiau nei vienoje transformacijoje ar užduočių sekoje.

Kalbant apie operacijas su failais, PDI gali nuskaityti ir įrašyti informaciją tiek iš/į struktūrizuotus, tiek iš/į pusiau struktūrizuotus (kai struktūra nėra aiškiai apibrėžta) failus. Ypač didelis operacijų ir transformacijų pasirinkimas yra Excel ir XML formatų failams.

2.3.1.3 Transformacijos

PDI transformacijos - tai tarpusavyje susietų loginių žingsnių tinklas. Iš esmės transformacija tai kryptinis grafas - duomenų kelias pradedant jų paėmimu iš šaltinio ir baigiant jų išsaugojimų galutinėje duomenų bazėje. Žemiau esančiame paveiksle pateikta pavyzdinė PDI transformacijos grafinė schema, kuri nuskaito tekstinį failą, atlieka filtravimą, rūšiavimą bei pakrauna duomenis į reliacinę duomenų bazę.



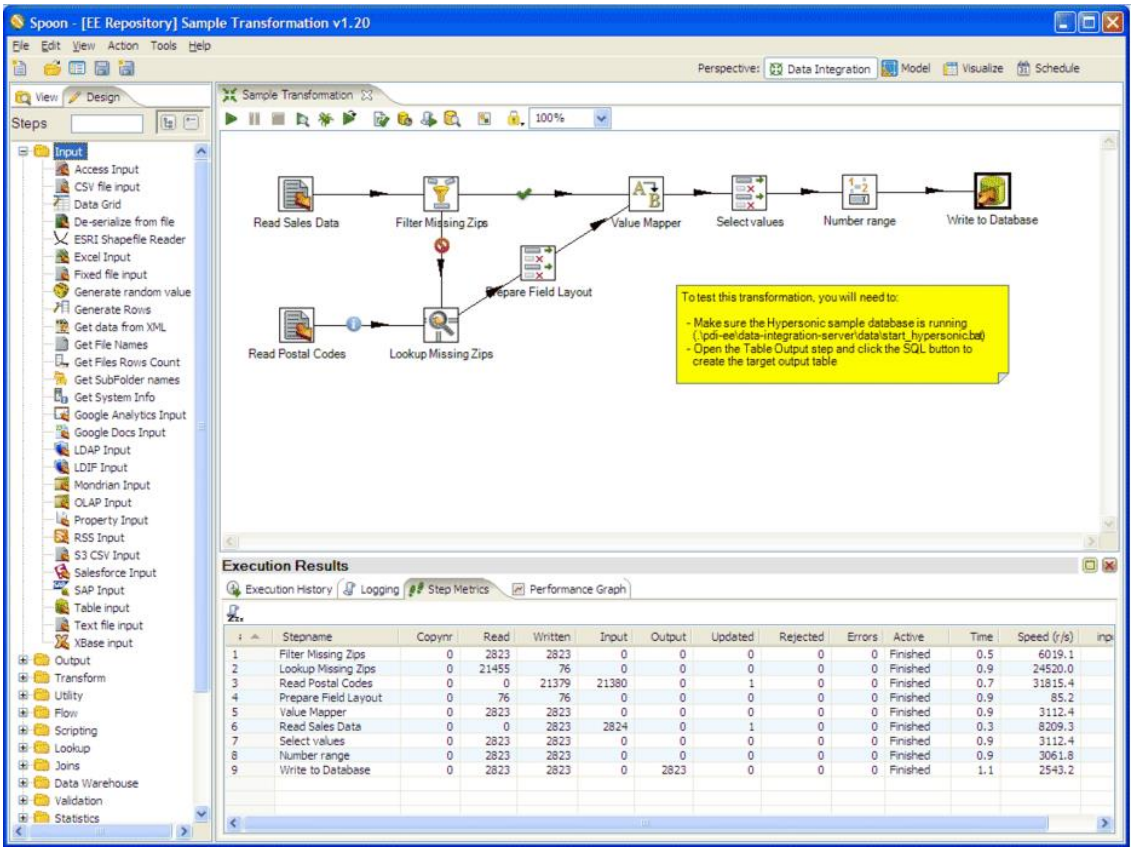
11 paveikslas. PDI transformacijos grafinės schema

Du pagrindiniai transformacijos komponentai yra žingsniai (angl. *steps*) ir perėjimai (angl. *hops*):

- Žingsniai – tai transformacijos blokai, kurie atlieka tam tikrą veiksmą pvz. nuskaito tekstinį failą arba įrašo duomenis į DB lentelę. PDI palaiko virš 100 skirtingų žingsnių, kurie sugrupuoti pagal atliekamų veiksmų tipą – duomenų paėmimas, programinio kodo vykdymas, duomenų įrašymas ir pan. Transformacijoje kiekvienas žingsnis yra atsakingas už specifinę funkciją, tokią kaip duomenų nuskaitymas, eilučių filtravimas, veiksmų žurnalizavimas. Administratorius gali nesunkiai konfigūruoti kokia funkcija bus vykdoma kiekviename iš žingsniu.
- Jungtys – tai elementai, kurie į bendrą tinklą sujungia atskirus žingsnius ir perduoda reikiamus metaduomenis iš vieno žingsnio į kitą bei apsprendžia duomenų judėjimo kryptį. Kiekvienas žingsnis gali turėti daugiau nei vieną jungti, pvz. viena jungtis apjungia du žingsnius, kita paima duomenis, o trečia juos išsaugo. Grafiniame redaktoriuje jungtys vaizduojamos rodyklėmis, kurios nurodo duomenų judėjimo kryptį.

12 paveikslas. PDI transformacijos jungtys

Žemiau esančiame paveiksle matoma transformacijos kūrimo naudotojo sąsaja Spoon įrankyje.



13 paveikslas. Transformacijos kūrimo naudotojo sąsaja

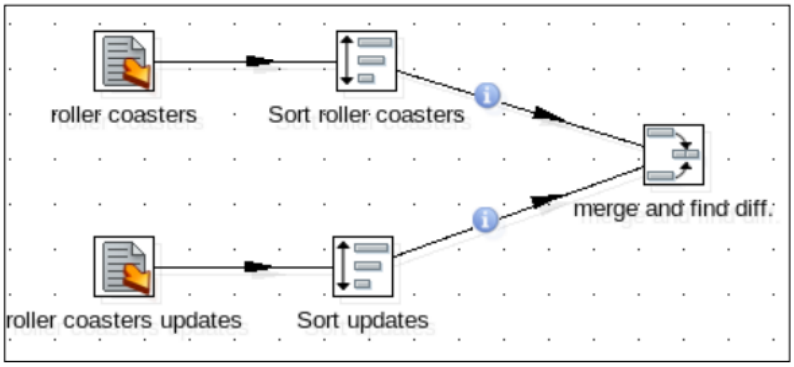
PDI platforma suteikia šias transformacijų vykdymo alternatyvas:

- Lokalus vykdymas – transformacijos vykdomos lokaliame kompiuteryje, kuriame yra įdiegtas Spoon įrankis. Ši alternatyva yra tinkama sistemos projektavimo, kūrimo ir testavimo stadijose.
- Vykdymas serveryje - kai transformacijos vykdomos nutolusiame serveryje. Nemokamas, atvirojo kodo Pentaho užduočių ir transformacijų vykdymo serveris vadinamas *Carte*.

Keletas uždavinių, kuriuos galima įgyvendinti naudojant PDI transformacijas:

- sąlyginis duomenų srauto padalinimas į kelis duomenų srautus;
- sąlyginis skirtingos struktūros eilučių apjungimas;
- kontrolinių sumų skaičiavimas, siekiant patikrinti duomenų objekto integralumą;
- dviejų duomenų srautų palyginimas;
- visų galimų porų suformavimas iš dviejų duomenų objektų;
- sąlyginis duomenų srautų apjungimas;
- eilučių įterpimas;
- kelių transformacijų vykdymas paraleliai.

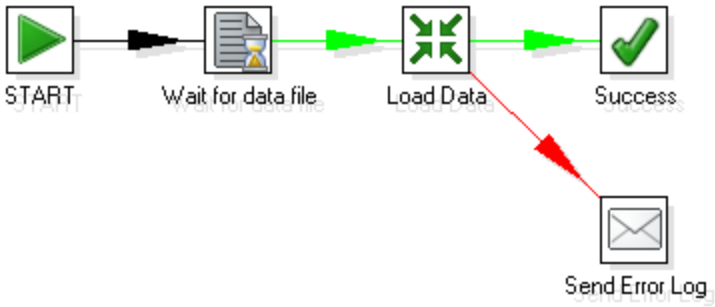
Žemiau esančiam paveiksle pateikiama dviejų duomenų srautų palyginimo PDI schemos pavyzdys.



14 paveikslas. Dviejų duomenų srautų palyginimo pavyzdys

2.3.1.4 Užduočių sekos

PDI platformoje užduočių sekos (angl. *jobs*), skirtos resursų, transformacijų vykdymo bei atskirų ETL veiksmų koordinavimui. Užduočių sekos į vieninga procesą apjungia atskirus ETL žingsnius. Žemiau esančiame paveiksle pateikiamas grafinis PDI užduočių sekos atvaizdavimas.



15 paveikslas. PDI užduočių sekos grafinis atvaizdavimas

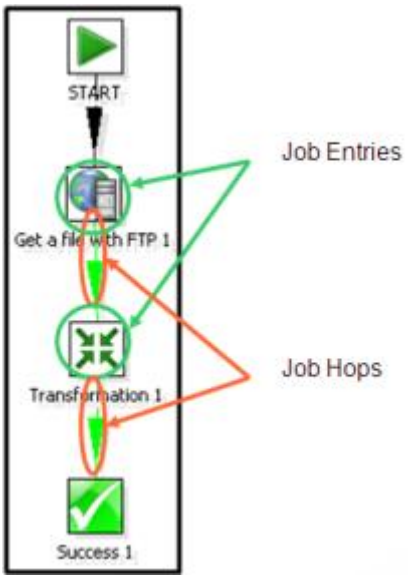
Užduočių sekos vykdymo metu gali būti atliekami įvairūs veiksmai, tokie kaip duomenų gavimas įvairiais protokolais (FTP, SOAP, REST ir pan.), būtinų sąlygų patikrinimas (pvz. ar egzistuoja reikiamos DB lentelės), transformacijų vykdymas, klaidos atveju el. laiško išsiuntimas ir pan. Dažnai aptinkamas užduočių sekos pavyzdys yra periodinis duomenų sandėlio atnaujinimas.

Užduočių sekos yra komponuojamos iš jungčių (angl. *job hops*), užduočių (angl. *job entries*) ir nustatymų. Užduotys - tai pagrindiniai užduočių sekos vykdymo blokai, kurie suteikia visą aibę funkcionalumų, tokių kaip duomenų paėmimas, transformacijos vykdymas ir pan. Užduočių sekos nustatymai – tai parametrai kontroliuojantys užduočių sekos eigą ir atliktų žingsnių žurnalizavimo metodus.

Jungtis apibrėžia ne tik užduočių vykdymo seką, bet ir sąlygas, kurioms esant gali būti vykdomas tolimesnis žingsnis. Jungtys gali būti kelių tipų:

- Nesąlyginės – sekantis žingsnis bus vykdomas visais atvejais, nepriklausomai nuo prieš tai buvusio žingsnio rezultatų.

- Vykdyti jeigu „True“ – sekantis žingsnis bus vykdomas tik tuo atveju, jeigu prieš tai buvęs žingsnis grąžino teigiamą rezultatą pvz. failas rastas, lentelė egzistuoja, klaidų nėra ir pan.
- Vykdyti jeigu „False“ - sekantis žingsnis bus vykdomas tik tuo atveju, jeigu prieš tai buvęs žingsnis grąžino neigiamą rezultatą pvz. failas nerastas, lentelė neegzistuoja, yra klaidų ir pan.



16 paveikslas. Užduotys ir jungtys

Užduočių sekos yra naudojamos šiems pagrindiniams veiksmas atlikti:

- apibrėžti transformacijų vykdymo tvarką ir eiliškumą;
- atlikti reikiamus patikrinimus prieš vykdant transformacijas pvz. patikrinti ar duomenų šaltinis pasiekiamas;
- pakrauti duomenis į duomenų bazę;
- atlikti operacijas su failais išsaugoti, parsisiųsti, kopijuoti, ištrinti ir pan.;
- el. paštu siųsti informaciją apie sėkmingai arba nesėkmingai atliktas užduotis.

2.3.1.5 PDI panaudojimo pavyzdžiai

HouseTrip yra vienas iš didžiausių nuomos paslaugų užsakymo tinklapių Europoje, siūlantis daugiau nei 250 tūkstančių nuomos pasirinkimo galimybių visame pasaulyje. Ši paslauga leidžia bet kuriam žmogui išnuomoti savo gyvenamąsias patalpas. Nuo *HouseTrip* veiklos pradžios 2010 metais, kuomet turėjo 200 nuomos pasiūlymų, yra įvykdyti keturi milijonai užsakymų įvairiose pasaulio vietose.

Tokios įmonės kaip *HouseTrip* yra priklausomos nuo klientų, todėl pagrindinis tikslas yra tobulinti siūlomų paslaugų prieinamumą klientams. Siekiant tikslų įmonė turi atlikti turimų duomenų analizę. Veiklos pradžioje įmonė naudojosi analizės sprendimais paremtais *Google Analytics* bei *Microsoft Excel*. Dėl didelių duomenų kiekių įmonė pasirinki *Pentaho* sprendimus. *HousTrip* turimi *Pentaho* sprendimai veikia *CentOS* operacinėje sistemoje ir naudoja *Amazon Redshift* duomenų sandėlius. Visa turima programinė įranga patalpinta *Amazon* debesyje.

Įmonė panaudojusi *Pentaho* sprendimus gali lengvai apdoroti didelius kiekius duomenų ir priimti reikiamus sprendimus gerinant paslaugas klientams. Daugiau informacijos adresu:

https://www.pentaho.com/sites/default/files/uploads/resources/housetrip_1.pdf

Šveicarijos nekilnojamo turto institutas, kuris atlieka nepriklausomus tyrimus, teikia mokymo ir konsultavimo paslaugas, bei atlieka mokslinius tyrimus. Institutas siekia skaidrinti vykdomus nekilnojamo turto sandorius ir visuomenei pateikti realias nekilnojamo turto rinkos kainas. Šiam tikslui institutas gauna duomenis apie atliktus sandorius iš bankų ir juos apdoroja. Šiai procedūrai institutas naudoja *Pentaho* duomenų integravimo ir analizės sprendimus, kurių dėka nesudėtingai galima importuoti bei apdoroti didelius kiekius duomenų ir pateikti išvadas. Institutas *Pentaho* duomenų integravimo sprendimus naudoja siekdamas užtikrinti gaunamų duomenų kokybei bei atrinkti klaidingus ir perteklinius duomenis. Daugiau informacijos adresu:

https://www.pentaho.com/sites/default/files/uploads/resources/sred_eng_final.pdf

IDBS teikianti paslaugas mokslo, farmacijos ir kitose pramonės šakose vykdo projektą, kuris sumažina išlaidas ir pagerina rezultatus farmacijos vykdomiems tyrimams. IDBS pasirinko *Pentaho* duomenų integravimo sprendimus siekiant sukurti veiksmingą bei lanksčią integruotą klinikinių duomenų gavimą iš 20-ies akademinių ir formacijos pramonės partnerių. IDBS pasirinko atviros architektūros *Pentaho* duomenų integravimo sprendimus, kurie gali būti lengvai prijungiami prie kitų atviro kodo platformų skirtų dalintis ir analizuoti biomedicininių tyrimų duomenis. IDBS taip pat pasirinko *Pentaho* duomenų integravimo sprendimus dėl gebėjimo atrinkti ir standartizuoti duomenis gautus iš partnerių visoje Europoje. IDBS sėkmingai baigė savo PDI projektą per penkias savaites. Sukurti sprendimai naudojami išgauti, transformuoti bei įkelti maždaug 20-ies tūkstančių pacientų įrašus. Daugiau informacijos adresu:

https://www.pentaho.com/sites/default/files/uploads/resources/idbs-case_study.pdf

Įmonei Soliditet reikėjo migruoti iš brangios ir nelanksčios COBOL aplinkos į šiuolaikinę, Java pagrindu veikiančią aplinką. Migravimas turėjo vykti palaipsniui, mažomis dalimis, todėl reikėjo duomenų integravimo priemonių, kurios palaikytų įvairius duomenų šaltinius, tokius kaip *PostgreSQL*, *MySQL*, *DB2* bei *MongoDB*. Šiam scenarijui puikiai tiko *Pentaho* duomenų integravimo sprendimai. *Pentaho* duomenų integravimo sprendimai leido gauti daugiau naudos iš IT komandos, įskaitant laiko taupymą, vykdomų užklausų optimizavimą ir naudojamas programavimo priemones. Daugiau informacijos adresu:

https://www.pentaho.com/sites/default/files/uploads/resources/soliditet.pdf

2.3.2 Topologinio žemėlapio atvaizdavimo biblioteka – Cytoscape.js

Tinklo topologinio žemėlapio atvaizdavimui numatoma naudoti atvirojo kodo JavaScript biblioteką Cytoscape.js (http://js.cytoscape.org). Ši biblioteka suteikia galimybę patogiai atvaizduoti ir manipuliuoti didelės apimties grafus (tinklus). Pagrindiniai bibliotekos privalumai:

- Suderinama su šiuolaikinėmis naršyklėmis (tiek stacionarių, tiek ir mobiliųjų įrenginių);
- Pilnai palaiko JSON duomenų formatą;
- Palaiko išdėstymo šablonus, kurie leidžia automatiškai išdėstyti tinklo mazgus;

- Palaiko grafo filtravimo užklausas;
- Palaiko veiksmus tiek naudojant pelę, tiek ir lietimui jautrius ekranus;
- Optimizuota grafo atvaizdavimo greitaveika;
- Plačiai naudojama visame pasaulyje.

2.3.3 Dinaminės duomenų analizės komponentai – Pentaho Mondrian ir Saiku CE

Dinaminei duomenų analizei numatoma naudoti OLAP (angl. *On-Line Analytical Processing*) programinius komponentus:

Pentacho Mondrian (https://community.hds.com/docs/DOC-1009853-mondrian) – tai atvirojo kodo OLAP serveris, kuris suteikia galimybę realiu laiku analizuoti didelius kiekius kompleksinių duomenų.

Saiku CE (https://community.meteorite.bi/) – tai atvirojo kodo OLAP analizei skirta naudotojo sąsaja. Saiku CE veikia interneto naršyklėje ir palaiko įvairius OLAP serverius, tame tarpe ir Mondrian.

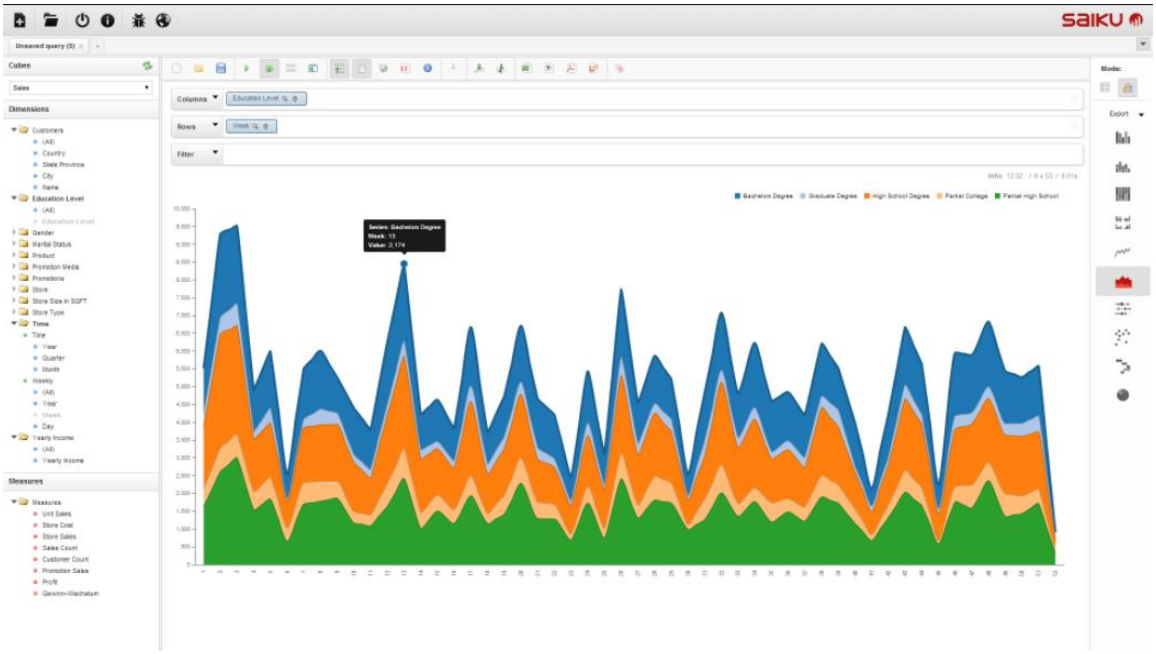
OLAP analizės priemonės sistemos naudotojams suteiks galimybę greitai ir interaktyviai analizuoti daugiamačius duomenis, t.y. pasirinkti pageidaujamus rodiklius bei dimensijas, atlikti duomenų filtravimą, pasukti duomenis, detalizuoti ir pan. OLAP principais paremti sprendimai suteikia šiuos pagrindinius privalumus:

- Greitas užklausų vykdymas esant dideliems duomenų kiekiams;
- Galimybė vykdyti sudėtingas skaičiavimo procedūras, panaudojant formules;
- Pateikimo lygmens lankstumas, kadangi metaduomenys saugomi serveryje.

OLAP tai specializuotos duomenų struktūros, kurios paspartina užklausų vykdymą daugiamatėje duomenų saugykloje. OLAP modulis suteikia tam tikrą loginį dimensijų modelį, kurio atžvilgiu naudotojai gali vykdyti užklausas. Dimensijų modelyje aprašomos dimensijos, matavimo rodikliai ir skaičiavimo operacijos, kurios yra prieinamos naudotojams. Toks modelis sukuria vieningą vaizdą, apimantį tiek OLAP struktūras, tiek ir reliacines struktūras. OLAP kubą galima apibrėžti kaip tam tikrą rinkinį sudarytą iš rodiklių ir su jais susietų dimensijų.

Vienas iš didžiausių OLAP analitinės sistemos privalumų yra tas, kad joje galima kurti ir naudoti sudėtingas skaičiavimo formules. Skaičiavimo procedūros gali būti saugomos tiek serveryje tiek ir kliento programinėje įrangoje, tačiau rekomenduojama procedūras saugoti centralizuotai, tam, kad jos būtų prieinamos visiems naudotojams. Serveryje vykdomos procedūros taip pat yra ir greitesnės už kliento aplikacijoje vykdomas procedūras.

Saiku CE taip pat suteikia galimybę naudotojui duomenis atvaizduoti grafiniu pavidalu, žemiau esančiam paveiksle pateikiamas grafinio duomenų atvaizdavimo pavyzdys:



17 paveikslas. Saiku CE grafinio duomenų atvaizdavimo pavyzdys

2.3.4 Statinių ataskaitų komponentas - Pentaho Reporting

Statinių ataskaitų modulio įgyvendinimui siūlome naudoti Pentaho Reporting atvirojo kodo ataskaitų rengimo įrankius (http://community.pentaho.com/projects/reporting/), kurie ataskaitų administratoriui leistų patogiai ir paprastai tvarkyti ataskaitų parametrus, taip pat stilizuoti ataskaitas.

Pentaho Reporting tai plačiai naudojamas ir patikimas, atviro kodo ataskaitų generavimo kompleksinis sprendimas, į kurio apimtį įeina šie komponentai:

- Report Designer – vizuali priemonė ataskaitų projektavimui;
- Reporting Engine – Java programavimo kalba parašyta biblioteka naudojama ataskaitų generavimui;
- Reporting Software Developers Kit (SDK) – programinės įrangos kūrėjams skirtas Reporting Engine bibliotekos, dokumentacijos ir visų reikalingų pagalbinių bibliotekų rinkinys, kuris į kuriamą sistemą leidžia įterpti Pentaho Reporting Engine.

Pentaho Reporting Engine – tai atvirojo kodo ataskaitų generavimo variklis, kuris sutiekia galimybę sugeneruoti dinamiškas ataskaitas iš įvairiuose duomenų šaltiniuose esančių duomenų. Šio variklio pagalba sugeneruotus ataskaitų dokumentus galima peržiūrėti, atsispausdinti bei eksportuoti į PDF, Excel, HTML, XML, CSV formatų failus.

Pentaho Reporting Engine palaiko įvairių tipų duomenų šaltinius. Palaikomų duomenų šaltinių pavyzdžiai:

- reliacines duomenų bazes (palaikančios JDBC ar ODBC);
- XML failai;
- OLAP duomenų šaltiniai;
- Rankiniu būdu nurodytos duomenų lentelės.

Pentaho Reporting Engine palaiko pagrindinius skaičiavimus:

- sumos (angl. *Sum*), vidurkio (angl. *Average*), kiekio (angl. *Count*), skirtingų objektų kiekio (angl. *Count-Distinct),* mažiausio elemento (angl. *Minimum*), didžiausio elemento (angl. *Maximum*) skaičiavimus;
- atviras formulių išraiškas (angl. *Open-Formula expressions*);
- skriptu užrašomas išraškas (angl. *Scriptable expressions*).

Pentaho Reporting leidžia skaičiavimus atlikti realiu laiku, todėl ataskaitos prisiderina prie gaunamų duomenų.

2.3.5 Naudotojo sąsajos kūrimo karkasas - AngularJs

Sistemos portalo naudotojo sąsajos elementai bus realizuojami naudojantis AngularJS programinio karkaso pagrindu. AngularJS – tai yra kompanijos Google sukurtas atvirojo kodo naudotojo sąsajos kūrimo karkasas, skirtas interneto puslapių kūrimui apjungiant HTML, CSS ir JavaScript naudotojo pusėje. Karkasas praplečia HTML ir padaro jį dinamiškesniu. AngularJS suteikia galimybę realizuoti MVC (angl. *Model-View-Controller*) architektūros principus kliento pusėje. AngularJS karkaso priemonėmis kuriama naudotojo sąsaja yra prienama standartinės interneto naršyklės terpėje.

AngularJS priemonėmis galima greitai kurti šiuolaikinių WEB technologijų pagrindu veikiančias sistemas, išlaikant vieningą ir nuoseklią naudotojo sąsają neatsisakant portalo stabilumo ir efektyvumo.

Portalo naudotojo sąsajos elementams realizuoti bus naudojami įvairūs AngularJS moduliai. Yra sukurta daug atvirojo kodo AngularJS modulių, tiekiančių įvairias pagalbines funkcijas - http://ngmodules.org/. Šių modulių pagalba, esant poreikiui, galima minimaliomis pastangomis sukurti papildomas naudotojui reikalingas funkcijas.

Sistemos portalas bus paremtas Single Page Application principu (http://en.wikipedia.org/wiki/Single-page_application), kurio pagrindą sudaro dinamiškai valdomi puslapio komponentai, nereikalaujantys viso puslapio perkrovimo. AngularJS karkasas kaip tik skirtas tokio tipo aplikacijų kūrimui.

AngularJS karkasas yra plačiai naudojamas praktikoje. Toliau pateikiame keletą pasauliniu mastu žinomų sistemų, kurios realizuotos naudojant *AngularJS:*

- www.youtube.com – populiariausias pasaulyje video medžiagos talpinimo portalas.
- www.lynda.com – internetinių pamokų portalas. Portale pateikiama įvairaus turinio ir formato mokomosios medžiagos.
- www.mifos.org – atvirojo kodo bankinė sistema. Sistema paremta šiomis technologijomis: AngularJS, Spring Framework, Hibernate.
- www.reviewmatters.com – portalas skirtas atsiliepimų apie kitus portalus talpinimui ir jų reitingavimui.

2.3.6 Programinės įrangos kūrimo karkasas - Spring Framework

Spring yra atvirojo kodo programinės įrangos kūrimo karkasas, kuris buvo sukurtas siekiant palengvinti ir supaprastinti informacinių sistemų kūrimą, naudojant Java technologijas. Spring buvo kuriamas kaip alternatyva standartiniam J2EE karkasui, kuris pasižymėjo sudėtingumu ir kompleksiškumu. Šiuo metu Spring yra vienas iš

labiausiai pasaulyje paplitusių Java programavimo karkasų, galima drąsiai teigti, kad tai yra *de facto* standartas Java Enterprise aplikacijų kūrimui. Nors dabartinį Spring karkasą sudaro daug komponentų, apimančių įvairius programinės įrangos kūrimo aspektus, tačiau nuo pat gyvavimo pradžios išlieka keturi pagrindiniai Spring principai:

- „Lengvasvoris“ programinės įrangos kūrimas naudojant paprastus Java objektus (POJO), t.y. siekiama, kad kodas nepriklausytų nuo Spring API;
- Objektų ryšių automatinio inicializavimo (angl. *Dependency Injection* – DI) ir sąsajų (angl. *Interfaces*) panaudojimas;
- bendrai naudojamo programinio kodo (aspektų) iškėlimas į atskirus modulius, t.y. realizuojamas aspektiškai orientuoto programavimo stilius (angl. *Aspect-oriented programming - AOP*);
- šablonų ir aspektų naudojimas siekiant sumažinti standartinio (perteklinio) kodo kiekį.

Java objektai - POJO

Dauguma programinės įrangos kūrimo karkasų programuotojus verčia naudoti specifines programines sąsajas (API), kas apsunkina kodo pernešamumą bei patį programavimą. Kuriant programinę įrangą *Spring* karkase yra vengiama (tiek kiek tai yra įmanoma) naudoti specifinius API, t.y. programuotojas nėra verčiamas naudoti specifines *Spring* sąsajas ar išplėsti specifines karkaso klases. Iš esmės *Spring* karkasui rašomos *Java* klasės yra paprastieji *Java* objektai – POJO ir praktiškai neturi jokių indikacijų, kad jos yra pritaikytos tik *Spring*.

Dependency Injection

Tipiškoje programoje objektų gyvavimo ciklą kontroliuoja pats kodas, DI būdu suprojektuotoje sistemoje objektų gyvavimo ciklą kontroliuoja programinis konteineris. Programuotojui reikia sukurti tik patį pirminį objektą, o visus kitus objektus, kai prireiks, sukurs konteineris. Konteineris objektui perduoda reikalingus komponentus per konstruktorių arba per set* metodus (JavaBeans properties). Trumpai tariant DI yra būdas susieti aplikacijos objektus taip, kad patiems objektams nereikėtų rūpintis kur yra susiję objektai ir kaip jie realizuoti. DI suteikia šiuos esminius privalumus:

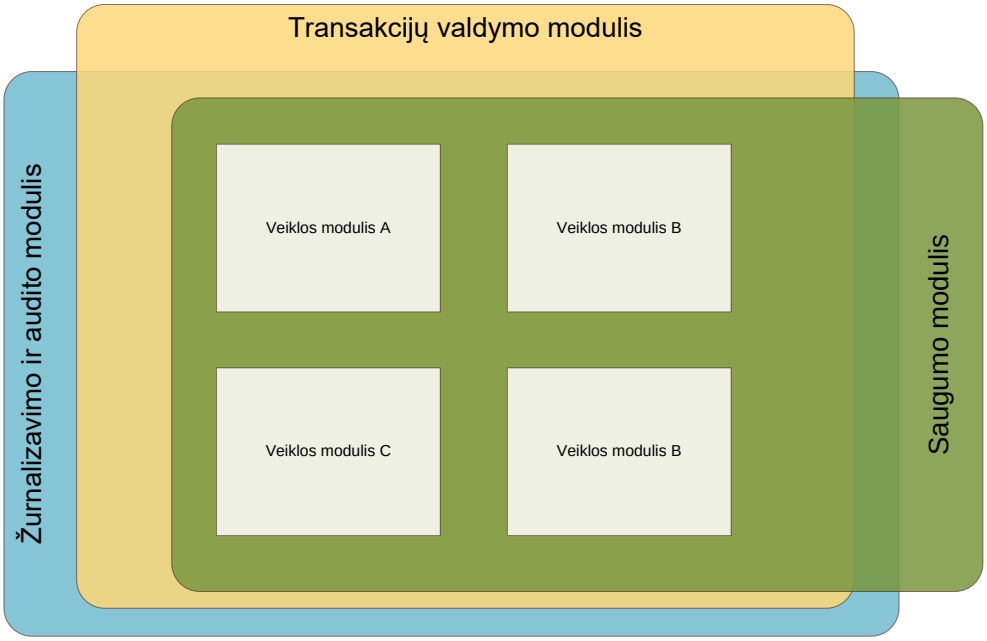
- leidžia turėti labiau atskirtus komponentus (angl. *loose coupling*);
- DI komponentus lengviau testuoti;
- DI reiškia paprastesnį ir lengviau suprantamą kodą.

Aspect-oriented programming – AOP

AOP programavimo stilius suteikia galimybę išskirti tam tikras sistemos funkcijas į atskirus komponentus (aspektus), kurie gali būti pakartotinai panaudojami. Praktiškai kiekvienos informacinės sistemos funkcionalumą galima išskaidyti į dvi stambias kategorijas:

- Pagrindinės veiklos funkcijos, t.y. tos funkcijos dėl kurių yra kuriama sistema;
- Pagalbinės funkcijos, t.y. sisteminės funkcijos reikalingos norint užtikrinti tokius sistemos aspektus kaip saugumas, atsekamumas ir pan.

Pagalbinės funkcijos sąveikauja su daugeliu sistemos komponentų, todėl siekiant išvengti kodo dubliavimo ir neapkrauti veiklos modulių bereikalingu funkcionalumu, tikslinga jas realizuojančius modulius atskirti į atskirus komponentus. Žemiau esančiame paveiksle pateikiamas AOP principą realizuojančios aplikacijos loginės architektūros schema.

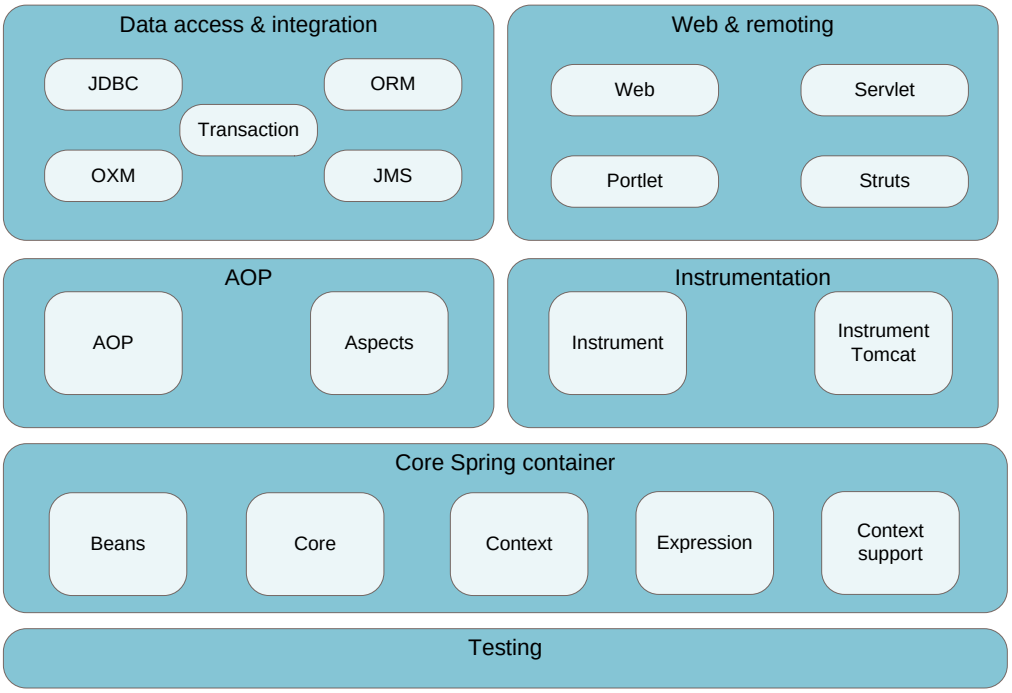


18 paveikslas. AOP principus realizuojančios aplikacijos loginė schema

Šablonų naudojimas

Kuriant programinę įrangą programuotojams dažnai tenka naudoti standartinį kodą, kuris nėra tiesiogiai susijęs su siekiamu funkcionalumu ir yra perteklinis. Šiai situacijai iliustruoti puikiai tinka pavyzdys kai norint atlikti paprastą duomenų bazės užklausą tenka parašyti nemažą JDBC API kodo gabalą vien tam, kad būtų užtikrintas sėkmingas prisijungimas prie duomenų bazės. *Spring* karkasas siekia eliminuoti arba kiek įmanoma sumažinti standartinio kodo kiekį naudojant šablonus, t.y. veiklos funkcija yra enkapsuliuojama į tam tikrą šabloną, todėl funkcijos vykdymui skirto kodo kiekis tampa ženkliai mažesnis ir paprastesnis.

Kaip matome iš aukščiau aprašytų aspektų, *Spring* karkaso pagrindinis tikslas yra supaprastinti informacinių sistemų kūrimą, panaudojant tokias technologijas kaip *Dependency Injection, AOP, POJO* ir pan. Pats *Spring* karkasas yra sudarytas iš šešių pagrindinių modulių, kurie praktiškai apima viską, ko reikia kuriant modernias informacines sistemas. Žemiau esančiame paveiksle pateikiama pagrindinių *Spring* modulių diagrama.



19 paveikslas. Spring karkaso moduliai

Core Spring container – tai centrinė *Spring* karkaso dalis, kuri valdo kitų programinių komponentų (Beans) sukūrimą, konfigūraciją ir administravimą. Šiame modulyje realizuotas vienas iš esminių *Spring* karkaso aspektų – *dependency injection.* Papildomai prie DI šis modulis suteikia daugybę papildomų funkcijų (el. paštas, JNDI, EJB integracija, scheduling), kurios yra būtinos kuriant informacines sistemas. Šis modulis funkcionuoja kaip pagrindas kitiems *Spring* karkaso moduliams.

AOP modulis – realizuoja aspektiškai orientuoto programavimo principus. Šis modulis tarnauja kaip pagrindas kuriant pakartotinai panaudojamus informacinės sistemos komponentus. Kaip ir DI, AOP suteikia galimybę kurti mažai susietus programinės įrangos objektus, tik šiuo atveju objektai dažniausiai apima ne vieną klasę bet platesnį IS funkcionalumą pvz. įvykių žurnalizavimas.

Data Access and integration - modulyje realizuoti šablonai darbui su duomenų saugojimo technologijomis. Šio modulio dėka programuojant galima išvengti perteklinio standartinio kodo (prisijungti prie DB, suformuoti užklausą, apdoroti rezultatą ir pa.). Modulyje realizuoti šablonai tiek darbui su reliacinėmis DB, tiek ir su objektinio – reliacinio susiejimo karkasais, tokiais kaip Hybernate, JPA, iBATIS ir pan. Šis modulis tai pat turi *Java* pranešimų serviso (angl. *Java Message Service – JMS*) šabloną, kuris puikiai tinka kuriant asinchroninę integraciją su išorinėmis IS. Taip pat reiktų paminėti ir tai, kad *Data Access and integration* modulyje yra realizuotas transakcijų valdymo servisas.

WEB and remoting – modulyje realizuoti komponentai skirti naudotojo sąsajos lygmens kūrimui. WEB aplikacijose, siekiant atskirti naudotojui pateikiamą vaizdą nuo aplikacijos veiklos logikos, yra plačiai naudojama modelio – vaizdo – kontrolerio (angl. *Model-View-Controller - MVC*) architektūra. Java platformoje egzistuoja nemažai populiarių MVC karkasų, tokių kaip *Apache Struts, JSF, WebWork* bei *Tapestry*. Nepaisant to, kad *Spring* puikiai integruojasi su populiariausiais MVC karkasais, jame taip pat yra realizuoti pilnaverčiai MVC architektūros komponentai.

Testing – modulyje realizuotos pagalbinės objektų bibliotekos, palengvinančios automatizuotų testų rašymą.

Taip pat norime pabrėžti, kad pilnas *Spring* karkasas turi gerokai daugiau komponentų, t.y. aukščiau paminėti moduliai yra *Spring* karkaso branduolys, kurio pagrindu atvirojo kodo bendruomenė kuria kitus komponentus (Spring web flow, Spring security, Spring web services ir t.t.).

2.3.7 Integracinis karkasas – Spring integration

Spring Integration yra atvirojo kodo integracinis karkasas, kurio paskirtis palengvinti aplikacijų integraciją.

Dviejų sistemų bendravimas gali tapti opia problema, jei jos naudoja skirtingus protokolus arba žinutes ir nėra galimybės jų pakeisti, šios problemos išsprendimui reikėtų naudoti daug įvairių įrankių arba patiems implementuoti ir konfigūruoti žinučių apdorojimą ir paskirstymą. Vietoje to galima naudoti *Spring Integration* karkasą, kuris apjungia daugelio įrankių funkcionalumą ir leidžia sutaupyti laiko bei neapkrauti sistemos rašant papildomą programinį kodą. *Spring Integration* gali būti konfigūruojamas naudojant *Spring* schemas.

Šis karkasas leidžia dviem skirtingoms sistemoms, kurios naudoja skirtingus protokolus ar žinutes, keistis duomenimis. Jis veikia kaip tarpininkavimo ir paskirstymo karkasas, kuris tarpininkauja siunčiant duomenis ir išsprendžia sistemų duomenų apsikeitimo skirtumus.

Pagrindiniai karkaso komponentai:

- Žinutės (angl. *Messages*) – objektai siunčiami iš vieno komponento į kitą. Žinute gali būti bet kuris Java objektas susietas su metaduomenimis, kuriuos karkasas naudoja objekto valdymui. Žinutės antraštėje gali būti saugomos bet kokios rakto ir reikšmės poros, paprastai joje laikoma dažniausiai užklausiama informacija (pvz. objekto identifikacinis numeris, pavadinimas, grąžinimo adresas ir pan.).
- Kanalai (angl. *Channels*) – gauna žinutes iš tiekėjo ir perduoda jas gavėjui. Priklausomai nuo kanalo tipo, jis gali turėti tik vieną arba daug gavėjų.
- Baigties taškai (angl. *Endpoints*) – sujungia sistemos programinį kodą su žinučių karkasu.

Spring Integration palaiko įvairių tipų žinučių baigties taškus. Pagrindiniai *Spring Integration* palaikomi žinučių baigties taškų tipai:

- *Transformer* – atsakingas už žinutės turinio ar struktūros konvertavimą ir modifikuotos žinutės grąžinimą. Taip pat, gali būti naudojamas žinutės antraštės koregavimui.
- *Filter* – nusprendžia, ar žinutė turėtų būti perduodama į išeities kanalą.
- *Router* – nusprendžia, kuris kanalas ar kanalai gaus kitą žinutę. Sprendimas priimamas pagal žinutės antraštėje esančius metaduomenis arba pagal žinutės turinį.
- *Splitter* – gauna vieną žinutę ir ją padaliną į kelias žinutes.
- *Aggregator* – gauna kelias žinutes ir jas sujungia į vieną žinutę.
- *Service Activator* – bendras baigties taškas, kuris naudojamas serviso sujungimui su žinučių sistema.
- *Channel Adapter* – sujungia žinučių kanalą su kokia nors kita sistema.

