**VILNIAUS UNIVERSITETAS**
**EKONOMIKOS IR VERSLO ADMINISTRAVIMO FAKULTETAS**
**EKONOMINĖS INFORMATIKOS KATEDRA**

**Tadas GERMANAVIČIUS**

***Strateginio informacinių sistemų valdymo programa***

**MAGISTRO DARBAS**

# SPECIALIZUOTOS DOKUMENTŲ VALDYMO SISTEMOS TESTAVIMO STRATEGIJOS KŪRIMAS

# CREATION OF TESTING STRATEGY FOR SPECIALIZED DOCUMENT MANAGEMENT SYSTEM

Leidžiama ginti _____
(parašas)

Katedros vedėjas prof. **R. Skyrius**

Magistrantas _____
(parašas)

Darbo vadovas _____
(parašas)

Doc. **M. Krutinis**

Darbo įteikimo data:

Registracijos Nr.

**Vilnius, 2020**

# TURINYS

# SANTRUMPŲ IR TERMINŲ PAAIŠKINIMAI

AEC (angl. *Architecture, Engineering & Construction*) – akronimas, apibūdinantis tris susijusias industrijas – architektūrą, inžineriją ir statybas.

BIM (angl. *Building Information Modelling*) – pastatų informacijos modeliavimas. BIM apibrėžiamas kaip technologijų, procesų ir strategijų rinkinys, leidžiantis keletui suinteresuotųjų šalių bendradarbiauti projektuojant, statant ir valdant pastatus virtualioje erdvėje. (*BIMdictionary.com*)

Brėžinys (angl. *drawing*) – dokumentas naudojamas grafinės informacijos pateikimui. (BS 1192:2007)

BS 1192 – standartas, apimantis AEC industrijos gamybinės informacijos vystymą, organizavimą ir valdymą, pasitelkiant disciplinuotą bendradarbiavimo ir detalizuotą pavadinimų suteikimo politiką. (*BIMdictionary.com*)

CAD (angl. *Computer-Aided Design*) – kompiuterinis projektavimas. CAD reiškia skaitmeninių įrankių naudojimą objektų ar erdvės kūrimui, keitimui, analizavimui ar optimizavimui. CAD apima visus prieš BIM naudojamus skaitmeninius įrankius ir jų kuriamus 2D bei 3D rezultatus. (*BIMdictionary.com*)

DVS (angl. *Document Management System, DMS*) – dokumentų valdymo sistema.

Metaduomenys (angl. *meta-data*) – duomenys, naudojami dokumentų ir kitų informacijos laukų aprašymui ir valdymui. (BS 1192:2007)

Modelis (angl. *model*) – AEC kontekste apibrėžiamas kaip organizuotų objektų rinkinys, skirtas reprezentuoti objektų, pavyzdžiui pastatų arba mechaninių įrenginių, fizines dalis. Modeliai gali būti dvimačiai (2D) arba trimačiai (3D) bei gali turėti grafinį ir ne grafinį turinį. (BS 1192:2007)

PĮ – programinė įranga.

# ĮVADAS

Informaciniame amžiuje kiekvienas kasdien susiduria su informacija, kurios vienas iš saugojimo bei perdavimo būdų – elektroniniai dokumentai arba kiti skaitmeniniai failai. Jeigu fiziniam asmeniui tenkantis dokumentų kiekis nėra be galo didelis ar nesuvaldomas, organizacijos jau seniai susiduria su dokumentų valdymo problemomis. Dokumentų ir kitų failų kiekiai dažnai yra labai dideli, be to organizacijoms kyla poreikis saugoti ne vieną to paties dokumento versiją, užtikrinti jų saugumą, pasiekiamumą, galimybę dalintis ir t. t. Fizinių dokumentų naudojimas tokiu atveju yra labai brangus ir nepatogus, kadangi reikalauja didelių kaštų ir laiko sąnaudų norint šiuos dokumentus tinkamai valdyti. Šią problemą spręsti padeda programinė įranga – dokumentų valdymo sistemos (DVS), kurios plačiai naudojamos įvairaus dydžio organizacijose. Paprastai dokumentų valdymo sistema privalo turėti keletą svarbiausių funkcijų, tokių, kaip failų saugojimas ir gavimas, failų operacijos, dokumentų versijų kūrimas ar metaduomenų saugojimas. Tačiau DVS tuo neapsiriboja ir dažniausiai turi žymiai platesnį funkcionalumą, kuris gali būti pritaikytas tam tikrai industrijai ar netgi organizacijai. Tokios dokumentų valdymo sistemos vadinamos specializuotomis ir gali būti itin sudėtingos bei turėti daug specifinių funkcijų.

Šiame darbe analizuojamos architektūros, inžinerijos ir statybų, kuri darbe toliau vadinama angliškų akronimu AEC (angl. *Architecture, Engineering & Construction*), industrijai pritaikytos specializuotos dokumentų valdymo sistemos. Dokumentų valdymas AEC organizacijose yra kritinis procesas, kadangi tiesiogiai daro įtaką vykdomų projektų kaštams, kokybei ir įgyvendinimo laikui. AEC industrija pasižymi konservatyvumu, o tai lemia, kad kompanijos nespėja laiku įsisavinti naujų technologijų, padedančių efektyviai valdyti procesus. 2016 m. tarptautinės verslo konsultavimo kompanijos *McKinsey & Company* pateiktoje ataskaitoje atskleidžiama, kad statybų sektorius pagal industrijos skaitmenizavimo indeksą, kuris sudarytas remiantis 27 metrikomis, užima 21-ąją vietą iš 22-jų ir lenkia tik žemdirbystę ir medžioklę (Agarwal *et al.*., 2016). Tai patvirtina, jog skaitmenizavimo lygis industrijoje kol kas yra vienas žemiausių, o tai reiškia, kad šis sektorius turi didelį potencialą išnaudoti skaitmenizavimo galimybes.

Kadangi AEC industrijoje generuojamas didelis kiekis dokumentų – ne tik įprastos dokumentacijos, bet ir specifinių failų – brėžinių bei modelių, kuriems kurti naudojama speciali programinė įranga, viena iš skaitmenizavimo sričių, kuri gali būti išnaudojama AEC industrijoje, ir yra DVS. Dar 2001 m. B.-C. Bjork teigė, kad DVS yra svarbiausia statybų industrijos IT technologija (Bjork, 2001). Bėgant laikui, DVS poreikis tik augo, kadangi šios sistemos gerokai išsiplėtė, atsirado galimybė integruoti DVS su kitomis organizacijos

sistemomis arba praplėsti sistemą prijungiant papildomus modulius ar paslaugas. Industrijos specifika (saviti procesai, specializuotos programinės įrangos palaikymas, projektų sudėtingumas) kelia tam tikrus iššūkius efektyviam dokumentų valdymui bei darbo pasidalijimui. AEC projektai DVS kelia ir kitų reikalavimų, kaip industrijos standartų palaikymas ar teisės aktų laikymasis, kuriuos užtikrinti gali padėti specializuota DVS. Taigi, siekiant užtikrinti industrijos dokumentų saugojimo ir valdymo procesus, vienas svarbiausių įrankių šioje industrijoje ir yra specializuota dokumentų valdymo sistema.

Kaip ir bet kokia kita sistema, dokumentų valdymo sistema turi būti kokybiška. Vienas iš pagrindinių programinės įrangos kūrimo procesų, padedančių užtikrinti programinės įrangos kokybę, yra testavimas. Testavimo metu naudojamos įvairios technikos – lygiai, tipai ir būdai bei remiamasi skirtingomis metodikomis. Kad testavimo procesas nebūtų chaotiškas, neefektyvus ir neapibrėžtas, organizacijoms rekomenduojama naudoti testavimo strategijas. Testavimo strategija padeda daryti testavimo procesą efektyviu, o tai leidžia ir ženkliai sumažinti programinės įrangos kūrimo kaštus. Nepaisant testavimo strategijos suteikiamų pranašumų, šį dokumentą sukuria ir atnaujina toli gražu ne kiekviena PĮ kurianti organizacija.

Kiekvienai sistemai taikoma testavimo strategija yra skirtinga. Egzistuoja ir ne vienas požiūris apie testavimo strategiją ir jos kūrimą – literatūroje jų galima rasti keletą. Šiame darbe apžvelgti trys populiarūs testavimo strategijos kūrimo metodai – ISO 29119 standartas, *TMMI Foundation* modelis bei Euristinis testavimo strategijos modelis. Tačiau, AEC industrijai specializuotos DVS specifiškumas bei kompleksiškumas reikalauja ir specialaus sudėtinio testavimo strategijos sprendimo, nes pritaikyti vieną universalią testavimo metodiką nebūtų efektyvu. Dėl tokios sistemos specifikos ir sudėtingumo, kyla poreikis naudoti būtent specializuotai DVS pritaikytą testavimo strategiją. Tačiau, informacijos apie AEC industrijai pritaikytos DVS testavimo strategijos kūrimą literatūroje rasti nepavyko. Dėl to, kyla poreikis sukurti testavimo strategiją, skirtą būtent tokiai sistemai, kurią būtų galima įgyvendinti specializuotas DVS kuriančiose organizacijose taip paverčiant jų testavimo procesą efektyviu.

Taigi, šio darbo tikslas – sukurti AEC industrijai specializuotos dokumentų valdymo sistemos testavimo strategiją.

Darbo uždaviniai:

1. Išanalizuoti testavimo proceso poreikį ir teorinius aspektus – technikas bei testavimo įrankius, siekiant pasirinkti tinkamą testavimo metodiką, reikalingą kuriant testavimo strategiją.
2. Atlikti testavimo strategijos kūrimo modelių analizę bei, remiantis jų gerosiomis praktikomis, sudaryti specializuotai DVS tinkantį testavimo strategijos kūrimo modelį.

3. Apibrėžti teorinius DVS bei specializuotų DVS aspektus ir naudą organizacijai, siekiant pagrįsti tokios sistemos svarbą AEC organizacijai.
4. Atlikti AEC industrijai specializuotų DVS palyginimo analizę, siekiant nustatyti specializuotos DVS funkcines grupes ir nefunkcinius reikalavimus (kokybės kriterijus).
5. Atlikti specializuotos DVS ypatumų ir gerųjų praktikų analizę, siekiant nustatyti sistemos testavimo strategijos kūrimui įtaką darančius veiksnius.
6. Pritaikyti sudarytą testavimo strategijos modelį ir sukurti specializuotos DVS testavimo strategiją remiantis išanalizuotais sistemos, testavimo technikų ir įrankių, kokybės kriterijų bei projekto aplinkos aspektais.
7. Pateikti išvadas ir rekomendacijas.

Darbo uždaviniams įgyvendinti panaudoti šie metodai: literatūros šaltinių analizė, testavimo strategijos kūrimo modelių, testavimo technikų ir įrankių analizė, gerųjų DVS panaudojimo praktikų AEC industrijoje analizė, specializuotų DVS lyginamoji analizė, specializuotos DVS funkcinių reikalavimų ir ypatumų analizė, specializuotos DVS nefunkcinių reikalavimų ir testavimo projekto aplinkos analizė, testavimo strategijos projektavimas.

Darbe pagrįstas testavimo poreikis ir svarba, testavimo eiga bei išanalizuoti trys testavimo strategijos kūrimo modeliai. Remiantis šių modelių gerosiomis praktikomis, sudarytas specializuotos DVS testavimo strategijos kūrimui tinkantis modelis, kartu atsižvelgiant į strategijos kūrimo apribojimus šio darbo kontekste. Pagrindinės šio modelio elementų grupės yra: testavimo technikos ir įrankiai, testuojamas produktas (specializuota DVS), projekto aplinka ir kokybės kriterijai. Darbe atliktas šių testavimo strategijos kūrimo modelyje apibrėžtų elementų grupių tyrimas. Tyrimo metu, apžvelgti ir palyginti testavimo technikų ir įrankių teoriniai aspektai, pagrįsta DVS nauda AEC organizacijai, atlikta panašių specializuotų DVS lyginamoji analizė, padėjusi nustatyti sistemos funkcines grupes bei jos unikalumą, lyginant su įprasta DVS. Darbe taip pat analizuoti specializuotos DVS ypatumai bei apžvelgti kiti testavimo strategijos kūrimui įtaką darantys veiksniai – kokybės kriterijai bei projekto aplinka. Remiantis šiais atliktais darbais, sukurtas testavimo strategijos dokumentas, skirtas AEC industrijai pritaikytos DVS testavimui. Sukurta strategija gali būti pritaikyta ir panaudota įgyvendinant ją organizacijose, kuriančiose specializuotas DVS. Pagal pateiktas rekomendacijas įdiegus testavimo strategiją, ją galima išplėsti pagal organizacijos ar produkto poreikius, atlikti testavimo brandos vertinimą bei testavimo strategijos efektyvumo organizacijoje tyrimą.

Darbe panaudoti 48 literatūros šaltiniai, pateikta 12 lentelių, 10 paveikslų bei 2 priedai.

# 1. TESTAVIMO PROCESAS IR TESTAVIMO STRATEGIJA

Programinės įrangos testavimo sąvoka atsirado dar XX a. viduryje. 1950 m. A. Turing straipsnis gali būti laikomas pirmuoju, kur buvo paminėtas programų testavimas. 1957 m. D. McCracken knygos *Digital Computer Programming* apžvalgoje, C. Baker atskyrė derinimą (angl. *debugging*) nuo testavimo. (Gelperin, Hetzel, 1998). Nuo to laiko, programinės įrangos testavimas, kaip ir visas programinės įrangos kūrimo procesas, keitėsi ir tobulėjo. Net ir dabar, kaip veikla ar procesas, programinės įrangos testavimas dažnai apibrėžiamas šiek tiek skirtingai. G. J. Myers teigia, kad testavimas yra procesas, kai programa vykdoma siekiant rasti klaidų (Myers, 2004). Y. Singh sako, kad testavimas yra naujai kuriamos programinės įrangos testavimas prieš jos faktinį naudojimą (Singh, 2012).

Testavimas yra platesnio proceso, vadinamo verifikavimu ir patikra (angl. *verification and validation, V&V*) dalis. Verifikavimo tikslas yra patikrinti, kad programinė įranga atitinka nurodytus funkcinius ir nefunkcinius reikalavimus, o patikros – užtikrinti, kad programinė įranga atitinka kliento lūkesčius. (Sommerville, 2011). Testavimo procesas negali vienareikšmiškai užtikrinti, kad programinė įranga neturi klaidų ir veikia tik taip, kaip tikisi naudotojas. Testavimas gali parodyti klaidų egzistavimą (pvz.: nustatyti defektai), tačiau negali garantuoti visiško klaidų nebuvimo. Taigi, testavimas padeda surasti programinės įrangos defektus bei netinkamą jos veikimą, tačiau negali visiškai garantuoti, kad programa neturi jokių klaidų.

Testavimo proceso veiklos gali būti skirstomos į dvi kategorijas: statinę analizę ir dinaminę analizę. Statinė analizė apima reikalavimų dokumentų, programinės įrangos modelių, dizaino dokumentų ir programinio kodo analizę ir peržiūrą. Tokio tipo veiklos metu programinės įrangos kodas nėra vykdomas, t. y. programa nepaleidžiama. Vykdant programinį kodą programa paleidžiama dinaminės analizės metu, galimai programai įvedant įvesties parametrus ir tikintis atitinkamų išvesties parametrų. (Naik, Tripathy, 2008; *ISTQB*, 2018).

Siekiant plačiau aptarti programinės įrangos testavimo procesą, reikalinga apibrėžti pagrindinius testavimo procese ir šiame darbe naudojamus terminus, kurie pateikiami žemiau.

**Defektas** arba **klaida** (angl. *fault; defect; bug*) – dėl tam tikrų priežasčių padaryta klaida programinės įrangos kode (Singh, 2012; *ISTQB,* 2018).

**Triktis** (angl. *failure*) – klaidos vykdymo rezultatas, kai tikėtina programinės įrangos išvestis nesutampa su gauta išvestimi (Singh, 2012).

**Testavimo atvejis** arba **testas** (angl. *test case; test*) – programinei įrangai pateikiamų įvesties ir tikėtinos išvesties veiksmų rinkinys (Singh, 2012).

**Testavimo planas** (angl. *test plan; test suite*) – testavimo atvejų rinkinys (Singh, 2012).

## 1.1. Testavimo poreikis ir tikslas

Programinės įrangos testavimo poreikis kyla pirmiausia dėl žmogiškojo faktoriaus. Programinę įrangą projektuoja ir kuria žmonės, kurie gali ne visada teisingai suprasti reikalavimus, neturėti reikiamų kompetencijų ar dėl kitų priežasčių palikti spragų ir klaidų programinės įrangos kode. PĮ testavimas reikalingas ir dėl aparatinės įrangos naudojimo niuansų bei kitų priežasčių. Programinės įrangos defektai, pasiekiami galutiniam sistemos naudotojui, gali baigtis laiko išteklių praradimu, finansinėmis išlaidomis ar net kelti pavojų. Norint užtikrinti, kad naudotoją pasieks kokybiškas produktas, o jo atliekamos funkcijos bus tokios, kaip ir tikėtasi, atsiranda poreikis testuoti programinę įrangą.

Programinės įrangos sukūrimo kaštai tiesiogiai priklauso nuo defektų skaičiaus ir jų nustatymo laiko. Kuo programinėje įrangoje daugiau defektų, tuo daugiau kaštų reikės jiems pataisyti. Defektų skaičiaus sumažinimas yra vienas efektyviausių būdų sumažinti viso programinės įrangos kūrimo proceso kaštus (Briski *et al.,* 2008). 2002 m. JAV Nacionalinio standartų ir technologijos instituto (NIST) atlikta studija rodo, kad reliatyvūs programinės įrangos defektų taisymo kaštai ženkliai priklauso nuo to, kuriame PĮ kūrimo etape defektai buvo rasti (Tassey, 2002, p. 5-4). Toks defektų taisymo kaštų palyginimas pateikiamas 1-oje lentelėje.

1 lentelė. **Reliatyvūs defektų, rastų skirtinguose programinės įrangos kūrimo etapuose, taisymo kaštai** (šaltinis: Tassey, 2002, p. 5-4)

| **Dizainas ir architektūra** | **Kūrimas / vieneto testavimas** | **Integracijos ir komponentų testavimas** | **Beta versijos testavimas** | **Po produkto išleidimo** |
|---|---|---|---|---|
| 1X* | 5X | 10X | 15X | 30X |
| * X – normalizuotas kaštų vienetas; gali būti išreikštas darbo valandomis, doleriais ir t.t. | | | | |

Kaip matoma 1-oje lentelėje, defekto taisymo kaštai, jeigu jis randamas po produkto išleidimo, gali išaugti netgi 30 kartų. Jeigu defektas buvo rastas ir pataisytas kūrimo stadijoje (t. y. vieneto (angl. *unit*) testavimo metu), jo taisymo kaštai, palyginus su dizaino ir architektūros etapu, išauga penkis kartus. Tuo tarpu vėlesniame etape, integracijos (angl. *integration*) ar komponentų (angl. *component*) testavimo metu rastus defektus pataisyti gali kainuoti 10 kartų daugiau. Programinės įrangos beta versijos metu rasti defektai pabrangsta 15 kartų. Tai reiškia, jog testavimo metu (bet kuriame programinės įrangos kūrimo etape) rasti ir

pataisyti defektai reikalauja ženkliai mažiau kaštų nei po produkto išleidimo, o idealiu atveju defektus reiktų aptikti kuo ankstesniame etape.

Kiek kitokį defektų taisymo kaštų vertinimą pateikė S. McConnell. Vidutiniai defektų taisymo kaštai pagal S. McConnell pateikti 2-oje lentelėje (McConnell, 2004).

2 lentelė. **Vidutiniai defektų taisymo kaštai pagal jų atsiradimo ir aptikimo laiką**
(šaltinis: McConnell, 2004, p. 29)

| **Aptikimo laikas** | | | | | |
|---|---|---|---|---|---|
| **Atsiradimo laikas** | **Reikalavimai** | **Architektūra** | **Kūrimas** | **Sistemos testavimas** | **Po išleidimo** |
| **Reikalavimai** | 1 k.* | 3 k. | 5 – 10 k. | 10 k. | 10 – 100 k. |
| **Architektūra** | – | 1 k. | 10 k. | 15 k. | 25 – 100 k. |
| **Kūrimas** | – | – | 1 k. | 10 k. | 10 – 25 k. |
| * k. – kartai, matavimo vienetas nurodantis, kiek kartų padidėja defektų taisymo kaštai. | | | | | |

Remiantis šiuo vertinimu, defektų taisymo kaštai po išleidimo gali išaugti net iki 100 kartų. Sistemos testavimo etape defektų taisymo kaštai yra 10 – 15 kartų didesni nei reikalavimų kūrimo etape. Tai yra sąlyginai nedideli kaštai, palyginus su defektų taisymo kaina, kai defektai randami po programinės įrangos išleidimo į rinką (nuo sistemos testavimo etapo kaštai gali išaugti iki 10 kartų). Apibendrinant šiuos du defektų taisymo kaštų vertinimo modelius, galima teigti, jog norint sumažinti programinės įrangos kūrimo kaštus, defektus reikia aptikti ir pataisyti kuo anksčiau. Tą padaryti padeda testavimo procesas.

Anksčiau minėtoje JAV Nacionalinio standartų ir technologijos instituto (NIST) atliktoje studijoje apytikriai suskaičiuota, kad JAV ekonomika kiekvienais metais praranda apie 60 milijardų JAV dolerių, kurie skiriami padengti išlaidoms, susijusioms su programinės įrangos pataisų (angl. *patch*) kūrimu ir išleidimu, užkrėstų sistemų diegimu iš naujo, taip pat prarastu produktyvumu, kylančiu dėl kompiuterių kenkėjiškų programų ir kitų problemų, kurioms kelią atveria programinės įrangos klaidos (Zhivich, 2009). Šioje studijoje preliminariai suskaičiuota tik JAV mastu patiriami nuostoliai, be to studija buvo atlikta 2002 m., kai programinės įrangos paplitimas buvo žymiai mažesnis. Žvelgiant į naujesnius šaltinius, atkreiptinas dėmesys į Austrijos programinės įrangos testavimo kompanijos *Tricentis GMBH* ataskaitą, kurioje teigiama, kad 2017 m. dėl programinės įrangos trikčių buvo prarasta apie 1,7 trilijono JAV dolerių. Tačiau, net ir ši studija neatspindi tikrosios statistikos, kadangi kompanija kiekvienais metais analizuoja tik viešus anglų kalba parašytus straipsnius ir naujienų pranešimus apie programinės įrangos triktis. 2017 m. kompanija įvertino 606 įvykusias programinės įrangos

triktis, paveikusias 314 organizacijų. Ataskaitoje skaičiuojama, kad dėl šių trikčių vienaip ar kitaip galėjo būti paliesti apie 3,6 milijardo žmonių. Be to, kompanija teigia, kad 2017 m. bendras programinės įrangos defektų skaičius, palyginus su ankstesniais metais, padidėjo. (*Tricentis GMBH*, 2018). Taigi, globaliai tikrieji dėl programinės įrangos trikčių kylantys nuostoliai nėra apskaičiuoti, tačiau akivaizdu, kad jie yra reikšmingi.

Programinę įrangą kurianti organizacija dėl PĮ defektų patiria ne tik tiesiogines išlaidas joms pataisyti, tačiau rizikuoja ir savo reputacija. Jeigu programinė įranga išleidžiama su klaidomis ar našumo problemomis, potencialiai gali nukentėti organizacijos reputacija ir būti prarastas klientų pasitikėjimas, o prarastas klientų pasitikėjimas gali tapti ir pajamų sumažėjimo priežastimi (Briski *et al.*, 2008). Organizacijai paskelbus apie didelį programinės įrangos defektą (pavyzdžiui, saugumo spragą), nukenčia ne tik organizacijos reputacija, tačiau gali kristi ir akcijų vertė, organizacija gali sulaukti teisinių pasekmių. Taigi, programinės įrangos testavimas reikalingas ne tik norint sumažinti kūrimo kaštus, bet ir nenukentėti finansiškai kitais būdais bei išlaikyti gerą reputaciją.

Programinės įrangos defektai gali atnešti ne tik finansinių ar reputacijos nuostolių, tačiau ir būti nelaimių priežastimi ar sukelti kitus pavojus. Programinės įrangos, naudojamos tokiose jautriose srityse kaip medicina, statybos, transportas, krašto apsauga ar astronautika, defektai gali sukelti pavojų žmonių sveikatai ar gyvybei. Tokių defektų neretai pasitaiko automobilių pramonėje. Pavyzdžiui, 2016 m. po keleto pranešimų apie sužeidimus dėl saugos oro pagalves valdančios programinės įrangos defekto (programinė įranga neatpažindavo, kad keleivio vietoje sėdi suaugęs žmogus, todėl saugos oro pagalvės neišsiskleisdavo), automobilių gamintoja *Nissan* atšaukė 3,5 mln. automobilių (Jensen, 2016). Akivaizdu, kad tokie programinės įrangos defektai yra ne tik pavojingi žmonėms, tačiau ir atneša didelius finansinius nuostolius. Netrūksta pavyzdžių, kai programinės įrangos defektas lėmė ir tragiškus įvykius. 1991 m. JAV armijos priešraketinės gynybos sistemos *Patriot* defektas lėmė, kad sistema nesusekė ir nesulaikė priešo raketos, paleistos į JAV armijos bazę Saudo Arabijoje. Šis triktis raketai leido pasiekti savo tikslą, dėl ko žuvo 28 ir buvo sužeista apie 100 amerikiečių karių (Zhivich, 2009). Istorijoje tokių įvykių, kai dėl programinės įrangos klaidos žmonės buvo sužeisti ar žuvo, yra ir daugiau. Tai įrodo, kad labai svarbu tinkamai atlikti testavimą, nes programinės įrangos defektai, kurie nebuvo rasti testavimo metu, gali sukelti ir tragiškų pasekmių.

Programinės įrangos testavimo poreikio svarba reikalauja tiksliau apibrėžti testavimo veiklos uždavinius. K. Naik ir P. Tripathy išskiria keturis testavimo uždavinius, pateikiamus toliau (Naik, Tripathy, 2008).

1. Parodyti, kad sistema veikia, o ne neveikia.
2. Bandyti priversti sistemą ar jos dalį neveikti.
3. Sumažinti neveikimo riziką iki priimtino lygio.
4. Kurti žemos rizikos programinę įrangą su mažiau testavimo atvejų.

Britų akademikas I. Sommerville išskiria du skirtingus testavimo proceso uždavinius (Somerville, 2011):

1. Įrodyti programuotojui ir klientui, kad programinė įranga atitinka reikalavimus.
2. Surasti situacijas, kuriose programinės įrangos elgsena yra neteisinga, nepageidaujama arba neatitinka savo specifikacijos.

Pirmasis uždavinys veda link patikros (angl. *validation*) testavimo ir naudojant testavimo atvejus, kurie atspindi laukiamą sistemos naudojimą, taip tikintis, kad sistema atliks savo darbą teisingai (Somerville, 2011). Tai reiškia, kad testavimo metu pirmiausiai reikia užtikrinti, jog PĮ elgiasi taip, kaip turėtų elgtis arba atitinka jai iš anksto nustatytus reikalavimus. Tam naudojama statinė analizė bei pagal reikalavimus iš anksto paruošti testavimo atvejai su aiškia veiksmų seka ir rezultatu, kurio tikimasi. Antrasis uždavinys veda link defektų paieškos testavimo, kur testavimo atvejai yra skirti atskleisti sistemos defektus. Įgyvendinant šį testavimo proceso uždavinį, siekiama surasti programinės įrangos defektus – klaidas, spragas, nenumatytą ar netikėtą veikimą. Šiuo atveju testavimo atvejai gali būti sąmoningai neaiškūs ir neprivalo atspindėti normalaus sistemos veikimo. (Sommerville, 2011).