2.3.8 Naudotojų autentikavimas ir autorizavimas - Spring security

Naudotojų autentifikavimui ir autorizacijai bus naudojamos *Spring Security* programinės priemonės. *Spring Security* modulis atsakingas už naudotojų autentifikavimą ir autorizaciją (arba prieigos/teisių kontrolę). *Spring* karkaso pagrindu sukurtų aplikacijų *de facto* saugumo standartu būtent ir yra laikomas *Spring Security* modulis. Pagrindinis *Spring Security* autentifikavimo mechanizmo privalumas yra jo lankstumas lyginant su J2EE servleto standartu arba EJB standartu. Dažniausiai šių standartų trūkumai išryškėja didelėse informacinėse sistemose. Svarbu paminėti, kad J2EE arba EJB saugumas yra aplikacijų serverio, o ne aplikacijos paketo (WAR ar EAR failo) dalis. Todėl diegiant ar plečiant sistemą autentifikavimą ir autorizaciją tektų konfigūruoti iš naujo. *Spring Security* šias problemas išsprendžia, taip pat teikia daug kitų privalumų.

Kalbant apie autentifikavimą ir autorizavimą, *Spring Security* palaiko daug įvairių modelių. Dauguma šių modelių yra kuriami trečiųjų šalių ir yra paremti įvairiais standartais. Taip pat *Spring Security* palaiko ir keletą nuosavų autentifikavimo modelių. Svarbiausi *Spring Security* palaikomi modeliai yra šie:

- OAuth(1a) ir OAuth2;
- HTTP BASIC autentifikacija;
- HTTP Digest autentifikacija;
- HTTP autentifikacija naudojant X.509 kliento sertifikatus;
- LDAP autentifikacija (populiariausia didelių sistemų autentifikacijos rūšis, ypač kai autentifikacija atliekama tarp skirtingų platformų);
- OpenID ir daugybė kt. autentifikacijos mechanizmų;

Planuojamos sistemos kūrimui numatoma panaudoti *Spring Security OAuth2* modulį, kuris leis kartu su *Spring Security* įgyvendinti dalį OpenId Connect protokolo, naudojant standartinius *Spring* ir *Spring Security* programavimo modelius ir konfigūracijas.

Sistemoje bus realizuotas vienas ar daugiau identifikacijos teikėjų (angl. *identity provider*), kurie bus atsakingi už vartotojo tapatybės patvirtinimą bei vienas autorizacijos serveris (angl. *authorization server*), kuris leis rinktis iš identifikacijos teikėjų. Kad suteiktų prieigą, autorizacijos serveris turės sukurti ir patvirtinti JWT raktus (informacija gaunama iš naudotojo pasirinkto identifikacijos teikėjo bei sistemos naudotojų mikroserviso), kurie bus suteikiami klientinei aplikacijai. JWT rakte gali būti talpinamas ne tik kliento identifikatorius, bet ir kita naudinga informacija (pvz.: vartotojo vardas, pavardė, rakto galiojimo laikas, vartotojo rolės ir pan.). Klientas prie apsaugotų išteklių bus prileidžiamas tik tuo atveju, jei turės galiojantį ir tinkamą JWT raktą. *Spring Security OAuth2* modulis leidžia įgyvendinti aukščiau aprašytą autentifikaciją ir autorizaciją tokiu būdu sustiprindamas apsaugą nuo neautorizuotos prieigos prie išteklių.

Internetinėse aplikacijose *Spring Security* modulis yra realizuotas panaudojant išskirtinai tik standartinius servletų filtrus ir jų grandines (angl. *security filter chain*). Pats *Spring Security* modulis nenaudoja jokių servletų, jų karkasų (pvz. Spring MVC), nėra jokių ryšių ribojančių *Spring Security* panaudojimą su norimu aplikacijos karkasu. *Spring Security* vienodai veikia tiek naršyklės, tiek tinklinių paslaugų kliento (angl. *web service client*) ar *AJAX* aplikacijos atveju. *Spring Security* pats yra atsakingas už filtrų grandinės valdymą. Atitinkami filtrai yra pridedami, pašalinami priklausomai nuo naudojamų *Spring Security* paslaugų, jie automatiškai sudėliojami į eilę. Tačiau reikia pabrėžti, kad rankinio konfigūravimo galimybė yra palikta, individualių metodų naudojimui.

Spring Security naudodama HTTP paketo antraštes, leidžia apsaugoti naudotoją nuo XSS, tarpininko (angl. *man in the middle*) ir panašių atakų. Pagrindinės konfigūruojamos antraštės yra šios:

- Kešo kontrolė – naršyklė informuojama nekešuoti informacinės sistemos turinio. Priešingu atveju yra tikimybė, kad naršyklė pateiks išsaugotą turinį netgi naudotojui užbaigus sesiją;
- Turinio tipo nustatymai – kai turinio tipas nėra nurodytas, naršyklės paprastai bando jį atspėti. Nustatymai leidžia uždrausti naršyklei spėlioti turinio tipą. Tokiu būdu naudotojas apsaugomas nuo nelaukto javascript ar kitos kalbos skripto vykdymo.
- HSTS (angl. *HTTP strict transport security*) – naršyklė informuojama, kad informacinė sistema veikia saugiu (HTTPS) protokolu ir naudotojų netikslios užklausos, prieš siunčiant užklausą, turi būti automatiškai perrašomos.
- X-Frame, X-XSS nustatymai – leidžia iš dalies apsaugoti naudotoją nuo XSS atakų bei nuo nepageidautinų naršyklės nukreipimų (angl. *browser redirect*).

Apibendrinus *Spring Security* teikia didelę autentifikavimo mechanizmų įvairovę, neribodama jų komponavimo ir individualizavimo, leisdama panaudoti ne tik nestandartines autentifikavimo, bet ir autorizacijas bei registracijos schemas.

2.3.9 Objektinio-reliacinio susiejimo karkasas – Hibernate

Klasikiniu atveju darbas su objektiškai orientuota aplikacija ir reliacinėmis duomenų bazėmis yra sunaudojantis daug laiko, nepatogus ir gremėzdiškas. Tokiu atveju sistema tampa sudaryta iš vienos pusės stipriai susietų, tačiau iš kitos pusės (kalbant apie palaikymą, atnaujinimą) atskirų dalių. Kūrimo išlaidos gerokai padidėja dėl neatitikimo tarp duomenų saugomų objektuose ir reliacinėse duomenų bazėse. Sistemos kūrimas paprastai vyksta ne tik objektinėje platformoje, bet ir duomenų bazės dalyje, kurioje, nesant labai aukštos kultūros komandoje, sunku užtikrinti versijų kontrolę, atsekti, struktūros pasikeitimo istoriją, bei autorius. Nėra patikimų priemonių užtikrinti, kad aplikacijos ir duomenų bazės sąsaja yra korektiška, kad padarytas pakeitimas kurioje nors sistemos dalyje, nesugeneruos naujų klaidų. Įprastai aplikacijos yra skirtos vienam konkrečiam duomenų bazės tipui, norint pakeisti duomenų bazę, paprastai būtinas visos aplikacijos nuodugnus pertvarkymas.

Šiems sunkumams spręsti dažniausiai yra naudojamas objektinis reliacinis susiejimas (ORM). Objektinis reliacinis susiejimas yra principas, kuriuo objektinis esybės modelis yra susiejamas su reliaciniu duomenų modeliu (ir atvirkščiai). *Hibernate* yra objektinio reliacinio susiejimo karkasas (angl. *Object-Relational Mapping*) Java aplinkai.

Hibernate yra vienas populiariausių objektinio reliacinio susiejimo karkasų. Aktyviai kuriamos ir palaikomos versijos yra ne tik Java platformoje, bet ir .NET, kur jo pavadinimas yra pakeistas į „nHibernate“. Pagrindiniu „Hibernate“ konkurentu yra laikomas „Microsoft“ vystomas „Entity framework“. „Hibernate“ pagrindinis pranašumas prieš kitus panašius karkasus yra tas, kad juo naudojantis galima ne tik susieti klasių objektus su duomenų bazės įrašais, bet ir tai, kad šie bazės objektai gali būti išrenkami naudojantis objektinėmis užklausomis (HQL kalba), „Hibernate“ yra tampriai integruotas su „Spring“ karkasu, turi anotacijomis paremtą transakcijų valdymą. „Hibernate“ karkaso tikslas yra dvejopas - iš vienos pusės siekiama panaikinti 95% rankinio darbo, kuris įprastai yra atliekamas kuriant duomenų apdorojimo sprendimą, iš kitos pusės siekiama išnaudoti kuo daugiau reliacinės duomenų bazės galimybių, nesupaprastinti sąsajas siekiant universalumo, kaip dažniausiai atsitinka kitų ORM karkasų atveju. „Hibernate“ karkasas leidžia aplikaciją kurti bet kokiam duomenų bazės tipui, t.y. neapriboja duomenų bazių valdymo sistemos pasirinkimo.

Hibernate karkasas leidžia optimizuoti ir pagreitinti aplikacijos užklausų veikimą. Yra palaikomas dviejų lygių spartinimas (angl. *caching*). Retai besikeičiantys duomenys, lentelės gali būti saugomi *Hibernate* karkaso valdomoje spartinančioje atmintyje, taip sistemoje išvengiant perteklinių užklausų į duomenų bazę.

Hibernate duomenų bazės užklausoms vykdyti naudoja HQL užklausų kalbą. Pagrindinis privalumas prieš SQL yra tas, kad šios užklausos kompiliavimo metu yra tikrinamos ir neatitikimai tarp klasių modelio ir užklausos yra iš karto identifikuojami. Taip sumažinama paslėptų klaidų tikimybė.

Transakcijų valdymas yra labai opi problema, reikalaujanti kuriant naują transakciją stebėti, ar nėra jau atidarytos transakcijos, gaudyti sistemos klaidas tam, kad neliktų neuždarytų transakcijų. Dėl šių priežasčių programinis kodas tampa sudėtingas, naudojami gaubiantys metodai, aplikaciją komplikuojančios klaidų gaudymo struktūros. *Hibernate* viso to nelieka įvedant valdiklio sąvoką (angl. *service*). Valdiklis yra speciali klasė, kuri yra anotuojama (pažymima), kad jos veiksmai turi būti atliekami transakcijoje. Programuotojui daugiau transakcijomis rūpintis nereikia, *Hibernate* atlieka visą likusį darbą – transakcijos inicializavimą, jos užbaigimą, transakcijos transakcijoje valdymą ir pan.

2.3.10 Indeksavimo ir paieškos komponentas - Apache Solr

Indeksavimo ir paieškos modulio funkcionalumui realizuoti bus naudojama *Apache Solr* atvirojo kodo paieškos platforma. *Apache Solr* yra didelio našumo, plečiama sistema suteikianti galimybę vykdyti turinio indeksavimą ir paiešką paskirstytose sistemose. *Solr* sukurtas naudojant Java programavimo kalbą ir kito populiaraus paieškos variklio *Apache Lucene* pagrindinę klasių biblioteką. Ši paieškos sistema palaiko programinius prievadus - saityno paslaugas (REST, HTTP/XML), kurios leidžia naudotis sistemos teikiamu funkcionalumu įvairiose technologinėse platformose. Tai pilnai konfigūruojama paieškos platforma, todėl gali būti pritaikoma įvairiose sistemose be didesnių programavimo darbų.

Apache Solr paieškos sistemos funkcionalumas ir privalumai:

- pilnatekstės (ang. *Full-text*) paieškos galimybės;
- facetinės (angl. *Faceted*) paieškos galimybės;
- automatiniai užklausų rašymo patarimai naudotojui (angl. *Autocomplete*);
- didelės spartos indeksavimas;
- dokumentų (pvz. Word, PDF) apdorojimas;
- integracija su duomenų bazėmis;
- standartais paremti programiniai prievadai – XML/XSLT, JSON ir HTTP;
- pateikiamos informacijos rūšiavimas ir grupavimas;
- WEB administravimo sąsaja;
- konfigūruojama indeksuojamų duomenų schema;
- konfigūruojamos spartinimo galimybės (angl. *Caching*);
- statistinės informacijos kaupimas ir pateikimas administratoriams.

Apache Solr paieškos variklis suteikia galimybę administratoriui aprašyti indeksuojamų dokumentų laukus ir laukų tipus arba pasinaudojant dinaminių laukų funkcionalumu automatiškai papildyti indeksuojamų dokumentų schemą. Pasinaudojant *Apache Solr* laukelių kopijavimo funkcionalumu (angl. *CopyField*) galima vieną šaltinio dokumento laukelį nukopijuoti į kelis skirtingus indeksuojamo dokumento laukelius ir tokiu būdu paspartinti atnaujinimo komandų vykdymą. *Apache Solr* paieškos variklis palaiko įvairias teksto apdorojimo ir analizės funkcijas, tokias kaip žodžių dalinimas, reguliariomis išraiškomis (angl. *regular expressions*) pagrįstas teksto filtravimas, panašiai skambančių žodžių paieška ir t.t.

Apache Solr turi labai plačias paieškos užklausų aptarnavimo ir rezultatų pateikimo galimybes:

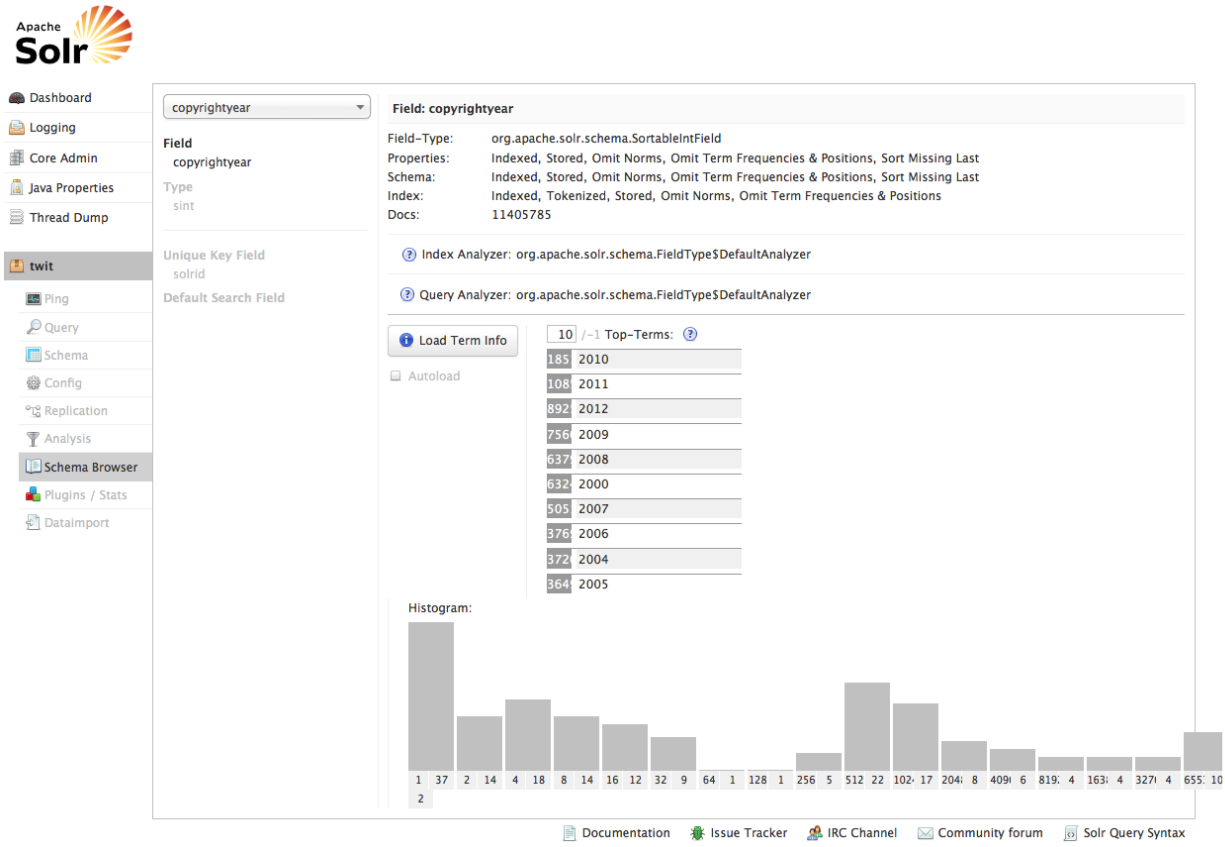
- HTTP sąsaja su konfigūruojamais atsakymo formatais (XML/XSLT, JSON, Python, Ruby ir kt.).
- Neaiškių paieškos užklausų apdorojimas (angl. *fuzzy search*). Šis funkcionalumas leidžia pateikti paieškos rezultatus, kurie yra panašūs į užklausą ir nebūtinai tiksliai atitinka užklausos tekstą. Panašumo nustatymas paremtas sudėtingais atstumo algoritmais (pvz. *Levenshtein Distance*).
- Rūšiavimas pagal neribotą skaičių laukelių arba pagal kompleksines skaitinių laukelių funkcijas.
- Naudotojo įvestų frazių padalijimas į atskirus žodžius, paieškos vykdymas per kelis indeksuotus laukelius ir svorių priskyrimas, remiantis kiekvieno laukelio svarba.
- Ieškomų žodžių ar frazių paryškinimas paieškos rezultatų sąraše.
- Facetinė paieška remiantis unikalių laukelių reikšmėmis, datos bei skaitinių reikšmių intervalais.
- Vykdant facetinę paiešką galimybė pasirinkti kelių filtrų reikšmes.
- Automatiniai užklausos rašymo patarimai naudotojui, remiantis panašiais terminais.
- Galimybė ieškoti daugiau panašių dokumentų (angl. *more like this*).
- Galimybė paieškos užklausose naudoti funkcijas, kurios perskaičiuoja paieškos rezultatų svorius ir gali įtakoti naudotojui pateikiamus rezultatus.
- Skaitinių laukelių statistinės formulės tokios kaip didžiausia/mažiausia reikšmė, vidurkis bei standartinis nuokrypis.

Greitesniam paieškos rezultatų pateikimui *Apache Solr* naudoja spartinimo mechanizmus (angl. *caching*). Spartinančioje atmintyje gali būti saugomi užklausų rezultatai, filtravimo rezultatai bei patys dokumentai. *Apache Solr* spartinančiojoje atmintyje saugomi objektai yra susiejami su konkrečiais indekso „vaizdais“, tad kol indekso „vaizdas“ yra galiojantis, rezultatų pateikimui bus naudojamas su juo susieti spartinančiosios atminties objektai. Siekiant išvengti paieškos sulėtėjimo pasikeitus indeksams, automatiškai vykdomi foniniai procesai, kurie paruošia naują indekso „vaizdą“ ir pateikia jį užklausų aptarnavimui.

Apache Solr taip pat turi pilnavertę naudotojo sąsają administravimui, kurioje galima stebėti su paieška susijusią informaciją bei konfigūruoti įvairius paieškos mechanizmo parametrus:

- Statistinė informacija apie spartinančiosios atminties išnaudojimą, atnaujinimą ir užklausas.
- Interaktyvi dokumentų schemos naršyklė.
- Replikavimo proceso informacija.
- Žurnalizavimo parametrų nustatymas.
- WEB užklausų sąsaja.
- Aukšto patikimumo *Solr* klasterių būsenos informacija.

Žemiau esančiame paveiksle pateikiamas *Apache Solr* administravimo konsolės pavyzdys.



20 paveikslas. Apache Solr administravimo konsolės pavyzdys

2.3.11 Duomenų bazių valdymo sistema - PostgreSQL

Išnagrinėję sistemai keliamus reikalavimus bei remdamiesi ankstesnių projektų patirtimi, mes rekomenduojame sistemos kūrimui naudoti atvirojo kodo *PostgreSQL* duomenų bazių valdymo sistemą (DBVS).

PostgreSQL yra nemokama, atvirojo kodo reliacinių duomenų bazių valdymo sistema, naudojama daugelyje pažangių projektų, pasižyminti našumu, patikimumu ir plečiamumu (angl. scalability). PostgreSQL palaiko ACID principus, o tai užtikrina duomenų vientisumą ir patikimumą. Be to, ši duomenų bazių valdymo sistema palaiko transakcijas, vaizdus (angl. views), duomenų tipus, operatorius, indeksavimo metodus, XML formato duomenis ir kt., todėl yra laikoma pažangia, patikima atvirojo kodo alternatyva komercinėms reliacinių duomenų bazių valdymo sistemoms.

PostgreSQL gali veikti įvairiose operacinės sistemose, pvz. Windows ir Linux. Užklausos duomenims talpinti, atrinkti, atnaujinti ir pašalinti yra rašomos SQL (angl. *Structured Query Language*) kalba. *PostgreSQL* palaiko SQL, suderinamą su SQL'92 specifikacija. Be to, yra galimybė programuoti įvairius duomenų bazės veikimą modifikuojančius plėtinius (angl. *extensions*), pavyzdžiui, specialius indeksavimo mechanizmus.

PostgreSQL palaiko daugiapakopį saugumo mechanizmą, leidžiantį:

- kurti roles, apibrėžti jų teises;
- kurti naudotojus, nustatyti, kurių rolių teisėmis jie gali naudotis;
- vienai rolei paveldėti kitos rolės teises;
- nustatyti visų naujai sukurtų objektų pradines teises ir kt.
- nustatyti prieigą prie duomenų bazės objektų (pavyzdžiui, lentelių), stulpelių ir kt.

Siūlomas architektūrinis sprendimas ir *PostgreSQL* saugumo mechanizmas leis užtikrinti, kad niekas, išskyrus administratorius, negalės betarpiškai dirbti - keisti įrašų, gauti duomenų iš duomenų bazės. *PostgreSQL* taip pat turi vidinius duomenų vientisumo užtikrinimo mechanizmus.

Kartu su standartiniais *PostgreSQL* įrankiais yra instaliuojami ir įvairūs pagalbiniai komponentai, veikiantys komandinės eilutės pagalba, pavyzdžiui, pg_dump. Šis komponentas leidžia atlikti duomenų ir/arba duomenų bazės schemos archyvavimą į duomenų failą, todėl gali būti naudojamas duomenų rezerviniam kopijavimui ir duomenų atkūrimui po gedimų.

Taip pat reiktų atkreipti dėmesį į *PostgreSQL* įvykių žurnalizavimo galimybes, kurios užtikrina operacijų žurnalų pildymą. Sistemos administratorius gali skaityti, analizuoti ir interpretuoti aktyvius galiojančius ir perkeltus į archyvą žurnalus.

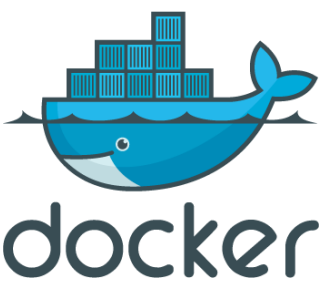
PostgreSQL duomenų bazių valdymo sistema, panaudodama technines serverio galimybes (procesoriaus branduolius), gali apdoroti daugiau nei vieną duomenų užklausą vienu metu (lygiagrečiai). Be to, yra daug įvairių būdų šią sistemą paleisti kaip duomenų klasterį (angl. *cluster*) keliuose įrenginiuose, pavyzdžiui, panaudojant bendrą diskų masyvą (angl. *shared storage*). Toks programinės įrangos lankstumas leidžia pritaikyti siūlomą sprendimą užsakovo poreikiams.

Darbui su *PostgreSQL* duomenų bazių valdymo sistema galima naudoti daug skirtingų įrankių, turinčių grafinę naudotojo sąsają, pavyzdžiui, *pgAdmin, Navicat* ir kt. Kartu su standartiniais *PostgreSQL* duomenų bazių valdymo sistemos komponentais yra siūloma įdiegti ir duomenų bazių administravimo įrankį *pgAdmin*.

PgAdmin įrankis leidžia atlikti visus įprastus duomenų bazių administravimo veiksmus:

- Duomenų bazių ir jų objektų peržiūrą, kūrimą, modifikavimą, šalinimą ir kt.;
- Duomenų lentelėse peržiūrą, modifikavimą, šalinimą ir kt.;
- Naudotojų teisių administravimą;
- Administruojamo serverio apkrovos ir kitų parametrų stebėseną;
- SQL užklausų vykdymą.

2.3.12 Konteinerių valdymas - Docker



Docker yra atvirojo kodo konteinerių valdymo programinė įranga. Ji veikia Linux operacinėje sistemoje.

Konteineriai yra tarsi lengvasvorės virtualios mašinos. Jie leidžia vykdyti programinį kodą atskirai nuo kitų konteinerių ir naudotis bendra technine įranga. Kitaip nei virtualiose mašinose, kiekvienam konteineriui nereikia paleisti atskiros operacinės sistemos, visi konteineriai naudojasi viena ir ta pačia operacine sistema, tai pagreitina aplikacijos paleidimo procesą. Kadangi konteineriai yra gerokai lengvesni (į juos įtraukiamos tik būtinos priklausomybės) už virtualias mašinas, todėl naudojant juos galima daug efektyviau išnaudoti serverių resursus.

> *Docker yra nemokama, atvirojo kodo konteinerių valdymo programinė įranga, kuri šiuo metu sparčiai populiarėja visame pasaulyje. Šia programine įranga naudojasi tokios pasaulinio garso įmonės kaip "eBay"," Spotify", "BBC News", "ING" ir dar daug kitų. Docker naudojimas supaprastina paskirstytųjų sistemų kūrimą, leisdamas įvairioms aplikacijoms ir procesams dirbti nepriklausomai vienas nuo kito toje pačioje fizinėje mašinoje ar keliose virtualiose mašinose. Tai leidžia paprastai panaudoti sistemai priskirtus naujus resursus, prijungti naujas fizines mašinas.*

Docker ypač palengvina mikroservisų architektūros įgyvendinimą, kadangi kiekvieną mikroservisą galima paleisti atskirame *Docker* konteineryje. Patalpinus aplikaciją (arba mikroservisą) ir jos priklausomybes į konteinerį, jį galima pernešti į bet kokį Linux serverį, nepriklausomai nuo to, kur tas serveris yra (kompiuteryje, debesyje ar kokioje nors kitoje aplinkoje), jei tik jame yra įdiegta *Docker* programinė įranga. Patalpinant aplikaciją į kitą serverį, nereikia rūpintis, kad jis būtų taip pat sukonfigūruotas, kadangi konteineris užtikrina visas reikiamas konfigūracijas.

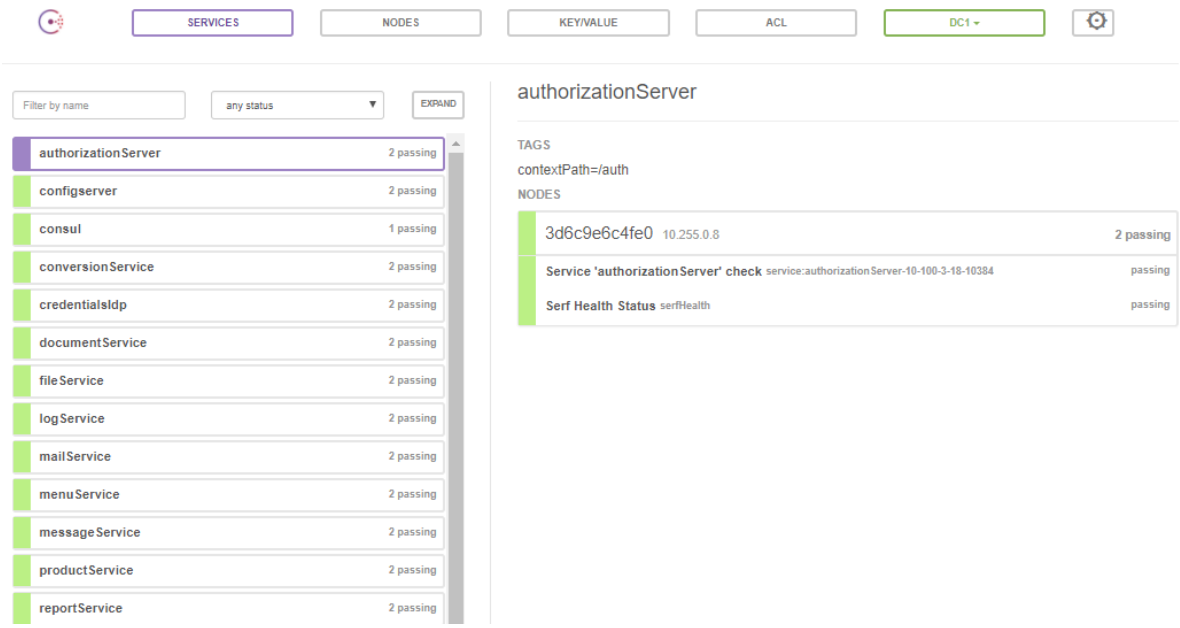
Naudojant *Docker* galima paleisti kelias skirtingas tos pačios programos versijas viename serveryje, tačiau kiekvienai versijai reikia priskirti skirtingus prievadus (angl. *port*), kad jas būtų galima pasiekti vienu metu.

Šios programinės įrangos kūrėjai daug dėmesio skiria saugumui. *Docker* naudoja sluoksninę apsaugą (angl. *layered security*), jis derina keletą apsaugos valdiklių, kad apsaugotų išteklius ir duomenis. Kai kurios Linux branduolio failų sistemos turi būti prijungtos konteinerio aplinkoje, tačiau didžiajai daliai šių failų sistemų galima nustatyti, kad jos būtų tik skaitomos, todėl privilegijuoti konteinerių procesai negali į jas rašyti ir paveikti serverio operacinės sistemos (angl. *host system*).

Docker naudoja kopijavimo rašant (angl. *copy-on-write*) failų sistemą. Konteineriai gali naudoti tą patį failų sistemos atvaizdą (angl. *file system image*), kaip pagrindą, bet kai konteineriai rašo, jie rašo į konteinerio specifinę failų sistemą. Taip užtikrinama, kad vienas konteineris negalės matyti kito konteinerio pakeitimų, net jei jie rašo į tą patį failo sistemos atvaizdą. Be to, vienas konteineris negalės pakeisti sistemos atvaizdo taip, kad įtakotų kitame konteineryje vykstančius procesus.

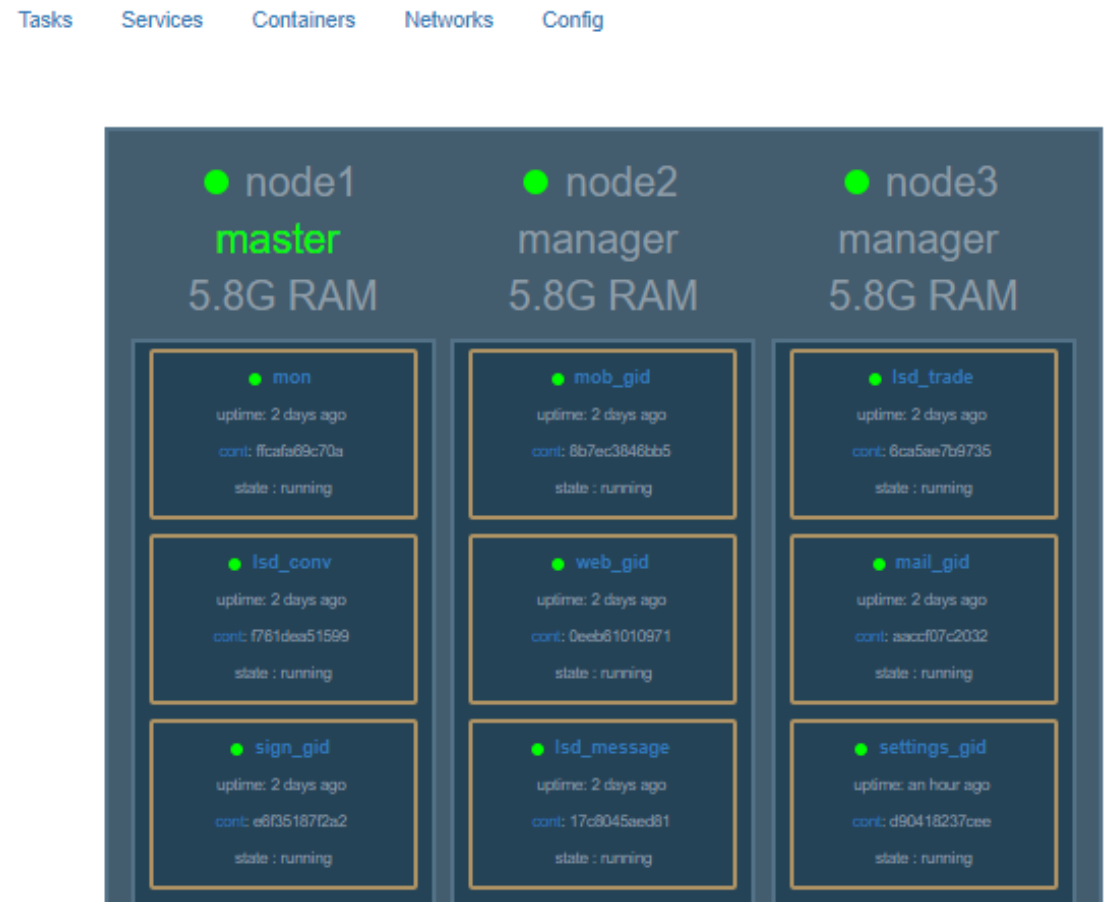
2.3.13 Servisų stebėsenos sprendimai

Kiekvienas sistemos servisas gali turėti būsenas „Aktyvus“ ir „Neaktyvus“, kurių visuma sudarys sistemos būseną. Jei visi servisai aktyvūs ir veikia tinkamai, tai sistema yra tinkamo veikimo būsenoje, kitu atveju sistema veikia ne pilnu pajėgumu ir administratorius turėtų sutvarkyti problemines sritis. Atskirų servisų būsenas turėtų būti galimybė peržiūrėti ir administruoti servisų registre (*discovery* mikroservisas). Žemiau esančiame paveiksle pateikiamas servisų registro ekranvaizdis.



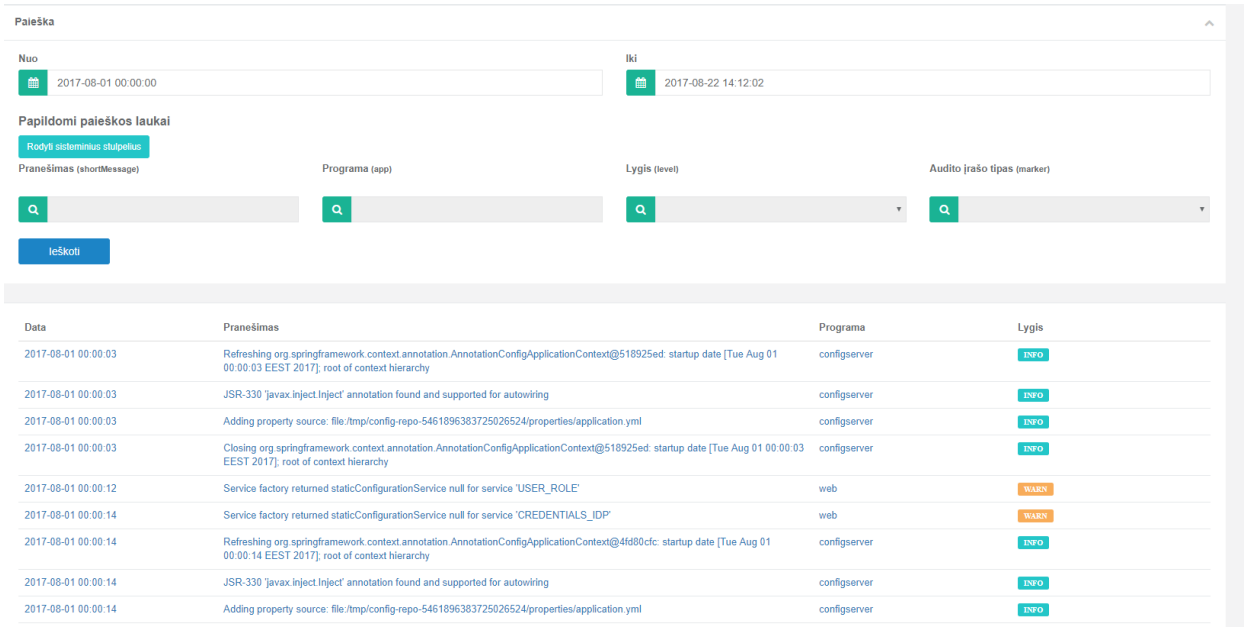
21 paveikslas. Servisų registro ekranvaizdis

Žvelgiant iš programinės įrangos konteinerių valdymo perspektyvos, administratoriui taip pat svarbu žinoti kokia yra kiekvieno *Docker* serverio ir jame veikiančių konteinerių būsena, todėl šiam tikslui numatyta speciali konteinerių būsenos stebėjimo konsolė, kurios pavyzdys pateiktas žemiau esančiame paveiksle.



22 paveikslas. Konteinerių būsenos stebėjimo konsolė

Būtina pažymėti ir tai, kad sistemos administravimui palengvinti, administratoriui bus sudaryta galimybė patogiai ieškoti ir peržiūrėti audito žurnalo įrašus. Žemiau esančiame paveiksle pateikiamas audito žurnalo paieškos ir peržiūros lango pavyzdys.



23 paveikslas. Audito žurnalo įrašų paieškos ir peržiūros langas

2.3.14 Duomenų saugos užtikrinimas

Sistema bus kuriama remiantis nepriklausomos saugumo organizacijos OWASP (angl. *Open Web Application Security Project*) teikiamomis rekomendacijomis. Konkrečios, Projektui aktualios, rekomendacijos bus derinamos analizės ir projektavimo etapų metu.

Bendru atveju duomenų apsaugos sprendimai turi užtikrinti, kad tik tinkamai identifikuoti ir autorizuoti naudotojai, kuriems suteiktos atitinkamos teisės, gali pasiekti sistemoje saugomus duomenis, atlikti jų peržiūrą bei redagavimą. Siekiant užtikrinti gerą sistemos duomenų apsaugą yra naudojami įvairūs naudotojų autentifikavimo bei perduodamų duomenų šifravimo sprendimai. Toliau trumpai aptarsime pagrindinius technologinius sprendimus, kuriuos numatome taikyti, siekdami užtikrinti duomenų apsaugą.

Naudotojų autentifikacija ir autorizacija

Sistemos saugumas bus užtikrinamas naudojant OAuth 2.0 protokolą bei kai kuriuos OpenId Connect protokolo principus. OpenId Connect – tai yra naujas identifikavimo protokolas paremtas pasaulyje plačiai paplitusia OAuth 2.0 specifikacija. Šis protokolas buvo kuriamas kaip pakaitalas senesniems protokolams tokiems kaip SAML ar WS-Federation. OIDC protokolas rūpinasi ir naudotojų identifikacija (skirtingai nuo OAuth 2.0, kuris rūpinasi tik autorizacija), tačiau neapriboja konkrečios realizacijos. Sistemoje turės būti realizuotas bent vienas tapatybės tikrinimo servisas (angl. *Identity Provider – IP*), kuris pasitelkdamas tam tikrą būdą turi atpažinti asmenį, pvz. naudojant tradicinį vartotojo vardo/slaptažodžio tikrinimą.

Sistemoje autentikacijai ir autorizacijai, skirtingai nuo įprasto OAuth 2.0, kuriame figūruoja prieigos raktai (angl. *access tokens*), kuriuos išduoda autorizacijos serveris (AS), bus naudojami tapatybės raktai (angl. *identity tokens*). Jie skirti tam, kad klientinė aplikacija (RP) galėtų ne tik „žinoti“, ar ji turi suteikti prieigą į tam tikrą resursą ar ne, bet ir žinoti:

- Kas yra tas naudotojas, kuris prašo prieigos;

- Kada, kur, kaip naudotojas buvo identifikuotas;
- Įvairūs papildomi atributai apie naudotoją.

Aukščiau paminėtas funkcionalumas OpenId Connect protokole įgyvendinamas pasitelkiant JWT formato tapatybės raktus. JWT (angl. *JSON Web Token*) - tai yra atviras standartas skirtas saugiai perduoti informaciją JSON formatu. OpenId Connect protokole JWT raktai yra pasirašomi ir jei yra poreikis užšifruojami. Visas tapatybės raktas gali būti užkoduotas į base64 simbolių eilutę, kuri gali būti naudojama ir perduodant raktą kaip URL parametrą.

Yra trys skirtingi būdai (angl. *flows*), kaip klientinei aplikacijai (RP) gauti tapatybės raktą. Pirmasis būdas „*authorisation code*“, antrasis „*implicit*“, trečiasis "*hybrid*". Pirmasis būdas yra aktualiausias, nes tinkamas naudoti tradicinėms tinklo bei mobilioms aplikacijoms ir yra saugiausias. Šio būdo įgyvendinimas susideda iš žingsnių:

1. *Klientinė aplikacija* gavusi užklausas iš *naudotojo naršyklės* prisijungti prie sistemos arba bandymą pasiekti apsaugotus servisus nukreipia naudotojo naršyklę į *autorizacijos serverį*. Nukreipiant nusiunčiama informacija, kur reikės nukreipti *naudotojo naršyklę,* kai naudotojas susiautentikuos bei papildomi parametrai saugumui užtikrinti (būsenos kodas (angl. *state*), kurį *autorizacijos serveriui* reikės panaudoti *naudotojo naršyklę* nukreipiant atgal).
2. Naudotojas *naršyklėje* pasirenka *tapatybės tikrinimo servisą. Naudotojo naršyklė* yra iškart nukreipiama į pasirinktą servisą.
3. Pasirinktame *tapatybės tikrinimo servise* naudotojas susiautentikuoja pagal pateiktas instrukcijas.
4. Po autentikacijos *naudotojo naršyklė* nukreipiama atgal į *autorizacijos serverį* kartu su identifikacine informacija apie susiautentikavusį asmenį (pvz. UUID ir tapatybės tikrinimo serviso tipas).
5. *Autorizacijos serveris* pagal naudotojo pasirinktą tapatybės tikrinimo serviso tipą ir UUID (gautus duomenis iš *tapatybės tikrinimo serviso*) atlieka užklausą į *naudotojų servisą* ir gauna informaciją, ar yra *sistemos naudotojas,* kuris gali jungtis su tokiu prisijungimu. Jei yra daugiau nei vienas, leidžiama *naudotojo naršyklės* lange pasirinkti, kurį *sistemos naudotoją* nori valdyti. Pagal gautus *sistemos naudotojo* duomenis parengiamas JWT formato raktas. Į jį įdedamos *sistemos naudotojo* rolės, *sistemos naudotojo* uuid ir kita būtina informacija. *Autorizacijos serveris* raktą pasirašo su savo privačiu raktu (angl. *private key*).
6. *Autorizacijos serveris* nukreipia *naudotojo naršyklę* atgal į pradinę įsimintą nuorodą kartu su įsimintu būsenos kodu. Kartu perduoda laikiną autorizacijos kodą.
7. *Klientinė aplikacija* patikrina ar nukreipiant į *autorizacijos serverį* nusiųstas būsenos kodas sutampa su gautu iš *autorizacijos serverio* kodu. Po to, su laikinu autorizacijos kodu kreipiamasi HTTP POST metodu į *autorizacijos serverį* ir gaunamas JWT raktas, kuris išsisaugo *klientinės aplikacijos* atmintyje susijusioje su HTTP sesija naršyklėje. *Naudotojo naršyklė* turi SESSIONID sausainiuką, kuris siunčiamas kiekvienos užklausos metu. Pagal jį, *naudotojo naršyklei* darant užklausą į *klientinę aplikaciją* ir atrenkamas *sistemos naudotojui* tinkamas JWT raktas ir kita būtina informacija.
8. *Klientinei aplikacijai* atliekant užklausas į kitus mikroservisus siunčiamas ir JWT raktas. Mikroservisuose jis validuojamas (su *autorizacijos serverio* viešu raktu (angl. *public key*)) ir panaudojama jame esanti reikalinga informacija.