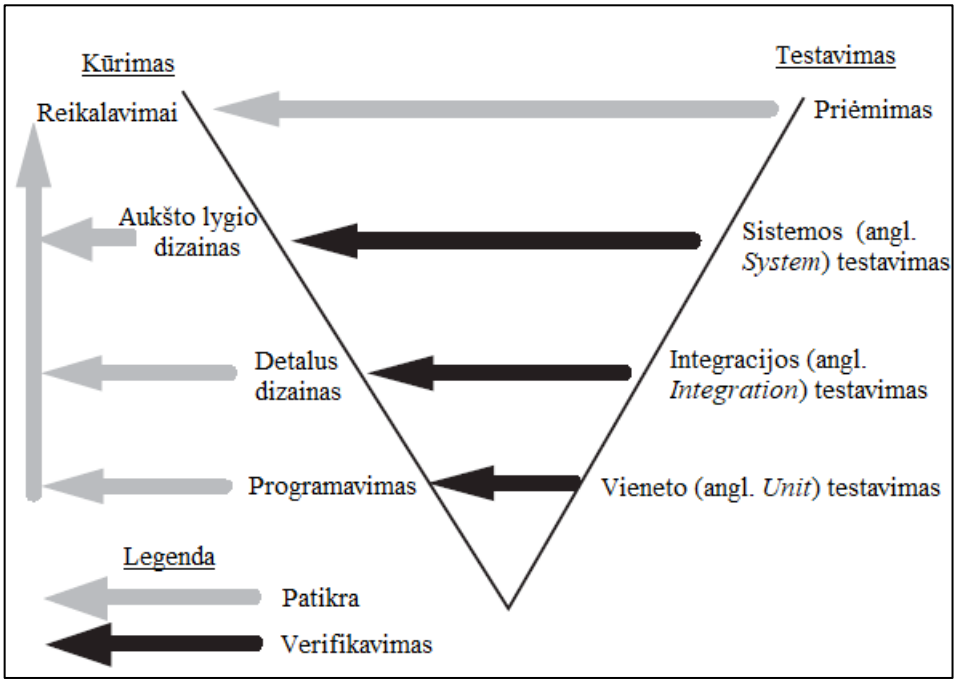
Taigi, skirtingi autoriai nevienodai apibrėžia testavimo proceso uždavinius. I. Sommerville akcentuoja programinės įrangos atitikimą reikalavimams (kitaip sakant, užtikrinimą, kad programinė įranga veikia pagal reikalavimus) ir situacijų radimą, kur elgsena yra neteisinga. K. Naik ir P. Tripathy be šių aspektų akcentuoja ir kitus testavimo proceso uždavinius: ne tik parodyti, jog sistema veikia ar priversti ją neveikti, tačiau ir mažinti sistemos trikčių tikimybę iki priimtino bei kurti tokią programinę įrangą, kuri reikalautų mažiau testavimo. Nepaisant tam tikrų skirtumų tarp testavimo uždavinių, dauguma mokslininkų ir praktikų sutaria, kad pagrindinis testavimo proceso tikslas yra užtikrinti tinkamą programinės įrangos veikimą ir surasti klaidas, dėl kurių sistema neveikia ar veikia netinkamai.

## 1.2. Testavimo proceso eiga

Programinės įrangos testavimo proceso eiga priklauso nuo PĮ kūrimo gyvenimo ciklo modelio. Tarptautinė programinės įrangos testavimo kvalifikacijos sertifikavimo organizacija (*International Software Testing Qualifications Board*, toliau darbe sutrumpintai vadinama *ISTQB*) šiuos modelius skirsto į dvi kategorijas: nuoseklūs (angl. *sequential*) kūrimo modeliai

ir iteracijų (angl. *iterative*) bei inkrementiniai (angl. *incremental*) kūrimo modeliai (*ISTQB*, 2018).

Nuoseklūs kūrimo modeliai apibrėžia programinės įrangos kūrimo procesą kaip linijinį procesą, nuoseklią veiklų seką, kur kiekviena kūrimo fazė prasideda pasibaigus prieš tai buvusiai fazei. Tokio modelio pavyzdys yra Krioklio (angl. *Waterfall*) metodas. Kadangi visos kūrimo veiklos (reikalavimų analizė, projektavimas, programavimas, testavimas) vykdomos iš eilės, testavimo procesas šio modelio atveju prasideda tik pasibaigus programavimo procesui. (*ISTQB*, 2018). Kitas nuoseklaus modelio pavyzdys yra V-modelis. Kitaip nei Krioklio modelis, šis modelis integruoja testavimo procesą į programavimo procesą. Be to, V-modelis apima skirtingus testavimo lygius kiekvienoje atitinkamoje programavimo fazėje. Grafinis V-modelio paaiškinimas pateikiamas 1-ame paveiksle (testavimo lygiai plačiau aprašyti 2.2 skyriuje). Nuoseklūs kūrimo modeliai leidžia pateikti visiškai užbaigto funkcionalumo programinę įrangą, bet paprastai reikalauja ilgo laiko tarpo, kad programinė įranga būtų išleista. (Sommerville, 2011; *ISTQB*, 2018).




1 pav. **Kūrimo ir testavimo fazės V-modelyje**

(šaltinis: Sommerville, 2011; p. 16)

Inkrementinis kūrimas užtikrina sistemos reikalavimų nustatymą, projektavimą, programavimą ir testavimą dalimis, o tai reiškia, kad programinės įrangos funkcionalumas auga palaipsniui. Kūrimas remiantis iteracijomis reiškia, kad ciklų (dažnai nustatytos trukmės) metu apibrėžiama, suprojektuojama, programuojama ir ištestuojama grupė programinės įrangos funkcijų. Kiekviena iteracija sukuria veikiančią programinės įrangos dalį, kuri yra augančio funkcionalumo dalis, kol sukuriama galutinė programinės įrangos versija. Tokių modelių

pavyzdžiai gali būti *Scrum, Kanban,* spiralės modeliai. Sistemos ar komponentai, kuriami pagal šiuos metodus, kūrimo metu dažnai turi persidengiančius ir iteracinius testavimo lygius. Idealiu atveju, kūrimo metu kiekviena funkcija yra testuojama keletu testavimo lygių. (Sommerville, 2011; *ISTQB*, 2018).

Testavimo procesas nėra vien tik testavimo atvejų vykdymas ir defektų ar trikčių paieška. Remiantis *ISTQB*, testavimo procesas susideda iš toliau pateiktų veiklų (*ISTQB*, 2018).

- **Testavimo planavimas**. Apima testavimo objektų, testavimo būdų ir apribojimų nustatymą.
- **Testavimo stebėjimas ir kontrolė**. Apima testavimo progreso stebėjimą pagal iš anksto sudaryta testavimo metrikų planą ir atitinkamų veiksmų ėmimąsi, kad plano tikslai būtų pasiekti.
- **Testavimo analizė**. Testavimo analizės metu analizuojama testavimo bazė siekiant identifikuoti testuojamas funkcijas ir nustatyti testų sąlygas.
- **Testavimo projektavimas**. Apima testavimo sąlygų plėtojimą į aukšto lygio testavimo atvejus, testavimo atvejų rinkinius ir kitą testavimo medžiagą.
- **Testavimo įgyvendinimas**. Apima testavimo procedūrų apibrėžimą, testavimo planų kūrimą, automatizavimą, testavimo duomenų generavimą ir kitus testavimui atlikti reikalingus veiksmus. Testavimo įgyvendinimas dažnai atliekamas kartu su testavimo projektavimu.
- **Testavimo vykdymas**. Pagal vykdymo tvarkaraštį vykdomi testavimo planai, sekamas progresas, analizuojami rezultatai, ruošiamos ataskaitos, registruojami defektai ir t. t.
- **Testavimo užbaigimas**. Surenkami baigtų testavimo veiklų duomenys siekiant įtvirtinti patirtį, testavimo medžiagą ir kitą svarbią informaciją taip patobulinant testavimo procesą. Patikrinama, ar visi testavimo metu rasti defektai buvo uždaryti, archyvuojami testavimo duomenys ir infrastruktūra.

Taigi, testavimo proceso eiga priklauso nuo pasirinkto PĮ kūrimo modelio. Be to, testavimo procesas apibrėžiamas ne tik kaip testavimo vykdymas, tačiau apima ir daugiau su testavimu susijusių veiklų. Todėl kuriant testavimo strategiją, svarbu atsižvelgti ir į kitas testavimo procesui priskiriamas veiklas.

## 1.3. Testavimo strategija ir jos kūrimo modeliai

Nepaisant testavimo procesui apibrėžtų veiklų, testavimo procesas organizacijose gali būti skirtingas. Jis priklauso tiek nuo organizacijos strategijos, tikslų, kuriamų produktų ar kokybės reikalavimų, tiek nuo pasirinktų programinės įrangos kūrimo metodų. Bendrąjį organizacijos požiūrį į testavimą nusako organizacijos testavimo politika. Testavimo politika apibrėžia organizacijos visuotinius testavimo uždavinius, tikslus ir strateginį požiūrį į testavimą, kuris turėtų būti susietas su bendra organizacijos kokybės politika (*TMMi Foundation*, 2018, p. 24). *ISTQB* nurodo, kad testavimo politika yra aukšto lygio dokumentas, aprašantis organizacijos testavimo principus, požiūrį ir pagrindinius tikslus (ISTQB, 2019).

Kadangi testavimo politika apima tik aukšto lygio organizacijos testavimo proceso gaires, reikalingas detalesnis žvilgsnis į testavimo procesą. Kad sistemos testavimo procesas nebūtų chaotiškas ir neefektyvus, svarbu turėti iš anksto apibrėžtą taisyklių, pasirinktų metodų ir būdų rinkinį. Testavimo procesą apibrėžti ir valdyti padeda testavimo strategija. Literatūroje galima rasti ne vieną testavimo strategijos apibrėžimą bei keletą požiūrių, atskleidžiančių testavimo strategijos elementus ir padedančių ją kurti. Literatūros apžvalgos metu buvo pasirinkta keletas šaltinių apie testavimo strategiją, kurie ir aptariami plačiau.

### Testavimo strategija pagal *ISTQB*

*ISTQB* nurodo, kad testavimo strategija yra su testavimo politika susieta dokumentacija, kuri aprašo bendrinius testavimo reikalavimus ir testavimo vykdymo organizacijoje detales (*ISTQB*, 2019). *ISTQB* skirsto testavimo strategiją į skirtingus tipus, tokius kaip analitinis, paremtas modeliu, metodinis ir kt. Derama testavimo strategija kuriama kombinuojant keletą iš šių tipų. Kadangi *ISTQB* dokumentacijoje testavimo strategijos elementai ir kūrimo aspektai detaliau neaprašyti, šiame darbe *ISTQB* požiūris į testavimo strategijos kūrimą plačiau aptariamas nebus.

### Testavimo strategija pagal ISO 29119 standartą

Vienas naujausių ir išsamiausių standartų, skirtų programinės įrangos testavimui, yra ISO 29119 standartas. Šiame standarte aptariamas ir testavimo strategijos kūrimas. Testavimo strategija ISO 29119 (2010) standarte apibrėžta kaip „aukšto lygio organizacijai ar programai (vienam ar keliems projektams) atliktinų testavimo lygių ir testavimo tų lygių ribose aprašymas“ (cituojama iš Kasurinen, 2011, p. 17). Remiantis standartu, testavimo procesas organizacijoje yra skirstomas į tris sluoksnius: organizaciniai testavimo procesai (testavimo politika ir strategija), testavimo valdymo procesai (planavimas, priežiūra, kontrolė ir užbaigimas) bei pamatiniai testavimo procesai (veiklos susijusios su testavimo atvejais bei jų

vykdymu) (Kasurinen, 2011, p. 17). Kadangi šiame standarte testavimo strategija suprantama kaip organizacijos procesas, teigiama, kad organizacija gali turėti tik vieną testavimo strategiją (kaip ir politiką).

ISO 29119 (2010) standarte nurodyta, kad testavimo strategija turėtų susidėti iš tokių elementų (cituojama iš Kasurinen, 2011, p. 25): bendrinis rizikų valdymas, įeities ir išeities kriterijai, nepriklausomumo laipsnis, testavimo organizacijos struktūra, testavimo dokumentacijos strategija, testavimo fazės, testavimo tipai, testavimo technikos, testų pasirinkimas ir prioretizavimas, testavimo aplinkos strategija, testavimo automatizavimas ir įrankiai, pakartotinis ir regresinis testavimas, testavimo užbaigimo kriterijai ir incidentų valdymo strategija. Akivaizdu, kad pagal ISO 29119 standarto požiūrį, testavimo strategija yra suprantama kaip labai platus apibrėžimas, padengiantis daug įvairių sričių. Toks požiūris yra gana komplikuotas ir praktikoje jį įgyvendinti nebūtų paprasta. Šis standartas susilaukė nemažai kritikos iš testavimo bendruomenės bei verslo organizacijų. Keletas iš kritikos priežasčių ir buvo standarto kompleksiškumas, per didelė orientacija į dokumentaciją ir dėmesys detalėms, kurios nėra tiek svarbios.

Taigi, ISO 29119 standarte apibrėžta testavimo strategija padengia labai platų su testavimo procesu susijusių veiklų spektrą ir yra suprantama kaip visos organizacijos, o ne atskirų produktų testavimo strategija.

### *TMMi Foundation* testavimo strategijos modelis

Kitas šaltinis, kuriame kalbama apie testavimo strategiją ir jos kūrimą yra „Testavimo brandos modelio integracijos“ gidas, kurį sukūrė ir atnaujina ne pelno siekianti organizacija *TMMi (Test Maturity Model Integration) Foundation*. Organizacija teigia, kad testavimo strategija yra kiekvieno projekto testavimo veiklų atspirties taškas (*TMMi Foundation,* 2018, p. 24), todėl testavimo strategijos procesų sritis yra aprašyta dar antrame brandos modelio lygyje iš penkių. Tai rodo, kad testavimo strategija yra vienas svarbiausių aspektų norint pasiekti testavimo brandą, nes be testavimo strategijos organizacija nepasiektų netgi antro brandos lygio iš penkių. Taigi, prieš pradedant testavimo proceso veiklas, privalu turėti testavimo strategiją, kuria remiantis ir būtų galima tinkamai vykdyti sistemos testavimą.

*TMMi Foundation* nurodo, kad organizacija, kuriai svarbu pagerinti testavimo procesą, pirmiausia turėtų aiškiai nustatyti savo testavimo politiką. Remiantis nustatyta visuotine testavimo politika, kuriamos testavimo strategijos, kurios gali būti pritaikytos projektams, programoms (projektų grupėms) ar produktams, priklausomai nuo organizacijos verslo specifikos. Taigi, kitaip nei ISO 29119 standarto atveju, *TMMi Foundation* modelis neapriboja testavimo strategijos kaip visos organizacijos strategijos.

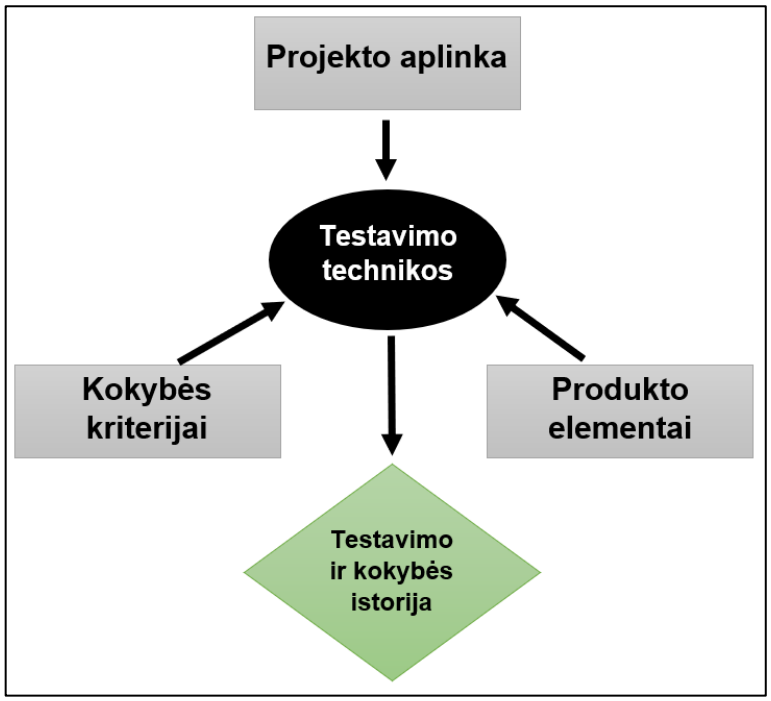
Remiantis *TMMi Foundation* modeliu, įprastai testavimo strategija susideda iš pritaikytinų testavimo lygių aprašymo. Kiekvieno lygio aprašymas turėtų susidėti bent jau iš tokių dalių: uždaviniai, atsakomybės, pagrindinės užduotys ir įvesties bei išvesties kriterijai. Gera testavimo strategija turėtų padengti ir daugiau sričių. *TMMi Foundation* modelyje nurodytas pavyzdinis testavimo strategijos padengiamų temų sąrašas pateiktas toliau (*TMMi Foundation*, 2018, p. 27).

- Bendrinės produkto kūrimo rizikos.
- Bendras testavimo modelis (V-modelis, inkrementinis modelis ar pan.).
- Uždaviniai, atsakomybės ir pagrindinės užduotys kiekviename testavimo lygyje.
- Testavimo atvejų, naudojamų kiekviename testavimo lygyje, kūrimo technikos.
- Testavimo tipai, kurie bus vykdomi kiekviename testavimo lygyje.
- Įvesties ir išvesties kriterijai kiekviename testavimo lygyje.
- Standartai, kurių privaloma laikytis.
- Testavimo nepriklausomumo lygis.
- Testų vykdymo aplinka.
- Požiūris į automatizavimą kiekviename testavimo lygyje.
- Požiūris į regresinį testavimą.

Remiantis šiomis testavimo strategiją padengiančiomis temomis, galima aiškiai pastebėti, jog *TMMi Foundation* modelio centrinė ašis yra testavimo lygiai. Be bendrųjų testavimo strategijos kūrimo aspektų, kaip rizikos ar standartai, modelyje akcentuojami testavimo strategijos faktoriai skirtinguose testavimo lygiuose. Detalesnis testavimo lygių paaiškinimas ir palyginimas pateikiamas 2.2 skyriuje.

**Euristinis testavimo strategijos modelis**

Dar vienas testavimo strategijos kūrimą aprašantis metodas yra Euristinis testavimo strategijos modelis. Šį modelį sukūrė amerikiečių testuotojas, keleto knygų apie testavimą autorius, nepriklausomas konsultantas bei įmonės *Satisfice, Inc* įkūrėjas J. Bach. Euristinis testavimo modelis yra periodiškai atnaujinamas remiantis gerosiomis praktikomis, o paskutinė 5.7.2 versija buvo išleista 2019 m. J. Bach teigia, kad Euristinis testavimo strategijos modelis yra taisyklių bei šablonų rinkinys, padedantis sukurti bei pasirinkti atliekamus testus. Kadangi modelis yra šablonų rinkinys, jis turėtų būti pritaikomas bei modifikuojamas kiekvienam atvejui individualiai. (Bach, 2019). Grafinė šio modelio diagrama pateikta 2-ame pav.




2 pav. **Euristinis testavimo strategijos modelis**

(šaltinis: Bach, 2019)

2-ame pav. pateiktas Euristinis testavimo modelis susideda iš 4 elementų grupių bei testavimo ir kokybės istorijos. Toliau šie aspektai aptariami plačiau.

- **Testavimo technikos** yra modelio centre, jos apibūdina, kokius testavimo metodus, būdus ar tipus naudoti. Modelyje pateikti testavimo technikų pavyzdžiai yra funkcinis testavimas, stresinis testavimas, automatinis testavimas ir t. t. Autorius pabrėžia, kad testavimo technikas galima kombinuoti taip gaunant didelę jų įvairovę, todėl specifinių testavimo technikų kiekis gali būti begalinis. Testavimo technikų parinkimas reikalauja tam tikros projekto aplinkos, produkto elementų bei kokybės kriterijų analizės, kurie modelyje ir supa testavimo technikas. (Bach, 2019).
- **Projekto aplinka** apima testavimo konteksto faktorius, tokius kaip resursai (testavimo komanda), turima įranga ir įrankiai, tvarkaraštis, pateiktini testavimo rezultatai ir kiti projekto aplinką sudarantys faktoriai, darantys įtaką testavimo procesui (Bach, 2019). Projektuojant testavimo strategiją, svarbu apsibrėžti viso projekto aplinką bei išorinius aspektus, kurie daro įtaką pačiam produktui bei jo testavimo strategijos kūrimui.
- **Produkto elementai** – tai testuojamo produkto dimensijos. Testuojant tam tikrą sistemą, privaloma suprasti jos struktūrą, funkcionalumą, naudojamus duomenis, platformą ir kitus produkto elementus. (Bach, 2019). Tik gerai išanalizavus patį

produktą, galima pritaikyti tinkamas testavimo technikas ir sukurti efektyvią testavimo strategiją.

- **Kokybės kriterijai –** tai tam tikri reikalavimai, kurie apibrėžia, kokius kokybės kriterijus turėtų atitikti produktas. Modelyje pateikti kokybės kriterijų pavyzdžiai yra tokie: patikimumas, panaudojamumas, patrauklumas, saugumas, plečiamumas, suderinamumas, efektyvumas ir kt. (Bach, 2019). Taigi, žvelgiant iš sistemų projektavimo perspektyvos, kokybės kriterijus daugeliu atvejų galima susieti su nefunkciniais sistemos reikalavimais. Nefunkciniai sistemos reikalavimai reikalingi kuriant testavimo strategiją tam, kad būtų galima apibrėžti kokybės kriterijus.
- **Testavimo ir kokybės istorija** yra testavimo rezultatas. J. Bach modelio aprašyme teigia, kad atliekant testavimą pagal suprojektuotą testavimo strategiją, atliekamas sistemos vertinimas, kuris ir yra tam tikra istorija, į kurią įeina ir aptikti defektai, neįprasti objektai ir t. t. (Bach, 2019).

Euristinis testavimo strategijos modelis, lyginant su ISO 29119 standartu ar *TMMi Foundation* modeliu, yra labiau struktūrizuotas, be to kiekvienai modelio elementų grupei yra pateikti elementų pavyzdžiai. Dėl šių priežasčių, Euristinį testavimo modelį yra paprasčiau modifikuoti ir pritaikyti praktiškai.

Apžvelgus tris testavimo strategijos kūrimo požiūrius (ISO 29119, *TMMi Foundation* modelį bei Euristinį testavimo strategijos modelį), galima daryti išvadą, kad visi požiūriai yra gana skirtingi, tačiau padeda atlikti tą patį uždavinį – sukurti testavimo strategiją, kuri padėtų testavimo procesą paversti efektyvesniu. Šie trys požiūriai iš esmės skiriasi savo padengiamų temų kiekiu ir detalumu. *TMMi Foundation* modelis taip pat naudingas tuo, kad pateikia ne tik testavimo strategijos kūrimo temas, bet ir žingsnius, kaip šią strategiją įgyvendinti organizacijoje ir kaip įvertinti testavimo brandą. Labiausiai išsiskiria ISO 29119 siūlomas testavimo strategijos požiūris, kadangi standarte pateikiama strategija yra labai plati, be to šis standartas, kitaip nei kiti analizuoti modeliai, teigia, kad testavimo strategija organizacijoje gali būti tik viena.

Testavimo strategija ne tik apibrėžia testavimo procesą, bet ir padeda jį kontroliuoti. Testavimo strategijos įgyvendinimas padeda testavimo procesą daryti efektyvesniu. Kai testavimo strategija yra apibrėžta ir jos laikomasi, mažiau tikėtini testavimo lygių persidengimai. Taip pat, kadangi testavimo uždaviniai ir kiekvieno testavimo lygio metodai yra aiškiai susieti, tai padeda išvengti testavimu nepadengtų sričių ir pagerinti produkto kokybę. (*TMMi Foundation*, 2018, p. 24).

## 1.4. Testavimo strategijos kūrimo apribojimai ir tyrimo metodika

Šio darbo objektas, kuriam bus kuriama testavimo strategija yra AEC industrijai specializuota dokumentų sistema. Darbas nėra orientuotas į konkrečios organizacijos ar sistemos testavimo strategijos kūrimą. Kadangi kiekvienas kuriamos strategijos atvejis yra unikalus (skiriasi organizacijos strateginiai tikslai, kokybės reikalavimai, testavimo politika, turimi resursai, programinės įrangos kūrimo metodai, sistemos architektūra ir t. t.), visų galimų atvejų padengti neįmanoma. Todėl siekiama sukurti universalią testavimo strategiją specifiniam produktui – būtent AEC industrijai pritaikytai DVS, kurią vėliau būtų galima papildyti ar koreguoti pritaikant konkrečiai organizacijai ar sistemai. Remiantis atliktomis tokių sistemų funkcionalumo palyginimo bei ypatumų analizėmis (aprašyta 3.4 ir 3.5 skyriuose), galima teigti, kad AEC industrijai specializuota DVS yra kompleksinė sistema, todėl testavimo strategija projektuojama sistemai, kuri yra kuriama pagal kliento – serverio architektūrą ir skirta *Windows* operacinėms sistemoms. Šie apribojimai kelia tam tikrų sąlygų ir testavimo strategijos kūrimo modelio pasirinkimui.

Dėl aprašytų apribojimų, šio darbo kontekste vieno iš aptartų testavimo strategijos kūrimo modelių pasirinkimas nėra galimas, nes kiekvienas iš jų rekomenduoja padengti tam tikras temas, kurios priklausytų nuo konkretaus atvejo, be to ne visi modeliuose aptariami aspektai aktualūs specializuotai DVS. Dėl šių priežasčių nuspręsta sudaryti testavimo strategijos kūrimo modelį, tinkantį specializuotai DVS. Iš to kyla ir darbo tikslas – sukurti AEC industrijai specializuotos dokumentų valdymo sistemos testavimo strategiją. Kuriant modelį daugiausiai remtasi Euristiniu testavimo strategijos modeliu (atsisakant tam tikrų elementų, kurie negali būti padengti dėl aprašytų apribojimų), tačiau neapsiribota vien tik šiuo modeliu bei panaudotos ir kitų dviejų modelių gerosios praktikos. Sudarytas specializuotos DVS testavimo strategijos kūrimo modelis pateikiamas 3-iame pav.


3 pav. **Specializuotos DVS testavimo strategijos kūrimo modelis**

(sudaryta autoriaus)

3-iame pav. pateiktame specializuotos DVS testavimo strategijos modelyje matoma, kad testavimo strategijos kūrimui reikalinga suprasti testuojamo produkto funkcionalumą ir ypatybes, projekto aplinką bei nustatytus kokybės kriterijus. Kokybės kriterijai iš dalies persidengia su projekto aplinka, kadangi kriterijai gali būti nustatomi ne tik iš projekto išorės, bet ir pagal projekto aplinkos veiksnius. Remiantis šiomis trimis strategijos elementų grupėmis, vėliau parenkamos testavimo technikos ir įrankiai. Visų šių elementų grupių visuma ir sudaro

specializuotos DVS testavimo strategiją. Pasirinktos modelio elementų grupės toliau aptariamos detaliau.

**Testavimo technikos ir įrankiai**

Bene svarbiausias akcentas, aprašomas kiekviename iš aptartų testavimo strategijos požiūrių, yra pasirinkta testavimo metodika. Būtent testavimo metodika, kuriant testavimo strategiją, padės atsakyti į klausimą „kaip testuoti sistemą?“ Apžvelgus strategijos kūrimo modelius, galima daryti išvadą, jog pagrindiniai testavimo teoriniai aspektai, kurie turėtų būti pasirinkti projektuojant strategiją, yra testavimo metodai, lygiai, tipai ir būdai. Šiame darbe jie apibendrintai vadinami testavimo technikomis. Su testavimo technikų pasirinkimu taip pat glaudžiai siejasi ir testavime naudojami įrankiai. Žinoma, dėl apribojimų kuriant strategiją, konkretūs įrankiai nėra siūlomi, tačiau nurodomi naudotini testavimo įrankių tipai ir jų naudojimo tikslas. Testavimo technikoms ir įrankiams analizuoti darbe pasitelktas literatūros analizės metodas.

**Produktas (AEC industrijai specializuota DVS)**

Kita labai svarbi testavimo strategijos dimensija – pati sistema (produktas), kuriam ir projektuojama testavimo strategija. Tik gerai suprantant sistemą, jos funkcionalumą, išskirtinumą bei ypatybes galima sukurti gerą testavimo strategiją. Būtent produkto suvokimas leidžia parinkti tinkamas testavimo technikas, be to padeda užtikrinti visapusišką produkto padengimą testavimo atvejais. Taigi, kitas privalomas testavimo strategijos modelio elementas – AEC industrijai specializuotos DVS.