Perduodamų duomenų šifravimas

Siekiant padidinti duomenų apsaugą, internetu perduodamos informacijos saugai užtikrinti bus naudojamas SSL kriptografinis protokolas.

Duomenų autentiškumas ir vientisumas

Visais sistemos panaudojimo atvejais turi būti užtikrinamas asmens duomenų autentiškumas ir vientisumas. Loginiam duomenų integralumui (vientisumui) užtikrinti bus naudojamos šios priemonės:

- duomenų įvesties validavimo taisyklių naudojimas el. formose;
- ACID principus palaikančios DBVS naudojimas – siūloma PostgreSQL DBVS pilnai palaiko ACID principus;
- išorinių raktų (angl. foreign key) naudojimas, siekiant užtikrinti referentinį integralumą;
- pirminių raktų (angl. primary key) naudojimas siekiant užtikrinti esybių integralumą;
- klasifikatorių naudojimas duomenų įvedimo formose;
- prieigos teisių prie duomenų valdymas.

Sistemoje bus realizuotas įvykių žurnalizavimo modulis, prie kurio bus prijungiami visi kiti moduliai. Tai leis nesudėtingai identifikuoti naudotojus, kurie atliko pakeitimus, bei patikrinti, kokie sistemos pakeitimai buvo įvykdyti.

Sistemos komponentų atskyrimas tinklo lygmenyje

Mūsų siūloma diegimo architektūra (2.2.3 skyrius) suteikia galimybę sistemos komponentus atskirti į atskiras tinklo zonas, priklausomai nuo komponento jautrumo. Taip pat norime pažymėti, kad skaidymą į atskiras tinklo zonas bus galima atlikti *Docker* priemonėmis, panaudojant tinklų virtualizavimo technologiją (VXLAN). Tokiu būdu bus galima atskirti viešai prieinamą sistemos dalį nuo vidinių servisų prie kurių yra ribojama prieiga.

2.3.15 Duomenų praradimo ir atstatymo scenarijai

Duomenų bazėse saugomi duomenys gali būti prarasti dėl visokių priežasčių – pradedant įsilaužimu į sistemos serverį ar duomenų saugyklos gedimu ir baigiant netyčiniu duomenų ištrynimu ar pakeitimu, kai tai padaro sistemos naudotojas ar administratorius. Siekiant išvengti šių scenarijų, organizacijoje turi būti įdiegtos techninės, programinės ir organizacinės duomenų praradimo prevencijos priemonės. Vis dėl to visų scenarijų neįmanoma numatyti ir nuo jų apsisaugoti, todėl turi būti realizuotos duomenų atstatymo priemonės. Šiuo tikslu sukurtoje sistemoje bus realizuotas rezervinis kopijavimo ir atstatymo sprendimas.

Docker infrastruktūroje rezervinis kopijavimas ir atstatymas skiriamas į dvi duomenų kategorijas: *Docker* šablonai (angl. *images*) ir *Docker* konteineriai. Toliau pateikiama apibendrinta *Docker* konteinerių rezervinio kopijavimo procedūra, kuri bus automatizuota panaudojant iš anksto paruoštą scenarijų (angl. *script*):

1. Surandami konteinerio identifikatoriai panaudojant *docker ps* komandą;
2. Panaudojant *docker commit* komandą padaromos pageidaujamų konteinerių rezervinės kopijos;
3. Konteinerio rezervinė kopija suspaudžiama į archyvą (pvz. .tar formato);
4. Konteinerio rezervinė kopija perkeliama į išorinį failų katalogą. Perkėlimui gali būti naudojamos standartinės operacinės sistemos priemonės tokios kaip *rsync* arba *scp*.

Norint atstatyti *Docker* konteinerio rezervinę kopiją vykdoma ši procedūra:

1. Panaudojant *docker load* komandą rezervinė kopija perkeliama į serverį;
2. Panaudojant *docker run* komandą, rezervinės kopijos pagrindu paleidžiamas naujas konteineris

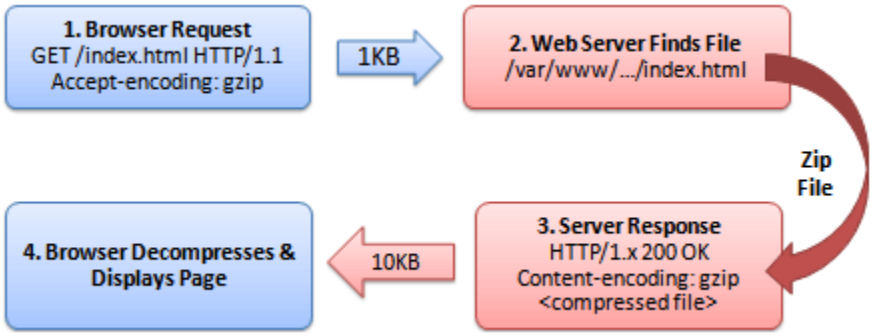
Kai kuriais atvejais (pvz. konteineriai, kuriuose veikia duomenų bazės) *Docker* konteinerio duomenys saugomi išorinėse failų saugyklose (angl. *data volumes*). Tokiu atveju vykdoma *docker run* komanda su atitinkamais parametrais, kuri padaro pageidaujamos failų saugyklos rezervinę kopiją.

Būtina pažymėti ir tai, kad duomenų bazių serverių rezervinės kopijos bus daromos SQL scenarijų (angl. *script*) pagalba, kurie bus vykdomi pačiame duomenų bazių valdymo sistemos konteineryje.

2.3.16 Duomenų apdorojimo efektyvumo užtikrinimas

Žemiau pateikiamos atskirų sistemos komponentų duomenų apdorojimo efektyvumo didinimo priemonės ir būdai:

1. Perduodamų duomenų kiekio mažinimas. Perduodamų duomenų kiekiui mažinti naudojamas išeinančio srauto suglaudinimas (naudojant šiuolaikinių naršyklių palaikomą GZIP algoritmą), statinių resursų sujungimas, taip pat gali būti naudojamos kitos puslapio dydžio sumažinimo rekomendacijos, pateiktos Google programuotų bendruomenės puslapyje adresu: https://developers.google.com/speed/docs/best-practices/request



24 paveikslas. HTTP srauto suglaudinimas naudojant GZIP

2. Duomenų bazių užklausų greitaveikos didinimas. Visos su duomenų ištraukimu susijusios operacijos bus atliekamos laikantis *PostgreSQL* bendruomenės pateikiamų duomenų bazės operacijų našumo rekomendacijų, pateikiamų adresu: http://wiki.postgresql.org/wiki/Performance_Optimization.
3. Bendro užklausų skaičiaus mažinimas. Bendras užklausų skaičius bus mažinamas dviem būdais:
    a. Perkeliant su duomenų ištraukimu nesusijusią logiką į naršyklėje vykdomą programinį kodą, naudojant *JavaScript* kalbą.
    b. Saugant statinius duomenis naudotojo naršyklėje, naudojant *expire headers* serverio formuojamo atsakymo atributus.
4. Sistemos apkrovos paskirstymas. Sistemos duomenų apdorojimo našumas gali būti didinamas plečiant sistemos techninius resursus. Sistema galės būti plečiama dviem būdais:
    a. naudojant vertikalų plečiamumą – didinant sistemos tarnybinių stočių pajėgumus, tai yra atminties kiekį, procesorių skaičių ir pan. Maksimalias plečiamumo galimybes nusako serverių techninės specifikacijos.
    b. naudojant horizontalų plečiamumą – didinant sistemą aptarnaujančių tarnybinių stočių skaičių.
5. Ilgiau trunkantys sistemos procesai bus vykdomi asinchroniniu būdu, o naudotojui ekrane bus pateikta informacija, apie proceso vykdymo progresą ir pabaigą. Tokiu būdu naudotojas galės pereiti į kitą puslapį proceso vykdymo metu, tačiau procesas tęsis.

2.3.17 Sistemos prieinamumas

Prieinamumas - tikimybė, kad sistema tam tikrame laiko taške bus dirbanti ir galinti pristatyti reikalaujančią paslaugą. Prieinamumą, kaip ir patikimumą, galima išreikšti kiekybiškai, tačiau skaičiuojant prieinamumo rodiklį yra įvertinimas ir sistemos taisymo laikas.

Kartais įmanoma įtraukti prieinamumo sąvoką į sistemos patikimumą (jei sistema neprieinama, ji ir nesuteikia sistemai apibrėžtų paslaugų). Tačiau įmanoma situacija, kai sistema yra prieinama, bet nevykdo specifikacijoje nurodytų funkcijų. Tokiose sistemose mažas patikimumas nebus problema, jei sistemos trūkumai galės būti greitai pataisomi ir nepažeis

duomenų. Siekdami didinti sistemos prieinamumą, projekto metu numatome naudoti eilę prieinamumo didinimo priemonių:

- Iš naršyklės ateinančio HTTP srauto balansavimo technologija Nginx, kuri užtikrina automatinį srauto perjungimą serviso sutrikimo atveju.
- Servisų katalogas ir užklausų balansavimo bibliotekos, kurių pagalba bus vykdomas vidinių sistemos užklausų balansavimas ir nukreipimą į veikiančius servisus. Esant vieno konteinerio sutrikimui užklausos gali būti nukreipiamos į kitą, dubliuojantį konteinerį.
- Automatinis *Docker* konteinerių perkrovimas sutrikimo atveju.
- Rezervinės duomenų bazės kopija, kuri gali būti talpinama į atskirą konteinerį. Sutrikus pagrindinei DB, užklausos gali būti nukreipiamos į dubliuotą duomenų bazę.
- Konteinerių valdymo platformos *Docker* klasterizavimo technologija, kuri leidžia valdyti konteinerius paskirstytus per kelis fizinius (arba virtualius) serverius.

2.3.18 Sistemos plečiamumas ir pakartotinis panaudojimas

Plečiamumas tai sistemos galimybė susitvarkyti su nuolat didėjančia apkrova pridedant naujų arba tobulinant/greitinant jau egzistuojančius resursus. Šį parametrą yra sunku išmatuoti konkrečia skaitine verte ir paprastai tai būna kompromiso ieškojimas konfigūruojant įvairias sistemos dalis. Norint pagreitinti vieną iš sistemos funkcijų dažnai neužtenka tik pridėti papildomą serverį pavyzdžiui į taikomųjų serverių telkinį, nes tai padarius gali paaiškėti, jog duomenų bazės serveriai nespėja aptarnauti taikomųjų programų serverių užklausų. Norint pakelti sistemos greitaveiką reikia žiūrėti į sistemą kaip į visumą, o ne vien tik į atskirus jos komponentus. Idealiu atveju visos sistemos dalys turėtų turėti maždaug vienodą apkrovimą – tai reikštų techninė ir programinė dalys yra puikiai suprogramuotos, sukonfigūruotos ir techninė įranga dirba optimaliu režimu. Plečiamumas gali būti matuojamas pagal įvairias dimensijas pvz. funkcinis plečiamumas, našumo plečiamumas, geografinis plečiamumas ir pan.

Išskiriami du būdai padidinti sistemos našumą techninės įrangos atžvilgiu:

- Vertikalus plėtimas (angl. *Scale-up*) – kai didinamos sistemą aptarnaujančio serverio arba serverių vidinės techninės galimybės (procesoriaus, disko greitis, operatyviosios atminties kiekis ir kt.);
- Horizontalus plėtimas (angl. *Scale-out*) – kai didinamas sistemą aptarnaujančių serverių kiekis. Šis būdas daugeliu atžvilgiu yra pranašesnis už vertikalųjį, kadangi leidžia pasiekti geresnį kainos/našumo santykį, turi aukštesnes išplėtimo ribas, tuo pačiu leidžia užtikrinti nepertraukiamą sistemos darbą. Tačiau, skirtingai nuo vertikalaus sistemos plečiamumo, horizontalus plečiamumas nėra įmanomas be tinkamos sistemos architektūros. Siūloma sistemos plečiamumo architektūra pilnai užtikrina sistemos horizontalų plečiamumą ir yra išbandyta didelės apkrovos projektuose.

Siūloma tokia bendra našumo plėtimo ir apkrovos paskirstymo architektūra:

- Klientinių užklausų balansavimo mazgas (Nginx). Tai komponentas per kurį naudotojų užklausos patenka į klientinių paslaugų sluoksnį. Šis mazgas atsakingas už tolygų užklausų paskirstymą tarp konteinerių, kuriuose veikia klientinių paslaugų modulis (API Gateway). Esant poreikiui apkrovos paskirstymo mazgas gali būti dubliuojamas, tam kad užtikrinti nepertraukiamą sistemos darbą, įvykus vieno mazgo sutrikimams. Būtina pažymėti ir tai, kad tokia architektūra yra visiškai skaidri programiniams klientams, jie dirba taip lyg jungtųsi į vieną serverį. Apkrovos balansavimui siūloma naudoti Nginx programinę įranga, kuri naudojama daugelyje didelių projektų ir organizacijų. Nginx – tai lengvasvoris HTTP serveris turintis labai dideles konfigūravimo ir panaudojimo galimybes. Jis bus naudojamas apjungimui ir susiejimui visų HTTP sąsajų ir jų pateikimui vartotojams. Keletas portalų kurie naudoja Nginx: Netflix, NetDNA, SoundCloud ir panašiai. Tai portalai sulaukiantys labai didelio lankytojų skaičiaus ir apkrovimo, o Nginx su tuo puikiai susitvarko.

- Vidinių užklausų balansavimas. Vidininės užklausos - tai užklausos, kurios vyksta iš vieno mikroserviso (A) į kitą (B) pvz. iš klientinių paslaugų modulio į dokumentų modulį. Vidinių užklausų balansavimas vykdomas tokiu principu:
    - Servisų katalogas periodiškai tikrina servisų prieinamumą ir jeigu servisas neprieinamas, jo URL yra pašalinamas iš balansavimo grupės.
    - Mikroservisas (A) kreipiasi į servisų katalogą.
    - Servisų katalogas grąžina aktyvius URL adresus, kuriais galima pasiekti mikroservisą (B);
    - Mikroservise (A) naudojama HTTP kliento biblioteka (pvz. https://github.com/Netflix/feign), gebanti užklausas balansuoti tarp visų aktyvių URL adresų.
- Esant poreikiui gali būti didinamas ir duomenų bazės našumas, panaudojant PostgreSQL duomenų bazės replikavimo galimybę. Tokiu būdu dalis duomenų nuskaitymo užklausų (SQL select) būtų nukreipiamos į duomenų bazės repliką.

Aprašytas principas leidžia horizontaliai plėsti sistemos našumą, t.y. jeigu nebepakanka konkretaus mikroserviso našumo, galima nesunkiai paleisti papildomus mikroserviso egzempliorius (išimtis taikoma tik mikroservisams saugantiems duomenis). Tuo pačiu yra užtikrinamas nepertraukiamas sistemos darbas: sutrikus vienam mikroserviso egzemplioriui, visas srautas gali būti nukreipiamas į kitą egzempliorių. Kaip minėta ankstesniuose skyriuose, sistemos komponentus planuojama diegti keliuose serveriuose, o kiekviename iš jų gali būti veikti skirtingi mikroservisai (konteineriai), todėl serverių apjungimui į bendrą telkinį numatoma panaudoti *Docker* klasterizavimo įrankį.

Sistemos plečiamumas funkcine prasme

Sistemos plečiamumas funkcine prasme apima esamų sistemos funkcijų tobulinimą bei naujų sukūrimą. Tikėtina, kad sprendimo realizavimui pasirinkus mikroservisų architektūrą, ateityje palengvės ir funkcinis sistemos plečiamumas, kadangi atisradus poreikiui praplėsti sistemos funkcionalumą bus galima sukurti naujus mikroservisus ir juos integruoti į klientinių paslaugų modulį bei, esant poreikiui, atlikti susijusių modulių pakeitimus.

Būtina pažymėti tai, kad funkcinio plečiamumo poreikis dažnai kyla dėl sistemoje naudojamų duomenų struktūrų ar veiklos esybių atributų pasikeitimų. Tokiais atvejais kai duomenų struktūrų pasikeitimai yra minimalūs (pvz. atsiranda vienas papildomas laukas), patogu turėti administravimo priemones, leidžiančias įgyvendinti šiuos pakeitimus, be programavimo darbų.

Vertinant Projekto kontekstą, ypač svarbus yra Sistemos plečiamumas surenkamų ir apdorojamų duomenų požiūriu, t.y. kad sukurta programine įranga galėtų naudotis kitos ES šalys arba ateityje atsiradus papildomiems duomenų šaltiniams, juos būtų galima nesunkiai integruoti į bendrą Sistemos duomenų bazę. Būtent dėl šios priežasties, Tiekėjas siūlo gana lankstų duomenų surinkimo ir apdorojimo komponentą – Pentaho data integration, kuris detaliai aprašytas skyriuje „Duomenų surinkimo ir apdorojimo komponentas - Pentaho data integration“.

Pakartotinis panaudojimas

Siūloma Sistemos architektūra ir jos realizacija palaiko Sistemos pakartotinį panaudojimą įdiegiant ją atskirai į reikalavimus atitinkančią techninę įrangą (pvz. įdiegiant Sistemą kitoje šalyje, siekiant Sistemą naudoti būtent tos šalies duomenų surinkimui, atvaizdavimui ir analizei).

Projekto metu bus paruošti *Docker* atvaizdai (angl. *images*), kurie sudarys galimybę Sistemą lengvai įdiegti naujoje techninėje įrangoje su minimaliu konfigūravimo kiekiu. Pakartotinai panaudota Sistema bus visiškai atskirta nuo pradinės Sistemos:

- Naudos visiškai atskirtu programiniu kodu – tik jai skirtais komponentais;
- Naudos tik jai skirtą duomenų bazę;
- Išlaikys visus pradinei Sistemai iškeltus reikalavimus ir gebėti atlikti tas pačias funkcijas.

Apibendrinant galima teigti, kad siūlomas architektūrinis sprendimas pasižymi šiomis savybėmis:

- siūlomas architektūrinis ir techninis sprendimas užtikrina galimybę plėsti sistemą tiek funkcine prasme, tiek ir našumo prasme;
- sistemos plečiamumą užtikrins ir atvirojo kodo programinių sprendimų naudojimas visuose sistemos architektūros komponentuose;
- sistemos pajėgumą bus galima didinti tiek vertikaliu, tiek horizontaliu būdu. Be to, mikroservisų architektūros naudojimas suteikia galimybę plėsti tik konkretaus mikroserviso pajėgumą;
- sistemos pajėgumus bus galima plėsti prijungiant papildomą techninę įrangą;
- *Docker* infrastruktūros dėka, bus užtikrinta galimybė pakartotinai panaudoti sukurtą programinę įrangą.

2.4 Informacinės sistemos veiklos rizikos

Šiame poskyryje aprašomos probleminės situacijos ir nustatytos sistemos veiklos (veikimo) rizikos, pateikiamos pagrįstos ir efektyvios sąnaudų prasme jų valdymo ir sprendimo priemonės.

Kiekvienos programinės įrangos naudojimas yra neatsiejamas nuo įvairių sistemos veiklos rizikų. Siekiant, kad šios rizikos turėtų kuo mažiau neigiamos įtakos, svarbu jas nustatyti kuo ankstesnėje stadijoje ir pasirinkti tinkamas rizikų valdymo priemones. Visų šios srities rizikų įvardinti praktiškai neįmanoma, tačiau mes išskyrėme pagrindines ir jas pateikiame žemiau esančioje lentelėje. Analizuojant rizikos veiksnius taikoma metodika aprašyta šio pasiūlymo 6 skyriuje.

12 lentelė. Pagrindiniai sistemos veiklos rizikos veiksniai ir jų valdymo priemonės

Eil. Nr.	Rizika	Rizikos poveikis	Rizikos tikimybė	Rizikos valdymo priemonės	Atsakingos šalys
1.	**Darbo praktikos rizikos**				
1.1.	Sistemos naudotojų nesugebėjimas vykdyti veiklos funkcijų kokybiškai ir efektyviai	M	M	Siekiant užtikrinti tinkamas sistemos naudotojų kompetencijas naudotis sukurtu sprendimu bus parengtas naudotojų gidas ir vykdomi naudotojų mokymai. Papildomai Perkančioji organizacija turės pasirūpinti, kad visi sistemos naudotojai dalyvautų suplanuotose mokymuose.	Paslaugų teikėjas, Perkančioji organizacija
1.2.	Sistemos administratorių nesugebėjimas tinkamai atlikti administravimo funkcijų, lemiantis sumažėjusį sistemos veiklos našumą	M	M	Siekiant užtikrinti tinkamas sistemos administratorių kompetencijas efektyviai administruoti sukurtą programinę įrangą bus parengtas administratorių gidas ir vykdomi administratorių mokymai. Papildomai Perkančioji organizacija turės pasirūpinti, kad visi sistemos administratoriai dalyvautų suplanuotose mokymuose.	Paslaugų teikėjas, Perkančioji organizacija
1.3.	Techninės įrangos resursų, reikalingų užtikrinti sklandų sistemos darbą, trūkumas	V	M	Tiekėjas jau pasiūlymo rengimo metu identifikavo preliminarų techninės įrangos poreikį. Projekto metu techninės įrangos reikalavimai bus patikslinami, kad įdiegta programinė įranga veiktų sklandžiai ir našiai	Perkančioji organizacija
2.	**Informacijos rizikos**				
2.1.	Nepakankama informacijos	V	M	Sistemoje saugomos informacijos kokybė ir prieinamumas gali būti užtikrinami įvairiomis	Paslaugų teikėjas,

Eil. Nr.	Rizika	Rizikos poveikis	Rizikos tikimybė	Rizikos valdymo priemonės	Atsakingos šalys
	kokybė ir prieinamumas			programinėmis ir techninėmis priemonėmis. Paslaugų teikėjas turi įgijęs daug patirties šioje srityje, todėl Paslaugų teikimo metu įdiegs logiškas ir efektyvias programines kokybės ir prieinamumo užtikrinimo priemones. Techninėmis priemonės rūpintis turėtų pati Perkančioji organizacija.	Perkančioji organizacija
2.2.	Netinkamas informacijos pateikimas (dizainas)	M	V	Siekiant, kad informacijos pateikimas atitiktų numatytus sistemos naudotojus, informacijos pateikimo būdai bus suprojektuoti Paslaugų teikimo metu remiantis tarptautiniais standartais ir gerosiomis praktikomis bei suderinti su Perkančiąja organizacija, o kaip jais naudotis bus pristatyta mokymų metu.	Paslaugų teikėjas
2.3.	Nepakankamas informacijos saugumas	D	V	Informacijos saugumo rizika apima daug įvairių smulkesnių rizikų, kurios nusako, kaip gali būti pakenkta informacijai. Remiantis tarptautiniais standartais ir gerosiomis praktikomis Paslaugų teikėjas įgyvendins efektyvias ir praktiškas programines saugos priemones, kurios pakels tvarkomos informacijos saugumo lygį. Techninių ir organizacinių saugos užtikrinimo priemonių naudojimą turės užtikrinti pati Perkančioji organizacija.	Perkančioji organizacija, Paslaugų teikėjas
3.	**Technologijų rizikos**				
3.1.	Naudojamos technologijos yra sudėtingos ir neefektyvios, neužtikrina našumo	M	M	Paslaugų teikėjas yra įgyvendinęs sistemų kūrimo projektus, kurių metu buvo išbandytos planuojamos naudoti technologijos ir įsitikinta jų efektyvumu, kokybe bei našumu.	Paslaugų teikėjas
3.2.	Programinės įrangos klaidos lemia sistemos strigimus, mažiną efektyvumą ir našumą	V	D	Siekiant išvengti PĮ klaidų, Paslaugų teikimo metu bus atliekamas sukurto sprendimo testavimas, o surastos klaidos pašalintos. Garantinio aptarnavimo metu atsiradus naujoms PĮ klaidoms ir sutrikimams, jie bus šalinami Paslaugų teikėjo garantinio aptarnavimo reglamente suderinta tvarka.	Paslaugų teikėjas
3.3.	Naudojamų technologijų palaikymas yra sudėtingas ir brangus	M	M	Kuriant sistemą bus naudojami tik praktiški, efektyvūs ir paprasti technologiniai sprendimai, todėl naudojamų technologijų palaikymas nebus sudėtingas ir brangus. Taip pat pirmaisiais metai po PĮ sukūrimo palaikymo paslaugas teiks Paslaugų teikėjas.	Paslaugų teikėjas
3.4.	Naudojamos technologijos nesuderinamos su kitomis susijusiomis sistemomis	V	M	Paslaugų teikėjas atliks susijusių sistemų analizę ir užtikrins jų suderinamumą su kuriama PĮ. Atsiradus susijusių sistemų pokyčiams po Paslaugų teikimo pabaigos. Perkančioji organizacija turės pati pasirūpinti suderinamumą užtikrinančių sprendimų diegimu ir priežiūra.	Paslaugų teikėjas, Perkančioji organizacija
4.	**Infrastruktūros rizikos**				
4.1.	Žmogiškoji, techninė ar informacinė infrastruktūra yra nepakankama užtikrinti nuolatinį sukurtos sistemos veikimą ir palaikymą	D	M	Paslaugų teikėjas pateiks reikiamo galingumo serverius ir reikiamos talpos duomenų saugyklą. Perkančiosios organizacija turės užtikrinti nepertraukiamą duomenų centro infrastruktūros (elektra, vėsinimas, interneto prieiga ir pan.) veikimą.	Perkančioji organizacija, Paslaugų teikėjas
5.	**Strategijos rizikos**				

Eil. Nr.	Rizika	Rizikos poveikis	Rizikos tikimybė	Rizikos valdymo priemonės	Atsakingos šalys
5.1.	Organizacijos strategija pasikeičia, sukurdama arba sustiprindama neatitikimus su sistemos strategija	M	M	Pasikeitus Perkančiosios organizacijos strategijai, tikslams ar poreikiams, Perkančioji organizacija turės suplanuoti ir įgyvendinti sprendimus, užtikrinančius atitiktį minėtiesiems aspektams.	Perkančioji organizacija

2.5 Gerųjų praktikų taikymas

Šiame poskyryje detaliai aprašomos Paslaugų teikimo metu planuojamos taikyti gerosios praktikos ir atvirųjų sistemų sąveikos principai.

Paslaugų teikėjas, remdamasis savo patirtimi, siūlo Projektą vykdyti pagal šiam Projektui adaptuotą programinės įrangos gyvavimo ciklo valdymo metodiką – *Open Unified Process* (toliau – OpenUP). OpenUP – tai supaprastinta iteracinės *Rational Unified Process* (RUP) / *Unified Process* (UP) metodikos versija, ji išlaiko esminius RUP/UP principus, tokius kaip iteracijos, panaudos atvejais paremtas programinės įrangos kūrimas, rizikos valdymas, orientacija į architektūrą ir pan., tačiau tuo pačiu atsisako arba apjungią dalį perteklinių disciplinų ir tokiu būdu labiau orientuojasi į judresnį (angl. *Agilė*), bendradarbiavimu paremtą programinės įrangos kūrimą.

OpenUP metodika pateikia tam tikrą rinkinį gerųjų praktikų, kurios gali būti taikomos įgyvendinant projektą. Būtina pažymėti tai, kad nebūtina taikyti visas metodikoje siūlomas praktikas, t. y. gerosios praktikos gali būti komponuojamos tarpusavyje, priklausomai nuo konkretaus projekto poreikių. Tolimesniuose poskyriuose trumpai aprašysime pagrindines praktikas, kurias planuojame taikyti projekto vykdymo metu.

Atkreipiame dėmesį į tai, kad mūsų pasirinkta metodika suderinama su LR LST ISO/IEC 12207:2008 standartu ir neprieštarauja Informacinės visuomenės plėtros komiteto prie Susisiekimo ministerijos direktoriaus įsakymui „Dėl valstybės informacinių sistemų gyvavimo ciklo valdymo metodikos patvirtinimo“, 2014-02-25, Nr. T-29.

2.5.1 Projekto vykdymo ir valdymo praktikos

Šiame skyriuje aprašomos projekto vykdymo ir valdymo gerosios praktikos, kuriomis vadovaujantis yra lengviau pasiekiamas užsibrėžtas tikslas, t. y. sukurti kokybiškai veikiančią sistemą.

2.5.1.1 Iteracinis programinės įrangos kūrimas

Iteracija yra santykinai mažos apimties veikla, kuri gali trukti nuo kelių dienų iki kelių savaičių. Dažniausiai iteracijos trukmė priklauso nuo projekto specifikos ir keliamų reikalavimų. Vykdant iteracinį programinės įrangos kūrimą kiekviena iteracija yra skirta tam tikros sistemos dalies įgyvendinimui. Iteracijos pabaigoje projekto komanda turi pasiekti numatytus tikslus, sukuriant arba modifikuojant suplanuotus sistemos modulius, adaptuojant įdiegtus produktus, ir paruošti tarpinę (nepilną, bet veikiančią) kuriamos sistemos versiją. Kiekvienoje iteracijoje gali būti atliekamas programavimas, reikalavimų detalizavimas, testavimas, bandomoji eksploatacija, projektavimo dokumentacijos patikslinimas. Realizavus paskutinės iteracijos reikalavimus, atliekamas visų iteracijų rezultatų testavimas.

Vykdant iteraciją, numatytais laiko tarpais, yra organizuojami iteracijos aptarimo susitikimai. Šių susitikimų metu yra aptariami atlikti ir vykdomi programavimo darbai. Taip pat, sprendžiamos problemos iškilusios vykdant užduotis. Vykdomų susitikimų metu paskirstomos nepadarytos užduotys.

Iteracijos planavimas, vertinimas bei progreso stebėjimas yra koncentruotas į užduotis. Atsižvelgiant į aukščiausio prioriteto užduotis yra sudaromas iteracijos planas. Kiekvienos iteracijos planavimui yra skiriamos kelios valandos. Planavimo metu yra nusprendžiama koks tikslas bus siekiamas, kiek laiko truks iteracija, kaip dažnai bus vykdomi iteracijos aptarimai ir pan. Atlikus iteracijos planavimą yra vykdomas reikalavimų detalizavimas ir užduočių paskirstymas programuotojams. Ši veikla dažniausiai trunka kelias dienas. Sudėliojus užduotims prioritetus ir paskirsčius jas programuotojams, pradedami programavimo darbai. Iteracijoje daugiausia laiko yra skiriama programavimo darbams. Vykdant programavimo darbus, iteracijos plane numatytu laiku, yra kuriamos stabilios sistemos versijos. Dažniausiai paskutinė savaitė ar paskutinės kelios iteracijos dienos yra skiriamos klaidų taisymui siekiant užtikrinti stabilią sistemos versiją.

Pati iteraciją yra skaidoma į programinio kodo mikroprieaugius (angl. *micro-increment*). Mikroprieaugis yra konkretaus projekto komandos darbo rezultatas, t. y. vieno ar kelių programuotojų darbo rezultatas atliktas per kelias valandas ar dienas. Mikroprieaugio koncepcija programuotojams leidžia išskaidyti atliekamas užduotis į mažesnės apimties užduotis ir tokiu būdu geriau įvertinti iteracijos vykdymą ir priimti korekcinius veiksmus.

Rekomenduojama, kad į iteracijos vykdymą būtų traukiama kiek įmanoma mažiau pakeitimų, kadangi iteracija turi būti laike apibrėžtos trukmės. Pakeitimais neturėtų būti laikomos klaidos ar netikslumai aptikti testavimo metu. Todėl, aptikti netikslumai turi būti taisomi einamosios iteracijos metu.

Lyginant iteracinę ir klasikinę (krioklio) projekto vykdymo metodiką galima išskirti šiuos iteracinio projekto vykdymo pranašumus:

- Iteracijų vykdymo metu anksčiau yra pastebimos rizikos keliančios pavojų sėkmingam projekto įgyvendinimui. Pavyzdžiui: parengtoje sistemos versijoje užsakovas pastebi jog funkcija yra realizuota netinkamai, tai programuotojai gali atlikti pakeitimus.
- Vadovaujantis iteracinio projekto vykdymo metodika kiekvienos iteracijos metu yra sukuriama nauja sistemos versija.
- Galimybė dirbti efektyviau, kadangi iteracijos metu gali būti atliekamos kelios gyvavimo ciklo veiklos nuo reikalavimų detalizavimo iki sukurtos sistemos testavimo.
- Lengviau aptinkamos klaidos, kurios gali būti ištaisomos vykdant tolimesnes iteracijas.

Planuojant iteraciją, rekomenduojama įvertinti užduotis, t. y. priskirti užduotims balus pagal sudėtingumą. Vertinant užduotis nereiktų kreipti dėmesio į laiką, kuris reikalingas jai atlikti. Atlikus užduočių vertinimą, galima nustatyti ir įvertinti iteracijos greitį. Greitis yra pagrindinis kriterijus į kurį atsižvelgiama planuojant iteraciją. Tarkim, iteracijoje buvo suplanuota atlikti užduočių, kurios atitinka 20 balų. Iteracijos pabaigoje buvo nustatyta jog atlikta užduočių, kurių balas atitinka 14. Taigi, iteracijos greitis yra 14. Planuojant sekančią iteraciją, greitis buvo sumažintas iki 18 balų atsižvelgiant į tai, kad komanda dirbs efektyviau.

Kiekvienos iteracijos greitis gali kisti priklausomai nuo joje vykdomų užduočių ir komandos narių turimos patirties. Taip pat iteracijos greičiui turi įtakos komandos struktūra, projekto vykdymo metu įgyti įgūdžiai ir pan. Dažniausiai projekto eigoje vykdymo greitis kyla, nes komanda geriau supranta kuriamą sistema bei dirba glaudžiau.

Rekomenduojama, kad iteracijos rezultatai būtų pristatomi užsakovui. Užsakovas yra supažindinamas su atliktais darbais ir pasiektais rezultatu - dokumentacija, nauja sistemos versija ir pan. Užsakovas gali pateikti komentarus apie pristatytus rezultatus, trūkstamas funkcijas bei idėjas tolimesniam sistemos vystymui. Iteracijos vertinimo metu turėtų būti atsakomi šie klausimai:

- Ar buvo pasiekti užsibrėžti tikslai? Ar sistemos versija atitinka numatytą funkcionalumą?
- Ar buvo sumažintos arba pašalintos rizikos? Ar buvo aptikta naujų rizikų?
- Ar visi suplanuoti darbai buvo įgyvendinti?
- Ar reikalingi papildomi projekto plano pakeitimai?
- Ar buvo atlikti išoriniai pakeitimai (pvz.: veiklos logikos pakeitimai, tesės aktai ir pan.) įtakojantys projekto vykdymą?

Projekto rizikų valdymas

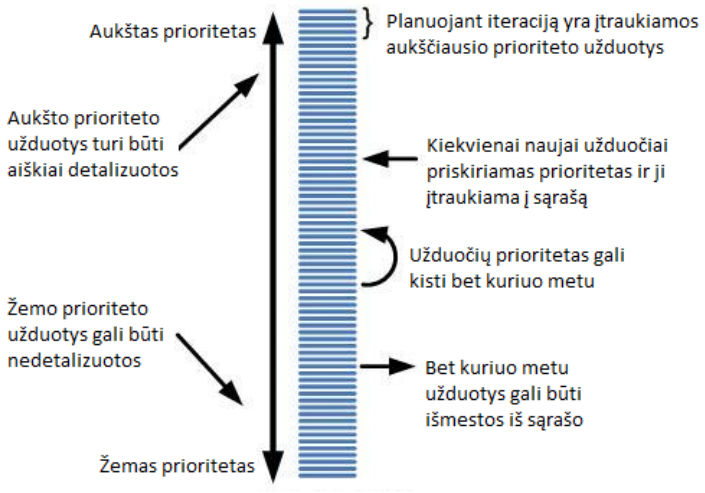
Projekto planavimo etape rekomenduojama identifikuoti rizikas, kurios gali įtakoti projekto eigą. Identifikuotas rizikas reikia suskirstyti pagal prioritetą. Didžiausio prioriteto rizikoms reikia skirti daugiau dėmesio ir sumažinti jų įtaką projekto eigai. Vykdant projektą reikia sudaryti tinkamą rizikų valdymo ir sprendimo metodiką, kuri leistų efektyviai sumažinti arba visai pašalinti rizikų įtaką. Vykdant projektą rekomenduojama reguliariai peržvelgti rizikų sąrašą ir, jei reikia, jį atnaujinti. Kiekvienos iteracijos planavimo metu reikia numatyti kokios galimos rizikos yra susijusios ir įtakoja iteracijos vykdymą. Taip pat, rekomenduojama rizikų sąrašą peržvelgti ir baigus iteraciją.

Projekto metu vykdomų užduočių valdymas

Planuojant projektą yra detalizuojamos stambios veiklos, kurios projekto vykdymo metu yra suskirstomos į mažesnes užduotis. Užduočių skirstymas pagal prioritetus gali būti vykdomas projekto ir iteracijos planavimo bei iteracijos vykdymo metu. Užduočių prioritetai gali būti skirstomi tokia tvarka:

- Analitikas pagal prioritetą surikiuoja užduotis reikalingas sistemos funkcijų bei panaudos atveju įgyvendinimui.
- Architektas ir programuotojų komanda identifikuoja architektūriškai svarbius panaudos atvejus ir patikslina jų prioritetą, tam kad būtų aišku kas pirmiausia turi būti padaryta.
- Testuotojai pagal prioritetą suskirsto užfiksuotas klaidas.
- Iteracijos vykdymo metu projektų vadovas atsižvelgdamas į vykdomų užduočių būsenas gali keisti užduoties prioritetą.

Suskirsčius visas užduotis pagal prioritetą, sąrašo viršuje pateikiamos aukščiausio prioriteto užduotys, apačioje – žemiausio prioriteto. Aukščiausio prioriteto užduotys turi būti aiškiai ir išsamiai aprašytos. Mažiausio prioriteto užduotys gali būti aprašomos minimaliai. Rekomenduojama, kad į iteraciją būtų įtraukiamos aukščiausio prioriteto užduotys.



25 paveikslas. Užduočių sąrašo sudarymas pagal prioritetą

Sudarytame užduočių sąraše esančios užduotys yra skirstomos į du tipus: suplanuotas ir nesuplanuotas užduotis. Priklausomai nuo šio užduoties tipo priklauso ir ją aprašančios informacijos pateikimas.

Nesuplanuotos užduotys – tai tokios užduotys, kurios yra numatytos atlikti projekto vykdymo metu bet dar nėra nuspręsta kurioje projekto iteracijos jos bus vykdomos. Detalizuojant šias užduotis rekomenduojama nurodyti šią užduoties informaciją:

- Užduoties pavadinimas;
- Užduoties aprašymas;
- Prioritetas;
- Užduoties būsena (nauja, priskirta, išspręsta ir pan.);
- Nuorodos į susijusią informacija (techninė specifikacija, projekto planas ir pan.).

Suplanuotos užduotys – tai tokios užduotys, kurios yra įtrauktos į iteraciją ir yra priskirtos konkrečiam projekto komandos nariui. Papildomai prie užduoties rekomenduojama nurodyti šią informaciją:

- Iteracija, kuriai priskirta užduotis;
- Už užduoties atlikimą atsakingas komandos narys;
- Numatomas laikas užduočiai atlikti;
- Darbo valandos panaudotos užduoties atlikimui.

Iteracijos/projekto retrospektyva

Įgyvendinus projektą ar iteraciją, rekomenduojama peržiūrėti kaip sekėsi siekti užsibrėžtų tikslų. Analizuojant įgyvendintą iteraciją/projektą gali būti priimami teisingi sprendimai sekančios iteracijos/projekto vykdymui. Retrospektyvos peržiūros suteikia galimybę geriau stebėti projekto komandos darbą, projekto progresą, lengviau priimti sprendimus reikalingus produktyvumo padidinimui bei galimų rizikų sumažinimui.

Retrospektyva rekomenduojama apžvelgti šiuose projekto stadijose:

- Kiekvienos iteracijos pabaigoje;
- Po naujos sistemos versijos parengimo;
- Po projekte netikėtai įvykusių reikšmingų įvykių ar incidentų.

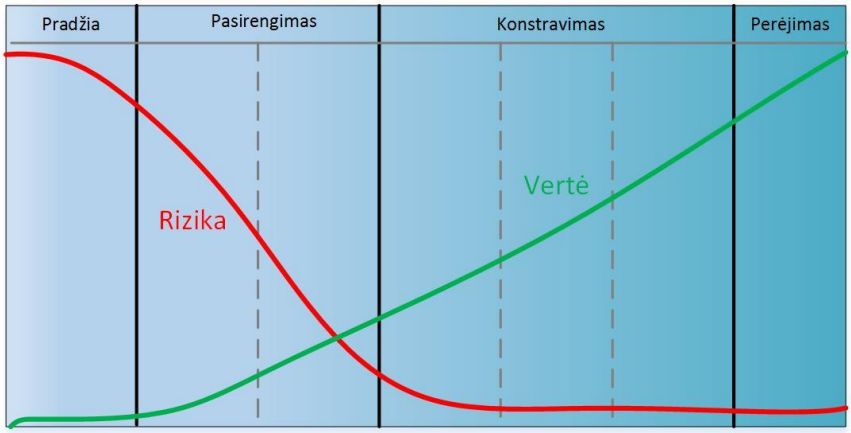
Sprendžiant iškilusias problemas komandoje yra užduodami trys pagrindiniai klausimai:

- Kokie darbai buvo sėkmingai atlikti vykdant iteraciją/projektą?
- Su kokiomis problemomis buvo susidurta vykdant iteraciją/projektą?
- Ką turėtume daryti kitaip ir kokius sprendimus turėtume priimti vykdant kitą iteraciją/projektą?

Remiantis šiais pagrindiniai klausimais yra priimami ir prioretizuojami sprendimai, kurie leistu pagerinti sekančio projekto ar iteracijos vykdymą.

2.5.1.2 *Rizikos – vertės balansu paremtas gyvavimo ciklas*

Siekiant geriau išbalansuoti projekto metu sukuriamą vertę ir kylančias rizikas, programinės įrangos kūrimo gyvavimo ciklą rekomenduojama skaidyti į fazes. Kiekviena fazė apima tik jai būdingus tikslus bei veiklas ir užduotis, reikalingas užsibrėžtiems tikslams pasiekti. Priklausomai nuo projekto specifikos kiekviena fazė gali būti skirstoma nuo vienos iki kelių iteracijų. Kiekvienos projekto fazės pabaigoje yra gaunamos atitinkamos gairės, kurios padeda nustatyti balansą tarp rizikų ir vertės, kadangi užsakovas yra orientuotas į du veiksnius: sumažinti riziką ir padidinti vertę.



26 paveikslas. Rizikos ir vertės santykis projekto vykdymo metu

26 paveiksle vaizduojama jog projekto pradinėse fazėse (inicijavimo ir pasirengimo) didelis dėmesys skiriamas rizikų mažinimui. Rizikos yra mažinamos su užsakovu sprendžiant esminius projekto klausimus, detalizuojant sistemai keliamus reikalavimus. Vėlesnėse fazėse (konstravimo ir perdavimo) yra kuriama vertė, t. y. realizuojama sistema, atliekamas testavimas ir sukurta sistema perduodama užsakovui.

Gairės projekto fazėms

Rekomenduojama, kad kiekvienos fazės pabaigoje užsakovui būtų pristatomi rezultatai, įrodantys jog fazės tikslai buvo pasiekti. Nors fazė gali būti skirstoma į kelias iteracijas, tačiau fazės pabaigą turėtų atspindėti bendras rezultatas. Jei gautas rezultatas tenkina užsibrėžtus tikslus, tuomet pereinam į kitą projekto fazę, o jeigu rezultatas netenkina - gali būti planuojama papildoma iteracija tikslams pasiekti.

Pradžios fazės pabaigoje turėtų būti gaunamos pirmosios gairės – projekto vykdymo tikslai. Šiame etape yra įvertinamas vykdomas projektas ir parengiamas jo planas.

Pasirengimo fazėje yra gaunamos kitos svarbios gairės – architektūriniai sistemos sprendimai. Šiame etape yra detalizuojami kuriamos sistemos reikalavimai bei apimtis, numatoma pagrindinių rizikų sprendimai. Pasirengimo fazėje yra parengiama sistemos architektūros specifikacija tam, kad tolimesnėse projekto fazėse neatsirastų netikslumų ir papildomų rizikų.

Pasirengimo fazė gali būti skirstoma į kelias iteracijas. Dažniausiai pirmoje iteracijoje yra rekomenduojama įgyvendinti ir ištestuoti maža dalį kritinių architektūros sprendimų, kad būtų galima nuspręsti kokius architektūrinius sprendimus naudoti. Tokiu būdu sumažinamos rizikos susijusios su sistemos architektūra. Vėlesnėse iteracijose yra detalizuojama tai kas buvo neįgyvendinta ankstesnėse iteracijose. Taip pat, realizuojami ir patikrinami likę architektūriniai sprendimai.

Konstravimo fazėje realizuojama visa sistema, atliekamas vidinis testavimas bei paruošiama reikalinga sistemos dokumentacija. Dažniausiai konstravimo fazė yra skirstoma į daugiau iteracijų nei kitos projekto fazės. Iteracijų skaičius priklauso nuo projekto tipo:

- Paprastas projektas – vykdoma viena iteracija, kurios metu parengiama ir ištestuojama bandomoji sistemos versija.
- Vidutinio dydžio projektas – fazė skirstoma į dvi iteracijas: viena iteracija skirta pradinės sistemos versijos parengimui, antra – sistemos paruošimui testavimui ir testavimo atlikimui.
- Didelis projektas – fazė skirstoma į tris arba daugiau iteracijų. Iteracijų skaičius priklauso nuo projekto specifikos, reikalavimų kiekio ir pan.

Perėjimo fazėje yra paruošiama galutinė sistemos versija. Šio etapo metu yra atliekamas priėmimo testavimas ir nusprendžiama ar sistemos kūrimo tikslai pasiekti. Perėjimo fazė gali būti skaidoma į vieną iteraciją, kuomet reikia atlikti tik klaidų taisymą, arba į kelias kai reikia atlikti svarbius sistemos pakeitimus, duomenų migravimą iš senos sistemos į naują ir panašiai. Užbaigus fazėje numatytas iteracijas ir įgyvendinus perėjimo fazės tikslus galima užbaigti projektą.

2.5.1.3 Projekto komandos valdymas

Sėkmingas projekto vykdymas dažniausiai priklauso nuo projekto komandos. Siekiant efektyviau prižiūrėti projekto komandos atliekamus darbus, rekomenduojama daryti periodinius projekto komandos susitikimus (angl. *scrum*), kurių metu yra aptariami atlikti bei vykdomi darbai. Rekomenduojama, kad susitikimai vyktų tuo pačiu metu, būti kiek įmanoma trumpesni (pvz. penkiolika minučių), o jame dalyvautų iki dešimt žmonių. Susitikimų metu, kiekvienas dalyvis turėtų atsakyti į šiuos pagrindinius klausimus:

- Kokias užduotis atlikau iki susitikimo? Atsakant į šį klausimą patikrinama ar komanda pagal planą vykdo priskirtas užduotis ir neatlieka į iteraciją neįtrauktų užduočių.
- Kokias užduotis ketinu daryti toliau? Atsakant į šį klausimą yra peržiūrimos ir paskirstomos iteracijoje likusios užduotys.
- Kas trukdo atlikti vieną ar kitą užduotį? Šiuo klausimu siekiama išsiaiškinti problemas, kurios įtakoja suplanuotų užduočių vykdymą. Iškilusių problemų sprendimui užduočių sąrašas gali būti papildytas naujomis užduotimis.

Tai yra pagrindiniai aptarimo klausimai, kurie leidžia apžvelgti vykdomos iteracijos būseną ir identifikuoti problemas.

Projekto eigoje būtina palaikyti subalansuotą komandos narių apkrovimą, t. y. kiek įmanoma vengti pikinių apkrovimo periodų, kuomet komandos nariai pervargsta, o jų našumas krenta. Žinoma, praktikoje dažniausiai sudėtinga išvengti „spurto“ periodų, tačiau jie turi būti trumpalaikiai. Siekiant subalansuoti komandos narių apkrovimą rekomenduojama:

- Kiek įmanoma labiau lygiagretinti veiklų ir užduočių vykdymą, pavyzdžiui, testavimą vykdyti lygiagrečiai su kūrimu, vietoj to, kad testavimą vykdyti tik projekto pabaigoje.
- Projektą suskirstyti į trumpas iteracijas, kas leidžia lengviau stebėti projekto progresą ir anksti aptikti klaidas. Taip pat suteikiama galimybė reguliariai pateikti atliktų darbų rezultatus, t. y. parengti ir pristatyti naują sistemos versiją.
- Naudoti nuolatinį integravimą (angl. *continuous integration*), kas leidžia lengvai sujungti ir ištestuoti sukurtą ar modifikuotą programinį kodą. Naudojant nuolatinį integravimą sumažinamas klaidų atsiradimas, kadangi kiekvienas komandos narys atskirai realizuoja tam tikras sistemos funkcijas ir vėliau prijungia prie visos sistemos.
- Vengti komandos narių viršvalandžių, kadangi tai dažniausiai blogo planavimo ar blogai paskirstytų resursų indikatorius.

Kita svarbi efektyvaus valdymo praktika - į komandą reikia žvelgti kaip į save organizuojantį darinį (angl. *self-organizing team*), t.y. pati komanda turi teisę spręsti kokias užduotis ir kaip jas vykdys. Svarbiausi aspektai, į kuriuos reikia atkreipti dėmesį taikant save organizuojančios komandos praktiką:

- Komanda pati pasirenka užduotis. Iteracijos pradžioje komanda atsižvelgdama į užduočių prioritetus ir laiką joms atlikti pasirenka užduotis.
- Komandos narys pats pasirenka užduotį. Dažniausiai asmenys turi daug patirties atliekant tam tikras užduotis ir gali efektyviai ir gerai atlikti pasirinktą užduotį.
- Komanda pasirenka kaip atlikti užduotis. Prieš pradedant užduočių atlikimą yra vykdomas komandos pasitarimas, kurio metu nusprendžiama kuri užduočių vykdymo strategija bus naudojama. Reikia nepamiršti, kad komanda atlikdama užduotis turi vadovautis numatytais organizacijos standartais, turima technine infrastruktūra ir pan.
- Reguliarūs komandos susitikimai. Siekiant užtikrinti jog pasirinktos užduotys yra vykdomos kaip suplanuota turi būti vykdomi reguliarūs užduočių aptarimai, kurių metu kiekvienas komandos narys pristato atliktus ir planuojamus darbus.

Vadovaujantis šia praktika, reikia nepamiršti ir projekto vadovo atsakomybių:

- Vadovavimas. Projekto vadovas prižiūri projekto eigą ir koordinuoja projekto komandos darbus, kad ji nenukryptu nuo parengto grafiko.
- Konfliktų sprendimas. Jei komandoje kyla nesutarimų dėl vykdomų užduočių ir pati komanda negali priimti sprendimo, projektų vadovas turi būti pasiruošęs spręsti iškilusius konfliktus.
- Užtikrinti komandos narių nuolatinį tobulėjimą. Nuolat panašias užduotis sprendžiantys komandos nariai turi būti motyvuojami įgyti naujų įgūdžių, t. y. projektų vadovas periodiškai turėtų priskirti naujo pobūdžio užduočių.
- Užtikrinti, kad komandos nariai neperžengia savo įgaliojimų. Nors save organizuojančios komandos nariai turi teisę patys pasirinkti kokias užduotis ir kaip vykdyti, tai nereiškią, kad jie turi teisę nesilaikyti nusistovėjusių standartų ir veiklos reikalavimų.
- Komunikuoti projekto būseną. Projekto vadovas periodiškai turi apžvelgti ir pristatyti projekto progresą užsakovui ar kitiems suinteresuotiems asmenims, kurie nėra aktyviai įsitraukę į projektą.

2.5.2 Techninės praktikos

2.5.2.1 Bendra vizija (angl. shared vision)

Bendros sprendžiamų problemų ir kuriamo produkto savybių vizijos sukūrimas ir palaikymas sumažina rizikas, susijusias su suinteresuotų asmenų ir rinkos palankumu produktui. Deja, nesutarimai ir prasta komunikacija apie produkto strategiją dažnai veda prie to, kad programuotojų komanda sukuria sistemą, kuri neatitinka suinteresuotų asmenų poreikių ir vizijos.

Yra itin svarbu, kad programuotojų komanda ir suinteresuoti asmenys turėtų tuos pačius lūkesčius. Produkto vizija identifikuoja suinteresuotus asmenis, suteikia bendrą identifikuotų problemų supratimą, užfiksuoja suinteresuotų asmenų

poreikius bei projekto apribojimus ir sukuria pagrindus reikalavimams, kurie būna detalizuoti vėliau. Bendra vizija tarnauja kaip pradinė projekto komunikacijos informacija ir sudaro strategiją, kurią turi atitikti visi ateities sprendimai.

Bendros vizijos sukūrimas ir palaikymas taip pat yra esminis dalykas programavimo apimties planavime ir sekime.

Bendros vizijos parengimas apima supratimą ir sutarimą dėl atsakymų į fundamentalius projekto klausimus: „kas?“ ir „kodėl?“. Tai užtikrina, kad suinteresuoti asmenys ir programuotojų komanda turi bendrą sprendžiamų problemų supratimą ir supranta poreikius.

Gairė: reikalavimų nustatymo technikos

Geri reikalavimai atsiranda iš gerų šaltinių. Keletas reikalavimų šaltinių pavyzdžių:

- Klientai;
- Naudotojai;
- Administratoriai;
- Partneriai;
- Srities ekspertai;
- Industrijos analitikai;
- Informacija apie konkurentus.

Identifikavus šiuos šaltinius, yra daugybė technikų, kurios gali būti panaudotos reikalavimų nustatymui:

- Diskusijų (angl. *brainstorming*) sesijos organizavimas;
- Naudotojų interviu;
- Klausimynų parengimas ir pateikimas;
- Darbas su tikslinėmis grupėmis;
- Analogiškų sistemų analizė;
- Pasiūlymų ir problemų ataskaitų egzaminavimas;
- Pokalbiai su palaikymo komandomis;
- Naudotojų atliktų patobulinimų studijavimas;
- Nenumatytų panaudos galimybių apžvalga;
- Seminarų organizavimas;
- Prototipų demonstravimas suinteresuotiems asmenims.

Kurios iš pateiktų technikų turėtų būti pasirinktos, priklauso nuo kelių faktorių:

- Suinteresuotų asmenų prieinamumas ir vieta;
- Programuotojų komandos žinios apie probleminę sritį;
- Klientų ir naudotojų žinios apie probleminę sritį;
- Klientų ir naudotojų žinios apie konstravimo procesą ir metodus.

Jei suinteresuoti asmenys nėra lengvai pasiekiami, tai tokios technikos kaip diskusijos, interviu ar seminarai, kurie reikalauja realaus žmonių dalyvavimo, gali būti sunkiai įgyvendinamos ar išvis neįmanomos. Jei jie yra lengvai pasiekiami, tai situacija yra daug geresnė ir praktiškai visos technikos gali būti pritaikytos. Šiuo atveju, suinteresuotų asmenų ir programuotojų komandos aktualios srities ir konstravimo patirtis yra esminis faktorius renkantis tinkamas technikas.

Gairė: efektyvi reikalavimų peržiūra

Reikalavimų peržiūra yra reikalinga tam, kad problemos su reikalavimais būtų nustatytos prieš pradedant eikvoti laiką ir kitus resursus neteisingų dalykų kūrimui. Negalima sakyti, kad reikalavimų rinkiniai privalo būti išbaigti iki konstravimo pradžios, tačiau turi būti užtikrinta, kad anksčiausiai įgyvendinami reikalavimai bus peržiūrėti pirmųjų iteracijų metu, siekiant užsitikrinti visų suinteresuotų asmenų pritarimą prieš investuojant daug resursų į konstravimą.

Reikalavimų peržiūra gali būti kelių tipų:

- Neformali. Neformali reikalavimų peržiūra – tai paprastas jų parodymas kolegoms ar prototipo demonstracija. Tokia peržiūra puikiai tinka reikalavimų struktūros įvertinimui ir akivaizdžių klaidų pašalinimui, tačiau gali praleisti detales.
- Formali. Formali reikalavimų peržiūra turėtų būti atliekama oficialių susitikimų metu. Tokiam susitikimui turi būti pasirengta iš anksto. Susitikimo metu turėtų būti priimami konkretūs sprendimai dėl reikalavimų, o po jo atliekami reikalavimų atnaujinimai.
- Dviejų lygių. Šiuo atveju pirmiausia atliekama neformali reikalavimų peržiūra, o po to ir formali.

2.5.2.2 Panaudos atvejais paremtas programavimas

Panaudos atvejais paremto programavimo praktika apibūdina, kaip nustatyti reikalavimus sistemai, kombinuojant panaudos atvejus ir bendruosius sistemos reikalavimus, ir tuomet vykdyti programavimą ir testavimą remiantis šiais panaudos atvejais. Ši praktika padeda sukurti konstravimo scenarijus, kurie aiškiai nusako sistemos veikimą. Panaudos atvejai kategorizuoja naudingas, išbaigtas, ištestuojamas ir apibendrintas veiklas, į kurias sistema yra įtraukta.

Pagrindiniai aspektai

Panaudos atvejai apibūdina sąveikas tarp vieno ar daugiau Veikėjų ir sistemos. Sistemos funkcionalumai yra apibrėžiami skirtingų panaudos atvejų, kurių kiekvienas atstovauja specifinius konkretaus aktoriaus tikslus. Panaudos atvejų rinkinys nustato visus galimus būdus, kuriais gali būti naudojama sistema.

Norint pilnai suprasti sistemos paskirtį, reikia žinoti, kam ji skirta, t. y. atsakyti į klausimą, kas naudosis sistema. Bendru atveju atsakymas yra: Veikėjai (angl. *Actors*). Veikėjas yra rolė, kurią asmuo ar išorinė sistema atlieka, kai sąveikauja su sistema. Kiekvienas Veikėjas turi unikalų ir svarbų požiūrį į sistemą. Šis požiūris yra bendras kiekvienam atskiram Veikėjo atvejui (angl. *instance of an Actor*). Veikėjai padeda identifikuoti išorines sąsajas ir nustatyti sistemos apimtis. Kiekvienas veikėjas dažniausiai yra susietas su panaudos atvejais, kurie apibūdina, ko šis veikėjas tikisi iš sistemos.

Panaudos atvejų diagramos yra naudojamos grafiškai atvaizduoti modelio poaibius tam, kad supaprastinti komunikaciją. Dažniausiai su konkrečiu modeliu būna susietos kelios panaudos atvejų diagramos, kurių kiekviena vaizduoja modelio poaibį, svarbų dėl tam tikrų priežasčių. Tas pats modelio elementas gali būti naudojamas keliose diagramose, tačiau kiekvienas atvejis turi būti nuoseklus.

Gairė: panaudos atvejų ir scenarijų detalizavimas

Kadangi panaudos atvejai modeliuoja reikalavimus, jie iš prigimties yra labai dinamiški. Iteracinis apibendrintasis metodas, kurio metu panaudos atvejai yra nuolatos tobulinami prieš juos detalizuojant, yra efektyvus būdas rašyti panaudos atvejus. Apibendrintasis metodas apima du aspektus: panaudos atvejų rinkinių rašymas ir atskirų panaudos atvejų rašymas.

Panaudos atvejų rinkinių rašymas: panaudos atvejai egzistuoja rinkiniais, o santykiai tarp įvairių panaudos atvejų ir Veikėjų yra labai svarbūs. Kuo daugiau yra suprantama apie Veikėjus, tuo daugiau yra sužinoma ir apie sistemos ribas ir transakcijas. Taip pat ir atvirkščiai. Todėl yra labiau efektyvu rengti kelis panaudos atvejus vienu metu nei vieną paskui kitą. Šiuo būdu galima suprasti įvairių panaudos atvejų poveikį vienas kitam jų rengimo metu, kas yra geriau, nei tai padaryti po visko, kas gali pareikalauti perrašymo ar visiško pradinio darbo atsisakymo.

Atskirų panaudos atvejų rašymas: yra logiška panašiai rašyti ir kiekvieną atskirą panaudos atvejį. Pradedant su kertiniu scenarijumi, galima identifikuoti įvairias alternatyvas ir klaidingas procesų sekas, tuomet jas įvertinti, pertvarkyti ar eliminuoti ir galiausiai detalizuoti išlikusius scenarijus.

Gairė: Veikėjų ir panaudos atvejų identifikavimas ir apibrėžimas

Identifikuojant Veikėjus pirmiausia turėtų būti nustatyti išoriniai objektai, su kuriais konstruojama sistema privalo sąveikauti. Kandidatai apima naudotojų grupes, kurioms būtų reikalinga sistemos pagalba atlikti savo užduotis ir kurios vykdytų sistemos pirmines ir antrines funkcijas, taip pat išorinę techninę įrangą, programinę įrangą ir kitas sistemas. Kiekvienas kandidatas turi būti apibrėžiamas suteikiant jam vardą ir parengiant aprašymą, kuris turėtų apimti veikėjų atsakomybės sritį ir tikslus, kuriuos veikėjas sieks atlikti naudodamasis sistema. Kandidatai, kurie neturi jokių tikslų, turi būti eliminuojami.

Identifikuojant veikėjus verta pasinaudoti tokiais klausimais:

- Kas teiks, naudos ir šalins sistemos informaciją?
- Kas naudosis sistema?
- Kam yra įdomios sistemos teikiamos savybės ir paslaugos?
- Kas atliks sistemos palaikymo funkcijas?
- Kokie yra sistemos išoriniai resursai?
- Kokioms kitoms sistemoms reikės sąveikauti su konstruojama sistema?

Geriausias būdas surasti panaudos atvejus yra apsvarstymas, ko kiekvienam veikėjui reikia iš sistemos. Kiekvienam veikėjui, ir žmogui ir ne tik, turėtų būti užduodami tokie klausimai:

- Kokie yra veikėjo tikslai, kuriuos jis bandys pasiekti naudodamasis sistema?
- Kokia yra pagrindinė užduotis, kurios norėtų veikėjas, kad sistema atliktų?
- Ar veikėjas kurs, saugos, keis, naikins ar peržiūrės sistemos informaciją?
- Ar veikėjui reikės informuoti sistemą apie staigius išorinius pokyčius?
- Ar veikėjas turi būti informuotas apie tam tikrus sistemos įvykius, tokius kaip tikslo resursų nepasiekiamumas?
- Ar veikėjas atliks sistemos paleidimą ir išjungimą?

Kiekvienam panaudos atvejui turi būti suteikiamas unikalus vardas ir aprašymas, kuris aiškiai nustato tikslus. Jei kandidatas į panaudos atvejus neturi tikslų, turėtų būti užduodamas klausimas, kodėl jis egzistuoja, ir tuomet arba identifikuoti tikslus, arba eliminuoti panaudos atvejį.

Gairė: bendrųjų sistemos reikalavimų specifikavimas

Atėjus laikui nustatyti bendruosius sistemos reikalavimus, savybes ar ribojimus, yra baigtinis rinkinys reikalavimų, į kuriuos turi būti atsižvelgta. Daugelis jų yra nežinomi suinteresuotiems asmenims, todėl gali būti sunku atsakyti į klausimus,

susijusius su prieinamumu, našumu, plečiamumu ar aplinka. Renkantis bendruosius reikalavimus pagal toliau pateikiamą sąrašą, svarbu užtikrinti, kad suinteresuoti asmenys suprastų kiekvieno iš jų vertė ir rinktųsi apgalvotai, o ne tiesiog visus iš eilės.

Funkcinių reikalavimų sritys:

- Auditas (žurnalizavimas);
- Autentifikacija;
- Spausdinimas;
- Ataskaitų rengimas;
- Periodinių įvykių planavimas;
- Saugumas.

Naudojamumo reikalavimų sritys:

- Paprastumas išmokti;
- Užduočių efektyvumas;
- Paprastumas prisiminti;
- Suprantamumas;
- Asmeninis pasitenkinimas.

Našumo reikalavimų sritys:

- Atsako laikas;
- Talpumas;
- Paleidimo ir išjungimo sąlygos.

Reikalavimų palaikymui sritys:

- Pritaikomumas;
- Suderinamumas;
- Konfigūruojamumas;
- Instaliavimas;
- Reikalingos pagalbos lygis;
- Palaikomumas;
- Plečiamumas;
- Ištestuojamumas.

Reikalavimų apribojimai:

- Dizaino ribojimai;
- Trečiųjų šalių ribojimai;
- Programavimo kalbos;
- Palaikymo platforma (aplinka);
- Resursų limitai;
- Fiziniai ribojimai.

Reikalavimų sąsajoms sritys:

- Naudotojo sąsaja:
  - Naudotojo sąsajos išvaizda;
  - Reikalavimai navigacijai;
  - Nuoseklumas;
  - Reikalavimai suasmeninimui.
- Sąsajos su išorinėmis sistemomis ar prietaisais:
  - Programinės įrangos sąsajos;
  - Techninės įrangos sąsajos;
  - Komunikacinės sąsajos.

2.5.2.3 *Evoliucionuojanti architektūra (angl. evolutionary architecture)*

Evoliucionuojančios architektūros praktika, kaip inkrementiškai kurti ir tobulinti sistemos architektūrą tuo metu, kai nustatinėjamos ir apibrėžiamos architektūros problemos programinės įrangos konstravimo metu. Tai sumažina techninių rizikų tikimybę, nereikalaujant didžiulių išankstinių pastangų architektūros konstravimui. Ši praktika:

- Padidina kokybę ir produktyvumą, sumažindama poreikį atlikti daug laiko reikalaujančius vėlai nustatytų klaidų taisymus. Tai įmanoma dėl to, kad architektūra yra išbandoma anksčiau ir pagrindinės problemos yra nustatomos prieš atliekant didžiąją dalį programavimo.
- Sumažina išleidimo trukmę, nes akcentuojamas pakartotinis panaudojimas. Tai pagerina sistemos nuoseklumą ir palaikomumą, nes atsižvelgiama į pamokas, išmoktas ankstesniuose projektuose ir pradinėse architektūros konstravimo stadijose.
- Ankstesnis didžiausių technologinių rizikų sričių nustatymas pagerina nuspėjamumą. Tai pagerina komandos reakciją į pokyčius, nes sutrumpėja architektūrinis ciklas ir sumažėja laiko, iššvaistomo architektūros perdarymui, kai atsiranda pakeitimai.

Evoliucionuojančios architektūros praktikoje yra analizuojami esminiai techniniai dalykai, kurie turi įtakos sprendimui, ir dokumentuojami architektūros sprendimo aspektai siekiant užtikrinti, kad jie yra įgyvendinti. Pagrindiniai evoliucionuojančios architektūros praktikos principai:

- Architektūrinės užduotys turi būti visada atliktos laiku;
- Pagrindiniai architektūriniai sprendimai ir neišspręstos problemos turi būti dokumentuojami;
- Pagrindinės architektūros konstravimo galimybės turi būti įdiegiamos ir ištestuojamos kaip būdas identifikuoti architektūros problemas.

Kaip projekte pritaikyti evoliucionuojančios architektūros praktiką?

Pirmiausia turi būti užtikrinta, kad projekto komanda ir užsakovas supranta, kas yra sistemos architektūra ir kokia jos užfiksavimo vertė.

Architektas neturėtų dirbti atskirai nuo programuotojų ir pateikti jiems architektūros specifikaciją be jokio bendradarbiavimo. Tai reikalauja per daug dokumentavimo ir komandos nariams apsunkina architektūros principų supratimą. Architektūros konstravimo veikla reikalauja bendradarbiavimo, kad būtų sėkminga. Svarbūs architektūriniai sprendimai turi būti dokumentuojami. Svarbių sprendimų neužfiksavimas didina klaidų ir problemų riziką vėlesniuose projekto etapuose, nes būsimi komandos nariai neturės galimybės būti esamų sprendimų priėmimo dalimi.

Pagrindiniai aspektai

Architektūriniai principai tai yra paplitę sprendimai skirti standartinėms problemoms spręsti programavimo metu, kad sumažėtų kompleksiškumas. Jie reiškia pagrindinius techninius aspektus, kurie yra standartiniai visam sprendimui. Architektūriniai principai žymiai palengvina sistemos architektūros vystymąsi. Jie leidžia komandai palaikyti darnią architektūrą, nors detalių įgyvendinimas yra atidedamas tol, kol jos nėra tikrai būtinos.

Architektūriniai principai yra naudojami išpildyti architektūrai reikšmingus reikalavimus. Įprastai tai yra ne funkciniai reikalavimai, tokie kaip našumo ar saugumo. Jei yra pilnai aprašyti, architektūriniai mechanizmai pateikia sistemos struktūros ir veikimo principų modelius. Jie formuoja bendruosius programinės įrangos pagrindus, kurie būna pritaikomi PĮ konstravimo metu. Jie taip pat formuoja pagrindus PĮ veikimo metodų standartizavimui, todėl yra svarbūs visoje sistemos architektūroje.

Architektūra gali būti pateikta įvairiais pjūviais, kurie gali būti kombinuojami siekiant sukurti holistinį sistemos vaizdą. Kiekvienas architektūrinis pjūvis išskiria specifinius architektūros aspektus, kurie gali būti skirti įvairiems sistema suinteresuotiems asmenims: naudotojams, projektuotojams, projekto vadovams, sistemos inžinieriams ir t. t.

Iš esmės, architektūriniai pjūviai vaizduoja abstrakcijas ir supaprastinimus, kuriuose išryškėja svarbios charakteristikos. Šios charakteristikos yra svarbios, kai yra kalbama apie:

- Sistemos vystymą ir perėjimą į kitą konstravimo etapą;
- Architektūros ar jos dalių pakartotinį panaudojimą;
- Papildomų savybių, tokių kaip našumas, prieinamumas, saugumas, vertinimą;
- Konstravimo užduočių priskyrimą komandai ar rangovui;
- Integravimą į didesnę sistemą.

Programinės įrangos architektūra yra sąvoka, kuri yra lengvai suprantama ir mažai patirties turintiems inžinieriams, tačiau ją sunku tiksliai apibrėžti. Visų pirma, yra sudėtinga atskirti architektūrą nuo projektavimo – architektūra yra vienas iš projektavimo aspektų, kuris daugiausiai dėmesio skiria kelioms specifinėms ypatybėms.

Architektūra gali būti naudojama įvairiais tikslais:

- Apibrėžti esminę sistemos struktūrą ir sprendimus, kuriais ji grindžiama.
- Identifikuoti ir sumažinti sistemos rizikas.
- Apibrėžti sistemos apimtį ir parengti gaires programuotojams.
- Apibrėžti bendrą sistemos vaizdą, kad ir kas atliktų sistemos architektūros palaikymą.
- Nustatyti projekto struktūrą ir komandos veiklos organizavimą.

Gairė: kompleksiškumo mažinimas abstrakcijomis

Programinės įrangos kūrimas yra veikla charakterizuojama kompleksiškumo. Tai gali būti įvairių formų, pavyzdžiui, sudėtingų reikalavimų, technologijų ar komandos pritaikymas. Abstraktumo lygio didinimas padeda suvaldyti šį kompleksiškumą ir turėti pamatuojamą progresą, nepaisant iškylančių sunkumų.

Kelios strategijos, kaip didinant abstraktumą sumažinti kompleksiškumą:

- Jau sukonstruotų ir daugelyje projektų išbandytų modelių naudojimas;
- Architektūros su įvairiais komponentais ir servisais projektavimas;

- Aktyvus komponentų pakartotinis panaudojimas;
- Esminių aspektų modeliavimas.

Gairė: architektūros modeliavimas

Kai konstruojama sistemos architektūra, vizualių architektūros modelių sukūrimas gali būti naudingas, kadangi tokie modeliai suteikia puikų aukšto lygio sistemos, jos komponentų ir jų tarpusavio sąsajų vaizdą. Ypač naudinga yra parengti įvairių tipų UML diagramas.

Gairė: programinės įrangos pakartotinis panaudojimas

Maksimalus pakartotinis panaudojimas visada buvo svarbus PĮ programuotojų tikslas. Yra daug geriau dar kartą kažką panaudoti nei padidinti kaštus kuriant kažką naujo. Programavimo kalbos, ypatingai objektinės, buvo sugalvotos siekiant padaryti pakartotinį panaudojimą lengvesniu. Bet vienos kalbos neužtenka, kad perpanaudojimas būtų efektyvus kainos atžvilgiu. Didžioji dalis pakartotinio panaudojimo būna pas patyrusius programuotojus ir architektus, kurie yra gabūs identifikuoti ir pasinaudoti pakartotinio panaudojimo galimybėmis.

Keletas resursų, kurie gali būti pakartotinai panaudojami:

- Architektūriniai karkasai;
- Architektūriniai principai;
- Architektūriniai sprendimai;
- Apribojimai;
- Aplikacijos;
- Komponentai.

2.5.2.4 Evoliucionuojantis sistemos projektas (angl. evolutionary design)

Sistemos projektavimas – tai vidinės struktūros ir veiklos modelio sukūrimas. Geras sistemos projektas padidina kokybę ir padaro sistemą lengviau palaikomą ir plečiamą, o prastas projektas gali nepalyginamai padidinti programinės įrangos išleidimo ir palaikymo kaštus.

Sistemos projektas yra programinio kodo abstrakcija, kuri vaizduoja sistemą iš perspektyvos, kuri palengvina struktūros ir funkcijų identifikavimą. Sistemos projektą turėtų sudaryti vizualiniai modeliai, paprasti eskizai, tekstiniai aprašymai ir t.t. Pagrindinė jo savybė yra ta, kad jis apibūdina, kaip skirtinga elementai sąveikauja tarpusavyje tam, kad išpildytų reikalavimus.

Evoliucionuojančio projektavimo praktika sumažina produkto išleidimo į rinką laiką palaipsniui atliekant projektavimą PĮ konstravimo metu. Tai padidina produktyvumą, inovatyvumą, modelių kokybę bei perpanaudojimo galimybes.

Kiekvienos projektavimo fazės metu sprendimas yra papildomas, patikslinamas ir performuojamas. Evoliucionuojančio projektavimo praktiką galima apibendrinti tokiais žingsniais:

- Naujų reikalavimų detalių supratimas;
- Projektavimo elementų identifikavimas;
- Elementų tarpusavio sąsajų nustatymas;

- Projektavimo sprendimų patikslinimas;
- Savybių projektavimas;
- Projektavimo komunikacija;
- Architektūros supratimas;
- Projektavimo vystymas.

Kaip pati sąvoka teigia, evoliucionuojantis projektavimas apima daugkartinį grįžimą prie esamo sistemos projekto ir tikslinimo, tobulinimo ar keitimo. Tai gali būti atliekama konstravimo ciklo pradžioje, jo metu, po jo arba kombinuotai.

Gairė: sistemos projekto ir programinio kodo susiejimas

Sistemos projektas turi pateikti pakankamai informacijos apie sistema, kad ši būtų sukurta vienareikšmiškai. Kas sudaro „pakankamai“, priklauso nuo konkretaus projekto.

Kai kuriais atvejais sistemos projektas primena tik eskizą, detalizuotą tik tiek, kad programuotojai galėtų pradėti darbą. Detalumo lygis dažnai priklauso nuo projektuotojo įgūdžių, projekto kompleksiškumo ir rizikos, kad projektas gali būti neteisingai įgyvendintas. Kitais atvejais sistemos projektas gali būti taip detalizuotas, kad jį galima automatiškai transformuoti į programinį kodą. Tai dažniausiai apima ir standartinių UML diagramų papildymus, kurie apibūdina specifinę programavimo kalbos ar aplinkos semantiką.

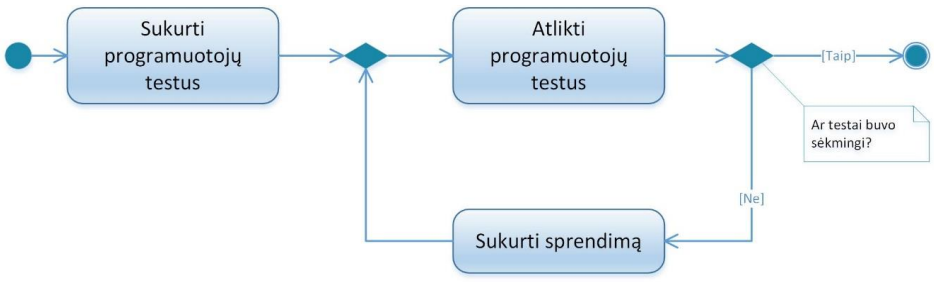
Sistemos projektas taip pat gali būti hierarchiškas, pavyzdžiui:

- Aukšto lygio projekto modelis, kuris apibendrintai aprašo visą sistemą;
- Subsistemų specifikacijos modelis, kuris tiksliai nustato reikalingas sąsajas ir pagrindinių subsistemų veikimą;
- Detalaus projekto modelis vidiniams subsistemų objektams aprašyti.

2.5.2.5 Testais paremtas programavimas

Testais paremtas programavimas yra praktika, kurios metu pirmiausia yra nustatomi testavimo atvejai, o tuomet lygiagrečiai yra konstruojami programuotojų testai ir programos kodas labai detaliame lygyje.

Šios praktikos taikymo metu, programuotojas parengia smulkius testus, tikrinančius nedidelius pakeitimus, paleidžia juos ir įsitikina, kad jie žlunga, ir tuomet parašo pakankamai programos kodo, kad šis testas būtų sėkmingas. Šis ciklas yra labai trumpas ir retai užtrunka ilgiau nei 10 min. Kai testas yra užbaigtas, programuotojas vis pereina prie kito, kol užbaigia visus.



27 paveikslas. Testais paremto programavimo praktikos ciklas

Testais paremto programavimo praktika pakeičia programuotojų mąstymą. Testai nėra rengiami kaip po visko papildomai sugalvoti dalykai. Vietoj to, programuotojų testai yra naudojami kaip kasdienis metodas konstruoti programinę įrangą.

2.5.2.6 *Nuolatinis testavimas (angl. concurent testing)*

Projekto metu taikant nuolatinio testavimo praktiką, testavimo veiklos yra vykdomos kiekvienos iteracijos metu lygiagrečiai kitoms programinės įrangos kūrimo veikloms. Ši praktika reikalauja nuolatinio ir glaudaus bendradarbiavimo tarp testuotojų ir programuotojų (PĮ kūrėjų), todėl norint ją taikyti svarbiausia yra atsižvelgti į testavimo apimtis (kas testuojama – posistemės, komponentai, funkcijos ir t. t.) bei projekto komandą (joje turi būti bent vienas testuotojas).

Kokybiniai testavimo reikalavimai

Programinės įrangos testavimas – tai ne tik funkcijų, sąsajų ar atsako laiko charakteristikų įvertinimas. Testai taip pat turi būti orientuoti ir į tokias charakteristikas bei savybes kaip:

- Integralumas (atsparumas klaidoms);
- Instaliavimo ir veikimo skirtinguose platformose galimybė;
- Gebėjimas vienu metu apdoroti daug užklausų.

Tam pasiekti turi būti sukurta ir įgyvendinta daug įvairių testų ir kiekvienas iš jų turi specifinius tikslus bei atlikimo technikas, o kiekviena technika orientuojasi į vieną ar daugiau testuojamo objekto charakteristikų ar savybių. Atsižvelgiant į tai, galima išskirti tokias testų grupes:

- Funkcionalumas (angl. *Functionality*);
- Naudojamumas (angl. *Usability*);
- Patikimumas (angl. *Reliability*);
- Našumas (angl. *Performance*);
- Palaikymas (angl. *Supportability*);

Funkcionalumo testų tipai:

- Funkcijų testai yra orientuoti į testuojamo objekto funkcijas sudarančius servisus, metodus ar panaudos atvejus.
- Saugumo testai skirti užtikrinti, kad testuojamo objekto duomenys ir funkcijos yra prieinamos tik tiems naudotojams ar sistemoms, kuriems jie yra skirti.
- Apimties testai (angl. Volume test) yra orientuoti įvertinti kaip testuojama objektas gali susitvarkyti su dideliais kiekiais duomenų, tiek įvesties ar išvesties metu, tiek ir esančiais duomenų bazėse. Apimties testai apima tokias testavimo strategijas kaip užklausų, kurios pateiktų visą duomenų bazės turinį, sukūrimas ar duomenų įvedimas, kai kiekvienas užpildytas laukas turi maksimalų įmanomą duomenų kiekį.

Naudojamumo testų tipai:

- Naudojamumo testai yra orientuoti į:
  - Žmogiškuosius faktorius;
  - Estetiką;
  - Naudotojo sąsajos nuoseklumą ir logiškumą;
  - Internetinius pagalbos puslapius ir naudotojų vadovus;
  - Mokymų medžiagą.
- Integralumo testai yra skirti įvertinti testuojamo objekto atsparumą klaidoms ir techninį atitikimą programavimo kalbos, sintaksės ir resursų naudojimui.

- Struktūros testai orientuojasi į testuojamo objekto dizaino ir struktūros vientisumą. Dažniausiai šie testai yra atliekami internetinėms aplikacijoms siekiant užtikrinti, kad visos reikalingos sąsajos yra sujungtos, o turinys atvaizduojamas tinkamai.

Patikimumo testų tipai:

- Stresinių situacijų testai (angl. *Stress test*) skirti įvertinti, kaip sistema reaguoja į nestandartines situacijas, tokias kaip ekstremali sistemos apkrova, nepakankama atmintis, nepasiekiami servisai ir techninė įranga ar riboti kiti resursai. Šie testai yra dažnai atliekami siekiant geriau suprasti, kaip ir kokiomis aplinkybėmis sistema gali užstrigti, kad būtų galima parengti nenumatytų atvejų bei veiklos tęstinumo planus ir suplanuoti jų biudžetus.
- Palyginamieji testai (angl. *Benchmark test*) yra skirti palyginti naujos nežinomos sistemos našumą su žinoma ir išbandyta sistema.
- Konkuravimo testai (angl. *Contention test*) yra orientuoti į testuojamos sistemos gebėjimo apdoroti konkurentines sesijas (kai skirtingi naudotojai nori pasinaudoti tais pačiais sistemos resursais) patikrinimą.

Našumo testų tipai:

- Apkrovos testai (angl. *Load test*) naudojami patikrinti ir įvertinti testuojamos sistemos apkrovos ribojimus ir limitus, ties kuriais sistema veikia tolygiai ir be trikdžių. Kartais šių testų metu sistemos apkrova yra pasirenkama tolygi, o keičiama sistemos konfigūracija. Įprastai matuojami būna apkrovos praeinamumas ir einamųjų transakcijų atsako laikas.
- Našumo profilio testai skirti nustatyti sistemos procesų vykdymo, duomenų prieigos, funkcijų, sisteminių užklausų laikines charakteristikas, kurios padėtų identifikuoti neefektyvius procesus.
- Konfigūracijų testai yra orientuoti į funkcijų, skirtų testuojamo objekto sukonfigūravimui skirtingoje techninėje ir programinėje įrangoje, veikimo užtikrinimą.

Instaliacijų testai – palaikymo testų tipas, skirtas įvertinti testuojamo objekto instaliavimo skirtingoje techninėje ir programinėje įrangoje galimybes ir nustatyti instaliavimo sąlygas.

Gairė: automatinių testų naudojimas

Automatiniai testai – svarbi testų rinkinių dalis. Dažniausiai jie rengiami siekiant palengvinti testavimo veiklas, tačiau visgi testų rinkinių palaikymas gali tapti sudėtingesnis.

Testų automatizavimas be planavimo didina tikimybę, kad testavimas gali būti neefektyvus ir nekokybiškas, todėl planavimui visada turi būti skirta dėmesio. Esminė kiekvieno testavimo plano vieta yra testų automatizavimo strategijos nustatymas. Parengtas planas turi būti pristatomas ir programuotojų komandai, nes dažniausiai ji nieko nežino apie tai, kaip tvarkomi testavimo resursai. Taip pat rekomenduotina programuotojų komandai paaiškinti ir tai, kad testų rinkinių palaikymas gali būti ir viso sistemos kūrimo proceso esminė dalis. Šios informacijos kaupimui turėtų būti naudojami įvairūs testavimo įrankiai – taip pat kaip ir kitiems testavimo resursams tvarkyti.

Siekiant palengvinti automatinių testų rinkinių palaikymą, testavimo resursai yra saugomi programuotojų komandai prieinamose saugyklose. Daugelis testų automatizavimo aplinkų kartu pateikia ir valdymo įrankius, skirtus paprastai organizuoti ir tvarkyti visus testavimo resursus bendroje saugykloje. Papildomai tam tikra prieigos teisių kontrolė yra vykdoma ir testų automatizavimo įrankių. Tai palengvina palaikymo rūpesčius ir užtikrina testų rinkinių integralumą.

Kiekvienos programinės įrangos palaikymas turi būti užtikrintas, tad testavimui naudojama PĮ nėra išimtis. Testavimo scenarijai ir kiti testavimo resursai, nesvarbu rašytiniai ar programiniai, taip pat turi būti palaikomi. Kaip ir bet kuri programinė įranga turi skirtingas palaikymo rūšis (pvz. korekcinis, prevencinis ar adaptyvusis), taip pat turi ir automatinių testų rinkinių resursai. Nustatant testų rinkinių gyvavimo ciklą, turi būti identifikuota, kaip planuojama užtikrinti korekcinį palaikymą (pvz. sintaksės klaidos scenarijuose), prevencinį palaikymą (pvz. kur įmanoma parengti apibendrintus testavimo scenarijus) ir adaptyvųjį palaikymą (pvz. kaip naudojant testavimo įrankius priskirti tuos pačius resursus keliems skirtingiems testų rinkiniams).

Testavimo resursai turi tobulėti taip pat kaip testuojamos aplikacijos. Keičiantis sistemos reikalavimams, keičiasi ir pati sistema, todėl testų rinkiniai turi būti tikrinami nuolatos. Jei įmanoma, tikrinimas turi būti atliekamas po kiekvienos naujos sistemos versijos išleidimo, o geriau ir dar dažniau. Testų rinkinių aktualumo užtikrinimas yra pilno etato darbas. Tarkime, kad PĮ pokyčiai lemia dalies testų netinkamumą, tuomet iškart tai nustačius, jų turi būti atsikratoma. Tai palaikymo naštą padarys daug labiau toleruotina. Kai kurie automatizuotų testų rinkinių tvarkymo įrankiai suteikia galimybę išskirti pasenusius ar neteisingus testus. Kai kuriais atvejais gali būti neaišku, ar geriau pašalinti tik testus iš testų rinkinių, ar visus rinkinius. Tokiu atveju galima sukurti pasenusių testų ar jų rinkinių paketus ir pažymėti juos kaip netinkamus, tačiau visiškai nesunaikinti.