Testuojamam produktui analizuoti darbe atliekamas specializuotos DVS tyrimas. Kadangi strategijos kūrimas reikalauja atskleisti sistemos funkcinius reikalavimus, šiai užduočiai įgyvendinti pasirinkta panašių sistemų lyginamoji analizė. Atlikus lyginamąją analizę, nustatytas sistemos funkcionalumas sugrupuojamas į atskiras funkcines grupes. Siekiant detaliau apžvelgti testuojamo produkto ypatumus ir potencialią įtaką testavimo strategijos kūrimui, taip pat atliekama geruųjų DVS panaudojimo praktikų AEC industrijoje analizė bei sistemos funkcionalumo ypatumų analizė, apžvelgianti specifinį tokios sistemos funkcionalumą ir skirtumus nuo įprastos DVS.

**Projekto aplinka**

Kaip akcentuojama aptartuose testavimo strategijos kūrimo modeliuose, testavimo strategijos kūrimui įtakos turi ir testuojamo objekto aplinka. Dėl nustatytų apribojimų, ši elementų grupė nebus analizuojama labai giliai ir atsižvelgiama tik į tuos elementus, kuriuos

galima apibrėžti darbo kontekste – suinteresuotas šalis, rizikas, testų vykdymo aplinką ir testavimo duomenis, defektų ir testų valdymo aspektus bei industrijos reikalavimus.

**Kokybės kriterijai**

Testavimo strategijos kūrimo metu svarbu žinoti kokybės kriterijus, t. y. ko tikimasi iš sistemos nefunkcine prasme, kaip ji turėtų veikti ar būti suderinama su kitomis sistemomis ir pan. Sistemai keliami kokybės kriterijai apibūdina, kokie nefunkciniai reikalavimai turės būti užtikrinti testavimo metu. Remiantis analizuotais modeliais, aprašytais apribojimais bei specializuotos DVS tyrimu, kuriant testavimo strategiją pasirinkta apžvelgti tokius kokybės kriterijus: suderinamumas, diegimo ir plečiamumo reikalavimai, panaudojamumas ir patrauklumas, lokalizavimas, saugumas bei pasiekiamumas ir greitaveika. Kokybės kriterijams, kaip ir testuojamo produkto funkcionalumui, pasitelktas toliau tęsiamas specializuotos DVS tyrimas – gerųjų praktikų analizė bei nefunkcinių reikalavimų analizė.

Galutiniam rezultatui, tai yra specializuotos DVS testavimo strategijai sukurti darbe naudojamas projektavimo metodas, kuris įgyvendinamas pagal sudarytą testavimo strategijos kūrimo modelį. Remiantis sukurtu modeliu, šio darbo metu testavimo strategijos kūrimas vykdomas pagal toliau pateiktą tvarką.

1. Testavimo technikų ir įrankių analizė.
2. Specializuotos DVS (testuojamo produkto) tyrimas.
3. Projekto aplinkos ir kokybės kriterijų apibrėžimas.
4. Testavimo strategijos dokumento kūrimas.

# 2. TESTAVIMO TECHNIKOS IR ĮRANKIAI

Vienas svarbiausių elementų kuriant testavimo strategiją yra testavimo technikos. Kadangi testavimo metodų, tipų ir būdų yra labai įvairių, svarbu pasirinkti tinkamiausius, kad testavimo procesas būtų aiškus ir efektyvus. Pasirinktas testavimo technikas įgyvendinti padeda testavimo procese naudojami įrankiai. Taigi, šiame skyriuje analizuojami testavimo technikų ir įrankių teoriniai aspektai, kurių pasirinkimas svarbus kuriant testavimo strategiją.

## 2.1. Testavimo metodai

Programinės įrangos testavimas skirstomas į skirtingus metodus. G. J. Myers, taip pat ir K. Naik su P. Tripathy, išskiria du pagrindinius metodus, pavadintus „dėžės“ principu: baltosios dėžės (angl. *white box*) ir juodosios dėžės (angl. *black box*) (Myers, 2004; Naik, Tripathy, 2008). Naujesniuose šaltiniuose išskiriamas ir trečias testavimo metodas – pilkosios dėžės (angl. *gray box*) testavimas (Sneha, Malle, 2017). Šis metodas yra efektyvi juodosios ir baltosios dėžių testavimo metodų kombinacija (Acharya, Pandya, 2013). Konceptualus šių trijų testavimo metodų palyginimas grafiškai pateikiamas 4-ame paveiksle.


4 pav. **Trijų testavimo metodų konceptualus palyginimas**
(šaltinis: Acharya, Pandya, 2013)

Juodosios dėžės testavimo metu testuotojas neturi jokių žinių apie vidinį programinį kodą ar kaip programinė įranga sukurta (Sneha, Malle, 2017). Tai reiškia, kad testuotojas suinteresuotas tik išorine programinės įrangos dalimi, tai yra įvestimi ir grafinėje sąsajoje matoma išvestimi (Naik, Tripathy, 2008). Baltosios dėžės testavimas, savo ruožtu, yra priešingas juodosios dėžės testavimui, kadangi vidiniai komponentai yra visiškai žinomi ir pasiekiami testavimo metu. Baltosios dėžės testavimo metu žiūrima į programinį kodą siekiant aptikti kitus galimus testus (Sommerville, 2011). Pilkosios dėžės testavimas yra tarpinis variantas tarp juodosios ir baltosios dėžės testavimo metodų. Testuotojai, naudojantys pilkosios

dėžės testavimo metodą, turi šiek tiek žinių apie programinės įrangos programinį kodą, tačiau ne visas, pvz. nežinoma, kaip sąveikauja sistemos komponentai (Sneha, Malle, 2017).

Platesnis testavimo metodų palyginimas pateikiamas 1-ame priede. Remiantis šiuo metodų palyginimu, galima daryti išvadą, kad kiekvienas testavimo metodas yra tinkamas ir naudojamas skirtingo tipo testavimuose. Juodosios dėžės testavimas labiau tinkamas grafinės naudotojo sąsajos, išvesties defektams aptikti, be to šio testavimo metu rezultatai gaunami greičiau, kadangi jis reikalauja mažiau laiko. Baltosios dėžės testavimas gerai tinka algoritmų, programinio kodo spragų, saugumo testavimui, tačiau užima daugiau laiko ir reikalauja kvalifikuotų išteklių. Pilkosios dėžės testavimas gali būti naudingas rasti stambius programinės įrangos defektus, kadangi remiamasi aukšto lygio programinės įrangos dizaino diagramomis. Taigi, skirtingi testavimo metodai gali padėti ištestuoti PĮ išsamiau, be to, atskleisti skirtingų tipų defektus.

## 2.2. Testavimo lygiai ir tipai

Programinės įrangos testavimas atliekamas skirtingais lygiais. Apibendrinus, bet kuri PĮ sistema pereina keturis testavimo lygius prieš ją diegiant (Naik, Tripathy, 2008; Singh, 2012, p. 368). Šie keturi lygiai yra vieneto, integracijos, sistemos ir priėmimo testavimas (Naik, Tripathy, 2008). Žemiau esančiame sąraše šie testavimo lygiai aptariami plačiau.

- **Vieneto (angl. *unit*) testavimas**. Naudojant funkcines ir (arba) struktūrines testavimo technikas testuojami nepriklausomi programinės įrangos vienetai. Toks vienetas yra mažiausias įmanomas programinio kodo eilučių rinkinys, kurį galima testuoti. Šie testai naudojami pamatiniams PĮ vienetams testuoti, pvz. moduliams, metodams ar pavieniams komponentams. Paprastai, šie testai rašomi programuotojų ir programinis kodas koreguojamas tol, kol visi vieneto testai būna sėkmingi. (Singh, 2012, p. 368; Sneha, Malle, 2017).
- **Integracijos (angl. *integration*) testavimas**. Sujungiama du ar daugiau programinės įrangos vienetų ir testavimas vykdomas siekiant rasti problemų tarp šių vienetų integracijų. Integracinis testavimas atliekamas tik po vieneto testavimo. Šiame testavime dirbama su savarankiškų programinės įrangos modulių (pvz.: procedūra, klasė, funkcija) integracija tarpusavyje, testuojama, kaip skirtingi moduliai veiktų vienoje posistemėje. (Naik, Tripathy, 2008, p. 158; Singh, 2012, p. 368).
- **Sistemos (angl. *system*) testavimas**. Testuojama visa vientisa sistema, naudojami funkciniai ir nefunkciniai testavimo tipai. Šis testavimo lygis parodo visos sistemos kokybę, kadangi tikrinami visi reikalavimai, atliekami patikimumo, saugumo, prieinamumo ir kiti svarbūs testavimai. (Singh, 2012, p. 368; Sneha, Malle, 2017).

- **Priėmimo (angl. *acceptance*) testavimas**. Paskutinis priėmimo testavimo lygis, atliekamas kliento arba naudotojo. Pagrindinis šio testavimo tikslas nėra rasti defektų, o užtikrinti, kad sistema iš galutinio naudotojo pusės veikia taip, kaip tikėtasi. (Singh, 2012, p. 368; Sneha, Malle, 2017).

Pagal testuojamą programinės įrangos dalį ar funkcionalumą, testavimas skirstomas į nemažai įvairių tipų. Vieno neginčijamo šaltinio, apibrėžiančios galutinį tokių tipų sąrašą nėra, be to testavimo tipai dažnai priklauso ir nuo testuojamos sistemos (darbalaukio programa, internetinis puslapis, mobili aplikacija, serveris ir t.t.). Kita vertus, testavimo tipus galima skirstyti pagal testavimo atvejų tipą. Tą padarė G. J. Myers (2004, p. 98), suskirstydamas testavimo tipus į 15 kategorijų. Šis sąrašas su trumpu paaiškinimu ir kiekvienam tipui būdingų savybių aprašymu pateikiamas žemiau.

- **Sugebėjimo (angl. *facility*) testavimas**. Tikrinama, ar programinė įranga gali atlikti iš anksto užsibrėžtus tikslus. Šiam testavimo tipui atlikti netgi nebūtinas kompiuteris, kadangi galima lyginti tikslus ir dokumentaciją. (Myers, 2004). Naujesnėje literatūroje dažnai apibrėžiamas kaip tiesiog funkcionalumo reikalavimų tikrinimas.
- **Apimties (angl. *volume*) testavimas**. Tai toks testavimo tipas, kur programai pateikiami didelės apimties duomenys ir jų kiekiai. Apimties testavimo tikslas yra pamatyti, kaip PĮ susitvarkys su jos ribas siekiančiais ar viršijančiais duomenų kiekiais. (Myers, 2004).
- **Streso (angl. *stress*) testavimas**. Streso testavimo metu PĮ sukeliama didelė apkrova arba kitaip, stresas. Streso testavimas skiriasi nuo apimties testavimo tuo, kad streso testavimo metu sistemai tenka apdoroti maksimalius duomenų srautus per trumpą laiką. (Myers, 2004; Khan, 2010).
- **Panaudojamumo (angl. *usability*) testavimas**. Panaudojamumas matuoja naudotojo pasitenkinimą sistema. Taip pat aiškinamasi, ar naudotojui lengva naudotis PĮ, mokytis ir ją suprasti. Pavyzdžiui, internetinės svetainės tinkamumo testavimas apima 6 svarbiausius kriterijus: tikslumą, efektyvumą, išbaigtumą, galimybę mokytis, pasitenkinimą ir pagalbos bei dokumentacijos aiškumą ir tikslumą. (Singh, 2012).
- **Saugumo (angl. *security*) testavimas**. Saugumo testai užtikrina, kad sistema atitinka pagrindinius saugumo reikalavimus: autentifikavimą, prieigos kontrolę, vientisumą, informacijos atskleidimą, patikimumą, duomenų šaltinio tikrumą. (Singh, 2012).
- **Efektyvumo (angl. *performance*) testavimas**. Efektyvumo testavimo tikslas yra įvertinti programinės įrangos efektyvumą atspindint realaus pasaulio scenarijus. Šio testavimo metu tikrinamas atsako greitis, taip pat sistemos apkrova. Efektyvumo

testavimas dažnai įvardijamas kaip platesnis procesas, apimantis apkrovos ir streso testavimo tipus. (Singh, 2012; Khan, 2010).

- **Atmintinės (angl. *storage*) testavimas**. Atmintinės testavimo metu tikrinama, ar programa teisingai naudoja laikinąją ir nuolatinę kompiuterio atmintį. (Myers, 2004).
- **Konfigūracijos (angl. *configuration*) testavimas**. Konfigūracijos testavimas yra testavimas, kai naudojamos skirtingos PĮ palaikomos aparatinės įrangos konfigūracijos, pavyzdžiui skirtingi atminties dydžiai, įvesties – išvesties įrenginiai. (Myers, 2004).
- **Suderinamumo / konfigūracijos / konvertavimo (angl. *compatibility / configuration / conversion*) testavimas**. Šis testavimo tipas užtikrina, kad programinė įranga gali būti suderinama veikti su senesnėmis versijomis arba atnaujinta iš senesnės į naujesnę. Taip pat testuojamas programinės įrangos suderinamumas su skirtingomis veikimo aplinkomis: operacinėmis sistemomis, mobiliaisiais įrenginiais, interneto naršyklėmis, duomenų bazėmis ir pan. (Myers, 2004; Singh, 2012).
- **Diegimo galimybių (angl. *installability*) testavimas**. Kadangi kai kurios sistemos turi sudėtingą diegimo procedūrą, diegimo galimybių testavimas padeda užtikrinti, kad šis procesas vyktų sklandžiai. (Myers, 2004).
- **Patikimumo (angl. *reliability*) testavimas**. Šis testavimo tipas padeda užtikrinti programos patikimumo reikalavimus, tokius kaip privalomas veikimo laikotarpis valandomis ar procentais (pvz.: per dieną, metus). (Myers, 2004).
- **Atkūrimo (angl. *recovery*) testavimas**. Atkūrimo testavimo metu testuojamas programinės įrangos atkūrimo scenarijus po patirtos aparatinės įrangos klaidos, duomenų praradimo ir pan. Šis testavimo tipas padeda sumažinti vidutinį laiką, reikalingą programinės įrangos atkūrimui (angl. *mean time to recovery*, *MTTR*). (Myers, 2004).
- **Tinkamumo aptarnauti (angl. *serviceability*) testavimas**. Tokio testavimo metu tikrinamos PĮ tinkamumo aptarnauti charakteristikos, pvz.: vidutinis laikas reikalingas problemos derinimui, priežiūros charakteristikos, vidinė dokumentacija. (Myers, 2004).
- **Dokumentacijos (angl. *documentation*) testavimas**. Dokumentacijos testavimas apima programinės įrangos kūrimo metu aprašytos dokumentacijos, tokios kaip diegimo ar pradedančiojo gidas, pamokos, vedliai ir kiti naudotojo vadovai, testavimą. (Singh, 2012).