Gairė: automatinių testų kūrimas

Nors atrodo, kad automatinių testų sukūrimas turėtų prisidėti prie pastangų, reikalingų testavimo įgyvendinimui, sumažinimo, realybė kartais būna kitokia. Gali būti, kad automatinių testų scenarijų sukūrimas užims daugiau laiko ne pats testavimas, todėl tokiais atvejais verta apsvarstyti, ar nebūtų efektyviau atlikti rankinį testavimą. Nusprendus kurti automatizuotus testus, turi būti išanalizuojama, kurie testų aspektai gali būti automatizuoti ir tada pradedamas testų projektavimas.

Žinoma, automatiniai testai turi ir savų problemų: nepatyrusiems testuotojams juos sunku parengti, o programuotojams reikia nemažai laiko ir pastangų juos realizuoti bei sudėtinga juose ieškoti klaidų. Daugelis testų valdymo įrankių šias situacijas daro mažiau problematiškomis, suteikdami testavimo scenarijų pradines funkcijas (šablonus), skirtas sudaryti testavimo objektų sąrašus, sistemingai būdais suprogramuoti tikrinimo taškus, įdiegti standartines komandas į scenarijus (pavyzdžiui, „užmigimo“ laikmačio), komentuoti ir dokumentuoti scenarijus. Kitas stambus pranašumas, kuris dažnai yra nepastebimas, testavimo valdymo įrankių naudojimas tam, kad esamų scenarijų dalis į naujus scenarijus.

Kai diskutuojama apie testavimo scenarijų automatizavimą, yra svarbu atskirti funkcinius ir našumo testus. Dažniausiai šiose diskusijose yra akcentuojami aplikacijų funkcionalumai. Našumo testų automatizavimas suteikia galimybę programiškai nustatyti darbines apkrovas pridedant naudotojų grupes, paleidžiant testus atsitiktinai ar tam tikru dažnumu, nustatant veikimo trukmę.

Automatiniai testai apima daug klaidų ieškojimo technikų, kurios naudojamos ir įprastose programose. Jų metu turi būti apgalvoti tiek veiklos procesų logika, tiek scenarijų duomenų aspektai. Automatinio testavimo įrankiai pateikia klaidų ieškojimo aplinkas bei testinių duomenų (angl. *datapool*) valdymo įrankius, kurie palengvina šio tipo testavimus. Testavimo scenarijų vykdymo metu, testinius duomenis naudojantys testai gali pakeisti konstantas kintančiomis vertėmis.

Gairė: testavimo tikslai

Nustatyti testavimo tikslai padeda parengti testus. Testavimo tikslai gali atsirasti iš daug įvairių šaltinių. Apskritai jie gali būti gauti skirtingais būdais priklausomai nuo programavimo aplinkos, programuojamos aplikacijos tipo ar testuotojų rafinuotumo. Nors testavimo tikslams nustatyti yra daug įvairių būdų, visgi galima išskirti kelias pagrindines kategorijas.

Štai kelios testavimo idėjos, skirtos išbandyti kvadratinės šaknies apskaičiavimo funkciją:

1. Skaičiaus, kuris vos mažesnis už nulį, įvedimas;
2. Nulio įvedimas;
3. Skaičiaus, iš kurio galima tiksliai ištraukti kvadratinę šaknį (pvz. 4 ar 16), įvedimas;
4. Atspausdinimas;
5. Testavimas, kai duomenų bazė užpildyta.

Pirmieji trys tikslai tikrina duomenų įvedimą, o paskutiniai du – aplinkos problemas. Nors šios formuluotės ir yra ganėtinai neišbaigtos, tačiau jos užtikrina, kad joks aspektas nebūtų užmirštas.

Nustačius konkrečius testavimo tikslus, atsiranda galimybė kelis jų įgyvendinti viename testavimo scenarijuje. Kuriant trumpus scenarijus, skirtus išpildyti keletą tikslų, atsiranda galimybė:

- Nukopijavus ir nežymiai patobulinus scenarijus, atitikti kitus testavimo tikslus;
- Lengviau surasti programoje įsivėlusias klaidas – jei scenarijus padengia du tikslus, tai vadinasi klaida yra viename iš jų, tačiau jei kokius septynis, tai klaidos paieškos laukas išsiplečia.

Vis dėlto reikėtų vengti kompleksinių testavimo scenarijų kūrimo. Štai kelios priežastys:

- Kompleksinius testus sudėtingiau suprasti ir palaikyti;
- Kompleksinius testus sudėtingiau sukurti;
- Sudėtingiau ieškoti klaidų kompleksiniuose testuose.

Gairė: testų rinkiniai

Testų rinkiniai suteikia galimybę suvaldyti testavimo vykdymo kompleksiškumą. Dažnai sistemos testavimo pastangos sužlunga, kad komanda pasimeta detalių testų smulkmenose ir praranda kontrolę. Panašiai kaip ir UML paketų diagramos, testų rinkiniai padeda valdyti testų vykdymą suteikdami testams hierarchinę tvarką. Jie suteikia galimybę valdyti strateginius testavimo aspektus, surinkdami testus kartu į susijusias grupes, kurias galima planuoti, valdyti ar vertinti. Testų rinkiniai reiškia konteinerius, skirtus testų scenarijų kolekcijų tvarkymui. Verta atkreipti dėmesį, kad patys testų rinkiniai gali būti sugrupuoti hierarchiškai – vienas rinkinys gali būti patalpintas kitame.

Kartais testų grupės gali būti susijusios tiesiogiai su posisteme ar kitu sistemos elementu, tačiau kitais atvejais jos bus susijusios su tokiais dalykais kaip kokybės charakteristikos, esminės programos funkcijos, reikalavimų ir standartų atitikimas ir daug kitų dalykų, kurie tvarkomi remiantis sistemos reikalavimais ir dėl to pasiskirstę po visus sistemos elementus. Kuriant testų rinkinius turi būti siekiama sutvarkyti visus esamus testavimo scenarijus keliomis skirtingomis kombinacijomis: kuo daugiau variantų turima, tuo labiau padidėja ištestuota sistemos sritis ir tikimybė joje surasti klaidų. Taigi testų rinkinių įvairovė padės užtikrinti, kad testuojamas objektas bus išbandytas maksimaliai.

2.5.2.7 Nuolatinis integravimas (angl. continous integration)

Pastangos, reikalingos sistemos komponentų tarpusavio integravimui, bėgant laikui auga eksponentiškai. Atliekant integravimo veiklas dažniau, integravimo problemos yra identifikuojamos anksčiau, o integravimui reikia mažiau pastangų. To rezultatas – aukštesnės kokybės produktas ir tiksliau nustatomas produkto išleidimo tvarkaraštis.

Nuolatinis integravimas suteikia tokius privalumus:

- Patogesnis projekto vykdymo veiklų sekimas;
- Geresnis ir ankstesnis klaidų nustatymas;
- Kokybiškesnis ir paprastesnis bendradarbiavimas tarp projekto komandos narių;
- Paprastesnis integravimas;
- Mažesnis skaičius lygiagrečių atnaujinimų, kuriuos reikia prijungti ir ištestuoti;
- Mažiau klaidų randama testavimo metu;
- Mažesnė techninių rizikų tikimybė;
- Mažesnė valdymo rizikų tikimybė.

Nuolatinio integravimo esmė gali būti apibūdinta tokiomis veiklomis:

- Programuotojai atlieka smulkius atnaujinimus naujausiai ištestuotai versijai savo darbo aplinkoje ir tuomet juos ištestuoja prieš pateikdami juos kitiems komandos nariams;
- Visų programuotojų atnaujinimų rinkiniai yra integruojami į darbinę integravimo aplinką ir nuolatos testuojami (bent kartą per dieną, o dar geriau – po kiekvieno atnaujinimo).

Pirmoji veikla užtikrina, kad atnaujinimai yra atliekami teisingai veikiančiai programos versijai ir yra ištestuojami prieš juos pateikiant. Antroji veikla identifikuoja integravimo problemas anksčiau, kad jos būtų išspręstos, kol atnaujinimai nėra užmiršti programuotojų.

Gairė: nuolatinis integravimas

Detalūs nuolatinio integravimo praktikos pritaikymo principai priklauso nuo naudojamų įrankių (konfigūracijų valdymo sistemos, automatizuoti kompiliavimo įrankiai, automatizuoti testavimo įrankiai ir kiti). Vis dėlto pagrindiniai žingsniai yra tokie:

1. Programuotojas pasirenka užduotį, kurią žada atlikti.
2. Programuotojas atnaujina savo darbinę versiją įsidiegdamas naujausią sistemos versiją iš integravimo aplinkos.
3. Programuotojas atlieka pakeitimus darbinėje aplinkoje įdiegtai sistemai bei atnaujina reikiamus testus ir ištestuoja atnaujinimus.
4. Prieš įdiegdamas atnaujinimus integravimo aplinkoje, programuotojas atnaujina savo darbinę aplinką (nes jo darbo metu galėjo kažkas įdiegti naują sistemos versiją integravimo aplinkoje) ir dar kartą atlieka atnaujinimų testavimą.
5. Jei testai yra sėkmingi, atnaujinimai yra įdiegiami integravimo aplinkoje.
6. Įdiegus atnaujinimus integravimo aplinkoje, yra atliekama šios sistemos versijos testavimas.
7. Jei yra randama klaidų, komanda yra apie tai informuojama nurodant testus su klaidomis.
8. Šis procesas kartojamas, kai komanda konstruoja ir nuolatos integruoja bei testuoja funkcionalumus smulkiais inkrementais.

Konceptualiai nuolatinis integravimas gali būti atliekamas labai paprastai. Vis dėlto praktikoje egzistuoja keletas apribojimų, kurių būtina laikytis:

- Visi pakeitimai turi būti įdiegimai į ištestuotą versiją, kuri garantuotai yra teisingai veikianti.
- Integravimo, kompiliavimo ir testavimo ciklas turi būti pakankami trumpas, kad būtų greitai įgyvendinamas, o komanda galėtų matyti rezultatus. Daugelyje metodikų yra rekomenduojamas 10 minučių trukmės ciklas.
- Atnaujinimų rinkiniai turi būti kuo mažesni, kad užduotys būtų užbaigiamos ir pakeitimai integruojami kelis kartus per dieną. Daugelyje metodikų yra rekomenduojamas 2 – 4 valandų trukmės laikotarpis tarp integravimų.

2.5.3 Diegimo praktikos

2.5.3.1 Produkto įvedimas į eksploataciją

Produkto įvedimo į eksploatacija praktika aprašo, kaip pasiruošti ir įvykdyti produkto (ar jo dalies) išleidimą į nuolatinę eksploataciją. Įvedimo į eksploatacija praktika yra skirta užtikrinti, kad sukurta aplikacija yra tinkamai paruošta įvedimui į nuolatinę eksploataciją be jokių nenumatytų pasekmių.

Ši praktika turi du komponentus: pasiruošimas įvedimui į eksploataciją ir diegimas. Pasiruošimo įvedimui į eksploataciją metu yra nustatomi pagrindiniai aspektai ir paruošiama visa reikalinga pagalbinė medžiaga, skirta produkto diegimui. Diegimas apima sistemos instaliavimą produkcinėje aplinkoje, patikrinimą, kad instaliavimas buvo sėkmingas, ir visų suinteresuotų asmenų informavimą, kad sistemos funkcijos yra prieinamos naudojimui.

Siekiant sėkmingai ir be didesnių sunkumų sistemą įvesti į nuolatinę eksploataciją, reikia atsižvelgti į tokius aspektus:

- Produkto įvedimo į eksploatacija veikla nėra laikas, skirtas poilsiui, t. y programuotojų komanda neturėtų atsipalaiduoti ir visas užduotis atlikti įprastais tempais.
- Testavimas nėra svarbesnis už dokumentavimą ir mokymus. Nors įvairių tipų testavimai, kurie būna vykdomi projekto apimtyje, yra svarbūs, dokumentavimas ir mokymai yra lygiai taip pat svarbūs, nes kaip lengvai ir patogiai naudotojai iš tiesų galės naudotis sistema yra kritiška sėkmingam produkto pristatymui.
- Sėkmingas sistemos užgrūdinimas (angl. *hardening*) yra svarbiau už funkcijų skaičių t. y. nereikėtų galvoti, kad platesnis pateiktų funkcijų sąrašas yra geriau. Nors visada yra pagunda išleidžiamame produkte pateikti kuo daugiau funkcijų, tam norui turėtų būti atsispiriama ir pateikiamos tik tos funkcijos, kurios yra užtikrintai išbaigtos ir puikiai ištestuotos bei integruotos.

Įvedimo į eksploataciją metu dažniausiai vykdomos tokios veiklos:

- Susikaupusių techninių problemų šalinimas;
- Stambių defektų šalinimas;
- Integracinių sąsajų su išorinėmis sistemomis užbaigimas;
- Įvairių produkto aspektų dokumentavimas, įskaitant pagalbos ir naudotojų dokumentaciją;
- Funkciniai, našumo, apkrovos, sisteminiai, integracinių sąsajų ir naudojamumo testavimai;
- Naudotojų ir administratorių mokymai;
- Diegimo plano rengimas.

3 Duomenų analizės, ataskaitų ir naudotojo grafinės sąsajos įgyvendinimas (P_2)

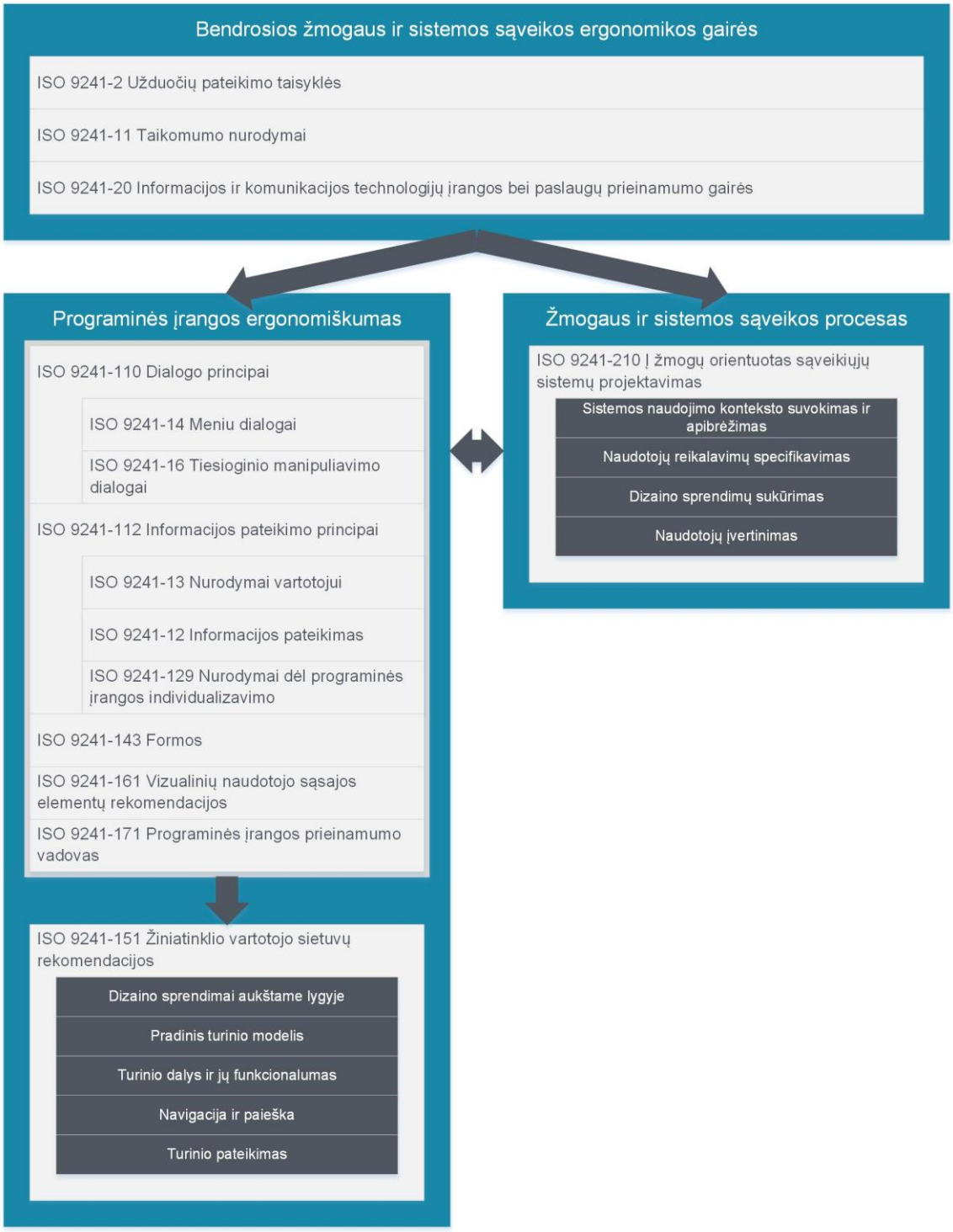
Šiame skyriuje pavaizduota ir aprašyta tinklo topologinio žemėlapio ir susijusių duomenų atvaizdavimo logika. Rengiant eskizus buvo siekiama, kad naudotojo sąsaja būtų paprasta ir aiški, leidžianti greitai išmokti visų pagrindinių sistemos funkcijų valdymo ir darbo eigos (angl. Workflow). Skyriuje taip pat aprašyti tarptautiniai standartai, kuriais bus remiamasi siekiant užtikrinti, kad visos funkcijos būtų pasiekiamos greitai ir efektyviai, užtikrinamas aukščiausio produktyvumo mažiausiomis laiko ir pastangų sąnaudomis.

Atkreipiame dėmesį, kad naudotojo sąsajos įgyvendinimui siūlomi technologiniai sprendimai detalizuoti šio pasiūlymo skyriuje „Sistemos realizacijai siūlomos programinės priemonės ir technologiniai sprendimai".

3.1 Ergonomiškumo užtikrinimas

Projekto metu kuriamas programinis sprendimas ir naudotojo sąsaja bus orientuoti į paprastumą, aiškumą ir patogumą, o įgyvendintas funkcijų valdymas ir darbo eiga užtikrins aukščiausią produktyvumą mažiausiomis laiko ir pastangų sąnaudomis bei visos funkcijos bus pasiekiamos greitai ir efektyviai. Šioms sistemos ergonomiškumo savybėms pasiekti projektuojant naudotojo sąsają ir joje pateikiamas funkcijas bus remiamasi tarptautiniuose standartuose pateikiamomis gerosiomis praktikomis.

Geriausiai ir kokybiškiausiai programinės įrangos ergonomiškumą (patogumą naudoti) aprašo tarptautinių standartų šeima ISO 9241. Šios šeimos standartų tarpusavio sąsajos pateikiamos žemiau esnačiame paveiksle.



28 paveikslas. Patogumą naudoti aprašančių standartų tarpusavio sąsajos

Didžiausias dėmesys bus skiriamas internete pateikiamų naudotojo sąsajų patogumą naudoti aprašančiam standartui „LST EN ISO 9241-151:2008 Žmogaus ir sistemos sąveikos ergonomika. 151 dalis. Žiniatinklio vartotojo sietuvų rekomendacijos“ (angl. *Ergonomics of human-system interaction -- Part 151: Guidance on World Wide Web user interfaces*) (toliau – ISO 9241-151), bei žmogaus ir sistemos sąveikos procesą apibūdinančiam standartui „LST EN ISO 9241-210:2011

Žmogaus ir sistemos sąveikos ergonomika. 210 dalis. Į žmogų orientuotas sąveikiųjų sistemų projektavimas“ (angl. *Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems*) (toliau – ISO 9241-210).

ISO 9241-151 standarte pateikiami patarimai apie į naudotoją orientuotos interneto prieigos kūrimą, siekiant padidinti jos tinkamumą naudoti. Standarte pateikiamose gerosiose praktikose akcentuojami šie interneto naudotojų prieigos kūrimo aspektai:

- aukšto lygio kūrimo sprendimai ir kūrimo strategijos;
- turinio kūrimas;
- navigacija ir paieška;
- turinio pateikimas.

ISO 9241-210 standarte pateikiami patarimai apie į naudotoją orientuotų informacinių sistemų projektavimą, įskaitant naudotojų atliekamą sistemos įvertinimą. Šio standarto 6.5 dalyje pateikiami sistemos vertinimo patarimai, skirti užtikrinti sukurtos sistemos atitikimą jų lūkesčiams ir reikalavimams. Čia pateikiami keli galimi sistemos vertinimo modeliai, kurie gali būti pasirenkami priklausomai nuo projektavimo tikslų ir finansinių galimybių. Išvardijami šie testavimo būdai:

- Informacinės sistemos testavimas, pasitelkiant tikslinius naudotojus;
- Inspekcijos tipo testavimas;
- Ilgalaikė stebėsena.

Atsižvelgiant į Projekto specifiką, projektuojant ir kuriant programinę įrangą, papildomai bus atsižvelgiama į šiuos standartus:

- LST EN ISO 9241-110:2006 Žmogaus ir sistemos sąveikos ergonomika. 110 dalis. Dialogo principai;
- ISO 9241-112:2017 Žmogaus ir sistemos sąveikos ergonomika. 112 dalis. Informacijos pateikimo principai;
- LST EN ISO 9241-12:2001 Ergonomikos reikalavimai, keliami įstaigos darbui su galiniais vizualizavimo įrenginiais (VDTs). 12 dalis. Informacijos pateikimas (ISO 9241-12:1998);
- LST EN ISO 9241-13:2001 Ergonomikos reikalavimai, keliami įstaigos darbui su galiniais vizualizavimo įrenginiais (VDTs). 13 dalis. Nurodymai vartotojui;
- LST EN ISO 9241-14:2001 Ergonomikos reikalavimai, keliami įstaigos darbui su galiniais vizualizavimo įrenginiais (VDTs). 14 dalis. Meniu dialogai;
- LST EN ISO 9241-16:2001/P:2008 Ergonomikos reikalavimai, keliami įstaigos darbui su galiniais vizualizavimo įrenginiais (VDTs). 16 dalis. Tiesioginio manipuliavimo dialogai;
- LST EN ISO 9241-129:2011 Žmogaus ir sistemos sąveikos ergonomika. 129 dalis. Nurodymai dėl programinės įrangos individualizavimo;
- LST EN ISO 9241-143:2012 Žmogaus ir sistemos sąveikos ergonomika. 143 dalis. Formos;
- ISO 9241-161:2016 Žmogaus ir sistemos sąveikos ergonomika. 161 dalis. Vizualinių naudotojų sąsajos elementų rekomendacijos;
- LST EN ISO 9241-171:2008 Žmogaus ir sistemos sąveikos ergonomika. 171 dalis. Programinės įrangos prieinamumo vadovas.

3.2 Sistemos naudotojo sąsaja

Šiame poskyryje pateikiami preliminarūs naudotojo sąsajos ekranvaizdžiai, iš kurių matosi, kad visos sistemos funkcijos yra pasiekiamos greitai ir efektyviai, užtikrinamas aukščiausio produktyvumas mažiausiomis laiko ir pastangų sąnaudomis.

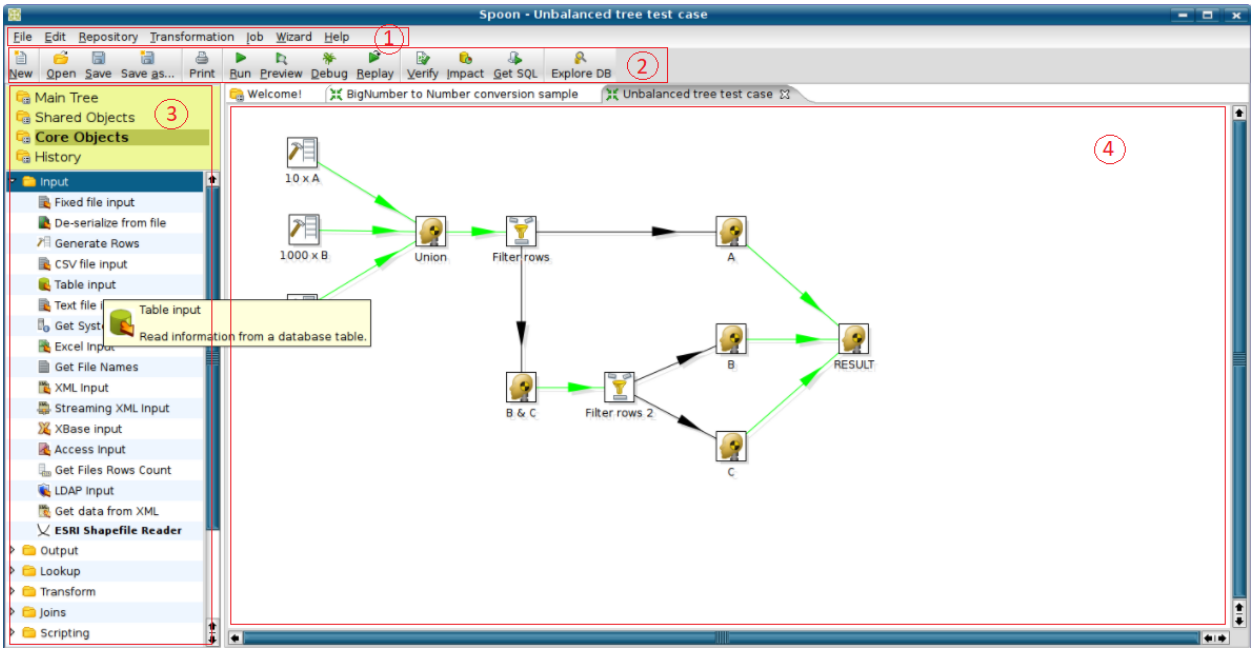
Remiantis 3.1 poskyryje aprašytuose standartuose pateikiamomis gerosiomis praktikomis ir rekomendacijomis buvo parengti pavyzdiniai sistemos naudotojo sąsajos ekranvaizdžiai, kuriuose vaizduojamas skyriuje „Panaudos atvejų pjūvis“ aprašytų funkcijų preliminarus realizavimas.

Atkreipiame dėmesį, kad dalies sistemos funkcijų realizavimui bus naudojami standartiniai programiniai įrankiai, todėl jų naudotojo sąsaja projekto metu nebus keičiama. Kitai daliai funkcionalumų, kuriuos realizuos pats Tiekėjas, šiame pasiūlyme pateikiami preliminarūs ekranvaizdžiai, suprojektuoti išanalizavus TS reikalavimus ir kitą turimą susijusią informaciją. Šie ekranvaizdžiai bus derinami ir tikslinami projekto metu.

Toliau pateikiamuose ekranvaizdžiuose raudonai pažymėti pagrindiniai dizaino ir funkciniai elementai, o žemiau paveikslų pateikiami jų aprašymai, apimantys funkcijos paskirtį ir alternatyvius įgyvendinimo būdus.

3.2.1 Duomenų surinkimo ir apdorojimo taisyklių kūrimas bei redagavimas

Kaip buvo minėta skyriuje „Duomenų surinkimo ir apdorojimo komponentas - Pentaho data integration“, duomenų surinkimo ir apdorojimo taisyklių kūrimui bei redagavimui bus naudojama Pentaho PDI Spoon programinė įranga, kuri yra lokaliame kompiuteryje instaliuojama aplikacija ir suteikia galimybę grafinės naudotojo sąsajos pagalba kurti ir konfigūruoti ETL procesus. Šis įrankis suteikia galimybę nerašant programinio kodo, „drag and drop“ principu, sukurti gana sudėtingas ETL užduotis. Žemiau esančiame paveiksle pateikia Spoon įrankio naudotojo sąsajos pavyzdys:



29 paveikslas. Duomenų surinkimo ir apdorojimo taisyklių kūrimo bei redagavimo ekranvaizdis

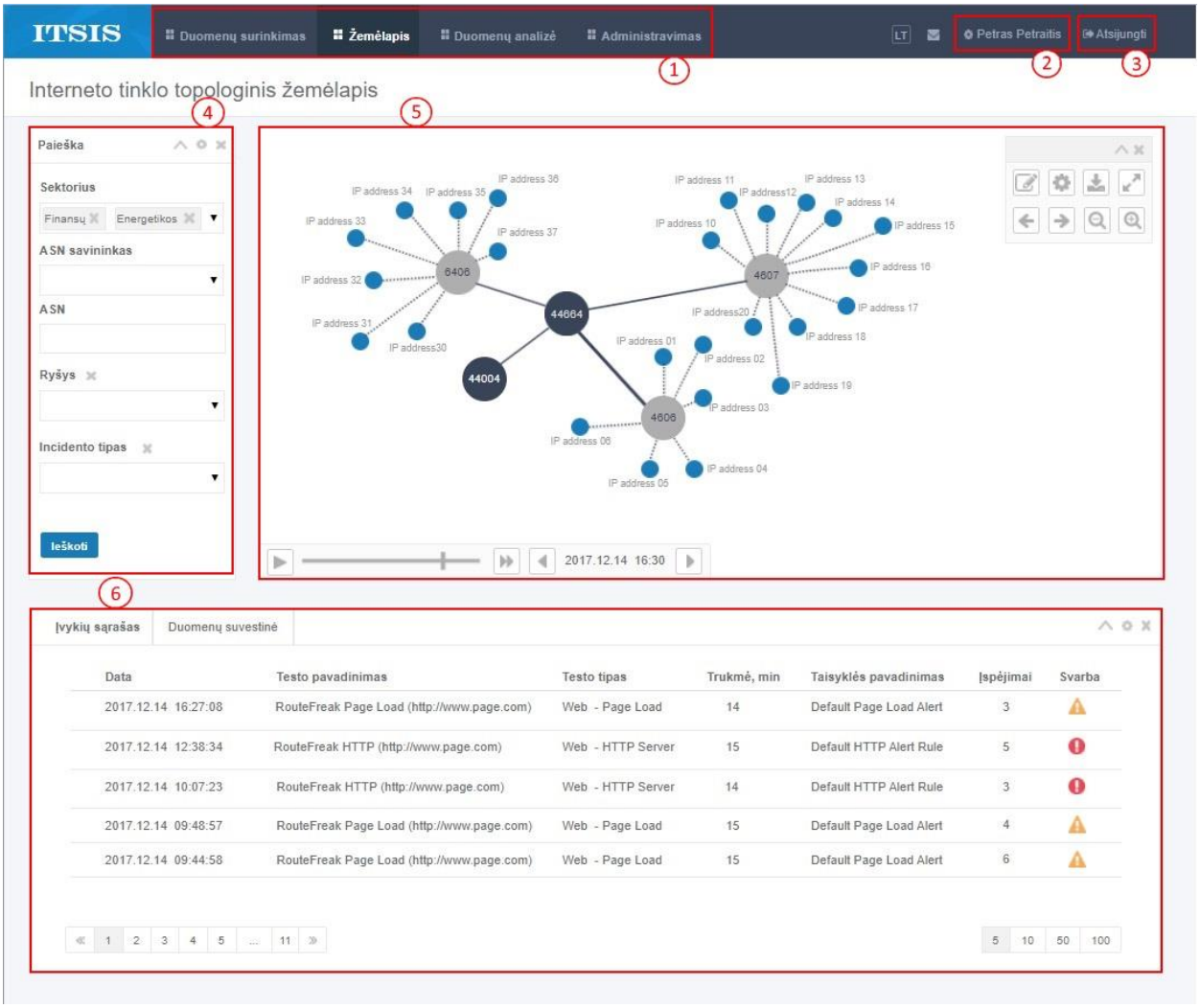
Ekranvaizdyje raudonai pažymėti šie funkciniai komponentai:

1. Pagrindinė įrankio meniu juosta.
2. Duomenų surinkimo ir apdorojimo (ETL) užduoties veiksmų juosta. Šioje juostoje galima inicijuoti naujos ETL užduoties kūrimą, išsaugojimą, testavimą, vykdymą ir pan.

3. ETL užduoties komponentai. Šioje dalyje galima pasirinkti ETL užduoties komponentus ir vykdymo žingsnius, t.y. duomenų šaltinius, transformacijas, filtravimus, rezultatų išsaugojimą ir pan.
4. Darbinė zona. Šioje zonoje visi ETL užduoties komponentai ir vykdymo žingsniai turi būti sudėlioti į vieną seką.

3.2.2 Tinklo topologinis žemėlapis

31 - 33 paveiksluose vaizduojamuose ekranvaizdžiuose raudonai pažymėti tinklo topologinio žemėlapio peržiūros pagrindiniai dizaino ir funkciniai elementai, o žemiau paveikslų pateikiami jų aprašymai, apimantys funkcijos paskirtį ir alternatyvius įgyvendinimo būdus.

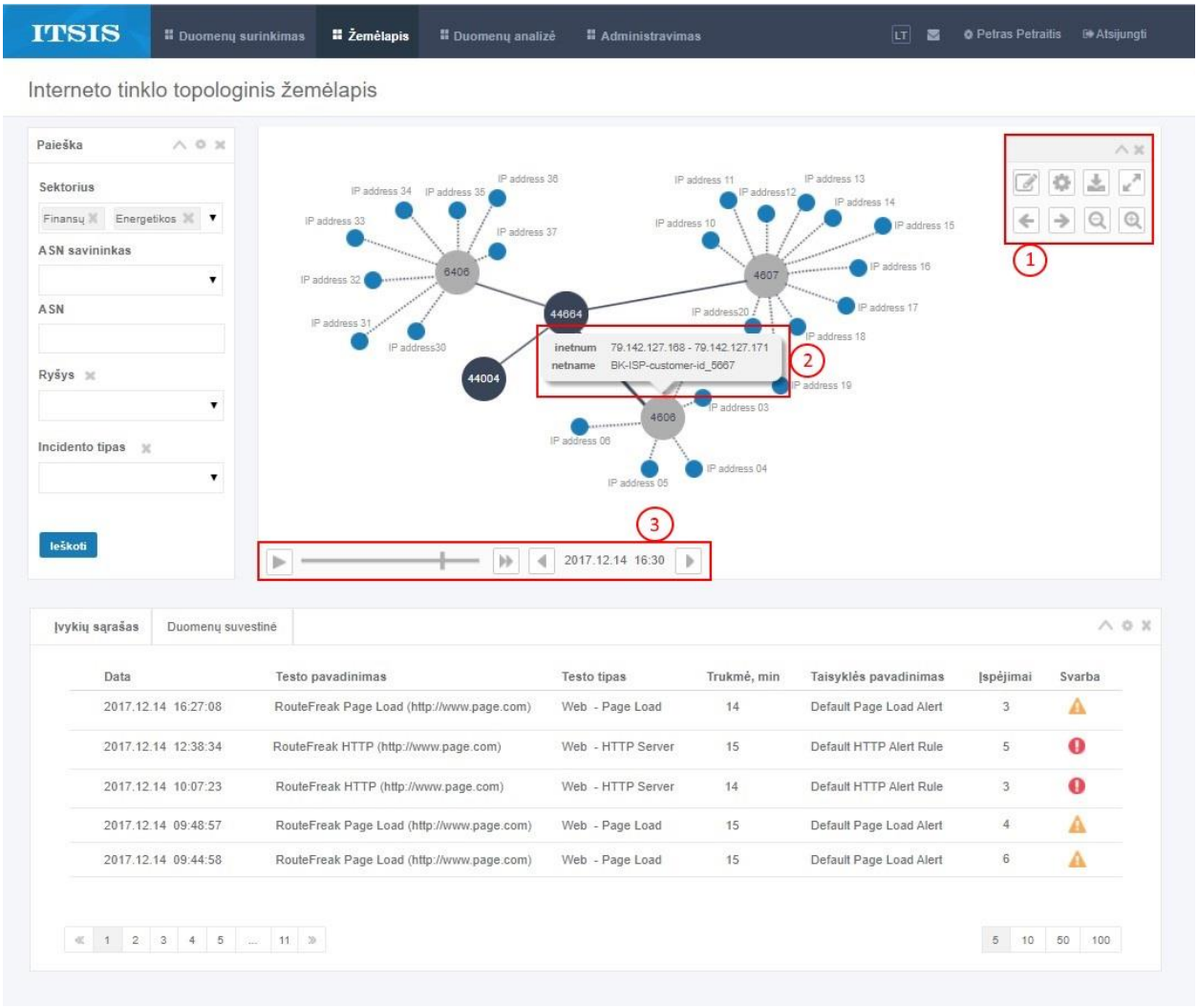


30 paveikslas. Tinklo topologinio žemėlapio peržiūra

31 paveiksle pateikiamas ekranvaizdis, skirtas atvaizduoti tinklo topologinio žemėlapio peržiūros funkcionalumą. Šiame ekranvaizdyje raudonai pažymėti šie funkciniai komponentai:

1. Pagrindinis meniu, kuriame pateikiamos nuorodos į pagrindines Sistemos funkcijas.
2. Prisijungusio naudotojo vardas bei pavardė, skirti pasiekti naudotojo informaciją ir jos tvarkymo funkcionalumą.
3. Mygtukas, skirtas atsijungti nuo Sistemos portalo.

4. Paieškos blokas skirtas atlikti topologiniame žemėlapyje pateikiamų objektų filtravimą pagal pasirinktus kriterijus. Paieškos blokas gali būti suskleidžiamas, konfigūruojamas, išjungiamas. Bloko vieta gali būti keičiama jį nutempiant į kitą vietą, bloko dydis gali būti didinamas/mažinamas.
5. Tinklo topologinio žemėlapio peržiūros, valdymo, analizes blokas. Bloko vieta gali būti keičiama jį nutempiant į kitą vietą, bloko dydis gali didinamas/mažinamas.
6. Informaciniai blokai – svarbių įvykių sąrašas ir sistemos duomenų suvestinė. Informaciniai blokai gali būti suskleidžiami, konfigūruojami, išjungiami. Blokų vieta gali būti keičiama jį nutempiant į kitą vietą, blokų dydis gali būti didinamas/mažinamas.



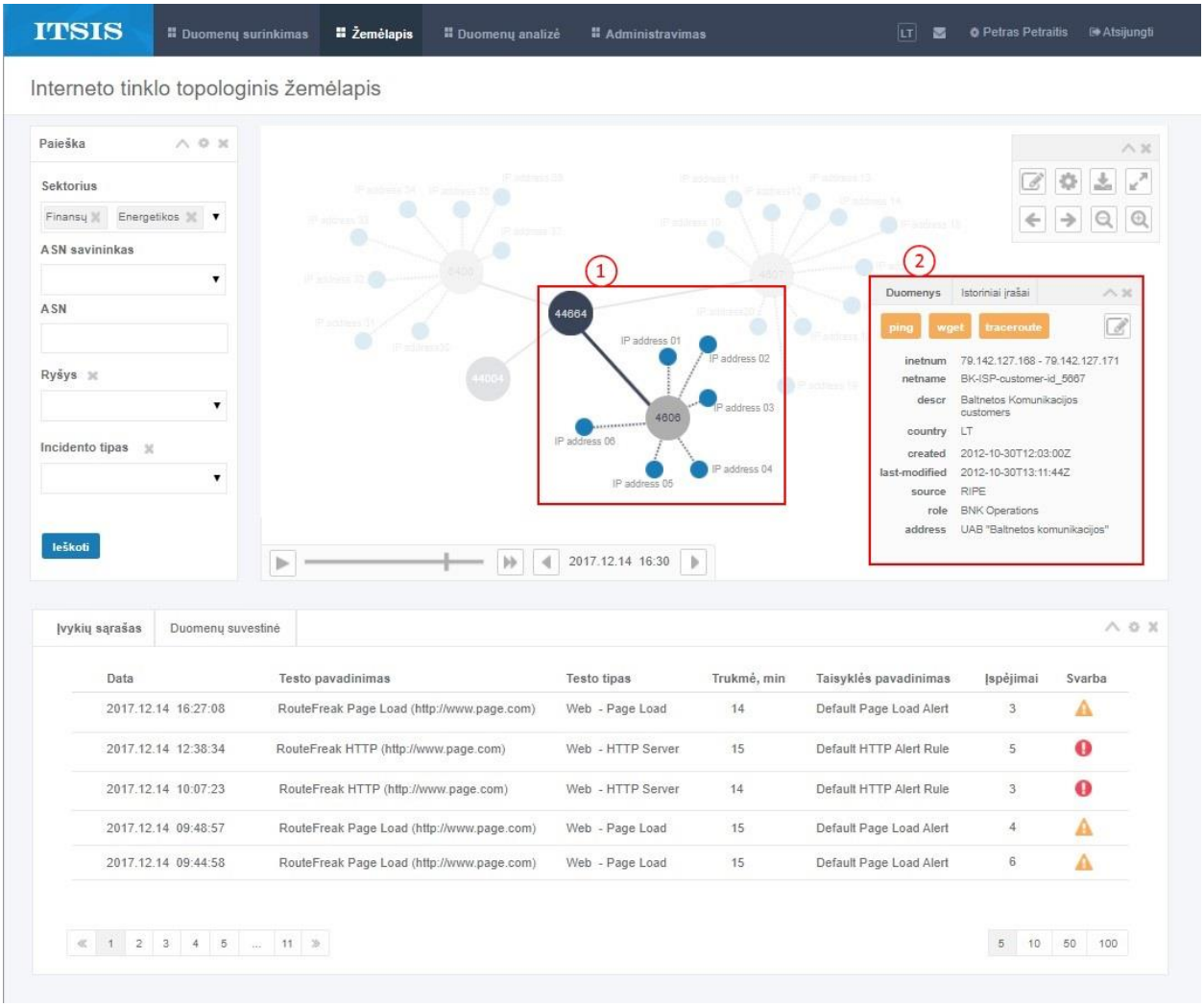
31 paveikslas. Tinklo topologinio žemėlapio valdymas

32 paveiksle pateikiamas ekranvaizdis, skirtas atvaizduoti tinklo topologinio žemėlapio valdymo bei objektų ir ryšių pokyčių laike analizės funkcionalumą. Šiame ekranvaizdyje raudonai pažymėti šie funkciniai komponentai:

1. Panelė skirta tinklo topologinio žemėlapio valdymui apimanti tokias funkcijas kaip: žemėlapio objektų ir ryšių atvaizdavimo valdymą, eksportavimą, žemėlapio peržiūros bloko išplėtimą paslepiant kitus blokus, vaizdo

slankiojimą į kairę ir dešinę, mastelio keitimą. Panelė gali būti suskleidžiama, išjungiama, vieta gali būti keičiama ją nutempiant į kitą vietą.

2. Virš objekto užvedus pele pateikiama pagrindinė objekto informacija (angl. „tooltip“).
3. Topologinio žemėlapio pokyčių analizei skirta panelė.



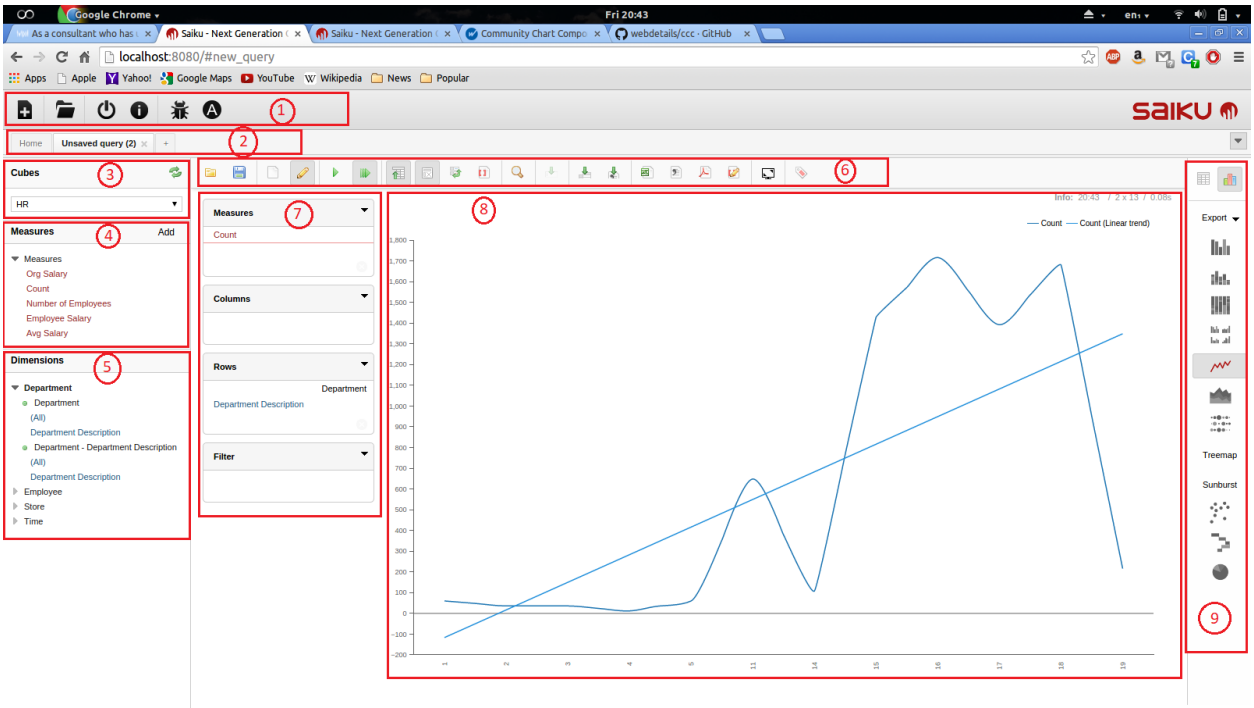
32 paveikslas. Tinklo topologinio žemėlapio valdymas

33 paveiksle pateikiamas ekranvaizdis, skirtas atvaizduoti tinklo topologinio žemėlapio valdymo bei objektų ir ryšių pokyčių laike analizės funkcionalumą. Šiame ekranvaizdyje raudonai pažymėti šie funkciniai komponentai:

1. Pasirinktos tinklo dalies išryškinimas žemėlapyje.
2. Panelė, kurioje pateikiama pasirinkto objekto detali informacija (objekto duomenys, istoriniai įrašai), objekto statusas ar kita naujausia informacija realiuoju laiku panaudojus turimą įrankį (pvz., ping) ir pagrindinių duomenų koregavimo funkcionalumas. Panelė gali būti suskleidžiama, išjungiama, vieta gali būti keičiama ją nutempiant į kitą vietą.

3.2.3 Duomenų analizė ir ataskaitos

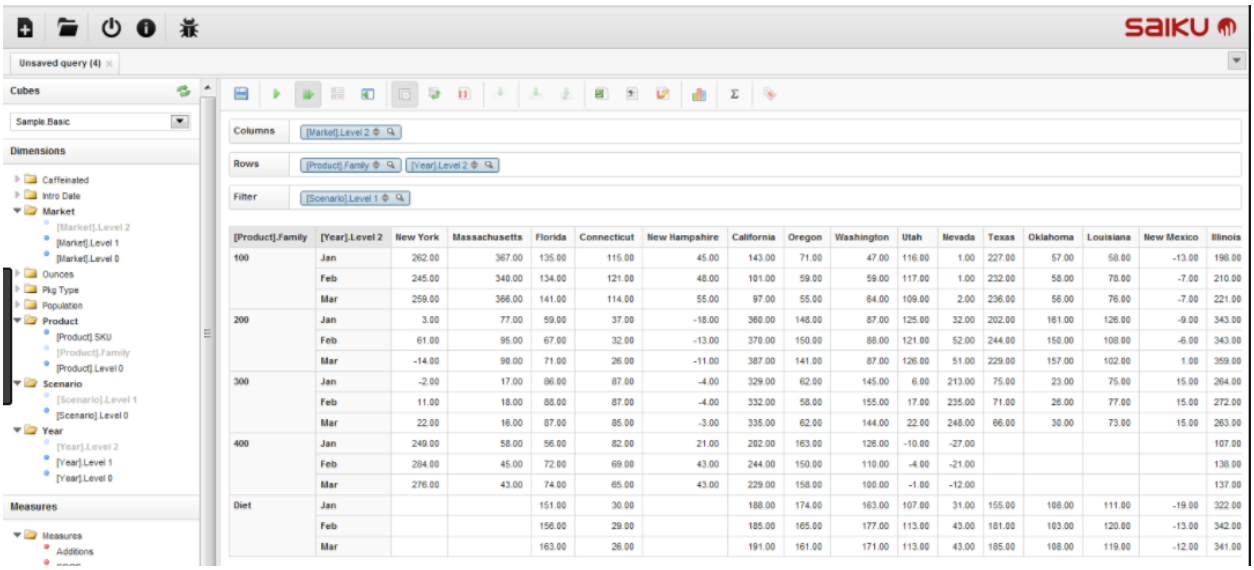
Kaip buvo minėta skyriuje „Dinaminės duomenų analizės komponentai – Pentaho Mondrian ir Saiku CE“, dinaminei duomenų analizei bus naudojama Saiku CE programinė įranga. Šis įrankis suteikia galimybę interaktyvios naudotojo sąsajos pagalba dinamiškai analizuoti daugiamačius duomenų rinkinius, t.y. „drag and drop“ principu, pasirinkti duomenų lentelių stulpelius, eilutes rodiklius, filtrus, duomenų atvaizdavimo formatą ir pan. Žemiau esančiuose paveiksluose pateikiama keletas Saiku CE įrankio naudotojo sąsajos pavyzdžių:



33 paveikslas. Saiku CE dinaminės duomenų analizės naudotojo sąsajos pavyzdys (grafinis duomenų atvaizdavimas)

Aukščiau esančiame paveiksle vaizduojamos naudotojo sąsajos pagrindiniai funkciniai blokai:

1. Pagrindinis analizės įrankio meniu.
2. Naudotojo sukurtų ar kuriamų dinaminių ataskaitų skirtukai.
3. Duomenų kubo pasirinkimo skiltis.
4. Rodiklių pasirinkimo skiltis.
5. Dimensijų pasirinkimo skiltis.
6. Ataskaitos įrankių juosta.
7. Ataskaitoje pasirinktų rodiklių, stulpelių, eilučių ir filtrų valdymo skiltis.
8. Ataskaitos formavimo zona.
9. Ataskaitos atvaizdavimo tipo valdymo meniu.



[Product].Family	[Year].Level 2	New York	Massachusetts	Florida	Connecticut	New Hampshire	California	Oregon	Washington	Utah	Nevada	Texas	Oklahoma	Louisiana	New Mexico	Illinois
100	Jan	262.00	367.00	135.00	115.00	45.00	143.00	71.00	47.00	116.00	1.00	227.00	57.00	58.00	-13.00	198.00
	Feb	245.00	340.00	134.00	121.00	48.00	101.00	59.00	59.00	117.00	1.00	232.00	58.00	78.00	-7.00	210.00
	Mar	259.00	366.00	141.00	114.00	55.00	97.00	55.00	64.00	109.00	2.00	236.00	56.00	76.00	-7.00	221.00
200	Jan	3.00	77.00	59.00	37.00	-18.00	360.00	148.00	87.00	125.00	32.00	202.00	161.00	126.00	-9.00	343.00
	Feb	61.00	95.00	67.00	32.00	-13.00	370.00	150.00	88.00	121.00	52.00	244.00	150.00	108.00	-6.00	343.00
	Mar	-14.00	90.00	71.00	26.00	-11.00	387.00	141.00	87.00	126.00	51.00	229.00	157.00	102.00	1.00	359.00
300	Jan	-2.00	17.00	86.00	87.00	-4.00	329.00	62.00	145.00	6.00	213.00	75.00	23.00	75.00	15.00	264.00
	Feb	11.00	18.00	88.00	87.00	-4.00	332.00	58.00	155.00	17.00	235.00	71.00	26.00	77.00	15.00	272.00
	Mar	22.00	16.00	87.00	85.00	-3.00	335.00	62.00	144.00	22.00	248.00	66.00	30.00	73.00	15.00	263.00
400	Jan	249.00	58.00	56.00	82.00	21.00	202.00	163.00	126.00	-10.00	-27.00					107.00
	Feb	284.00	45.00	72.00	69.00	43.00	244.00	150.00	110.00	-4.00	-21.00					138.00
	Mar	276.00	43.00	74.00	65.00	43.00	229.00	158.00	100.00	-1.00	-12.00					137.00
Diet	Jan			151.00	30.00		188.00	174.00	163.00	107.00	31.00	155.00	108.00	111.00	-19.00	322.00
	Feb			156.00	29.00		185.00	165.00	177.00	113.00	43.00	181.00	103.00	120.00	-13.00	342.00
	Mar			163.00	26.00		191.00	161.00	171.00	113.00	43.00	185.00	108.00	119.00	-12.00	341.00

34 paveikslas. Saiku CE dinaminės duomenų analizės naudotojo sąsajos pavyzdys (duomenų atvaizdavimas lentelės pavidalu)

4 Paslaugų teikimo planas (P_3)

Šiame poskyryje pateiktas aiškus, detalus ir racionalus paslaugų teikimo planas, kuriame paslaugų teikimas suskirstytas į atskirus etapus. Šių etapų išskyrimas ir seka labai gerai pagrįsta darbų tarpusavio sąsaja. Paslaugų teikimo planu optimizuotas žmogiškųjų išteklių naudojimas, numatomos alternatyvos sprendžiant problemas dėl vėlavimo ar veiklų persidengimo, parodyta plano sąsaja su kitomis pasiūlymo dalimis. Taip pat pateiktas aiškus ir detalus paslaugų teikimo grafikas, paslaugų teikimui skirtas laikas ir terminai pagrįsti ir išsamiai paaiškinti, numatytos galimos alternatyvos.

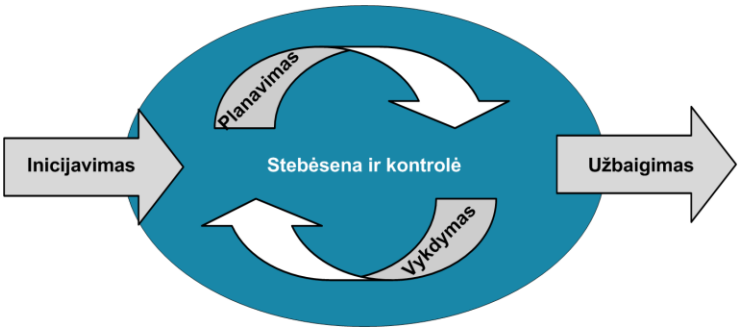
4.1 Projekto valdymo metodika ir principai

Siekiant užtikrinti tinkamą Paslaugų teikimo valdymą ir aukštą galutinių rezultatų kokybę, Projekto valdymui numatoma taikyti atitinkamai adaptuotą IT projektų valdymo metodiką, parengtą remiantis gerosios praktikos principais, išdėstytais Projektų valdymo instituto (angl. *Project Management Institute*) projektų valdymo metodinėse gairėse (angl. *Project Management Body of Knowledge,* toliau – PMBOK).

PMBOK metodika aprašo rekomenduojamus taikyti visuotinai pripažintus projektų valdymo procesus ir žinių sritis, kurioms turėtų būti skiriama daugiausiai dėmesio įgyvendinant projektą. Šioje metodikoje išskirti projekto valdymo procesai suskirstyti į penkias procesų grupes:

- inicijavimo procesus, susijusius su naujo etapo arba proceso apibrėžimu;
- planavimo procesus, susijusius su projekto apimties nustatymu, valdymo plano sudarymu ir projekto veiksmų, kuriuos reikia atlikti projekto vykdymo eigoje, nustatymu bei planavimu;
- vykdymo procesus, susijusius su darbų, numatytų projekto valdymo plane, atlikimu;
- stebėsenos ir kontrolės procesus, susijusius su proceso inicijavimo, planavimo, vykdymo ir užbaigimo priežiūra;
- užbaigimo procesus, susijusius su oficialiu visų projekto arba etapo veiksmų užbaigimu ir užbaigtų rezultatų perdavimu.

Šių procesų grupių tarpusavio sąveika pateikta paveiksle žemiau.



35 paveikslas. Projekto valdymo procesai

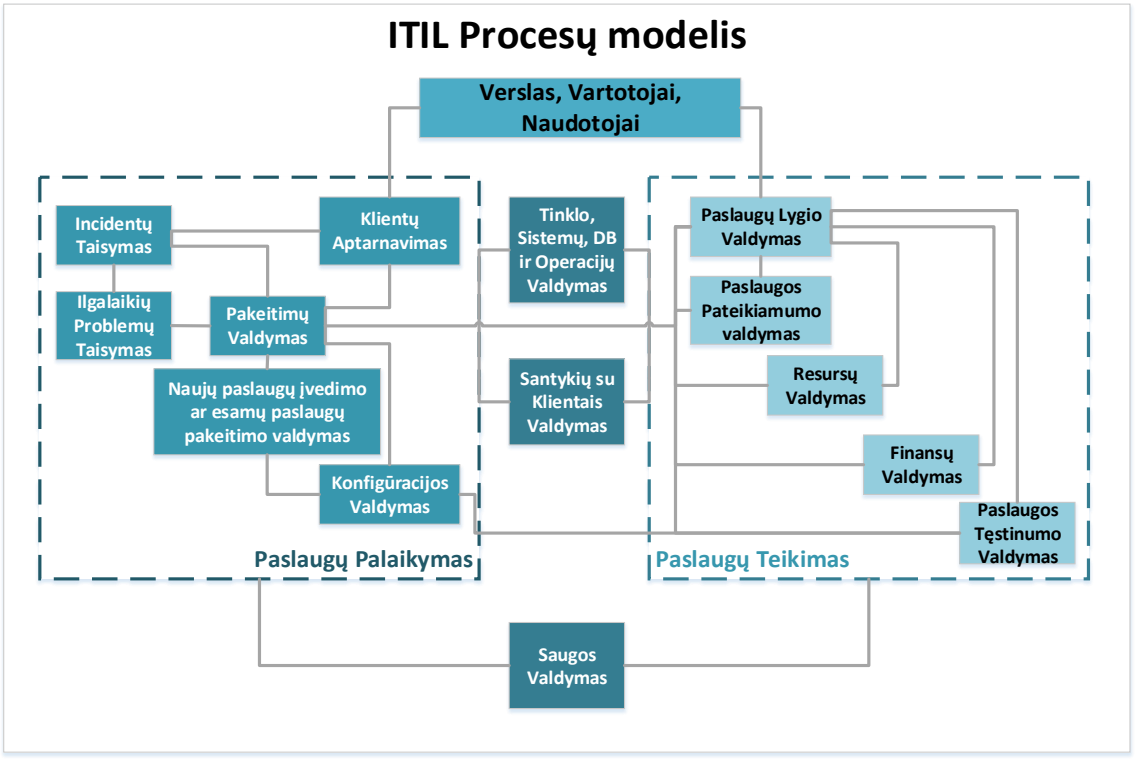
Remiantis PMBOK metodikoje pateikiamomis rekomendacijomis bei vadovaujantis Paslaugų pirkimo techninėje specifikacijoje nustatytais reikalavimais:

- Pasiūlymo 4.4 skyriuje pateikiamas ITIL metodika paremtas pakeitimų valdymo aprašymas;

- Pasiūlymo 5.1 skyriuje pateikiamas komunikacijos planas, detalizuojantis Projekto organizacinę struktūrą ir jo metu numatomus taikyti komunikacijos būdus bei jų taikymo aplinkybes;
- Pasiūlymo 4.2 skyriuje pateikiamas išsamus Paslaugų teikimo grafikas, apimantis Paslaugų teikimo etapus bei detalų numatytų darbų įgyvendinimo grafiką;
- Pasiūlymo 5.2 skyriuje aprašomas kokybės valdymo planas, apimantis Projekto veiklų ir rezultatų kokybės užtikrinimą;
- Pasiūlymo 6 skyriuje pateikiamas Projekto rizikų valdymo planas, apimantis bendrą rinkų valdymo metodiką ir jos taikymą bei konkrečius rengiant Pasiūlymą identifikuotus rizikos veiksnius ir reikalingas jų valdymo priemones;
- Pasiūlymo 4.3 skyriuje pateikiamas žmogiškųjų išteklių valdymo planas, kuriame pristatoma Paslaugų teikėjo ekspertų komanda bei pateikiamas detalus jos nariams numatytų funkcijų ir atsakomybių Projekte aprašymas.

Projekto valdymo veiklose taip pat bus taikomi ITIL veiklos valdymo principai. ITIL (angl. *Information Technology Infrastructure Library*) – veiklos valdymo teorija, orientuota į darbo optimizavimą bei kokybės užtikrinimą IT įmonėse. Tai labiausiai paplitusi IT įmonių valdymo metodologija, kurta remiantis jau egzistuojančiomis IT metodologijomis (pvz. konfigūracijos valdymas), konsultacinių firmų patirtimi, patvirtinta Didžiosios Britanijos standartu BS15000.

Skirtingai nei daugelis kitų teorijų, ITIL yra paremta keliomis teorinio apibendrinimo idėjomis: visų pirma, siekiant tiksliai aprašyti galimus procesus, ITIL sukuria specifinę terminiją, skirtą procesų aprašymui. Daugelis ITIL naudojamų sąvokų ne visiškai tiksliai atitinka įprastas atitinkamų žodžių reikšmes, pvz., *incidentas* ITIL terminologijoje reiškia bet kokį paslaugos sutrikimą, nepriklausomai nuo jo kilmės, o *problema* – ne vartotojo patiriamą paslaugos sutrikimą, o tik *incidento* priežastį. Tokia terminija garantuoja tikslų ir vienareikšmį įmonėje vykstančių procesų aprašymą, tačiau kažkiek trukdo greitą pačios ITIL teorijos įsisavinimą.



36 paveikslas. ITIL procesų modelis

Kitas ITIL bruožas – tai abstrakcija. Skirtingai nuo taikomųjų metodologijų, tokių, kaip APM ar MOF, ITIL nekonkretizuoja procesų ar, juo labiau, naudojamų priemonių. Pvz., ITIL nustato, kad *CMDB* (angl. *Configuration Management Data Base*) gali būti keičiama tik pakeitimų valdymo *(*angl. *Change management)* proceso metu, tačiau nekonkretizuoja *CMDB* laikomų duomenų struktūros ar jų paskirties. Nepaisant to, ITIL stengiasi pateikti labiausiai vykusius IT valdymo metodologijų pavyzdžius.

Nors daugelis dalykų ITIL teorijoje atitinka COBIT sampratą, ITIL yra orientuota visų pirma į kokybės gerinimą, o ne į taupų lėšų panaudojimą, todėl labiau tinka valstybinėms įstaigoms bei firmoms, besiorientuojančioms į viršutinį rinkos sektorių. Aktuali ITIL sąsaja ir su ISO 20000 standartu. ITIL Service Strategy (Paslaugų strategijos parengimo knygoje) teigiama: „*Published in 2005, ISO/IEC 20000 is the first formal international standard specific to IT service management. An organization comfortable with ITIL will find no difficulty in interpreting ISO/IEC 20000.*“ Egzistuoja kelios modifikuotos ITIL versijos. APM (angl. *Alignability Process Model*) – konkretizuota, supaprastinta ITIL versija su pavyzdinėmis procedūromis bei konfigūracija HP ServiceDesk programai. Viena iš ypatybių – aptarnavimo skambučių *(*angl. *Service call)* ir incidentų valdymo sujungimas. Skirtingai nuo įprastinio ITIL, APM modelis gali būti įdiegtas žymiai lengviau, tokiu būdu tapdamas tarpiniu laipteliu, „ITIL-izuojant“ įmonę. MOF (angl. *Microsoft Operations Framework*) – labai supaprastinta, dalies modulių neturinti ITIL versija, labiau orientuota į programinės įrangos kūrimą bei palaikymą, kuris šioje teorijoje išnagrinėtas daug išsamiau, nei ITIL.

4.2 Paslaugų teikimo grafiko valdymas

Paslaugų teikimo grafikas sudarytas vadovaujantis PMBOK metodikoje išdėstytais laiko planavimo principais. Planuojant laiką, šioje metodikoje rekomenduojama: apibrėžti numatomus darbus, nustatyti šių darbų eiliškumą, įvertinti numatomų darbų trukmes, nustatyti Projekto dalyvių atsakomybes ir parengti darbų įgyvendinimo grafiką.

Vykdomi darbai, jų eiliškumas, numatomas Perkančiosios organizacijos indėlis bei preliminarūs vykdymo terminai pateikiami žemiau esančioje lentelėje.

13 lentelė. Paslaugų teikimo etapai ir darbai

Etapas	Darbai	Pateiktys / rezultatai	Perkančiosios organizacijos indėlis	Planuojama pradžia	Planuojama pabaiga
Iniciavimas	▪ Paslaugų teikimo sutarties pasirašymas ▪ Įvadinis susirinkimas ▪ Pasiūlyme pateikto kalendorinio projekto grafiko atnaujinimas	▪ Pasirašyta sutartis ▪ Įvykęs įvadinis susirinkimas ▪ Atnaujintas projekto kalendorinis grafikas	Sutarties pasirašymas ir dalyvavimas įvadiniame susirinkime	2018.02.01	2018.02.02
Analizė	▪ Atliekama detali poreikių ir susijusių sistemų analizė ▪ Patikslinami TS nurodyti Sistemos reikalavimai ▪ Parengiami reikalavimai kompiuterinei techninei ir sisteminei programinei įrangai ▪ Pagrindžiama planuojama naudoti programinė įranga	Sistemos analizės dokumentacija apimanti: ▪ Sistemos paskirtis, tikslai, informaciniai srautai, detalūs funkciniai ir nefunkciniai reikalavimai. ▪ reikalavimai kompiuterinei techninei ir sisteminei programinei įrangai. ▪ Planuojamos naudoti programinės įrangos	Informacijos teikimas, atsakymų į pateiktus klausimus teikimas, dokumento derinimas, pastabų aptarimas	2018.02.05	2018.04.03

Etapas	Darbai	Pateiktys / rezultatai	Perkančiosios organizacijos indėlis	Planuojama pradžia	Planuojama pabaiga
		pasirinkimo pagrindimas.			
Projektavimas	▪ Parengiama Sistemos projektavimo dokumentacija; ▪ Parengiamas modernizuotos Sistemos testavimo planas.	Sistemos projektavimo dokumentacija apimanti: ▪ detalizuotų funkcinių, techninių ir kitų reikalavimų įgyvendinimo aprašymą. ▪ Įgyvendinamų komponentų sąrašai, jų tipai, įgyvendinamos Sistemos funkcijos, sąryšiai su kitais komponentais, komponentų vaizdai, funkcinė logika, o taip pat Sistemos parametrų aprašymai, duomenų bazės modelio aprašymai, sąsajų tarp sistemų ir jų valdymo aprašymai bei kita reikalinga informacija. Detalus testavimo planas, kuriame aprašytos testavimo metodikos.	Informacijos teikimas, atsakymų į pateiktus klausimus teikimas, dokumento derinimas, pastabų aptarimas	2018.04.04	2018.05.29
Kūrimas	**I iteracija:** ▪ Kūrimo aplinkos parengimas ▪ Duomenų surinkimo ir apdorojimo, pagrindinių tinklo topologinio žemėlapio bei duomenų paieškos funkcionalumų programavimas ▪ Sistemos analizės ir projektavimo dokumentacijos tikslinimas ▪ Tarpinis rezultatų pristatymas	▪ Parengta kūrimo aplinka ▪ Suprogramuoti duomenų surinkimo ir apdorojimo, pagrindinių tinklo topologinio žemėlapio bei duomenų paieškos funkcionalumai ▪ Patikslinta Sistemos analizės ir projektavimo dokumentacija tikslinimas ▪ Atliktas tarpinis rezultatų pristatymas	Informacijos teikimas, atsakymų į pateiktus klausimus teikimas, pastabų aptarimas, dalyvavimas tarpinių rezultatų pristatymuose	2018.05.30	2018.08.13
	II iteracija ▪ Likusių topologinio žemėlapio, duomenų analizės, grėsmių aptikimo ir administravimo funkcionalumų programavimas ▪ Sistemos analizės ir projektavimo dokumentacijos tikslinimas ▪ Tarpinis rezultatų pristatymas	▪ Suprogramuoti likusių topologinio žemėlapio, duomenų analizės, grėsmių aptikimo ir administravimo funkcionalumai ▪ Patikslinta Sistemos analizės ir projektavimo dokumentacija ▪ Atliktas tarpinis rezultatų pristatymas	Informacijos teikimas, atsakymų į pateiktus klausimus teikimas, pastabų aptarimas, dalyvavimas tarpinių rezultatų pristatymuose	2018.08.14	2018.10.16
	▪ Vidinis testavimas ir klaidų taisymas ▪ Vidinio testavimo ataskaitos parengimas ▪ Testavimo scenarijų rengimas	▪ Atliktas vidinis testavimas ir klaidų taisymas ▪ Parengta vidinio testavimo ataskaita	Dalyvavimas rengiant testavimo aplinką	2018.10.16	2018.10.31

Etapas	Darbai	Pateiktys / rezultatai	Perkančiosios organizacijos indėlis	Planuojama pradžia	Planuojama pabaiga
	▪ Sistemos versijos diegimas testavimo aplinkoje	▪ Sistemos versija įdiegta testavimo aplinkoje ▪ Parengti galutiniai testavimo scenarijai			
Testavimas	▪ Parengiamos Sistemos naudotojų ir administratorių instrukcijos ▪ Vykdomas galutinis Sistemos testavimas ▪ Vykdomas Sistemos saugumo testavimas ▪ Atliekamas integracinis testavimas ▪ Atliekami testavimo metu išaiškėjusių klaidų taisymai.	▪ Sistemos naudotojų ir administratorių instrukcijos, parengtos pagal Sistemos funkcines sritis ir atskirų naudotojų grupių atliekamas funkcijas ▪ Eksploatacijai parengta Sistema, atitinkanti testavimo plane nurodytus testavimo priėmimo kriterijus ▪ Parengta galutinio testavimo ataskaita.	Testavimo veiklų vykdymas	2018.11.01	2019.01.01
Bandomoji eksploatacija	▪ Parengiama Sistemos gamybinė aplinka; ▪ Vykdomas duomenų surinkimas, pagal kūrimo etapo metu sukurtas integracijas; ▪ Atliekama duomenų analizė; ▪ Suformuojamos suvestinės ataskaitos; ▪ Suarchyvuojami duomenys; ▪ Atnaujinama Sistemos dokumentacija. ▪ Teikiamos konsultacijos Sistemos eksploatacijos klausimais, reaguojama į eksploatacijos metu nustatytus defektus. ▪ Šalinami bandomosios eksploatacijos metu nustatyti defektai. ▪ Parengiama bandomosios eksploatacijos rezultatų ataskaita.	▪ Parengta Sistemos gamybinė aplinka, atlikti naudotojų teisių nustatymai; ▪ Atliktas bandomasis duomenų surinkimas, sukurtos ir išbandytos duomenų surinkimo taisyklės, pasinaudota duomenų apdorojimo būdais, išbandytos duomenų paieškos galimybės, tinklo topologinio žemėlapio atvaizdavimo būdai ir atvaizdo valdymo galimybės, atlikta duomenų analizė, parengtos bent 5 skirtingos ataskaitos, išbandyti administravimo ir pranešimų komponentai. ▪ Pašalinti bandomosios eksploatacijos metu nustatyti defektai ▪ Atnaujinta Sistemos dokumentacija ▪ Bandomosios eksploatacijos rezultatų ataskaita ▪ Sistema parengta naudojimui.	Bandomosios eksploatacijos vykdymas	2019.01.02	2019.03.01

Pasibaigus pagrindiniams Paslaugų teikimo darbams, Paslaugų teikėjas teiks garantinio aptarnavimo paslaugas. Šios paslaugos bus teikiamos 36 mėn., laikantis TS pateikiamų reikalavimų. Garantinė priežiūra bus vykdoma pagal suderintą garantinės priežiūros reglamentą. Garantinės priežiūros metu bus teikiamos

mėnesinės garantinės priežiūros ataskaitos bei esant poreikiui atnaujinama Sistemos dokumentacija ir programinė įranga.

Sudarant detalų Paslaugų teikimo grafiką, darbų trukmės ir pradžios bei pabaigos datos nustatytos remiantis Paslaugų pirkimo techninės specifikacijos reikalavimais. Darbų trukmės, kurios Paslaugų pirkimo techninėje specifikacijoje nebuvo nurodytos, sudarant Paslaugų teikimo grafiką, įvertintos ekspertiniu būdu, atsižvelgiant į sukauptą panašių projektų įgyvendinimo patirtį.

Numatant darbų vykdymo laikotarpius, **vadovautasi šiomis pagrindinėmis prielaidomis**:

- Paslaugų teikimo laikotarpis – 13 mėnesių nuo Paslaugų teikimo sutarties įsigaliojimo datos;
- numatoma Paslaugų teikimo sutarties vykdymo pradžia turėtų būti 2018 m. vasario 1 d. (darant prielaidą, kad visos pirkimo procedūros bus įvykdytos operatyviai ir neatsiras papildomų kliūčių).

Siekiant kiek įmanoma optimizuoti Projekto darbams reikalingą laiką ir išvengti laikotarpių, per kuriuos būtų neefektyviai naudojami numatyti žmogiškieji ištekliai, rengiant Paslaugų teikimo grafiką, nemažą dalį darbų numatyta vykdyti lygiagrečiai, taip sumažinant bendrą jų trukmę. Taip pat numatyta kiek įmanoma grįsti vienus darbus kitų darbų metu surinkta informacija, priimtais sprendimais ir pan., taip išvengiant galimo darbų dubliavimo.

Įvertinus visus šiuos aspektus, parengtas detalus kalendorinis Paslaugų teikimo grafikas, apimantis numatytus vykdyti darbus, jų rezultatus (rezultatų pateikimą), darbų vykdymo trukmes, pradžios ir pabaigos datas bei darbų tarpusavio priklausomybes, pateikiamas žemiau esančiame paveiksle. Grafike taip pat nurodomi ir už veiklas atsakingi Tiekėjo ekspertai. Ekspertų sutrumpinimų reikšmės:

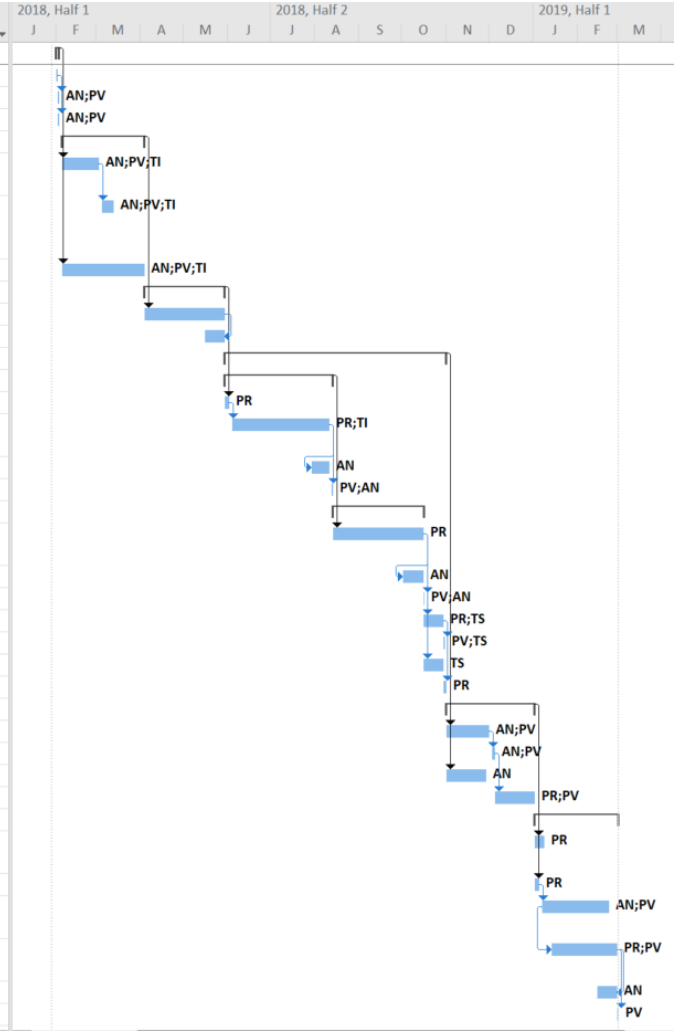
- Projekto vadovas (PV)
- Analitikas (AN)
- Programuotojas (PR)
- Kompiuterių tinklų inžinierius (TI)
- Testuotojas (TS)

Pažymėtina, kad pateiktas grafikas yra preliminarus ir turės būti keičiamas, atsižvelgiant į realius Paslaugų teikimo sutarties įgyvendinimo terminus ir Perkančiosios organizacijos poreikius.

Esant poreikiui perplanuoti parengtą Paslaugų teikimo grafiką ir/ ar žmogiškųjų išteklių paskirstymą dėl identifikuotų vėlavimo rizikų, numatomos šios **galimos alternatyvos**:

- Projekto darbų perstūmimas etapų viduje, visiškai sulygiagretinant darbus, kuriems reikalingi ne tie patys žmogiškieji ištekliai;
- atskiriems darbams numatyto laiko perskirstymas, sutrumpinant nekritiniam darbui numatytą laiką iki minimalios įmanomos ribos;
- žmogiškųjų išteklių perskirstymas, sutelkiant visą dėmesį į darbus, kuriuose galimi vėlavimai.

	WBS	Task Name	Duration	Start	Finish	Resource Names
1	**1**	**Iniciavimas**	**2 days**	**Thu 18-02-01**	**Fri 18-02-02**	
2	1.1	Paslaugų teikimo sutarties pasirašymas	1 day	Thu 18-02-01	Thu 18-02-01	
3	1.2	Įvadinis susirinkimas	1 day	Fri 18-02-02	Fri 18-02-02	AN;PV
4	1.3	Pasiūlyme pateikto kalendorinio projekto grafiko atnaujinimas	1 day	Fri 18-02-02	Fri 18-02-02	AN;PV
5	**2**	**Analizė**	**42 days**	**Mon 18-02-05**	**Tue 18-04-03**	
6	2.1	Detali poreikių ir susijusių sistemų analizė, tikslinami TS nurodyti Sistemos reikalavimai	20 days	Mon 18-02-05	Fri 18-03-02	AN;PV;TI
7	2.2	Reikalavimų Sistemos kompiuterinei techninei ir sisteminei programinei įrangai specifikavimas. Planuojamos naudoti PĮ pagrindimas	6 days	Mon 18-03-05	Mon 18-03-12	AN;PV;TI
8	2.3	Sistemos analizės dokumentacijos rengimas ir derinimas	42 days	Mon 18-02-05	Tue 18-04-03	AN;PV;TI
9	**3**	**Projektavimas**	**40 days**	**Wed 18-04-04**	**Tue 18-05-29**	
10	3.1	Sistemos projektavimo dokumentacijos parengimas ir derinimas	40 days	Wed 18-04-04	Tue 18-05-29	
11	3.2	Sistemos testavimo plano rengimas ir derinimas	10 days	Wed 18-05-16	Tue 18-05-29	
12	**4**	**Kūrimas**	**111 days**	**Wed 18-05-30**	**Wed 18-10-31**	
13	**4.1**	**1 iteracija**	**54 days**	**Wed 18-05-30**	**Mon 18-08-13**	**AN;AR**
14	4.1.1	Kūrimo aplinkos parengimas	3 days	Wed 18-05-30	Fri 18-06-01	PR
15	4.1.2	Duomenų surinkimo ir apdorojimo, pagrindinių tinklo topologinio žemėlapio bei duomenų paieškos funkcionalumų programavimas	50 days	Mon 18-06-04	Fri 18-08-10	PR;TI
16	4.1.3	Sistemos analizės ir projektavimo dokumentacijos tikslinimas	10 days	Mon 18-07-30	Fri 18-08-10	AN
17	4.1.4	Tarpinis rezultatų pristatymas	1 day	Mon 18-08-13	Mon 18-08-13	PV;AN
18	**4.2**	**2 iteracija**	**46 days**	**Tue 18-08-14**	**Tue 18-10-16**	
19	4.2.1	Likusių topologinio žemėlapio, duomenų analizės, grėsmių aptikimo ir administravimo funkcionalumų programavimas	45 days	Tue 18-08-14	Mon 18-10-15	PR
20	4.2.2	Sistemos analizės ir projektavimo dokumentacijos tikslinimas	10 days	Tue 18-10-02	Mon 18-10-15	AN
21	4.2.3	Tarpinis rezultatų pristatymas	1 day	Tue 18-10-16	Tue 18-10-16	PV;AN
22	4.3	Vidinis testavimas ir klaidų taisymas	10 days	Tue 18-10-16	Mon 18-10-29	PR;TS
23	4.4	Vidinio testavimo ataskaitos parengimas	1 day	Tue 18-10-30	Tue 18-10-30	PV;TS
24	4.5	Testavimo scenarijų rengimas	10 days	Tue 18-10-16	Mon 18-10-29	TS
25	4.6	Sistemos versijos diegimas testavimo aplinkoje	2 days	Tue 18-10-30	Wed 18-10-31	PR
26	**5**	**Testavimas**	**44 days**	**Thu 18-11-01**	**Tue 19-01-01**	
27	5.1	Sistemos funkcionalumų ir integracijų testavimas	22 days	Thu 18-11-01	Fri 18-11-30	AN;PV
28	5.2	Sistemos saugumo testavimas	2 days	Mon 18-12-03	Tue 18-12-04	AN;PV
29	5.3	Naudotojų ir administratorių instrukcijų rengimas	20 days	Thu 18-11-01	Wed 18-11-28	AN
30	5.4	Klaidų taisymas ir pakartotinis testavimas	20 days	Wed 18-12-05	Tue 19-01-01	PR;PV
31	**6**	**Bandomoji eksploatacija**	**43 days**	**Wed 19-01-02**	**Fri 19-03-01**	
32	6.1	Sistemos diegimo (pakartotinio panaudojimo) ir konfigūravimo instrukcijos rengimas	5 days	Wed 19-01-02	Tue 19-01-08	PR
33	6.2	Gamybinės aplinkos parengimas	3 days	Wed 19-01-02	Fri 19-01-04	PR
34	6.3	Vykdoma Sistemos bandomoji eksploatacija (duomenų surinkimas, analizė, ataskaitos, archyvavimas)	35 days	Mon 19-01-07	Fri 19-02-22	AN;PV
35	6.4	Šalinami bandomosios eksploatacijos metu nustatyti defektai ir teikiamos konsultacijos	34 days	Mon 19-01-14	Thu 19-02-28	PR;PV
36	6.5	Sistemos analizės ir projektavimo dokumentacijos atnaujinimas	10 days	Fri 19-02-15	Thu 19-02-28	AN
37	6.6	Bandomosios eksploatacijos ataskaitos rengimas	1 day	Fri 19-03-01	Fri 19-03-01	PV



37 paveikslas. Paslaugų teikimo grafikas

4.3 Žmogiškųjų išteklių valdymas

Atsižvelgiant į Paslaugų pirkimo dokumentuose keliamus reikalavimus bei įgyvendinamo Projekto specifiką, 4.2 skyriuje aprašytų darbų vykdymui suburta Paslaugų teikėjo ekspertų komanda, kuri dalyvaus siekiant kokybiškai ir laiku pasiekti Projekto tikslus ir įgyvendinti konkrečius Paslaugoms keliamus uždavinius. Į Projektą įtraukti ekspertai turi reikiamą kvalifikaciją ir komandinio darbo patirtį vykdant didelius IT srities projektus, taip pat nuolat dalyvauja kompleksinių sprendimų projektų vykdymo grupėse.

Siekiant išvengti dubliavimo ir į visus darbus numatyta įtraukti tik tuos ekspertus, kurių dalyvavimas tuose darbuose yra būtinas, iš anksto aiškiai apibrėžtos pagrindinės kiekvieno eksperto funkcijos ir atsakomybės. Numatyti Paslaugų teikėjo ekspertai, jų vaidmuo, pagrindinės funkcijos ir atsakomybės Projekte nurodyti lentelėje žemiau.

14 lentelė. Paslaugų teikėjo komanda

Eil. Nr.	Ekspertas	Vaidmuo	Pagrindinės funkcijos ir atsakomybės
Pagrindiniai ekspertai			
1.	Skirmantas Šermukšnis	Paslaugų teikimo vadovas (Projekto vadovas)	▪ Vadovavimas Paslaugų teikėjo darbo grupei, bendras Paslaugų teikimo darbų koordinavimas ir kontrolė; ▪ darbų apimties ir rezultatų derinimas su Perkančiąja organizacija; ▪ darbinių susitikimų organizavimas ir Perkančiosios organizacijos informavimas apie Projekto eigą; ▪ Projekto rizikų valdymas; ▪ Projekto pakeitimų valdymas; ▪ Projekto įgyvendinimo strategijos parengimas ir projekto įgyvendinimo veiklų analizė.
2.	Ramunė Braukaitė	Analitikė	▪ Perkančiosios organizacijos poreikių išsiaiškinimas, analizė; ▪ Programavimo užduočių apibrėžimas ir paskirstymas programuotojams; ▪ Analizės ir projektavimo dokumentacijos rengimas; ▪ Kitos projektinės dokumentacijos rengimas; ▪ Naudotojų ir administratorių vadovų rengimas; ▪ Dalyvavimas testavimo ir bandomosios eksploatacijos veiklose.
3.	Aurimas Savickas Adomas Jatužis	Programuotojas	▪ Programinės įrangos funkcijų ir procesų konstravimas; ▪ Testinės ir gamybinės aplinkos rengimas; ▪ Programinės įrangos diegimas.
4.	Aurimas Savickas	Duomenų bazių programavimo specialistas	▪ duomenų bazių programavimas; ▪ duomenų bazių diegimas; ▪ Testinės ir gamybinės aplinkos rengimas.
5.	Giedrius Trapkauskas	Kompiuterių tinklų inžinierius	▪ BGP ir RIPE duomenų surinkimo ir apdorojimo sprendimų parinkimas ir konsultavimas susijusiais klausimais.
6.	Žydrūnas Skardžius	Informacinės sistemos testavimo specialistas	▪ Vidinių testavimų valdymas; ▪ Priėmimo testavimo organizavimas ir valdymas. ▪ Programinės įrangos funkcijų ir procesų testavimas; ▪ Testavimų dokumentacijos rengimas; ▪ Testavimų metu nustatytų klaidų ir jų būsenų fiksavimas tam skirtame žurnale.