- **Procedūrų (angl. *procedure*) testavimas**. Tai toks testavimas, kai tikrinamos sistemos naudotojų (pvz., sistemos administratoriaus) su programine įranga atliekamos procedūros. (Myers, 2004).

Be šių testavimo tipų dar dažnai išskiriamas programinės įrangos internacionalizacijos arba lokalizacijos testavimas. Kadangi programinės įrangos pritaikymas kitoms rinkoms gali būti defektų, ypač grafinėje naudotojo sąsajoje, priežastimi, šio testavimo tipo metu tikrinama tarptautinei rinkai ar konkrečiai šaliai (arba kalbai) pritaikyta PĮ.

Kitas testavimo tipų skirstymo būdas yra priskiriant testavimo tipus funkciniam bei nefunkciniam testavimui. Nefunkcinis sistemos testavimas apima nefunkcinių sistemos reikalavimų patikrą ir padeda patikrinti, „kaip gerai“ sistema atlieka savo darbą. Funkcinis testavimas apima visus testavimo tipus, kur testuojamos sistemos atliekamos funkcijos. Funkcinis testavimas vykdomas remiantis funkciniais sistemos reikalavimais ir gali būti skirstomas į toliau pateiktus tipus.

- **Dūmų (angl. *smoke*) ir paviršinis (angl. *sanity*) testavimas.** Dūmų ar paviršinio testavimo metu, testai vykdomi kertiniam komponento ar sistemos funkcionalumui ir stabilumui patikrinti, taip įsitikinant, kad sistema veikia tinkamai ir galima pradėti tolimesnį jos testavimą (*ISTQB,* 2019).
- **Regresijos (angl. *regression*) testavimas.** Su pakeitimais susijęs testavimo tipas, kurio metu siekiama išsiaiškinti, ar dėl naujo sistemos funkcionalumo arba kitų pakeitimų įtakos neatsirado defektų nepakeistose sistemos srityse (*ISTQB,* 2019).
- **Beta testavimas.** Priėmimo testavimo tipas, kai išleidžiama ir sistemos naudotojams pateikiama negalutinė sistemos versija, siekiant leisti naudotojams susipažinti su kuriama sistema ir iš anksto aptikti problemas, kylančias iš naudotojo perspektyvos (Sommerville, 2011).

Apžvelgus testavimo lygius ir tipus, galima teigti, kad šie elementai yra reikšmingi kuriant testavimo strategiją, kadangi testavimo lygių ir tipų pasirinkimas lemia, kokie sistemos funkciniai ir nefunkciniai reikalavimai bus testuojami bei kokių lygių testavimas taikomas sistemai.

## 2.3. Testavimo būdai

Priklausomai nuo testų vykdymo būdo, testavimo procesas gali būti klasifikuojamas į dvi kategorijas: rankinis ir automatinis testavimas (Sharma, 2014). Rankiniame testavime, testuotojas vykdo programą su tam tikrais testavimo duomenimis ir palygina gautus rezultatus su lūkesčiais, o automatinio testavimo metu šios veiklos yra automatizuotos (Sommerville, 2011). Abu testavimo būdai turi savų teigiamų ir neigiamų aspektų.

### 2.3.1. Rankinis testavimas

Rankinis testavimas yra kruopščiausias ir seniausias testavimo būdas (Sharma, 2014). Tai reiškia, kad visas testavimo proceso darbas, nuo reikalavimų analizės ir testavimo duomenų ruošimo iki rezultatų analizės, yra atliekamas žmogaus rankomis. Rankinis testavimas yra sunki veikla, reikalaujanti testuotoją turėti tam tikrą rinkinį savybių: būti kantriu, pastabiu, įtariu, kūrybingu, inovatyviu ir plačių pažiūrų (Sharma, 2014). Rankiniai testai yra tinkami giliai užslėptų ar specifinių atvejų suradimui, kurių automatiniai testai negalėtų atspėti (Leitner *et al.*, 2007). Tačiau, pasikartojantis rankinis testavimas yra neefektyvus didelėms sistemoms ar programoms, turinčioms labai didelę duomenų rinkinio aprėptį (Sharma, 2014). Todėl didelės apimties PĮ testuoti vien tik rankiniu būdu yra sudėtinga ir tai reikalauja daug išteklių. R. M. Sharma išskiria keturis rankinio testavimo neigiamus aspektus (Sharma, 2014):

- Reikalaujantis daug laiko – kadangi žmogiškieji resursai vykdo testavimo atvejus, rankinis testavimas yra labai lėtas ir varginantis.
- Reikalingas didelis investavimas į žmogiškuosius išteklius – rankiniam testavimui reikia daugiau testuotojų, todėl, kad testavimo atvejai vykdomi rankiniu būdu.
- Mažiau patikimas – rankinis testavimas yra mažiau patikimas, todėl, kad testai ne kiekvieną kartą gali būti atliekami preciziškai dėl žmogiškųjų klaidų.
- Neprogramuojamas – nėra galimybės pasitelkti programavimą sudėtingų testų rašymui, kurie gali atskleisti paslėptą informaciją.

Taigi, rankinis testavimas turi aiškių trūkumų, tačiau ginčijamasi ir dėl jo teigiamų savybių. V. Kumar išskiria net 9 rankinio testavimo pranašumus (Kumar, 2012):

- Taupumas. Rankinis testavimas yra taupesnis nei automatinis, nes nereikalauja sudėtingo testavimo atvejų programavimo.
- *Ad-hoc* testavimas. Testuotojas gali atlikti daugiau *Ad-hoc* (atsitiktinio) testavimo. *Ad-hoc* testavimo metu randama daugiau defektų nei automatiniame testavime. Kuo daugiau laiko testuotojas praleidžia eksperimentuodamas, tuo daugiau defektų gali būti rasta.
- Žvalgomasis testavimas. Dėl žmogaus galimybės mąstyti, testuotojas ras būdų ir priemonių kaip geriausiai tyrinėti produktą be jam iš anksto pateiktų būdų. Trumpai tariant, asmuo gali greitai atlikti žvalgomąjį testavimą.
- Rankinis testavimas gali būti naudojamas tiek mažuose, tiek dideliuose projektuose.
- Testavimo atvejai gali būti lengvai pridedami ir ištrinami pagal projekto pokyčius.
- Rankinio testavimo procesas yra lengvai suprantamas naujam testuotojui.

- Rankinis testavimas yra patikimesnis nei automatinis testavimas (daugeliu atvejų automatinis testavimas nepadengia visų galimų atvejų).
- Rankinis testavimas nėra susietas su jokia programavimo kalba.
- Rankiniam testavimui padengti užtenka ribotų išlaidų.

T. Wyher straipsnyje „5 priežastys, kodėl rankinis testavimas vis dar labai svarbus“ pabrėžia ir dar keletą rankinio testavimo pranašumų (Wyher, 2016):

- Rankinis testavimas vykdomas iš žmogaus perspektyvos, testuotojai gali greitai pastebėti, kai kažkas atrodo ne taip. Automatiniai testavimo atvejai neaptinka vizualinių problemų.
- Kaip ir bet koks programinis kodas, automatiniai testai gali turėti klaidų. Tai reiškia, kad automatinis testavimas gali raportuoti klaidingai teigiamus (angl. *false-positives*) arba klaidingai neigiamus (angl. *false-negatives*) rezultatus.
- Kai kurių scenarijų automatizavimas nėra įmanomas techniškai arba per daug kainuoja.
- Rankinis testavimas padeda suprasti problemą konceptualiame ir emociniame lygmenyje. Jis susieja testuotoją su galutiniu naudotoju ir leidžia jam įsijausti į jo rolę.

Apibendrinant aptartus rankinio testavimo pranašumus ir trūkumus, galima teigti, kad rankinis testavimas yra gana sudėtinga, laikui imli veikla, reikalaujanti ne tik žmogiškųjų pastangų, bet ir tinkamų testuotojo savybių bei pasiruošimo. Nors rankinis testavimas ir nėra efektyvus būdas laiko atžvilgiu, vis dar negalima nuginčyti ir jo teikiamų pranašumų – šis testavimo būdas gerai tinka didelėms sistemoms testuoti bei užslėptiems defektams aptikti, taip pat naujo funkcionalumo testavimui bei tam tikriems testavimo tipams. Specializuotos DVS atveju, dėl sistemos specifikos, aptartos tolesniuose skyriuose, rankinio testavimo pranašumai gali būti išnaudoti naujo funkcionalumo testavimui, specifinėms sistemos sritims, žvalgomajam, panaudojamumo ar dokumentacijos testavimui.

### 2.3.2. Automatinis testavimas

Paprastai automatinis testavimas yra tuo metu naudojamo rankinio testavimo proceso automatizavimas (Maurya, 2012). Automatinis testavimas automatizuoja ne tik testavimo atvejų vykdymą, bet ir testavimo atvejų generavimą ir rezultatų tikrinimą (Leitner *et al.*, 2007). Taigi, automatinis testavimas gali apimti plačią testavimo vykdymo proceso dalį. Geriau suprasti automatizavimo vaidmenį testavimo procese padeda V. Garousi ir F. Elberzhager paaiškinimas. Sutelkdami dėmesį į testų specifikaciją (projektavimas ir scenarijų rašymas),

vykdymą ir analizę (įvertinimas ir ataskaitų teikimas), jie išskyrė šešias testavimo proceso veiklas, kur automatizavimas turi didelį potencialą. Jos pateikiamos žemiau. (Garousi, Elberzhager, 2017).

- Testavimo atvejų projektavimas – testavimo atvejų ar testavimo reikalavimų, tenkinančių aprėpties kriterijus (angl. *coverage criteria*), kitus inžinerinius tikslus ar paremtų žmogiškąja patirtimi (pvz., žvalgomasis testavimas) sąrašo išskyrimas.
- Testavimo scenarijų rašymas – testavimo atvejų rankinio testavimo planuose ar automatizuoto testavimo kode dokumentavimas.
- Testų vykdymas – testuojamos PĮ testavimo atvejų vykdymas ir rezultatų saugojimas.
- Testų įvertinimas – testavimo rezultatų (sėkmingas arba ne) įvertinimas.
- Testavimo rezultatų ataskaitų teikimas – įvertintų testavimo rezultatų ataskaitų teikimas programuotojams, pvz. naudojantis defektų sekimo sistemomis.
- Testų valdymas ir kitos testavimo inžinerinės veiklos. Testų valdymas susideda iš tokių veiklų, kaip planavimas, kontrolė, stebėjimas ir pastangų įvertinimas. Kitos testavimo veiklos apima testavimo planų minimizavimą ir regresinių testų parinkimą.

Automatizavimas leidžia organizacijai sukurti turtingą daugkartinio naudojimo testavimo atvejų biblioteką ir palengvina nuoseklių testavimo atvejų rinkinio vykdymą (Naik, Tripathy, 2008). Visiškai automatizuota testavimo sistema geba testuoti programinę įrangą tokią, kokia ji yra, be naudotojo įsikišimo (Leitner *et al.*, 2007). Tai reiškia, kad gerai automatizuotai sistemai reikia minimalių žmogaus darbo išteklių – tik rezultatų peržiūrai, testų priežiūrai ir pan., kadangi testavimo procesą sistema gali atlikti be naudotojo įsikišimo.

Kaip ir rankinis testavimas, automatinis testavimas turi tiek privalumų, tiek trūkumų. Automatinis testavimas yra geras „į plotį“ žymiai geriau nei „į gylį“ (Leitner *et al.*, 2007). Tai reiškia, kad automatinis testavimas naudingas padengiant didelius kiekius testavimo atvejų, kurie yra gana paprasti ar net paviršutiniški, taip sutaupant daug laiko vykdant rankinį testavimą, tačiau nėra toks naudingas, kai PĮ testuojama „į gylį“: ieškoma specifinių atvejų, tikrinamos neįprastos situacijos, netipiniai duomenys ir panašiai.

R. M. Sharma teigia, jog automatinis testavimas yra greitesnis už rankinį testavimą ir be šios teigiamos savybės išskiria dar 7 automatinio testavimo pranašumus, kurie pateikiami žemiau (Sharma, 2014).

- Taupumas – testavimo atvejai vykdomi automatizavimo įrankio, taigi reikalauja mažiau testuotojo darbo.
- Pakartojamumas – tas pats testavimo atvejis (įrašymo ir pakartojimo būdu) gali būti pakartotas naudojant testavimo įrankius.

- Pakartotinis panaudojimas – testavimo planai gali būti panaudoti pakartotinai skirtingoms programinės įrangos versijoms.
- Programavimas – testuotojai gali programuoti sudėtingus testus, kurie gali atskleisti paslėptą informaciją.
- Visapusiškumas – testuotojai gali sukurti testavimo planus, kurie padengia kiekvieną programinės įrangos funkciją.
- Patikimumas – automatiniai testai kiekvieną kartą atlieka tiksliai tą pačią operaciją.
- Testavimo aprėptis – platesnė PĮ funkcijų testavimo aprėptis.

K. Naik ir P. Tripathy taip pat atskleidžia gana panašius automatinio testavimo pranašumus pateikiamus toliau (Naik, Tripathy, 2008).

- Testuotojo produktyvumas.
- Regresinio testavimo (angl. *regression testing*) padengimas.
- Testavimo atvejų pakartotinis panaudojimas.
- Testavimo atvejų nuoseklumas (pastovumas).
- Testavimo intervalų sumažinimas.
- Sumažėję PĮ palaikymo kaštai.
- Didesnis testų efektyvumas.

Kita vertus, esama ir automatinio testavimo trūkumų, kuriuos atskleidžia K. Naik ir P. Tripathy (Naik, Tripathy, 2008) bei K. Karhu ir kiti (Karhu *et al.*, 2009):

- Laikas – automatinis testavimas reikalauja nemažai laiko, ypač pradžioje, kol automatizuojamas pagrindinis programinės įrangos funkcionalumas.
- Lėšos ir resursai – reikalingos papildomos lėšos ir žmogiškieji ištekliai, ypač automatizavimo pradžioje, kol testavimas vis dar vykdomas rankiniu būdu, tačiau papildomai išleidžiama automatizavimui.
- Testuojamos sistemos stabilumas – automatiniai testai tinkami tik stabiliai sistemai, kuri negali nuolat keistis ar strigti.
- Testuotojų kvalifikacijos kėlimas – automatinius testus ruošiantys testuotojai privalo turėti programavimo patirties, nes testų automatizavimui naudojamos programavimo kalbos.
- Nuolatinė automatizuotų testų sistemos priežiūra – sistema turi būti nuolat prižiūrima ir adaptuojama pagal naujausius PĮ pokyčius, funkcionalumą.