Detaliai Paslaugų teikėjo ekspertų funkcijų sąsajos su konkrečiais Projekto darbais parodytos atsakomybių (angl. *Responsible, Accountable, Consulted, Informed* – *RACI*) matricoje, pateikiamoje lentelėje žemiau. Sudarant šią matricą, nustatyti už kiekvieną darbą atsakingi, atskaitingi ir kiti juose dalyvaujantys ekspertai. Siekiant optimaliai paskirstyti Projekto veikloms numatytus žmogiškuosius išteklius ir užtikrinti aiškią Projekto įgyvendinimo kontrolę, už kiekvieną darbą paskirta po atsakingą ir atskaitingą asmenį, o atsakomybės už vienu metu vykdomus darbus, kur tai įmanoma, numatytos skirtingiems ekspertams.

Pateiktoje lentelėje naudojami šie žymėjimai:

AT - **AT**sakingas (angl. *responsible*) – ekspertas, kuris yra atsakingas už tinkamą konkretaus darbo įvykdymą ir galutinį jo rezultatą;

A - **A**tskaitingas (angl. *accountable*) – ekspertas, kuris yra atskaitingas Perkančiajai organizacijai už konkretaus darbo rezultatus (jų peržiūrėjimą tvirtinimą ir pateikimą Perkančiajai organizacijai);

K - **K**onsultuojantis (angl. *consulted*) – ekspertas, kuris dalyvauja Projekto darbuose, atlikdamas konkrečias jam pavestas užduotis ir (ar) teikdamas tam tikros srities pagalbą;

I - **I**nformuojamas (angl. *informed*) – ekspertas, kuris tiesiogiai nedalyvauja konkrečiame darbe, bet yra informuojamas apie jo rezultatus ir (ar) bendrą Projekto eigą.

15 lentelė. Projekto atsakomybių matrica

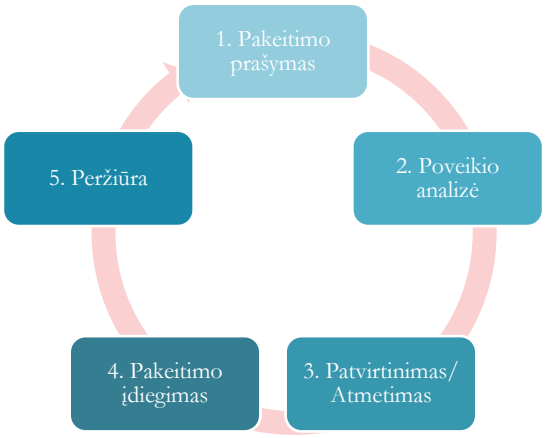
Darbai	Paslaugų teikimo vadovas	Analitikas	Programuotojas	Duomenų bazių programavimo specialistas	Kompiuterių tinklų inžinierius	Testavimo specialistas
Analizė ir projektavimas						
Detali poreikių ir susijusių sistemų analizė, tikslinami TS nurodyti Sistemos reikalavimai	A	AT	I	I	K	
Reikalavimų Sistemos kompiuterinei techninei ir sisteminei programinei įrangai specifikavimas. Planuojamos naudoti PĮ pagrindimas	A	AT	I	I	K	
Sistemos analizės ir projektavimo dokumentacijos rengimas ir derinimas	A	AT	I	I	K	
Sistemos testavimo plano rengimas ir derinimas	A	I				AT
Kūrimas						
Kūrimo aplinkos parengimas	A		AT	K		
Funkcionalumų programavimas	A	I	AT	AT		
Sistemos analizės ir projektavimo dokumentacijos tikslinimas	A	AT				
Tarpinis rezultatų pristatymas	A	AT				
Vidinis testavimas ir klaidų taisymas	A	I	AT			AT
Vidinio testavimo ataskaitos parengimas	A	I				AT
Testavimo scenarijų rengimas	A	I				AT
Diegimas testavimo aplinkoje	A	I	AT		K	
Testavimas						
Sistemos funkcionalumų ir integracijų testavimas		K	I		K	
Sistemos saugumo testavimas	A	K	I			AT
Naudotojų ir administratorių instrukcijų rengimas	A	AT				
Klaidų taisymas ir pakartotinis testavimas	A		AT	K		I
Bandomoji eksploatacija						
Sistemos diegimo (pakartotinio panaudojimo) ir konfigūravimo instrukcijos rengimas	A		AT	K	K	
Gamybinės aplinkos parengimas	A		AT		K	
Vykdoma Sistemos bandomoji eksploatacija (duomenų surinkimas, analizė, ataskaitos, archyvavimas)		K			K	K
Šalinami bandomosios eksploatacijos metu nustatyti defektai ir teikiamos konsultacijos	A	K	AT	K	K	

Darbai	Paslaugų teikimo vadovas	Analitikas	Programuotojas	Duomenų bazių programavimo specialistas	Kompiuterių tinklų inžinierius	Testavimo specialistas
Sistemos analizės ir projektavimo dokumentacijos atnaujinimas	A	AT				
Bandomosios eksploatacijos ataskaitos rengimas	A/AT					

Siekiant užtikrinti savalaikį Projekto įgyvendinimą, esant veiklų vėlavimams ar kitoms nenumatytoms aplinkybėms, lemiančioms Paslaugų teikėjo komandos nariams tenkančio darbo krūvio išaugimą, į suplanuotų darbų vykdymą numatoma įtraukti papildomus ekspertus, reikalingus konkrečių darbų įvykdymui. Taip pat numatoma, kad jei dėl tam tikrų nuo Paslaugų teikėjo nepriklausančių aplinkybių kuris nors pagrindinis ekspertas negalėtų tęsti darbo Projekte, bus iš anksto numatytas jį galintis pakeisti (ne žemesnę kvalifikaciją ir darbo patirtį turintis) ekspertas.

4.4 Pakeitimų valdymas

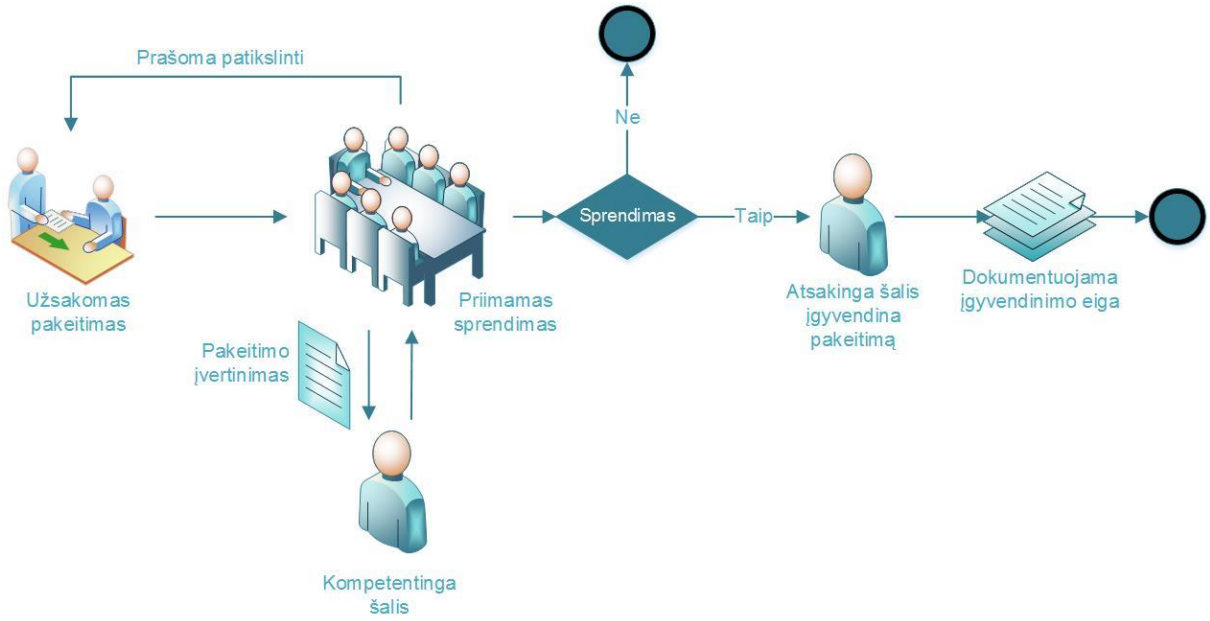
Pakeitimams valdyti taikome pripažintą ITIL pakeitimų valdymo (angl. *change management*) procesą. Jį sudarantys žingsniai pavaizduoti žemiau esančiame paveiksle.



38 paveikslas. Pakeitimų valdymo procesas pagal ITIL

1. Pakeitimo užsakymas – kiekvienas pakeitimas užsakomas pildant pateikimo užsakymo formą;
2. Poveikio analizė – išanalizuojamas pakeitimo neigiamas ir teigiamas poveikiai, bei įtaka veiklai neįgyvendinus pakeitimo;
3. Patvirtinimas/ atmetimas – pakeitimo užsakymas patvirtinamas arba atmetamas;
4. Pakeitimo įdiegimas – organizacijoje, jos veikloje ar sistemoje įdiegiamas/ įgyvendinamas pakeitimas;
5. Peržiūra – peržiūrimi pakeitimo įgyvendinimo rezultatai.

Pakeitimų valdymas ir įgyvendinamas pavaizduotas žemiau esančioje schemoje.



39 paveikslas. Pakeitimo įgyvendinimas

Kilus pakeitimo poreikiui, pateikiamas dokumentuotas pakeitimo užsakymas užpildant pakeitimo užsakymo formą, o užsakytas pakeitimas registruojamas į pakeitimų žurnalą.

Pakeitimai gali būti identifikuojami visuose organizacinės struktūros lygmenyse (žr. 5.1.1 poskyrį).

Sėkmingai užregistravus ir priėmus pakeitimo užsakymą, analizuojama pakeitimo įtaka Projekto eigai ir rezultatams. Įvertinus įtaką, pakeitimas priskiriamas kategorijai. Registruojami pakeitimai skirstomi į šias kategorijas, atsižvelgiant į pakeitimų svarbą, aktualumą ir poreikį:

16 lentelė. Pakeitimų kategorijos

Kategorija	Aprašymas
Standartiniai pakeitimai	Pakeitimai, kurie nekelia rizikos kokybiškam informacinių technologijų paslaugų teikimui arba visos informacinių technologijų infrastruktūros veikimui.
Skubūs pakeitimai	Pakeitimai, kurie skirti aukščiausio prioriteto sutrikimams arba problemoms šalinti ir kuriems reikia ypatingos įvertinimo, patvirtinimo ir atlikimo skubos, taip pat avariniai pakeitimai.
Plėtros pakeitimai	Pakeitimai, kai kuriamos arba modernizuojamos informacinių technologijų paslaugos ir su tuo susiję jų atlikimo veiksmai nėra visiškai aiškūs, o pakeitimų atlikimas yra susijęs su tam tikra rizika informacinių technologijų paslaugų teikimui arba visos informacinių technologijų infrastruktūros veikimui

Esant poreikiui Projekto valdymo grupė prieš priimdama sprendimą dėl pakeitimo, gali kreiptis į kompetentingas Projekto vidines arba išorines šalis dėl pakeitimo užsakymo įvertinimo. Tokiu atveju sprendimai dėl pakeitimo priimami tik gavus įvertinimą.

Skubiems pakeitimams prieš patvirtinimą rekomenduotina parengti ir pakeitimo įgyvendinimo veiksmų planą. Standartiniams ir plėtros pakeitimams veiksmų planas gali būti sudaromas po patvirtinimo. Pakeitimo poreikis tvirtinamas Projekto valdymo grupėje. Po patvirtinimo, jei reikalinga, patikslinama pakeitimo kategorija bei veiksmų planas, jei jis buvo

parengtas. Patvirtinus pakeitimo poreikį, priskiriamas pakeitimo įgyvendinimą kontroliuojantis specialistas. Įgyvendinus pakeitimą paskirtas specialistas atlieka detalią pakeitimo peržiūrą ir patvirtina pakeitimo atlikimą.

Pakeitimai gali turėti šias būsenas, kurios numatytos Informacinių technologijų paslaugų valdymo metodikos, patvirtintos Informacinės visuomenės plėtros komiteto prie Susisiekimo ministerijos direktoriaus 2013 m. birželio 19 d. įsakymu Nr. T-83 „Dėl informacinių technologijų paslaugų valdymo metodikos patvirtinimo“, 5 priede:

17 lentelė. Pakeitimų būsenos

Būsena	Aprašymas
Užregistruotas	Ši būsena rodo, kad pakeitimo užsakymas buvo sėkmingai užregistruotas.
Priimtas	Ši būsena rodo, kad pateiktas pakeitimo užsakymas yra tinkamai suformuotas ir bus vykdomi tolesni jo apdorojimo veiksmai. Priešingu atveju užsakymai grąžinami užsakymo teikėjui patikslinti.
Patvirtintas	Ši būsena rodo, kad pateiktas pakeitimo užsakymas buvo patvirtintas Projekto valdymo grupėje (arba Projekto priežiūros komitete).
Atmestas	Ši būsena rodo, kad pateiktas keitimo užsakymas Projekto valdymo grupėje (arba Projekto priežiūros komitete) buvo atmestas.
Atliktas	Ši būsena rodo, kad pateiktas pakeitimas buvo sėkmingai atliktas.
Neatliktas	Ši būsena rodo, kad pateikto pakeitimas įgyvendinimas nebuvo sėkmingas.
Uždarytas	Ši būsena rodo, kad pakeitimo atlikimo procedūra yra sėkmingai užbaigta.

Esant nepatvirtintiems užsakytiems pakeitimams turintiems būseną „Priimtas“ arba vykdant patvirtintus pakeitimus, Projekto valdymo grupės susitikimuose turi būti svarstomas, prieš susitikimą atnaujintas, Projekto pakeitimų žurnalas.

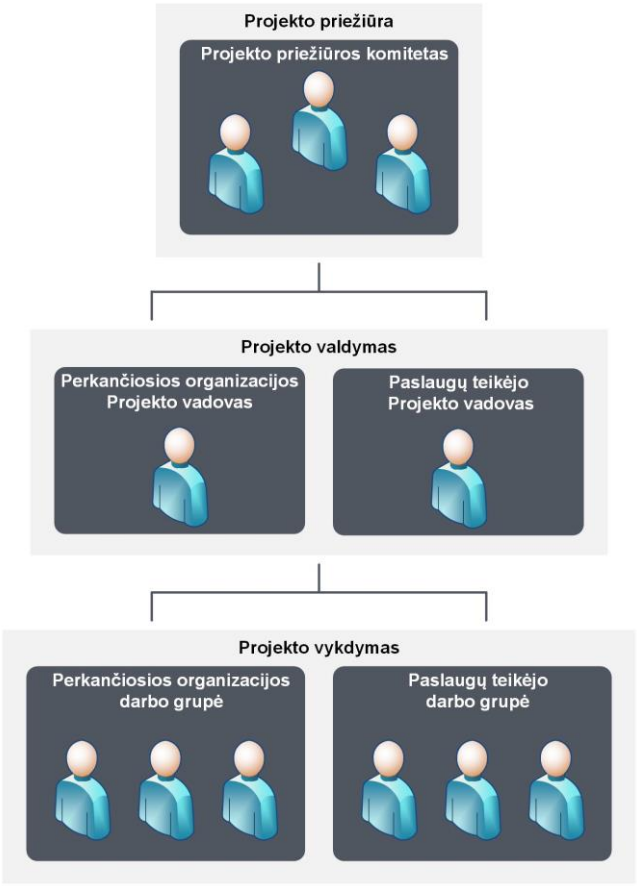
5 Koordinavimo ir kokybės valdymo planas (P_4)

Šiame poskyryje pateiktas aiškus, detalus ir racionalus koordinavimo ir kokybės valdymo planas, susietas su kitomis pasiūlymo dalimis. Plane aiškiai aprašyta, kaip bus užtikrinama kiekvieno paslaugų teikimo plane, išskirto darbų rezultatų kokybė. Taip pat aiškiai aprašytas Perkančiosios organizacijos vaidmuo koordinavimo ir kokybės valdymo procese, aprašyta logistika bei sprendimų derinimo procedūros.

5.1 Koordinavimo planas

5.1.1 Projekto organizacinė struktūra

Remiantis sukaupta panašių projektų įgyvendinimo patirtimi bei 4.1 skyriuje aprašytos PMBOK metodikos principais, sklandžiam Projekto įgyvendinimui užtikrinti rekomenduojama naudoti trijų lygių organizacinę struktūrą, t. y. sudaryti už aukščiausio lygmens (strateginę) Projekto kontrolę atsakingą Projekto priežiūros komitetą, paskirti už bendrą Projekto valdymą ir koordinavimą atsakingus projekto vadovus ir suburti už Projekto darbų įgyvendinimą atsakingas Projekto darbo grupes. Principinė šios organizacinės struktūros schema pateikta paveiksle žemiau.



40 paveikslas. Projekto organizacinė struktūra

Žemiau esančioje lentelėje pateikiamas kiekvieno organizacinės struktūros elemento aprašymas, apimantis pasiūlymus dėl jų funkcijų ir atsakomybių Projekte.

18 lentelė. Projekto dalyvių funkcijos ir atsakomybės

Eil. Nr.	Ekspertas	Vaidmuo	Pagrindinės funkcijos ir atsakomybės
1.	Projekto priežiūros komitetas	Strateginis Projekto valdymas (priežiūra)	▪ Projekto eigos stebėsena ir kontrolė; ▪ svarbiausių Projekto sprendimų priėmimas; ▪ pagrindinių Projekto vykdymo metu parengtų rezultatų tvirtinimas; ▪ Perkančiosios organizacijos įsipareigojimų vykdymas Projekto metu; ▪ klausimų, kurie negali būti išspręsti žemesniuose Projekto valdymo organuose, sprendimas.
2.	Perkančiosios organizacijos projekto vadovas	Perkančiosios organizacijos darbo grupės veiklos valdymas. Projekto vykdymo priežiūra. Tarpininkavimas tarp Projekto priežiūros komiteto bei Perkančiosios organizacijos darbo grupės	▪ Atskaitomybė už Projekto įgyvendinimą Projekto priežiūros komitetui; ▪ bendras Projekto veiklų planavimas ir koordinavimas; ▪ kasdienis Projekto valdymas ir operatyvių sprendimų, susijusių su Projekto įgyvendinimu, priėmimas; ▪ Perkančiosios organizacijos darbo grupės veiklos planavimas ir koordinavimas; ▪ Projekto rezultatų priėmimas ir teikimas Projekto priežiūros komitetui susipažinti ir/ ar tvirtinti; ▪ pastabų dėl Paslaugų teikėjo pateiktuose dokumentuose rastų trūkumų ir pasiūlymų dėl jų taisymo teikimas.
3.	Paslaugų teikėjo projekto vadovas	Paslaugų teikėjo darbo grupės veiklos valdymas. Projekto vykdymo koordinavimas ir valdymas (teikiamų paslaugų apimtyje)	▪ Sutartyje nustatytų Paslaugų teikėjo įsipareigojimų vykdymas; ▪ bendras Projekto veiklų planavimas ir organizavimas, kasdienis Projekto valdymas ir sprendimų, susijusių su Projekto įgyvendinimu, priėmimas (teikiamų Paslaugų apimtyje); ▪ susitikimų inicijavimas ir organizavimas; ▪ klausimų, susijusių su Projekto įgyvendinimu, sprendimas, problemų iškėlimas, pasiūlymų dėl problemų šalinimo teikimas; ▪ Perkančiosios organizacijos informavimas apie Projekto eigą, iškylančias problemas, rizikas, tolimesnius uždavinius; ▪ atsiskaitymas Perkančiajai organizacijai už teikiamas paslaugas ir galutinius jų rezultatus.
4.	Perkančiosios organizacijos darbo grupė	Paslaugų teikėjo parengtų rezultatų peržiūra ir kontrolė. Aktualios informacijos ir metodinės pagalbos teikimas.	▪ Dalyvavimas darbiniuose susitikimuose ir Paslaugų teikėjui reikalingos informacijos teikimas; ▪ pastabų ir pasiūlymų dėl Paslaugų teikėjo darbo grupės parengtų dokumentų ir kitų rezultatų teikimas.
5.	Paslaugų teikėjo darbo grupė	Paslaugų įgyvendinimo plane numatytų darbų vykdymas	▪ numatytų Projekto darbų ir 4.3 skyriuje aprašytų funkcijų vykdymas; ▪ dalyvavimas Projekto darbo grupių susitikimuose.

5.1.2 Komunikacijos valdymo planas

Efektyvi suinteresuotų šalių komunikacija, leidžianti suderinti įvairius organizacinius skirtumus, skirtingus kompetencijų lygius, perspektyvas ir interesus, siejamus su projekto įgyvendinimu ir galutiniu jo rezultatu, yra vienas svarbiausių bet kokio projekto sėkmę lemiančių veiksnių. Atsižvelgiant į tai, iš anksto numatytos efektyvią komunikaciją tarp Paslaugų teikėjo ir Perkančiosios organizacijos užtikrinančios komunikacijos valdymo priemonės. Trumpi jų aprašymai, apimantys kiekvienos komunikacijos priemonės naudojimo aplinkybes, pateikiami žemiau esančioje lentelėje.

19 lentelė. Komunikacijos valdymo priemonės

Eil. Nr.	Komunikacijos priemonė	Naudojimas
1.	Telefonas	Telefonu bus keičiamasi nesvarbia informacija: derinami susitikimų laikai ir panašaus pobūdžio informacija, kurios nebūtina fiksuoti ir išsaugoti.
2.	Elektroninis paštas	Elektroniniu paštu bus derinami visi rengiami dokumentai, siunčiama visa svarbi, išliekamąją vertę turinti informacija.
3.	Susitikimai	Paslaugų teikėjas viso Projekto metu pagal poreikį organizuos Projekto darbo grupių susitikimus, kurių metu bus sprendžiamos pagrindinės Projekto įgyvendinimo problemos, tvirtinami Paslaugų teikimo rezultatai. Darbo grupių susitikimuose bus svarstomi aktualūs Projekto įgyvendinimo klausimai, planuojamos veiklos ir aptariami jų rezultatai. Paslaugų teikimo metu Paslaugų teikėjo ekspertai taip pat neatlygintinai bendradarbiaus ir dalyvaus susitikimuose bei darbo grupėse su Perkančiąja organizacija ir/ ar kitomis suinteresuotomis šalimis.
4.	Ataskaitos ir pristatymai	Paslaugų teikėjas parengs Paslaugų teikimo planą, į kurį įeis Paslaugų teikimo grafikas ir darbų įgyvendinimo aprašymai, taip pat sutarties įgyvendinimo, rizikų ir kokybės valdymo bei komunikavimo planai. Paslaugų teikėjas pateiks galutinę Paslaugų teikimo ataskaitą, kurioje bus aprašytos atliktos veiklos, pašalintos ir po Projekto užbaigimo aktualios rizikos ir kita aktuali informacija.
5.	Dokumentacija	Siekiant kokybiško Projekto įgyvendinimo, visa svarbiausia informacija bus dokumentuojama, o rengiama dokumentacija derinama su atsakingais Perkančiosios organizacijos specialistais. Visi Projekto metu rengiami dokumentai bus prieinami Projektą įgyvendinantiems asmenims, juose pateikiama informacija, esant poreikiui, atnaujinama.

Nurodytos komunikacijos priemonės bus naudojamos keičiantis visa su Projekto įgyvendinimu susijusia informacija. Šių priemonių naudojimo dažnumas ir kiti svarbūs aspektai bus aptarti ir suderinti su Perkančiąja organizacija.

5.2 Kokybės valdymas

Projekto kokybės valdymas apima procesus, leidžiančius užtikrinti, kad projektas patenkintų poreikius, dėl kurių jis buvo pradėtas vykdyti. Tai visi valdymo veiksmai, susiję su kokybės politika, tikslais bei atsakomybe, kurie yra vykdomi tam skirtomis kokybės užtikrinimo priemonėmis. Siekiant didžiausios teikiamų Paslaugų kokybės, Projekto įgyvendinimo metu numatoma naudoti dviejų tipų **kokybės užtikrinimo priemones**:

- technines priemones, tokias kaip ekspertų darbo grupės turima patirtis, kompetencijos ir specializacija;
- administracines priemones, tokias, kaip Projekto įgyvendinimo stebėsena ir įvairios atskaitomybės tvarkos ir procedūros.

Paslaugų teikėjo pusėje bus taikomos griežtos kokybės užtikrinimo procedūros, atitinkančios LST EN ISO 9001 kokybės vadybos standarto reikalavimus:

- Paslaugų teikėjo ekspertų komanda bus sudaryta iš kvalifikuotų ir panašių projektų įgyvendinimo patirtį turinčių specialistų;
- Projekto vykdymui ir valdymui bus naudojamos visuotinai pripažintos bei plačiai taikomos metodikos, kurias Paslaugų teikėjo specialistai jau yra praktiškai pritaikę anksčiau sėkmingai įgyvendintuose projektuose;
- visos Projekto įgyvendinimo veiklos bus vykdomos vadovaujantis parengtu ir su Perkančiąja organizacija suderintu Paslaugų teikimo planu ir Paslaugų teikimo grafiku. Už konkrečius numatytus darbų rezultatus bus atsakingi Paslaugų teikėjo reikiamą kvalifikaciją ir patirtį turintys ekspertai;

- Paslaugų teikėjo projekto vadovui bus priskirta aiški atsakomybė už bendrą Projekto rezultatų kokybę (atskaitomybė Perkančiajai organizacijai už visas vykdomas veiklas ir jų rezultatus);
- visi Paslaugų teikėjo rengiami dokumentai ir kiti rezultatai bus nuolatos aptariami su Perkančiąja organizacija ir rengiami/ peržiūrimi bent 2 Paslaugų teikėjo ekspertų, o galutiniai Paslaugų teikimo rezultatai derinami laikantis iš anksto aptartos ir suderintos tvarkos.

Siekiant užtikrinti Projekto rezultatų (rengiamų dokumentų, kurie įvardinti 4.2 skyriuje 13 lentelėje) atitikimą nustatytiems kokybės kriterijams, visi parengti dokumentai bus derinami su Perkančiąja organizacija tokia tvarka:

- Paslaugų teikėjo parengti dokumentai pateikiami elektronine forma (MS Word (*.docx) formatu) su Perkančiąja organizacija suderintame Paslaugų teikimo grafike nustatytais terminais;
- Pateiktus dokumentus Perkančioji organizacija peržiūri ir įvertina per su Paslaugų teikėjų suderintą terminą (numatoma – per 5 darbo dienas nuo jų pateikimo);
- gavus pastabų dėl pateiktų dokumentų kokybės, parengti dokumentai pataisomi per su Perkančiąja organizacija suderintą terminą (numatoma – per 5 darbo dienas nuo pastabų gavimo);
- jei per nustatytą terminą Paslaugų teikėjo pateiktiems dokumentams pastabų nepateikiama, laikoma, kad pateiktiems rezultatams neprieštaraujama ir jų kokybė atitinka reikalavimus.

Ataskaitų ir kitų dokumentų galutinės versijos Perkančiajai organizacijai bus pateiktos su Perkančiąja organizacija suderintais formatais: elektroniniu parašu pasirašytame dokumente (PDF-LT arba ADOC formatu) ir / arba popieriniu formatu. Visa dokumentacija pateikiama lietuvių kalba.

Sukurtos programinės įrangos kokybė bus užtikrinama įgyvendinant testavimo veiklas. Planuojamo įgyvendinti testavimo strategija aprašoma 5.4 skyriuje

5.3 Problemų valdymas

Nepriklausomai nuo to kaip sklandžiai yra vykdomi projektai, neišvengiamai didelių projektų įgyvendinimo metu yra susiduriama su tam tikromis problemomis. ITIL paslaugų teikimo gyvavimo cikle problemų valdymas yra tolimesnis žingsnis po incidentų (sutrikimų) valdymo. Incidentų valdymas rūpinasi bet kokiais neplanuotais trukdžiais IT paslaugai ar jos kokybės kritimu, o problemų valdymas rūpinasi esmine įvykusio incidento priežastimi.

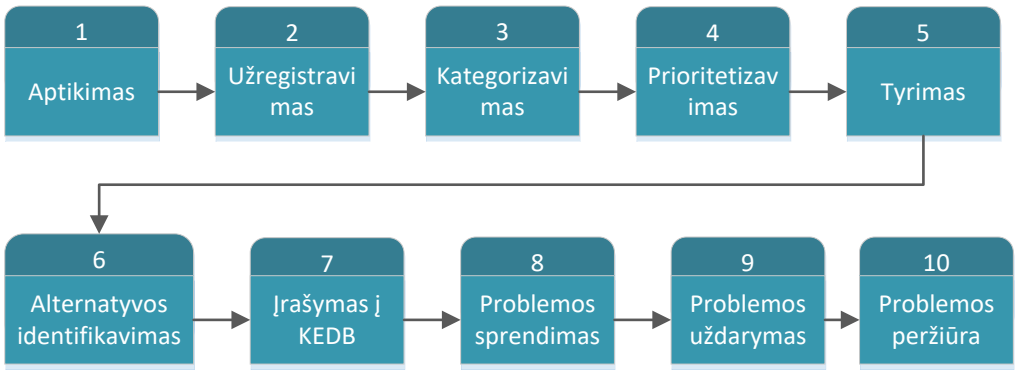
Informacinių technologijų paslaugų valdymo metodikoje, patvirtintoje Informacinės visuomenės plėtros komiteto prie Susisiekimo ministerijos direktoriaus 2013 m. birželio 19 d. įsakymu Nr. T-83 yra pateiktos incidento ir problemos sąvokos:

Incidentas (sutrikimas) – nenumatytas informacinių technologijų paslaugų sutrikimas, informacinių technologijų paslaugų kokybės pablogėjimas arba įvykis, kuris gali sutrikdyti informacinių technologijų paslaugų teikimą;

Problema – vienas didelę įtaką informacinių technologijų paslaugų teikimui turintis sutrikimas arba keli sutrikimai, kuriems būdingi tokie pat požymiai, o priežastis, dėl kurios jie įvyko, nežinoma ir jai išsiaiškinti būtina išsami analizė.

Matome, kad tokio tipo projektų įgyvendinimo metu, nėra tikslinga išskirti šių dviejų procesų ir juos traktuoti kaip vieną. Nes problemos apima incidentus ir išsprendžiant problemą, kartu bus išvengta ir incidentų pasireiškimo. Taigi šiame poskyryje detaliau bus aprašomas tik problemų valdymo procesas.

Problemoms valdyti taikome pripažintą ITIL problemų valdymo (angl. *problem management*) procesą. Šis procesas turi daug žingsnių. Kiekvienas jų yra labai svarbus, siekiant proceso sėkmės ir kokybės. Jį sudarantys žingsniai pavaizduoti žemiau esančiame paveiksle ir aprašyti lentelėje.

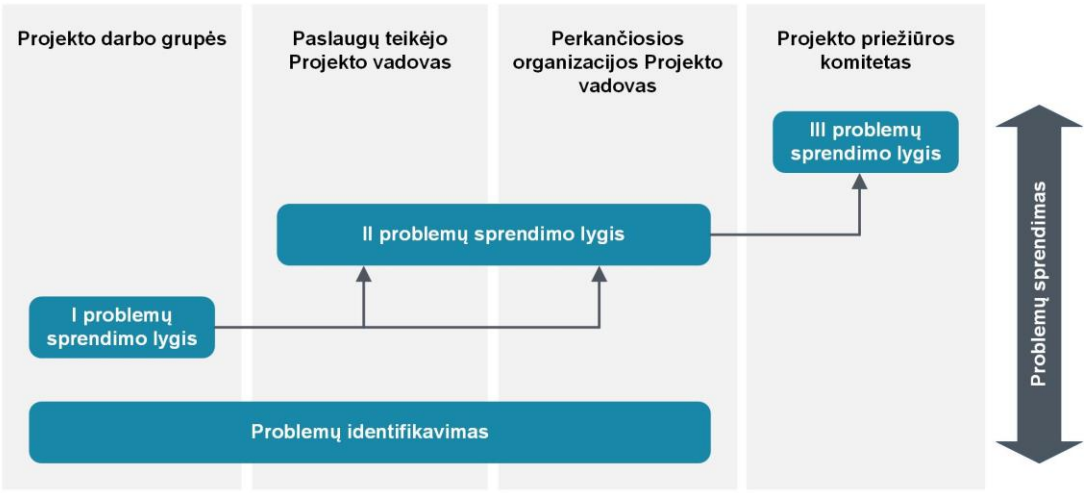


41 paveikslas. ITIL problemų valdymo procesas

20 lentelė. ITIL problemų valdymo proceso žingsniai

Eil. Nr.	Žingsnis	Aprašymas
1.	Aptikti (angl. *detect*)	Pastebima, identifikuojama problema stebint besikartojančius incidentus (sutrikimus);
2.	Užregistruoti (angl. *log*)	Problema yra užregistruojama problemų registre (angl. *problem record*) kartu, pagal galimybes, įrašant su problema susijusius duomenis, tokius, kaip pasireiškimo laikas ir data, susiję incidentai, simptomai, problemos kategorija ir kt.;
3.	Kategorizuoti (angl. *categorize*)	Kategorizuojant problemas atsiranda galimybė jas rūšiuoti. Taip pat padeda sekti reguliariai atsirandančių problemų tendencijas ir jas automatiškai prioritetizuoti;
4.	Prioritetizuoti (angl. *prioritize*)	Problemos prioritetas nustatomas, atsižvelgiant į problemos skubumą bei daromą poveikį vartotojams, organizacijai;
5.	Tyrimas ir tikslus apibūdinimas (angl. *investigating and diagnosing*)	Apima detalesnę problemos analizę;
6.	Identifikuoti alternatyvą probleminei sričiai (angl. *identify a workaround for the problem*)	Alternatyvaus būdo identifikavimas.
7.	Palikti įrašą žinomų klaidų duomenų bazėje (angl. *raise a known error record*)	Identifikavus problemą ir radus alternatyvią veiklą probleminei sričiai palaikyti, ji užregistruojama problemų registre.
8.	Spręsti problemą (angl. *resolve the problem*)	Spręsti problemą (incidentų priežastį) atsiradus galimybei ir išvengti jų pasikartojimo;
9.	Uždaryti problemą (angl. *close the problem*)	Problemos klausimo uždarymas, tai proceso žingsnis, kuris yra vykdomas tada, kai yra įvykdyti visi prieš tai minėti žingsniai. Tačiau pagal ITIL tai dar nėra paskutinis proceso žingsnis;
10.	Peržiūrėti problemą (angl. *review the problem*)	Tai yra paskutinis ITIL problemų valdymo proceso žingsnis, žinomas kaip pagrindinė problemos peržiūra (angl. *major problem review*). Šios peržiūros metu problemų valdymo komanda įvertina problemos dokumentaciją ir pakartotinai identifikuoja kas ir kodėl įvyko ir kaip to būtų galima išvengti ateityje.

Projekto įgyvendinimo metu kylančių problemų sprendimui bus taikoma problemų sprendimo procedūra, apimanti keletą problemų sprendimų lygių. Principinė šios procedūros schema pateikiama paveiksle žemiau.



42 paveikslas. Problemų sprendimo procedūra

Vadovaujantis šia procedūra:

- Paslaugų teikėjo ir Perkančiosios organizacijos darbo grupės bus atsakingos už einamųjų klausimų ir su tuo susijusių problemų sprendimą;
- Projekto vykdymo lygmenyje negalimi išspręsti klausimai bus perduodami Paslaugų teikėjo ir Perkančiosios organizacijos Projekto vadovams;
- itin sudėtingi ar su kertiniais Projekto įgyvendinimo sprendimais susiję klausimai ir problemos bus perduodami Projekto priežiūros komitetui.

5.4 Testavimo strategija

Sistemos testavimas – tai procesas, apjungiantis statinius ir dinaminius veiksmus, susijusius su programinės įrangos kūrimo ciklu, t. y. testavimas yra pradedamas ankstyvoje programinės įrangos kūrimo stadijoje ir baigiamas visiškai baigus kurti programinę įrangą. Testavimas leidžia nustatyti, ar programinė įranga atitinka konkrečius jai keliamus reikalavimus, ar atitinka savo paskirtį bei leidžia išsiaiškinti programinės įrangos defektus. Kitaip tariant programinės įrangos testavimas apima priemones, skirtas programinės įrangos kokybei įvertinti ir užtikrinti. Sistemos testavimo procesą galima suskaidyti į tris pagrindinius etapus:

43 paveikslas. Testavimo etapai

- Pasirengimas testavimui. Šio etapo metu rengiamas testavimo planas, testavimo metodiką bei testavimo scenarijai. Taip pat paruošiama testinė aplinka, kurioje yra vykdomas testavimas.

- Testavimo vykdymas. Šio etapo metu yra atliekamas sistemos testavimas. Kūrimo metu dažniausiai yra testuojami atskiri sistemos moduliai bei funkcijos, o sukūrus visą sistema yra atliekamas bendras visų sistemos modulių ir funkcijų testavimas.
- Testavimo užbaigimas. Šio etapo metu yra įvertinamas testavimo metu gautas rezultatas. Atsižvelgiant į gautus testavimo rezultatus yra nustatomas ar sistema yra pilnai sukurta ir gali būti pradėta naudoti. Jei gauti rezultatai netenkina, atliekamas sistemos tobulinimas ir pakartotinis testavimas.

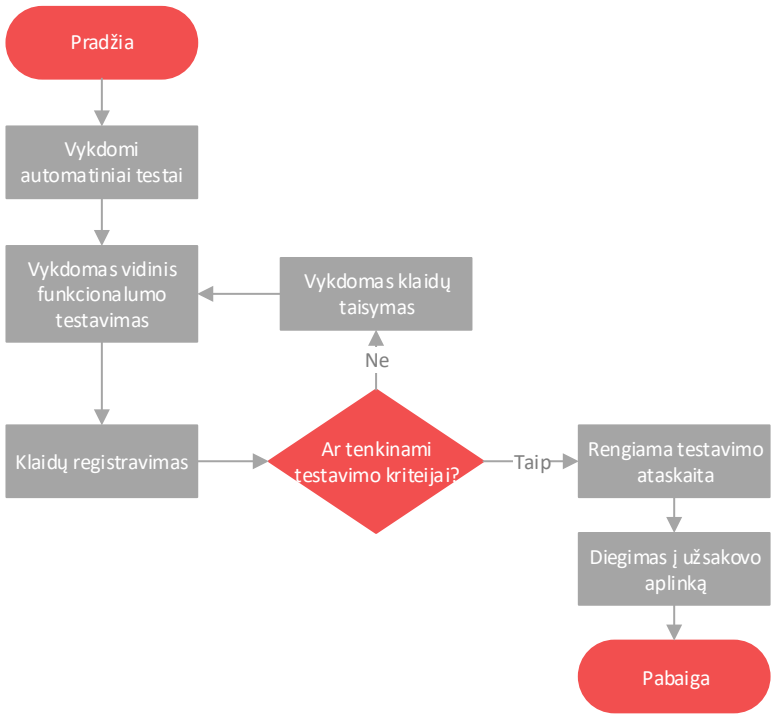
5.4.1 Vidinio testavimo procedūra

Vidinį testavimą sudaro elementų (angl. *Unit tests*), e2e (angl. *End-to-End tests*), vidinio bei išorinio integravimo, našumo ir funkcionalumo testavimas. Elementų, e2e, integravimo ir našumo testavimą atlieka programinės įrangos kūrimo komanda, o funkcionalumo testavimą atlieka testuotojų komanda.

Pasirengimas testavimui:

- Parengiama Paslaugų teikėjo testinė aplinka. Šioje aplinkoje yra vykdomas vidinis sistemos testavimas (angl. *Alpha Test*). Testinė aplinka naudojama vidinio testavimo tikslams. Vykdytojo testinėje aplinkoje yra vykdomi automatiniai testai bei funkcionalumo testai, atliekami testuotojų komandos.

Testavimo vykdymas:



44 paveikslas. Vidinio testavimo vykdymas

- Vykdomi automatiniai testai - komponentų testai (angl. *Unit tests*), integraciniai testai (angl. *integration tests*) bei e2e testai (angl. *End-to-End tests*). Šiuos testus parengia programuotojai kurdami sistemą. Dažniausiai dar prieš kurdami sistemos modulį ar funkciją, programuotojai parašo automatinius testus. Programuotojų sukurti testai yra vykdomi automatiškai kiekvieno sistemos atnaujinimo metu t.y., programuotojui atlikus sistemos pakeitimus ir sudiegus į testinę aplinką. Automatiniai testai yra vykdomi naudojant *Jenkins Continuous Integration* įrankį. Jeigu automatiniai testai praėjo sėkmingai sistema yra sudiegiama į testinę aplinką. Jei automatiniuose testuose buvo aptikta klaidų yra informuojamas programuotojas ir testinė aplinka nėra atnaujinama.

- Sėkmingai įvykdžius automatinius testus testuotojų komanda atlieka papildomą sistemos funkcionalumo testavimą, kurio neapima automatiniai testai. Testuotojai vykdydami testavimą vadovaujasi programuotojų padarytų užduočių bei išspręstų klaidų sąrašu, t.y. tikrinami tik užduotyse nurodyti sistemos funkcionalumai bei ištaisytos klaidos. Papildomai testuotojai įvertina sistemos greitaveiką, patogumą bei dizaino netikslumus, atliekami duomenų įvedimo ir saugojimo funkcijų patikrinimas, kuomet yra tikrinama ar išsaugomi korektiški ir leistini duomenys.

Klaidų registravimas. Testavimo metu testuotojai aptiktas klaidas registruoja klaidų valdymo sistemoje. Užregistruotos klaidos yra priskiriamos programuotojams, kurie ištaiso klaidas ir vėl perduoda testavimui. Detaliau apie klaidų registravimą žiūrėkite 5.4.3 skyriuje.

- Pakartotinis testavimas. Testuotojai atlieka pakartotinį testavimą pagal programuotojų išspręstas klaidas. Taip pat atliekamas ir kitų sistemos funkcijų testavimas, kurių veiklą galėjo įtakoti išspręstos klaidos. Pasikartojančias klaidas testuotojas grąžina programuotojas taisymui, o išspręstas – uždaro.
- Esant poreikiui, gali būti parengiama ir užsakovui pateikiama vidinio testavimo ataskaita (išspręstų ir likusių klaidų sąrašas). Dažniausiai ši ataskaita yra rengiama MS Excel formatu, kad būtų patogu peržiūrėti registruotų klaidų būsenas.

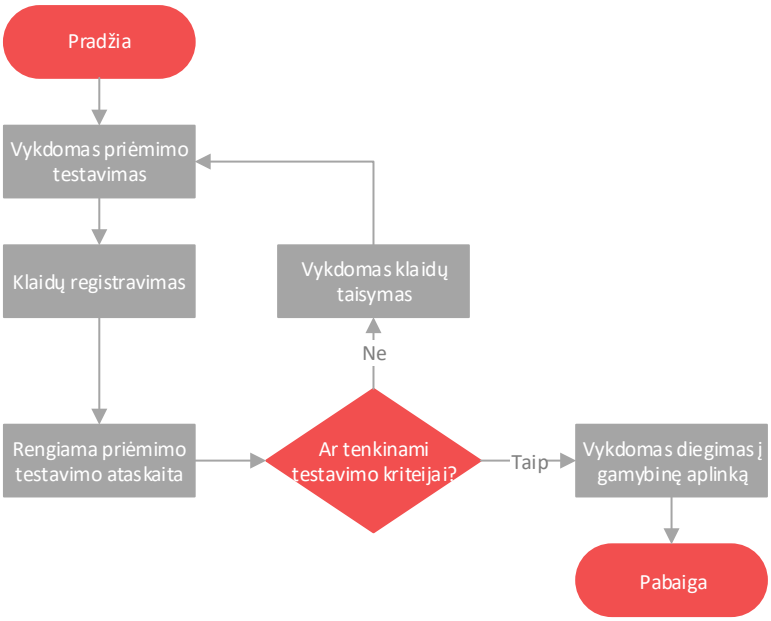
5.4.2 Priėmimo testavimo procedūra

Priėmimo testavimo metu sistema yra testuojama užsakovo testinėje aplinkoje, o patį testavimą vykdo užsakovo atstovai. Būtina pažymėti ir tai, kad priėmimo testavimas gali būti vykdomas tiek kiekvienos kūrimo iteracijos pabaigoje, tiek ir pilnai užbaigus sistemos funkcijų kūrimo darbus.

Pasiruošimas testavimui

- Parengiama testavimo metodika ir planas. Atsižvelgdami į kuriamos sistemos specifiką ir reikalavimus yra parengiama testavimo metodika. Šioje metodikoje nurodoma kaip bus vykdomas testavimas ir kokius rezultatus norima pasiekti. Metodikoje nurodomi klaidų valdymo bei suinteresuotų asmenų komunikavimo principai. Testavimo planas gali būti rengiamas konkrečiai iteracijai atskirai arba vienas visam sistemos testavimui. Testavimo planas apima šias dalis:
  - Testavimo seka – kadangi sistemų kūrimas dažniausiai vykdomas keliais etapais, tai nurodoma kokie sistemos komponentai ir funkcijos yra testuojamos tam tikrame projekto etape.
  - Pradžios sąlygos – nurodomos būtinos sąlygos testavimui pradėti t.y., parengta testinė aplinka, testavimui reikalingi duomenys ir pan.
  - Priėmimo kriterijai – nurodomos sąlygos, kurias turi tenkinti testavimo rezultatai, kad būtų galima pereiti į kitą projekto etapą. Kiekvienam testavimo etapui gali būti keliamos skirtingos pabaigos sąlygos.
  - Testavimo įrankiai – nurodoma kokie testavimo įrankiai turėtų būti naudojami atliekant sistemos testavimą.
- Parengiami ir suderinami detalūs testavimo scenarijai. Remiantis sistemos dokumentacija yra rengiami testavimo scenarijai apimantys testuojamas sistemos funkcijas. Testavimo scenarijai gali būti rengiami atskirai kiekvienai testavimo iteracijai. Testavimo scenarijus susideda iš šių dalių:
  - Pradžios sąlygos – nurodoma kokios sąlygos turi būti įgyvendintos pradedant vykdyti testavimo scenarijų.
  - Atliekami žingsniai – nurodomi detalūs žingsniai, kurių metu yra patikrinama testuojama sistemos funkcija. Taip pat, nurodomi duomenys reikalingi konkrečiame žingsnyje.
  - Pabaigos sąlygos – nurodoma kokių rezultatų tikimasi atlikus testavimo scenarijų.
- Rezultatai – nurodomi faktiniai rezultatai, kurie buvo gauti atlikus testavimo scenarijų.
- Parengiama užsakovo testinė aplinka, kurioje vykdomas iteracinis bei galutinis priėmimo testavimas (angl. *Beta test*).

Testavimo vykdymas



45 paveikslas. Priėmimo testavimo vykdymas

- Vykdomas priėmimo testavimas. Testavimą vykdo užsakovo atstovai, kuriuos testavimo klausimais konsultuoja sistemos kūrėjas.

Klaidų registravimas. Testavimo metu aptiktos klaidos registruojamos klaidų valdymo sistemoje. Užregistruotos klaidos yra priskiriamos programuotojams, kurie ištaiso klaidas ir vėl perduoda testavimui. Detaliau apie klaidų registravimą žiūrėkite 5.4.3 skyriuje:

- Ataskaitos rengimas. Atlikus priėmimo testavimą yra rengiama priėmimo testavimo ataskaita, kurioje nurodomos kokios klaidos buvo aptiktos ir užregistruotos, kiek klaidų ištaisyta ir kiek liko. Ataskaitoje taip pat nurodoma ar tenkinami priėmimo kriterijai. Jeigu priėmimo kriterijai nėra tenkinami, papildomai gali būti nurodomas klaidų taisymo laikotarpis bei numatomas pakartotinis testavimas.
- Klaidų taisymas. Užregistruotos klaidos yra perduodamos taisymui. Ištaisytos klaidos pirmiausia yra testuojamos sistemos kūrėjo aplinkoje, o po to diegiama į kliento aplinką.
- Pakartotinis testavimas. Atlikus klaidų taisymą ir atnaujinus sistemą kliento aplinkoje vėl vykdomas testavimas. Pakartotinis testavimas skirtas patikrinti ar visos užregistruotos klaidos yra ištaisytas. Jei klaidos ištaisytos tai vykdomas testavimo užbaigimas.

Testavimo užbaigimas:

- Testavimo pabaiga. Testavimas laikomas baigtu, jeigu testavimo rezultatai tenkina priėmimo kriterijus. Priėmimo kriterijai dažniausiai yra išreiškiami tam tikru procentiniu kiekiu klaidų, kurios gali likti pasibaigus testavimui ir bus taisomos bandomosios eksploatacijos bei garantinio aptarnavimo metu. Jeigu sistema tenkina priėmimo kriterijus, ji yra diegiama į gamybinę aplinką.

Priėmimo kriterijų pavyzdys

Sistema gali būti diegiama į gamybinę aplinką jei:

- *nėra testavimo rezultatų su kritinėmis klaidomis*
- *yra ne daugiau kaip 10 procentų testavimo scenarijų su vidutinėmis klaidomis*
- *yra ne daugiau kaip 50 procentų testavimo scenarijų su mažomis klaidomis*

Likusios vidutinės ir mažos klaidos, kurios nedaro tiesioginės įtakos korektiškam sistemos veikimui, gali būti taisomos bandomosios eksploatacijos ir garantinio aptarnavimo metu.

5.4.3 Klaidų registravimas ir valdymas

Visos klaidos, aptiktos sistemos testavimo metu, atliekant tiek vidinį, tiek priėmimo testavimą, registruojamos specialioje programinėje įrangoje *JIRA*, kuri yra prieinama internetu. Klaidos, aptiktos po testavimo, t.y. vykdant jos eksploataciją, taip pat registruojamos *JIRA* sistemoje. Tiesiogiai registruoti klaidas *JIRA* sistemoje gali tik tam tikrą prieigą turintys asmenys. Paveiksle žemiau pateiktas apibendrintas klaidos valdymo procesas:

46 paveikslas. Apibendrintas klaidos valdymo procesas

Klaidos, rastos sistemos testavimo metu, yra registruojamos, kiekvienai klaidai užpildant tokią informaciją:

- Santrauka (angl. *Summary*) – trumpas klaidos aprašymas.
- Prioritetas (angl. *Priority*) – rastos klaidos prioritetas parodo, kokio sudėtingumo yra klaida ir kaip greitai ji turi būti peržiūrėta programuotojo, ir priimtas tam tikras sprendimas tos klaidos atžvilgiu. Galimi registruojamų klaidų tipai:
  - Kritinė klaida – aukščiausio prioriteto klaida dėl kurios sistema ar jos modulis/funkcija neveikia ar veikia nekorektiškai. Šio tipo klaidos yra sprendžiamos pirmiausia.
  - Vidutinė klaida – nedidelis sistemos trūkumas ar funkcinis defektas, klaidos įmanoma išvengti. Klaidos sprendžiamas pašalinus kritines klaidas.
  - Maža klaida – ši klaida neįtakoja sistemos veikimo. Tai gali būti dizaino klaida (netinkama spalva, šriftas ir pan.). Klaidos sprendžiamos pašalinus aukštesnio prioriteto klaidas.
- Tipas (angl. *Type*) – nurodomas tipas (klaida, naujas funkcionalumas, pakeitimas ir pan.).
- Atsakingas asmuo – nurodomas programuotojas, kuris bus atsakingas už klaidos taisymą.
- Komponentas (angl. *Component*) – posistemė ar modulis, kuriame buvo rasta klaida.
- Registruojantis asmuo (angl. *Reporter*). Užpildoma automatiškai pagal registruojančio asmens prisijungimo duomenis.
- Aplinka (testinė, produkcinė ir pan.).
- Aprašymas (angl. *Description*) – išsamus klaidos aprašymas.
- Priedai (angl. *Attachment*) – pridedami dokumentai (aprašymas, paveikslėlis) padedantys greičiau identifikuoti ir pašalinti klaidą.

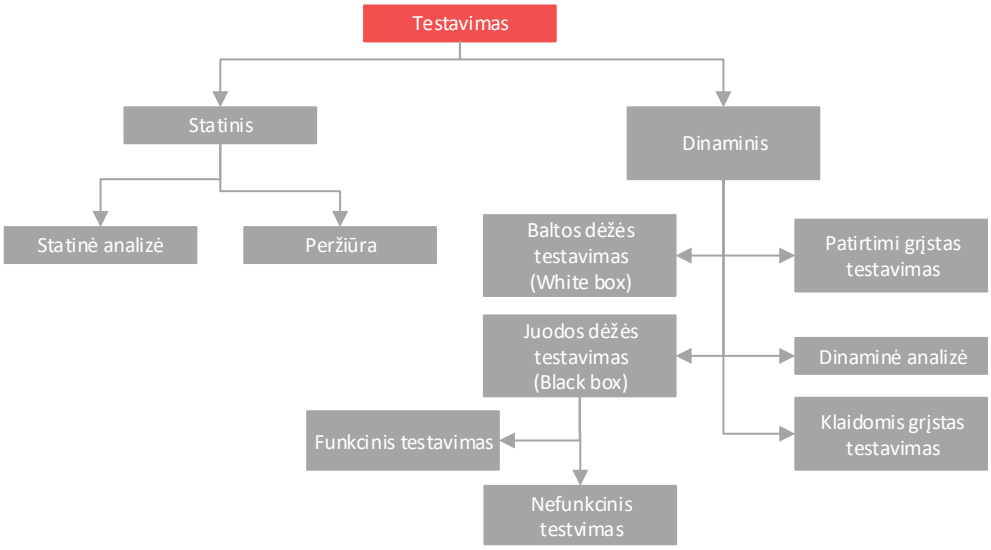
Kiekvienai įrašytai klaidai automatiškai suteikiamas unikalus identifikacinis numeris. Naudojant šį numerį yra užtikrinamas klaidų atsekamumas kadangi, JIRA programoje gali būti registruojamos kelių projektų klaidos.

Klaidos aprašymą dažnai reikia patikslinti, pridedant kokią nors bylą, naudotojo sąsajos lango kopiją. Esant tokiam poreikiui, prie klaidos aprašymo gali būti pridedama (prisegama) byla.

Kai programuotojas išsprendžia problemą, apibrėžtą klaidos aprašyme, jis klaidų sistemoje pažymi, kad klaida išspręsta ir nurodo klaidos rezoliuciją. Testuotojas peržiūri visas programuotojų išspręstas klaidas ir patikrina, ar tikrai jos nepasikartoja sistemoje. Testuotojas gali sutikti su programuotojo sprendimu arba jį atmesti. Jei klaida yra teisingai išspręsta, testuotojas pakeičia klaidos būseną į „Uždaryta“. Jeigu klaida nėra tinkamai išspręsta, testuotojas ją atidaro pakartotinai.

5.4.4 Testavimo metodikos

Kiekvienas testavimas yra vykdomas remiantis tam tikra metodika, kuri nurodo kaip pasiruošti ir atlikti testavimą siekiant surasti sistemos klaidas. Žemiau esančiame paveiksle pateikiamos testavimo metodikų hierarchija.



47 paveikslas. Testavimo metodikų hierarchija

Testavimo metodikos yra skirstomos į dvi stambias grupes: statinis testavimas ir dinaminis testavimas

Statinis testavimas

Statinis testavimas – tai testavimas atliekamas nevykdant jokio programinio kodo ar sistemos funkcijų. Statinį testavimą apima šios testavimo metodikos:

- Peržiūra – peržiūros naudojamas ankstyvoje projekto stadijoje siekiant sumažinti galimas rizikas ir identifikuoti sistemos dalis, kurioms reiks skirti daugiau dėmesio vykdant vėlesnes testavimo veiklas. Peržiūrų metu yra peržiūrimi projekto artefaktai t.y., dokumentacija, programinis kodas ir pan. Peržiūrose dažniausiai dalyvauja kokybės užtikrinimo specialistai, testuotojai, dokumentų autoriai bei kiti suinteresuoti asmenys. Peržiūros dar gali būti skirstomos į šiuos tipus:

- Reikalavimų peržiūra (angl. *Inspection*) – viena plačiausiai dokumentuotų formalių technikų, tačiau praktikoje dažniau naudojamos neformalios technikos. Reikalavimų analizę būtina atlikti pradinėje sistemos kūrimo stadijoje, šis procesas būtinas, norint teisingai suprojektuoti sistemos funkcijas bei patikrinti, ar sistemos funkcijos atitinka jai keliamus reikalavimus.
- Kolegų peržiūra (angl. *Peer Review*) – programinės įrangos aptarimas tarp kolegų ar toje srityje dirbančių žmonių, tai padeda išsiaiškinti klaidas bei galimus patobulinimus.
- Techninė peržiūra (angl. *Technical Review*) – panaši į kolegų peržiūrą tik labiau įsigilinant į pasirinktas technines priemones ir sprendimus.

- Statinė analizė – atliekama siekiant įvertinti parašyta programinį kodą. Sukurtos sistemos funkcijų atitikimą reikalavimams. Programinio kodo analizei dažniausiai naudojami automatiniai kodo analizės įrankiai tokie kaip kodo kompiliatoriai ir pan. Šių įrankių pagalba yra aptinkamos programinio kodo klaidos tokios kaip nenaudojamas kodas, nesibaigiantys ciklai ir pan.

Dinaminis testavimas

Dinaminis testavimas – tai testavimas, kurio metu yra praktiškai išbandomos sukurtos sistemos funkcijos. Dinaminį testavimą apima šios testavimo metodikos:

- Patirtimi grįstas testavimas – testavimas atliekamas remiantis turima patirtimi. Testavimo metu kreipiamas dėmesį į panašiose sistemose dažnai pasitaikančias klaidas. Taikant šią metodiką, patyrę testuotojai parengia detalesnius testus užtikrindami pilną sistemos modulio/funkcijos patikrinimą.
- Dinaminė analizė – testavimas, kurio metu yra stebimas ir analizuojamas sistemos veikimas įvykus sistemos klaidai t.y., stebimas sistemos klaidos įtaka atminties, procesoriaus naudojimui, tinklo apkrovai. Dažniausiai dinaminėje analizėje yra naudojami įrankiai, kurių pagalba yra atliekami automatiniai testai.
- Klaidomis grįstas testavimas – testavimo metu yra orientuojamasi į vieno tipo klaidų grupes. Šio testavimo metu yra siekiama ištestuoti tam tikras sistemos funkcijas, kurios gali įtakoti tų pačių klaidų atsiradimą.
- Baltos dėžės testavimas (angl. *White box*) – tai toks testavimas kuomet yra žinoma sistemos architektūra ir programinis kodas. Šio testavimo metu yra nagrinėjamos sistemos būsenos bei sistemos veiksmai atliekant testuojamas funkcijas. Teatavimo metu yra taikomi įvairūs testavimo principai tokie kaip:
  - Testų apimtis – testų apimtis išreiškiama procentu, kuris apibūdina, kiek parametrų išsamiai ištestuojamas tam tikras objektas. Šis dydis paskaičiuojamas pagal formulę: *Apimtis = Patikrinta parametrų / Viso parametrų * 100*.
  - Kodo sakinių testavimo apimtis – tai ištestuotų kodo sakinių procentas lyginant su visais kodo sakiniais: *Kodo sakinių apimtis = Patikrinta sakinių / Viso sakinių * 100*.
- Juodos dėžės testavimas (angl. *Black box*) – tai toks testavimas kuomet nėra gilinamasi į sistemos architektūrą bei programinį kodą. Juodos dėžės testavimas yra skirstomas į dvi kategorijas: funkcinis testavimas ir nefunkcinis testavimas. Funkcinio testavimo metu yra testuojamos sistemos funkcijos ir tikrinama ar gautas rezultatas yra tinkamas nesigilinant kaip sistema atliko vykdomas funkcijas.
- Funkcinio testavimo metu yra taikomi įvairūs testavimo principai tokie kaip:
  - Ekvivalentus skirstymas (angl. *Equivalence Partitioning*): Galimas teisingas testavimo duomenų intervalas suskaidomas dalimis, atlikti testavimą su visomis galimomis testavimo reikšmėmis prireiktų daug laiko ir tai nėra būtina, todėl testavimui imamos tik kelios reikšmės iš kiekvienos testavimo duomenų dalies, rezultatas su visais duomenimis turi būti teisingas. Pvz. testuojame formos duomenų įvedimo laukelį, kurio galimos teisingos reikšmės yra nuo 1 iki 100. Šį reikšmių intervalą galima suskirstyti į 1 – 50 ir 50 – 100. Tada sistemą testuojame su iš kiekvieno intervalo pasirinktomis reikšmėmis: 1, 16, 50, 67 ir 100, o taip pat ir su neteisingomis atsitiktinėmis reikšmėmis kaip pvz. -10, 125, 0, 0.85.
  - Ribinių reikšmių analizė (angl. *Boundary Value Analysis*): Labai svarbu atlikti testavimus su ribinėmis intervalo reikšmėmis bei su reikšmėmis mažiausiu žingsniu išeinančiomis iš intervalo. Pvz. jei

testuojamo objekto galimų reikšmių intervalas 1 – 99, tada testavimus būtina atlikti su reikšmėmis: 0, 1, 99, 100.

- Būsenų pasikeitimo testai (angl. *State Transition Testing*): Kai kurie sistemos objektai turi apibrėžtą baigtinę galimų būsenų aibę. Ši testavimo technika apima objekto galimų būsenų patikrinimą bei objektų elgseną pereinant iš vienos būsenos į kitą. Čia būtina patikrinti tiek galimus objekto būsenos pasikeitimus, tiek negalimus, pvz. objektas gali keisti būseną iš 1 į 2, tačiau negali iš 2 į 1. Taip pat būsenų keitimas priklausomai nuo pradinės būsenos ir būsenos, į kurią keičiamas objektas, gali iškviesti skirtingus veiksmus, o tai sąlygoti skirtingą galutinę būseną.

- Nefunkcinio testavimo metu yra tikrinama sistemos savybės nesusijusios su naudotojui prieinamomis funkcijomis. Nefunkcinis testavimas susijęs su sistemos patikimumu, saugumu, palaikymu, patogumu, suderinamumu ir pan. Šis testavimas dažniausiai yra vykdomas naudojant testavimo įrankius. Nefunkcinio testavimo metu gali būti atliekami tokie testai kaip:
  - Našumo (angl. *Performance*) testavimas - skirtas patikrinti sistemos našumą, t.y. ar nestringa sistemos veikimas kuomet ji apdoroja tam tikrą užklausų kiekį. Šio testavimo metu testuojamos atskiros sistemos funkcijos ir įvertinamas jų veikimas.
  - Apkrovos (angl. *Stress*) testavimas - skirtas patikrinti ar sistemos veikimas nenukenčia kai ja intensyviai naudojamasi. Testavimo metu yra simuliuojamas didelis aktyvių naudotojų kiekis.
  - Patogumo (angl. *Usability)* testavimas - testuojamas sistemos patogumas naudotojui t.y., ar sistemos funkcijos patogiai prieinamos ir intuityviai įvykdomos. Taip pat testuojama ar sistemos išvaizda priimtina visoms naudotojų grupėms.
  - Sistemos saugumo (angl. *Security*) testavimas – atliekamas sistemos testavimas atsižvelgiant į jai keliamus saugumo reikalavimus. Testavimo metu tikrinama ar sistemą galima „nulaužti“ ir prieiti prie viešai neteikiamų duomenų.

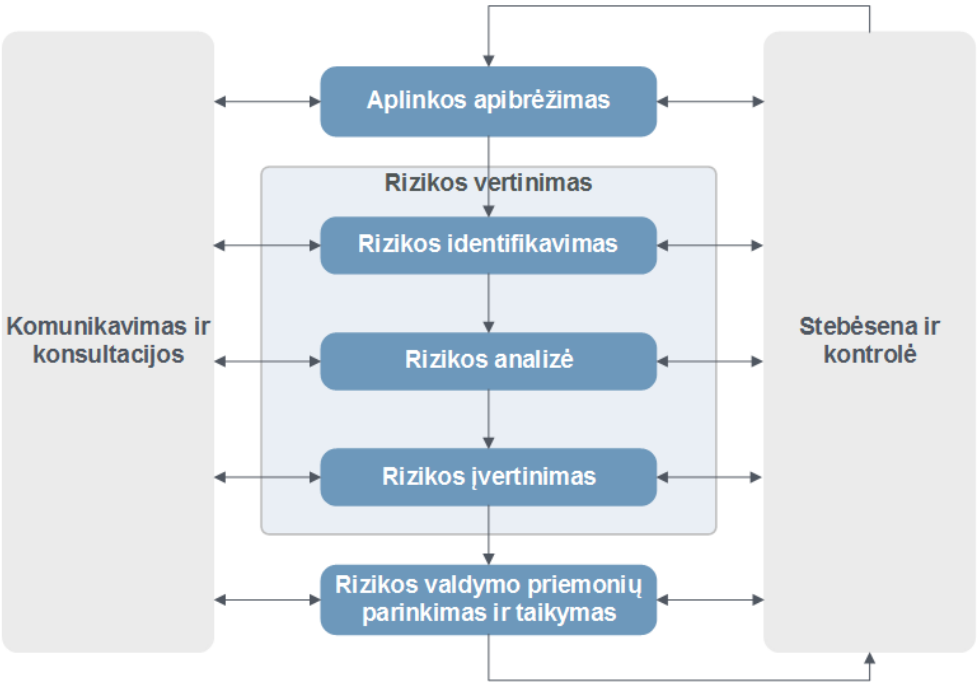
6 Rizikos valdymas (P_5)

Šiame poskyryje pateiktas rizikos valdymo planas su išsamia rizikos veiksnių analize, visi rizikos veiksniai susiję su vykdomu projektu; analizė apima ir laikotarpį po paslaugų įgyvendinimo, ir yra pagrįsta išskirtinėmis paslaugų srities žiniomis. Pasiūlytos efektyvios priemonės rizikai valdyti, pažangūs, įvairius aspektus apimantys, problemų sprendimo būdai. Atsakomybės rizikos valdyme aprašytos aiškiai, detaliai ir nedviprasmiškai.

6.1 Rizikų valdymo metodika

Įgyvendinant net ir labai gerai suplanuotą projektą, galimos netikėtos probleminės situacijos, kurių ankstyvas numatymas ir atitinkamas pasirengimas padeda išvengti rimtesnių kliūčių projekto tikslams pasiekti. Siekiant sumažinti projekto biudžetui, grafikui ar įgyvendinimui neigiamos įtakos turinčių veiksnius ir sustiprinti teigiamų veiksnių poveikį, reikalingas tinkamas projekto rizikų valdymas. Standarte ISO 31000 *Risk management – Principles and guidelines* (toliau – ISO 31000), aprašomoje rizikų valdymo metodikoje pateikiami svarbiausi rizikų valdymo principai ir gairės. Šis standartas yra tinkamas bet kokio tipo organizacijoms, nepriklausomai nuo jų dydžio, vykdomos veiklos ar sektoriaus ir yra plačiai naudojamas ir pripažįstamas visame pasaulyje.

Principinė ISO 31000 aprašomo rizikos valdymo proceso schema pateikiama žemiau esančiame paveiksle.



48 paveikslas. Rizikos valdymo procesas

Remiantis ISO 31000 standarte aprašyta metodika, rizikų valdymo procesą sudaro:

1. **Komunikavimas ir konsultacijos** – viso rizikų valdymo proceso metu vykdomas komunikavimas ir konsultavimasis tarp išorinių ir vidinių suinteresuotų šalių, leidžiantis užtikrinti abipusį su rizikomis susijusių sprendimų priėmimo ir konkrečių atsakomųjų veiksmų poreikio suvokimą;

2. **Aplinkos apibrėžimas** –leidžia vykdant rizikų valdymą susikoncentruoti ties organizacijos ar projekto tikslais, atsižvelgiant į išorinę ir vidinę organizacijos bei rizikų valdymo aplinką. Aplinkos apibrėžimas padeda nustatyti tinkamus rizikų vertinimo kriterijus ir apimtį visam likusiam rizikų valdymo procesui;
3. **Rizikos vertinimas**, kurį sudaro:
    3.1. **Rizikos identifikavimas** – identifikuojami visi galimi rizikų šaltiniai, pažeidžiamos vietos, galimi situacijos pasikeitimai ir jų padariniai bei galimos pasekmės. Šio etapo tikslas yra sudaryti išsamų rizikų, pagrįstų įvykiais kurie gali paveikti organizacijos ar projekto tikslų siekimo eigą, sąrašą.
    3.2. **Rizikos analizė** – atliekama išsami identifikuotų rizikų analizė, kurios metu nustatoma kurios rizikos turi būti valdomos ir kokios jų valdymo strategijos bei metodai yra labiausiai tinkami. Analizės metu turi būti nustatomos visos galimos teigiamos ir neigiamos rizikos veiksnių pasekmės bei jų tikimybės ir identifikuojami visi faktoriai galintys joms daryti įtaką. Pagal galimas rizikų pasekmes ir jų tikimybes nustatomi rizikų lygiai, kurie nusako jų keliamas grėsmes.
    3.3. **Rizikos įvertinimas** – rizikų analizės rezultatai yra palyginami su aplinkos apibrėžimo metu nustatytais rizikų vertinimo kriterijais. Remiantis šiuo palyginimu yra priimami sprendimai dėl kiekvienos rizikos valdymo poreikio ir prioritetiškumo.
4. **Rizikos valdymo priemonių parinkimas ir taikymas** – atsižvelgiant į rizikos lygį parenkama ir pritaikoma viena ar daugiau rizikos valdymo priemonių. Prieš pradedant taikyti rizikų valdymo priemones yra parengiamas rizikų valdymo planas, kuriame pagrindžiamas jų valdymo priemonių pasirinkimas bei aprašoma kokiais būdais ir tvarka jos bus pritaikomos ir įgyvendinamos. Gali būti pažymimos ir likutinės rizikos, kurios gali išlikti ir pritaikius rizikų valdymo priemones.
5. **Stebėsena ir kontrolė** – viso rizikų valdymo proceso metu vykdoma visas rizikų valdymo veiklas apimanti stebėsena ir kontrolė. Pastovi stebėsena leidžia įsitikinti, kad naudojamos rizikų valdymo priemonės veikia tinkamai bei padeda surinkti informaciją, reikalingą rizikų valdymo proceso gerinimui, laiku pastebint besikeičiančius rizikų veiksnius ir esamo proceso silpnąsias vietas.

6.2 Rizikų valdymo metodikos taikymas

Remiantis ISO 31000 standarte aprašytu procesu, rengiant Pasiūlymą, identifikuoti ir įvertinti rizikos veiksniai, galintys turėti įtakos sėkmingam Projekto rezultatų pasiekimui (Projekto įgyvendinimo metu) ir/ ar jų panaudojimui (po Projekto) bei numatytos galimos (siūlomos) identifikuotų veiksnių rizikos valdymo priemonės. Atliekant rizikos vertinimą, atsižvelgta į identifikuotų rizikos veiksnių pasireiškimo tikimybę (didelė, vidutinė ar maža) ir poveikį (didelis, vidutinis ar mažas). Rizikos vertinimui naudotų tikimybių ir poveikio kategorijų aprašymai pateikiami žemiau esančiose lentelėse.

21 lentelė. Rizikos poveikio kategorijos

Rizikos poveikis	Poveikio aprašymas
Didelis (D)	Gali padaryti Projektui esminės žalos. Dėl šių įvykių gali tekti pakartotinai planuoti darbus ir skaičiuoti išteklius. Ypatingais atvejais gali žlugti visas projektas.
Vidutinis (V)	Gali sukelti vėlavimus arba papildomo darbo poreikį, todėl gali būti viršyti specialieji nenumatytiems atvejams skirtas laikas ir kiti ištekliai.
Mažas (M)	Gali sukelti vėlavimus arba papildomo darbo poreikį, tačiau šį darbą galima atlikti neviršijant esamų specialiųjų išteklių.

22 lentelė. Rizikos tikimybių kategorijos

Tikimybė	Aprašymas
Didelė (D)	Tikimybė įvykti yra didesnė nei neįvykti.
Vidutinė (V)	Yra reali tikimybė įvykti.
Maža (M)	Nelabai tikėtina, bet įmanoma.

Įvertinus rizikos poveikį ir tikimybę, nustatytas rizikos lygis, kuriuo remiantis didžiausią dėmesį Projekto įgyvendinimo metu bus galima skirti rizikingiausioms sritims, tam numatant reikalingas prevencines ir/ ar poveikio mažinimo priemones. Naudotų rizikos lygių aprašymai pateikiami žemiau esančioje lentelėje.

23 lentelė. Rizikos lygiai

Rizikos lygis	Poveikis ir tikimybė	Aprašymas	Prevencinės ir poveikio mažinimo priemonės
Maksimalus	D-D	Rizika gali lemti Projekto žlugimą	Būtina skirti daugiau išteklių Projekto vykdymui. Projekto vykdymo komandą formuoti iš specialistų, turinčių išskirtines specifinės dalykinės srities žinias, ilgametę patirtį bei tinkamą kvalifikaciją. Maksimaliai išnaudoti galimybę kuriamoje posistemėje naudoti gerai veikiančias, pilnai ištestuotas šiuo metu naudojamas komponentes, kurios tenkins jos būsimus naudotojus. Paskirti konkretų atsakingą asmenį.
Aukštas	D-V V-D	Rizika gali lemti žymų Projekto vėlavimą, gali prireikti papildomų išteklių, Projekto veiklų perplanavimo	Skirti papildomus išteklius Projekto vykdymui. Į Projekto vykdymo komandą pritraukti specialistus, turinčius išskirtines specifinės dalykinės srities žinias, ilgametę patirtį bei tinkamą kvalifikaciją. Pasinaudoti galimybe kuriamoje posistemėje naudoti gerai veikiančias, pilnai ištestuotas šiuo metu naudojamas komponentes, kurios pilnai tenkins jos būsimus naudotojus. Paskirti konkretų atsakingą asmenį rizikos kontrolei.
Žemas	D-M M-D V-V	Rizika gali lemti nežymius nukrypimus nuo Projekto vykdymo laiko ir biudžeto	Skirti papildomus išteklius Projekto vykdymui. Į Projekto vykdymo komandą pritraukti specialistus, turinčius reikalingas dalykinės srities žinias. Bent dalinai pasinaudoti galimybe kuriamoje posistemėje naudoti gerai veikiančias, pilnai ištestuotas šiuo metu naudojamas komponentes, kurios pilnai tenkins jos būsimus naudotojus. Paskirti konkretų atsakingą asmenį rizikos kontrolei.
Minimalus	V-M M-V M-M	Rizika numatyta ir valdoma	Papildomi ištekliai nėra reikalingi.

Šiame skyriuje aprašyti rizikos valdymo pricipai bus taikomi ne tik rengiant Pasiūlymą identifikuotiems rizikos veiksniams (žr. 6.3 poskyrį), bet ir viso Projekto įgyvendinimo metu, nustatant ir vertinant naujus rizikos veiksnius bei parenkant jiems valdyti reikalingas priemones.

Visos Projekto įgyvendinimo metu identifikuotos rizikos, jų lygis ir kita aktuali informacija, esant poreikiui bus fiksuojama Projekto rizikų registre. Visi Projekto rizikų registre užfiksuoti rizikos veiksniai bus reguliariai (Projekto eigos aptarimui skirtų susitikimų metu) aptariami su Perkančiąja organizacija. Esant poreikiui, bus reguliariai aptariamas ir reikalingų ir/ ar pritaikytų rizikos valdymo priemonių įgyvendinimas.

6.3 Projekto rizikos ir jų valdymo priemonės

Skirtingoms rizikoms reikalingos skirtingos valdymo priemonės, todėl, įvertinus potencialius Projekto rizikos veiksnius, numatyti galimi veiksmai jų neigiamam poveikiui mažinti. Taip pat siekiant užtikrinti tinkamą siūlomų rizikos valdymo priemonių taikymą, numatytos už kiekvieną priemonę atsakingos šalys.

Projekto rizikų aprašymas, apimantis identifikuotus rizikos veiksnius, jų poveikio ir tikimybės įvertinimą bei atsižvelgiant į tai Paslaugų teikėjo siūlomas rizikos valdymo priemones ir už jas atsakingas šalis, pateikiamas žemiau esančioje lentelėje. Pažymėtina, kad pateiktas identifikuotų rizikos veiksnių sąrašas nėra nekeičiamas – Projekto įgyvendinimo metu reikalingas

nuolatinis nustatytų rizikos veiksnių ir jų valdymo priemonių stebėjimas, naujų rizikų identifikavimas bei rizikų valdymo proceso vertinimas.

24 lentelė. Identifikuoti Projekto rizikos veiksniai ir jų valdymo priemonės

Eil. Nr.	Rizika	Rizikos poveikis	Rizikos tikimybė	Rizikos valdymo priemonės	Atsakingos šalys
1.	**Organizacinės rizikos**				
1.1.	Projektu suinteresuotų šalių interesų konfliktai	V	V	Siekiant išvengti interesų konfliktų Projekto metu, bus iš anksto identifikuojami suinteresuotų šalių poreikiai ir lūkesčiai, visoms suinteresuotoms šalims aiškiai išaiškinami Projekto tikslai ir uždaviniai. Projekto įgyvendinimo metu suinteresuotų šalių interesai bus derinami pabrėžiant bendrą Projekto tikslą ir kontekstą bei prioritetizuojant jų poreikius pagal objektyvius kriterijus.	Paslaugų teikėjas, Perkančioji organizacija
1.2.	Organizaciniai pokyčiai Perkančiojoje organizacijoje	D	M	Paslaugų teikėjo panašių projektų vykdymo srities žinios ir ilgametė patirtis leistų dalį darbų vykdyti vėluojant Perkančiosios organizacijos sprendimams, tačiau organizacinių pokyčių atveju Perkančiosios organizacijos vadovybė turėtų užtikrinti sklandų projektinių darbų perdavimą naujai Projekto darbo grupei.	Perkančioji organizacija
1.3.	Keletas tuo pat metu vykstančių projektų/ vykdomų sutarčių Perkančiojoje organizacijoje	V	M	Paslaugų teikėjo panašių projektų vykdymo srities žinios ir ilgametė patirtis leistų dalį darbų vykdyti be didesnės Perkančiosios organizacijos pagalbos. Tačiau tokiu atveju Perkančioji organizacija turėtų labai aiškiai nustatyti Projekto prioritetų paskirstymą, aiškiai apibrėžti Projekto tikslus ir reikalavimus bei atsakomybių paskirstymą Projekte.	Perkančioji organizacija
1.4.	Keletas tuo pat metu Paslaugų teikėjo vykdomų projektų	V	V	Paslaugų teikėjas turi ilgametę panašių projektų valdymo patirtį ir pakankamai kvalifikuotų žmogiškųjų išteklių, todėl gali užtikrinti sėkmingą kelių projektų vykdymą vienu metu. Dėl kitų Paslaugų teikėjo vykdomų projektų nenumatytai išaugus Projekto komandos specialistų darbo krūviui, bus skiriami papildomi Projekto kontekstą ir vykdomas veiklas išmanantys specialistai.	Paslaugų teikėjas
1.5.	Nepakankamas žmogiškųjų išteklių skyrimas Projektui	V	M	Sprendimas dėl būsimos Paslaugų teikėjo Projekto komandos priimtas rengiant šį pasiūlymą. Numatyti visi tinkamam perkamų Paslaugų teikimui reikalingi ekspertai. Esant poreikiui, Paslaugų teikėjas skirs papildomus žmogiškuosius išteklius konkrečių Projekto įgyvendinimo veiklų vykdymui. Efektyviam Projekto koordinavimui ir valdymui užtikrinti reikalinga organizacinė struktūra bus detaliai aptarta ir suderinta Projekto įgyvendinimo pradžioje. Tiksli Projektą vykdančių, valdančių ir prižiūrinčių grupių sudėtis ir jų atsakomybės bus nurodytos Paslaugų teikimo plane.	Paslaugų teikėjas, Perkančioji organizacija
1.6.	Projekte dalyvaujančių pagrindinių ekspertų pasikeitimas Projekto įgyvendinimo metu ar prieš jo pabaigą	V	V	Siekiant išvengti rizikų, susijusių su ekspertų pasikeitimais, bus iš anksto numatyti pagrindinius ekspertus galintys pavaduoti ar pakeisti specialistai, vykdomas jų rengimas, supažindinant su pagrindinių ekspertų funkcijomis ir vykdoma veikla Projekte.	Paslaugų teikėjas, Perkančioji organizacija
2.	**Technologinės rizikos**				

Eil. Nr.	Rizika	Rizikos poveikis	Rizikos tikimybė	Rizikos valdymo priemonės	Atsakingos šalys
2.1.	Neįgyvendinamų arba sunkiai įgyvendinamų reikalavimų apibrėžimas.	D	V	Projekto apimties valdymas atsižvelgiant į projekto laiko, finansinius išteklius ir siekiamą kokybę. Reikalavimų prioritetizavimas ir dėmesio į aukščiausio prioriteto reikalavimus sutelkimas.	Paslaugų teikėjas, Perkančioji organizacija
2.2.	Sprendimų technologinis nesuderinamumas	D	V	Technologijų analizė. Alternatyvių sprendimų numatymas. Atvirų ir visuotinai pripažintų standartų naudojimas.	Paslaugų teikėjas, Perkančioji organizacija
2.3.	Įdiegtų funkcionalumų nepatogumas ir kompleksiškumas bei neatitikimas naudotojų poreikiams	D	M	Bendradarbiavimas su būsimais įdiegtų sprendimų naudotojais.	Paslaugų teikėjas, Perkančioji organizacija
2.4.	Išorinių reikalavimų pasikeitimas įgyvendinant Projektą	D	M	Nuolatinis Projekto aplinkos stebėjimas. Atvirų standartų naudojimas, leidžiantis nedidelėmis sąnaudomis atlikti pakeitimus.	Paslaugų teikėjas, Perkančioji organizacija
2.5.	Paslaugų teikėjo nesugebėjimas realizuoti specifinių ar komplikuotų reikalavimų	V	V	Nuolatinė komunikacija ir mediavimas tarp Perkančiosios organizacijos ir Paslaugų teikėjo dėl kuriamų technologinių sprendimų; Paslaugų teikėjo kuriamų sprendimų atitikimo Perkančiosios organizacijos poreikiams įvertinimas ir kontrolė.	Paslaugų teikėjas
2.6.	Projekto tikslų, uždavinių, apimties ir/ ar reikalavimų sudėtingumas ir/ ar pasikeitimai	D	M	Projekto tikslai ir uždaviniai bus kaip galima tiksliau apibrėžti įvadiniame Projekto etape ir įtvirtinti parengtoje ir su Perkančiąja organizacija suderintoje įvadinėje Paslaugų teikimo ataskaitoje. Bet kokie vėlesni pakeitimai bus galimi tik bendru Perkančiosios organizacijos ir Paslaugų teikėjo sutarimu.	Paslaugų teikėjas, Perkančioji organizacija
3.	**Projektinės rizikos**				
3.1.	Per didelės ir per daug kompleksinės Projekto apimties nustatymas	V	M	Teikiamų Paslaugų apimtis kiek įmanoma tiksliau bus apibrėžta Projekto inicijavimo etape. Iš anksto netinkamai įvertinus Paslaugų apimtį ar esant jos pasikeitimams Projekto metu bus taikomos Paslaugų įgyvendinimo plane numatytos alternatyvos ir/ ar skiriami papildomi ekspertai.	Perkančioji organizacija
3.2.	Netinkamas Projekto ar atskirų jo dalių įgyvendinimui reikalingo laiko įvertinimas	V	M	Bus skiriamas išskirtinis dėmesys netinkamai įvertintai veiklai vykdyti, taikant didesnių išteklių skyrimo (angl. crashing) ar lygiagretaus veiklų atlikimo (angl. fast tracking) metodus. Taip pat bus ieškoma bendro Perkančiosios organizacijos ir Paslaugų teikėjo sutarimo dėl galimų netinkamai įvertintos Projekto dalies apimties mažinimo, kiek įmanoma mažiau paveikiant Projekto įgyvendinimą.	Perkančioji organizacija
3.3.	Projekto veiklų vėlavimas	V	V	Bus vykdomas nuoseklus veiklų planavimas ir nuolatinė Projekto įgyvendinimo eigos stebėsena. Bus imamasi visų įmanomų priemonių, kad visos Projekto veiklos būtų įvykdytos ir dokumentai parengti laiku. Esant numatomiems veiklų	Paslaugų teikėjas

Eil. Nr.	Rizika	Rizikos poveikis	Rizikos tikimybė	Rizikos valdymo priemonės	Atsakingos šalys
				vėlavimams, bus taikomi didesnių išteklių skyrimo (angl. crashing) ar lygiagretaus veiklų atlikimo (angl. fast tracking) metodai, taip pat imamasi kitų alternatyvų.	
3.4.	Perkančiosios organizacijos ir Paslaugų teikėjo komunikacijos klaidos	V	M	Projekto įgyvendinimo pradžioje bus aptartas ir suderintas formalus komunikacijos procesas. Komunikacijos valdymo planas, apimantis numatomas naudoti komunikacijos priemones, keitimosi informacija periodiškumą ir kitus svarbius aspektus.	Paslaugų teikėjas, Perkančioji organizacija
3.5.	Žemas Perkančiosios organizacijos suinteresuotumas kuriamais sprendimais	M	M	Bus vykdoma nuolatinė komunikacija su Perkančiosios organizacijos darbuotojais, nuolatinis jų įtraukimas į Projekto veiklas.	Paslaugų teikėjas, Perkančioji organizacija
4.	**Rizikos po Projekto**				
4.1.	Papildomos (iš anksto nenumatytos) išlaidos reikalingos projekto apimtyje sukurtiems technologiniams sprendimams palaikyti	M	M	Pasireiškus šiai rizikai gali padidėti Perkančiosios organizacijos išlaidos, susijusios su sukurtos infrastruktūros palaikymu. Galimi rizikos valdymo būdai: 1) skirti papildomas lėšas, siekiant užtikrinti projekto tęstinumą; 2) atsisakyti šios veiklos vykdymo savais resursais ir pirkti infrastruktūros palaikymo paslaugas iš išorės, jeigu paskelbus konkursą bus pasiūlyta prieinama kaina (mažesnė už faktines išlaidas, susijusias su darbuotojų išlaikymu).	Perkančioji organizacija
4.2.	Nenaudojami projekto apimtyje sukurti rezultatai	D	M	Priežasčių analizė ir greitas reagavimas, tobulinant sistemą, didinant jos patogumą. Numatomas papildomas tikslinių grupių informavimas, skatinimas naudotis projekto rezultatais.	Perkančioji organizacija