Apžvelgiant automatinio testavimo pranašumus ir trūkumus, pastebėta, kad automatinis testavimas efektyviai panaudojamas pakartotiniams testavimams bei padengiant sistemos testavimą „į plotį“. Nepaisant aiškių automatinio testavimo pranašumų, pastebėtina, kad šis

būdas reikalauja didesnių investicijų testavimo automatizavimo pradžioje bei programavimo žinių. Specializuotos DVS testavime automatinis testavimo būdas gali būti naudojamas naujų versijų dūmų testavimui, sistemos funkcionalumo regresiniam testavimui bei tokiems testavimo tipams, kaip suderinamumo, konfigūracijos, saugumo, efektyvumo testavimas ir kt.

### 2.3.3. Testavimo būdų apibendrinimas

Tiek rankinis, tiek automatinis testavimas turi tam tikrų privalumų ir trūkumų. Rankinis testavimas yra nuobodus ir reikalauja didelių laiko sąnaudų ir žmogiškųjų išteklių investicijų (Sharma, 2014). Šias pagrindines rankinio testavimo problemas spręsti padeda automatizavimas. Tačiau, automatinio testavimo įdiegimas taip pat reikalauja didelių laiko, lėšų, žmogiškųjų išteklių investicijų, be to negali būti pritaikytas visiems testavimo tipams. Tam tikri testavimo tipai, tokie kaip panaudojamumo, interaktyvumo, gyvybingumo ar suderinamumo testavimas, dažnai netinkami automatizavimui (Naik, Tripathy, 2008). Taigi, automatizavimas nėra nemokamas ir norint užtikrinti sėkmingą rezultatą turi būti įgyvendinamas atsakingai (Garousi, Elberzhager, 2017).

Rankinis testavimas pasižymi savo įvairiapusiškumu. 2012 m. Švedijoje atliktas tyrimas parodė, kad iš 115 apklausoje dalyvavusių specialistų 80 % nesutiko, jog automatinis testavimas pakeis rankinį testavimą (Rafi *et al.*, 2012). Tam pritaria ir K. Naik su Tripathy, teigdami, jog automatizavimas negali imituoti žmogaus kūrybiškumo, nepastovumo ir stebėjimo galimybių (Naik, Tripathy, 2008). Taigi, dažnai geriausi rezultatai pasiekiami kombinuojant šiuos du testavimo būdus ir pasikartojantį ar varginantį darbą automatizuojant, o naujus, specifinius ar itin sudėtingus atvejus bei tam tikrus testavimo tipus vykdant rankiniu būdu.

## 2.4. Testavimo įrankiai

Programinės įrangos testavimo procese naudojama daug įvairių įrankių, kurie padeda kurti testavimo planus, sekti defektus, atlikti testavimą, jį automatizuoti ir t. t. Pavyzdžiui, automatinių testų inžinierius gali naudoti daugybę testavimo įrankių, tokių kaip statinio kodo analizatorius, testavimo duomenų generatorius, tinklo analizatorius (Naik, Tripathy, 2008, p. 24). Programinės įrangos testavimo įrankiai gali padėti sumažinti laiko, reikalingo testavimui, sąnaudas ir padaryti testavimą lengvu ir smagiu (Singh, 2012).

Remiantis *ISTQB*, testavimo įrankiai apima (*ISTQB*, 2018):

- įrankius, kurie tiesiogiai naudojami testavimui, tokie kaip testų vykdymo ir testavimo duomenų paruošimo įrankiai;

- įrankius, kurie padeda valdyti reikalavimus, testavimo atvejus, testavimo procedūras, automatizuoto testavimo scenarijus, testų rezultatus, testavimo duomenis, defektus; taip pat įrankius, skirtus testų vykdymo ataskaitų rengimui ir testų vykdymo stebėjimui;
- įrankius, kurie naudojami įžvalgoms ir vertinimui;
- bet kokius kitus įrankius, kurie padeda testavimui.

Taigi, testavimo įrankių samprata yra gana plati, kadangi bet koks įrankis, atnešantis naudą programinės įrangos testavimo procese, gali būti laikomas testavimo įrankiu. Testavimo įrankiai dažnai klasifikuojami pagal skirtingus kriterijus, tokius kaip tikslas, kaina, licencijavimo tipas ar naudojama technologija (*ISTQB,* 2018). K. Sneha ir G. M. Malle teigia, kad pagrindiniai programinės įrangos testavimo įrankiai gali būti skirstomi į tris kategorijas: testų valdymo įrankiai, funkcinio testavimo įrankiai ir apkrovos testavimo įrankiai (Sneha, Malle, 2017). *ISTQB* testavimo įrankius klasifikuoja į 6 tipus pagal testavimo veiklų palaikymą. Šie tipai pateikiami 3-ioje lentelėje.

3 lentelė. **Testavimo įrankių tipai**

(sudaryta autoriaus, remiantis *ISTQB,* 2018; Singh, 2012)

| **Testavimo įrankio tipas** | **Aprašymas** |
|---|---|
| **Testavimo valdymo įrankiai** | Valdymo įrankiai gali būti naudojami bet kokioms testavimo veikloms viso PĮ kūrimo metu. Tokių įrankių pavyzdžiai gali būti:<br>• testų valdymo įrankiai ir PĮ gyvenimo ciklo valdymo įrankiai;<br>• reikalavimų valdymo įrankiai (pvz.: testavimo objektų atsekamumas);<br>• defektų valdymo įrankiai (pvz.: defektų sekimo įrankiai, skirti kurti defektams ir sekti jų statusą bei istoriją);<br>• konfigūracijos valdymo įrankiai (testavimo aplinkų, programinės įrangos konfigūravimui) bei nuolatinės integracijos įrankiai. |
| **Statinio testavimo įrankiai** | Šie įrankiai naudojami statinio testavimo veikloms atlikti, t. y. tokiam testavimui, kuris nereikalauja programinės įrangos vykdymo, o remiasi rankiniu būdu vykdoma peržiūra arba įrankiais paremtu programinio kodo ar kitų darbo produktų vertinimu. Pavyzdžiui, tai gali būti statinės analizės įrankiai arba įrankiai, palaikantys peržiūras. |

3 lentelės tęsinys. **Testavimo įrankių tipai**

(sudaryta autoriaus, remiantis *ISTQB,* 2018; Singh, 2012)

| **Testavimo įrankio tipas** | **Aprašymas** |
|---|---|
| **Testų kūrimo ir įgyvendinimo įrankiai** | Testų kūrimo įrankiai padeda atlikti testavimo produktų kūrimo darbą, pvz.: testavimo atvejų, testavimo procedūrų ir testavimo duomenų kūrimą. Tokių įrankių pavyzdžiai gali būti:<br>• testų kūrimo įrankiai;<br>• modeliais paremti testavimo įrankiai;<br>• testavimo duomenų paruošimo įrankiai;<br>• priėmimo testavimo būdu paremto kūrimo (angl. *acceptance test driven development, ATDD*) ir elgsena paremto kūrimo (*angl. behavior driven development, BDD*) įrankiai;<br>• testavimu paremto kūrimo (angl. *test driven development, TDD*) įrankiai. |
| **Testų vykdymo ir registravimo įrankiai** | Tai įrankiai palaikantys ir pagerinantys testų vykdymo ir registravimo veiklas. Įrankių pavyzdžiai:<br>• testų vykdymo įrankiai (pvz.: skirti vykdyti regresijos testavimą);<br>• aprėpties (angl. *coverage*) įrankiai (pvz.: reikalavimų aprėptis, programinio kodo aprėptis);<br>• testų įrenginiai (angl. *test harnesses*);<br>• vieneto testų karkasiniai įrankiai. |
| **Našumo matavimo ir dinaminės analizės įrankiai** | Našumo matavimo ir dinaminės analizės įrankiai yra būtini efektyvumo ir apkrovos testavimo lygių veikloms, kadangi šios veiklos negali būti efektyviai atliekamos rankiniu būdu. Tokie įrankiai gali būti:<br>• efektyvumo testavimo įrankiai;<br>• stebėjimo įrankiai;<br>• dinaminės analizės įrankiai. |

3 lentelės tęsinys. **Testavimo įrankių tipai**

(sudaryta autoriaus, remiantis *ISTQB,* 2018; Singh, 2012)

| **Testavimo įrankio tipas** | **Aprašymas** |
|---|---|
| **Specializuotų testavimo poreikių įrankiai** | Specifiniams ar specializuotiems testavimo poreikiams naudojama ir daugiau testavimo įrankių, kurie gali padėti testuoti tokias sritis:<br>• duomenų kokybės vertinimas;<br>• duomenų konversija ir migravimas;<br>• tinkamumo testavimas;<br>• prieinamumo testavimas;<br>• lokalizavimo testavimas;<br>• saugumo testavimas;<br>• perkeliamumo (angl. *portability*) testavimas. |

Taigi, remiantis 3-iąja lentele, matoma, kad testavimo įrankiai gali apimti plačią testavimo proceso dalį. Pagrindiniai testavimo įrankių uždaviniai yra (*ISTQB,* 2018; Singh, 2012):

- pagerinti testavimo veiklų efektyvumą automatizuojant pasikartojančias užduotis arba užduotis, kurias atliekant rankiniu būdu reikia ženklių resursų (pvz.: testų vykdymas, regresijos testavimas);
- pagerinti testavimo veiklų efektyvumą paremiant rankines testavimo proceso veiklas;
- pagerinti testavimo veiklų kokybę sudarant sąlygas nuoseklesniam testavimui ir aukštesnio lygio defektų atkuriamumui (angl. *reproducibility*);
- automatizuoti veiklas, kurios negali būti atliekamos rankiniu būdu (pvz.: didelio masto efektyvumo testavimas);
- padidinti testavimo patikimumą (pvz.: automatizuojant didelių duomenų palyginimus ar simuliuojant elgseną);
- pagreitinti testavimą (pvz.: atlikti automatinius testus nakties ar savaitgalio metu, kai testuotojai nedirba).

Šie uždaviniai parodo, kad testavimo procese naudojami įrankiai padeda pagerinti, pagreitinti ir tinkamai valdyti testavimo proceso veiklas. Testavimo strategijos kūrimo metu, nurodomi naudotinų įrankių tipai. Tai vėliau padeda efektyviai vykdyti testavimo valdymo procesus (planavimą, kūrimą ir priežiūrą).

# 3. SPECIALIZUOTA DOKUMENTŲ VALDYMO SISTEMA

Remiantis sudarytu testavimo strategijos kūrimo modeliu, reikalinga atlikti testuojamo produkto, t. y. specializuotos dokumentų valdymo sistemos, tyrimą. Pirmiausia, apžvelgiamos įprastos dokumentų valdymo sistemos ir jų funkcionalumas bei teikiami pranašumai. Vėliau apibrėžiamos specializuotos DVS, taip pat AEC industrijai pritaikytos DVS ir jų svarba organizacijoms. Tada atliekama tokios DVS funkcinių reikalavimų analizė, o galiausiai ir detalesnė tokios sistemos ypatumų analizė.

## 3.1. Dokumentų valdymo sistemos, jų funkcijos ir pranašumai

Sąvoka „dokumentas“ gali būti vartojama bet kokiam raštui, kuris turi informacijos. Dokumentas gali būti tekstinis, skaičiuoklės, prezentacijos failas ar CAD brėžinys. (Austerberry, 2006). BS 1192 standartas teigia, kad dokumentas yra informacijos, kurią galima valdyti ir dalintis kaip vienetą, talpykla (BS 1192:2007). Informacijos valdymas yra būtinas bet kokio projekto sėkmingam užbaigimui. Efektyviai valdyti informaciją ir ja dalintis padeda elektroninės dokumentų valdymo sistemos. (Sommerville, Craig, 2006).

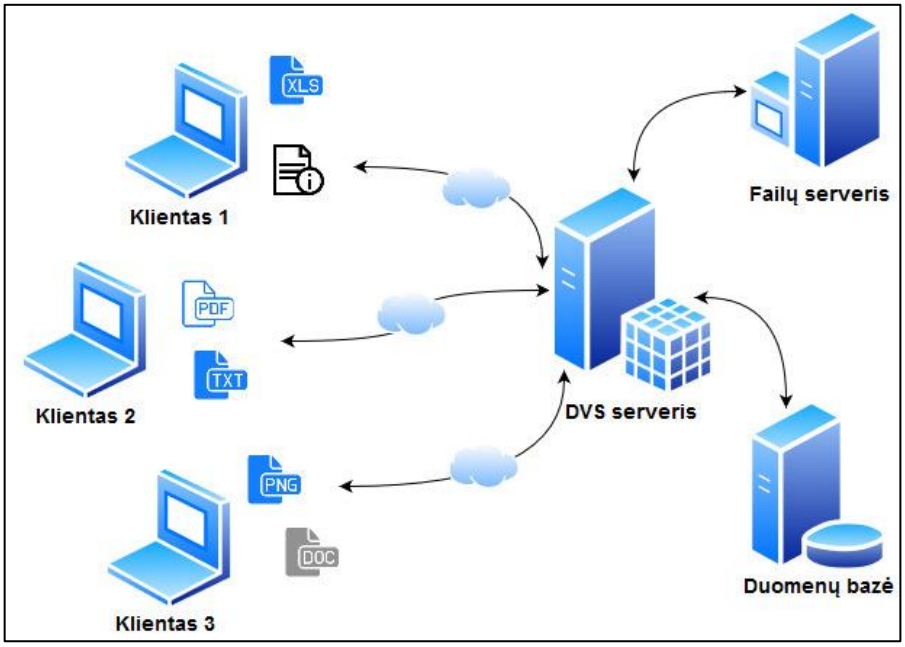
Žemiau pateikiama keletas DVS apibrėžimų, rastų literatūroje.

- Dokumentų valdymo sistema gali būti apibūdinama kaip rankinė ar elektroninė sistema, naudojama dokumentų generavimui, sekimui, valdymui ir saugojimui, bei turinti įvairių dokumentų versijų, sukurtų ir redaguotų skirtingų naudotojų, sekimą. (Koekemoer, Von Solms, 2018).
- Dokumentų valdymo sistema yra programinės įrangos sistema (arba programų rinkinys), naudojama saugoti, ieškoti ir gauti elektroninius dokumentus ir (arba) skenuotus paveikslėlius ar popierinius dokumentus. Sistema taip pat seka skirtingas dokumentų versijas, sukurtas skirtingų naudotojų. (Khan *et al.*, 2015).
- Dokumentų valdymo sistema yra programinės įrangos sprendimas, skirtas elektroninių resursų (jų pirminiu, originaliu formatu) ir jų metaduomenų saugojimui, gavimui ir darbo eigų vykdymui naudojant centrinę saugyklą. Darbo eiga paprastai apima verslo taisykles, padengiančias teisių, dokumentų įsiregistravimo ir išsiregistravimo bei patvirtinimo procesus. (BIMdictionary.com).

Taigi, remiantis dokumentų valdymo sistemos apibrėžimais, galima apibendrinti, kad DVS yra programinė įranga, skirta įvairių dokumentų (failų) saugojimui centrinėje saugykloje ir užtikrinanti tokias funkcijas, kaip dokumentų versijų sekimas, patvirtinimas, dokumentų darbo eigos ir kt.

Pirmosios dokumentų valdymo sistemos atsirado dar praėjusio amžiaus aštuntajame dešimtmetyje, kuomet organizacijos pradėjo kurti DVS, skirtas fizinių, popierinių dokumentų valdymui bei skaitmenizavimui. Vėliau, devintajame dešimtmetyje, DVS toliau tobulėjo ir buvo įgyvendintas ir elektroninių dokumentų valdymas.

Dokumentų valdymo sistemos dažnai įgyvendinamos naudojant kliento – serverio architektūrą. Tokia specializuotos DVS architektūra bus analizuojama ir šio darbo kontekste. Bazinis DVS modelis pateikiamas 5-ame pav.




5 pav. **Dokumentų valdymo sistemos modelis**

(sudaryta autoriaus)

Dokumentų valdymo sistemos aplikacijos serveris yra sujungiamas su duomenų bazės serveriu bei failų serveriu, kur saugomi dokumentai. Naudodami DVS kliento programą interneto ar intraneto pagalba į DVS serverį jungiasi DVS naudotojai, kurie dirba su dokumentais. DVS serveriui konfigūruoti bei administruoti gali būti naudojamas papildomas specialus įrankis, su kuriuo dirba sistemos administratorius.

Programinė įranga, skirta dokumentų valdymui, turi turėti tam tikras funkcijas, kad galėtų būti laikoma dokumentų valdymo sistema. Pagrindiniai elektroninės DVS funkciniai komponentai pateikiami 4-oje lentelėje.

4 lentelė. **Dokumentų valdymo sistemos funkcijos**

(sudaryta autoriaus, remiantis *Electronic...*, 2012*;* Mateika, 2008)

| Funkcija | Aprašymas |
|---|---|
| **Saugumo kontrolė** | Užtikrinamas reikalaujamas dokumentų saugumas, taip pat naudotojų prieigos kontrolė (angl. *access control*). |
| **Dokumentų operacijos** | Palaikomas dokumentų importavimas ir eksportavimas bei standartinės dokumentų operacijos, tokios kaip *Atidaryti, Redaguoti, Ištrinti* ir pan. Taip pat DVS gali leisti naudotojui įsiregistruoti dokumentą (užrakinti, kad jis nebūtų keičiamas kito naudotojo) bei vėliau jį išregistruoti. |
| **Versijos** | DVS užtikrina dokumento versijų kūrimą bei valdymą. |
| **Saugykla** | DVS užtikrina dokumentų saugojimo galimybę. Dokumentų saugojimas dažnai būna centralizuotas naudojant vieną pagrindinę saugyklą. |
| **Metaduomenų saugojimas ir naudojimas** | DVS palaiko dokumento metaduomenų atpažinimą, saugojimą ir naudojimą. |
| **Paieška** | DVS turi dokumentų, aplankų, atributų paiešką. Taip pat gali turėti ir pilno teksto paiešką, kai paieška atliekama dokumentų viduje. |
| **Dokumento darbo eiga (angl. *workflow*)** | DVS leidžia apibrėžti skirtingas dokumentų darbo eigas, kurios apima automatinį dokumento priskyrimą kitam naudotojui, dokumento prieigos kontrolės keitimą, pasirašymą ir t. t. |
| **Dokumentų įkėlimas ir konvertavimas** | DVS turi galimybę įkelti fizinius dokumentus juos konvertuojant į elektroninius. |
| **Integracija su kitomis sistemomis** | DVS gali būti integruota su kitomis sistemomis, pvz. el. pašto klientu, teksto redaktoriumi ir kitomis taikomosiomis programomis. |

Dokumentų valdymo sistema be abejonės suteikia organizacijai pranašumų. J. Sommerville ir N. Craig išskiria toliau pateiktus DVS pranašumus (Sommerville, Craig, 2006, p. 130).

- Projekto informacijos centrinė saugykla.
- Greitesnis projekto informacijos dokumentų apdorojimas ir gavimas.
- Informacijos nuoseklumas tarp sistemos naudotojų.
- Reikalaujama mažiau žmogaus dėmesio informacijos pateikimui ir platinimui.
- Sumažintos administravimo ir siuntimo išlaidos.
- Skirtingo lygmens prieiga prie kontroliuojamos informacijos viso projekto mastu.
- Padidintas problemų sprendimo ir sprendimų priėmimo produktyvumas.
- Nepertraukiama prieiga prie sistemos visiems projekto dalyviams.
- Struktūruotos dokumentų darbo eigos.
- Neribotas sistemos naudotojų skaičius.

Taigi, sistema, kurioje įgyvendintas DVS funkcionalumas, leidžia organizacijai ne tik saugoti ir lengvai valdyti reikalingus dokumentus, tačiau ir užtikrina atitinkamą saugumą ir prieigą bei padeda organizacijai struktūrizuoti veiklą ir procesus.

## 3.2. Specializuotos dokumentų valdymo sistemos

Standartinės universalios dokumentų valdymo sistemos tinka daugumai organizacijų ir leidžia valdyti įvairius jų dokumentus. Tačiau, universalios DVS nėra lanksčios, jos užtikrina tik pagrindines funkcijas, todėl yra tinkamos paprastam dokumentų saugojimui ir baziniam jų valdymui. Ypač dažnai toks dokumentų valdymas naudojamas palaikančiosioms, tiesioginės vertės nekuriančioms, organizacijų veikloms. Kai kurios organizacijos turi sudėtingus verslo procesus, kurie reikalauja atitinkamų dokumentų valdymo sistemos funkcijų – kompleksinių darbo eigos kelių, automatizuotų dokumentų valdymo procesų, plačių konfigūravimo galimybių ir pan. Be to, skirtingose industrijose veikiančių organizacijų procesai ir specifika nėra vienoda, taip pat skiriasi ir organizacijų kitų naudojamų sistemų profilis, su kuriomis DVS gali turėti integraciją. Taip kyla poreikis diegti ir naudoti specializuotas DVS, kurios dažnai naudojamos jau ne palaikančiosioms, o pagrindinėms organizacijos veikloms.

Specializuotos DVS gali būti pritaikomos įvairioms sritims, tačiau mokslinėje literatūroje dažniausiai kalbama apie medicinos, žmogiškųjų išteklių valdymo, AEC bei finansų srityse naudojamas specializuotas DVS. Kiekvienu atveju naudojamai DVS yra būdinga tam tikra specifika, nuo kurios priklauso sistemos funkcionalumas. Pavyzdžiui, medicinos srityje DVS gali būti naudojamos medicininiams įrašams saugoti, finansų – ataskaitoms, sąskaitoms ir kitiems finansiniams dokumentams. Pirmuoju atveju DVS saugoma elektroninė paciento kortelė, todėl kritiniais sistemos reikalavimais tampa integracija su kitomis specializuotomis sistemomis (medicinine PĮ), platus dokumentų tipų spektras (nuotraukos, vaizdo įrašai,

grafiniai vaizdai, garso įrašai ir t.t.) bei sistemos greitaveika ir gebėjimas apdoroti didelius kiekius bei daug vietos užimančius dokumentus (Mateika, 2008). Antruoju atveju, labai svarbus dokumento kelias ir prieigos kontrolė – finansiniai dokumentai turi būti patvirtinti, pasirašyti bei apmokėti, tas turi būti atliekama iki nustatyto termino, be to turi būti užtikrintas aukštas saugumas, taigi reikalinga galimybė konfigūruoti prieigos kontrolę prie tam tikrų dokumentų ar atitinkamų jų darbo eigos žingsnių.

Taigi, norint kuo geriau išnaudoti DVS privalumus, atsiranda poreikis naudoti specializuotą, tam tikrai industrijai, verslo specifikai ar netgi konkrečiai organizacijai pritaikytą dokumentų valdymo sistemą.

## 3.3. AEC industrijai pritaikytos dokumentų valdymo sistemos ir jų svarba

AEC industrijoje naudojamų dokumentų standartai yra nusistovėję ir daug nesikeitė dar nuo XX a. vidurio. Planai, brėžiniai, specifikacijos ir t.t. atrodo labai panašiai, kaip ir prieš dešimtmečius. Tačiau, tokių dokumentų kūrimo, valdymo, dubliavimo ir paskirstymo technologijos keitėsi iš esmės. (Bjork, 2001). 1980-ųjų pabaigoje ir 1990-ųjų pradžioje kompiuterių tinklai leido pradėti naudoti DVS projektų dokumentacijos saugojimui ir valdymui. Tai AEC industrijos dokumentų valdymą padarė žymiai paprastesnį. Dar 2001 m. B.-C. Bjork teigė, kad DVS yra svarbiausia statybų industrijos IT technologija (Bjork, 2001). Ir nors pagrindinės to meto specializuotų DVS specifinės funkcijos tebuvo trys – failų gavimo mechanizmas remiantis hierarchine aplankų struktūra arba metaduomenimis, peržiūrų ir pakeitimų valdymas bei CAD failų peržiūra su specialia programine įranga – AEC industrijai pritaikytos DVS sparčiai tobulėjo ir siūlė vis didesnį funkcionalumą bei vis labiau plito šioje industrijoje.

### 3.3.1. AEC projekto gyvavimo ciklas

AEC industrija turi savitus verslo procesus ir dokumentų valdymui kelia atitinkamus reikalavimus. Norint apibrėžti šiuos reikalavimus, svarbu suprasti AEC projekto gyvavimo ciklą, kurio metu ir generuojami dokumentai. Vienas iš plačiai naudojamų modelių AEC industrijoje yra RIBA (*Royal Institute of British Architects*) planas, kurio etapai aptariami 5-oje lentelėje.

5 lentelė. **RIBA plano etapai**

(sudaryta autoriaus, remiantis *RIBA,* 2013)

| Nr. | Etapo pavadinimas | Aprašymas |
|---|---|---|
| **0.** | **Strateginis apibrėžimas** | Identifikuojamas kliento verslo atvejis, strateginė santrauka bei kiti svarbiausi projekto reikalavimai. |
| **1.** | **Pasiruošimas ir paskyra** | Nustatomi projekto tikslai (įskaitant kokybės tikslus, projekto rezultatus, biudžetą) ir kiti parametrai bei apribojimai. Sudaroma pirminė projekto paskyra, atliekama galimybių analizė bei statybų aikštelės apžvalga. |
| **2.** | **Konceptualus projektavimas** | Paruošiamas konceptualus projektas, kartu su struktūrinio dizaino pasiūlymu bei kaštų informacija. Nustatomos projekto strategijos. Sutariami projekto paskyros pakeitimai bei sudaroma galutinė projekto paskyra. |
| **3.** | **Išvystytas projektavimas** | Paruošiamas išvystytas projektas, kartu su atnaujintais struktūrinio dizaino pasiūlymais, statinio aptarnavimo sistemomis, specifikacijomis, kaštų informacija bei projekto strategijomis. |
| **4.** | **Techninis projektavimas** | Paruošiamas techninis projektas, įskaitant visą architektūrinę, struktūrinę bei statinio aptarnavimo sistemų informaciją. Paruošiami projektai ir specifikacijos subrangovams. |
| **5.** | **Statyba** | Vykdoma nuotolinė gamyba, statybų aikštelėje vyksta statyba. |
| **6.** | **Perdavimas ir užbaigimas** | Statinio perdavimas ir projekto užbaigimas pasirašant statinio kontraktą. |
| **7.** | **Eksploatacija** | Pagal paslaugų tvarkaraščio įsipareigojimus perimama pastato eksploatacija. |

Akivaizdu, kad kiekvieno iš RIBA plano etapų metu yra kuriami ir naudojami dokumentai, kuriuos reikalinga saugoti bei valdyti. Taip pat galima teigti, jog viso projekto metu generuojama dokumentacija turi būti derinama su klientu. Tai reiškia, kad DVS privalo palaikyti dokumentų bendrinimą. Dokumentų tipai, kiekis ir dydžiai priklauso nuo plano etapų. Pavyzdžiui, 0-nio, 1-ojo bei 6-ojo etapų metu daugiausiai generuojami įprasti tekstiniai dokumentai – reikalavimai, sutartys, planai, sąskaitos, ataskaitos ir t. t. 5-ojo bei 7-ojo etapų metu atsiranda poreikis realiu laiku keisti jau sukurtus dokumentus – pasižymėti komentarus,

pastabas. Tačiau, techninė bei teisinė projekto dokumentacija nėra pagrindinis specializuotos DVS iššūkis. Specialieji poreikiai dokumentų valdymo sistemai geriausiai atsispindi 2-ojo – 4-ojo etapų metu, kuomet vyksta projektavimo darbai ir DVS naudojama brėžinių ir kitų projektavimo failų valdymui.

### 3.3.2. Specializuotos dokumentų valdymo sistemos nauda AEC organizacijai

Specializuotos DVS įdiegimas AEC organizacijai gali atnešti nemažai naudos – standartizuoti dokumentų valdymą, įgalinti bendradarbiavimą dirbant su dokumentais bei sutaupyti kaštų ir laiko. 6-oje lentelėje pateikiamos pagrindinės problemos, su kuriomis organizacija susiduria nenaudodama DVS, ir jų poveikis projektams.

6 lentelė. **Neigiama įtaka AEC projektui nenaudojant DVS**
(šaltinis: Devanand, 2015)

| Veiksnys | Rezultatas | Poveikis |
|---|---|---|
| Nepakankamas komunikacijos lygis tarp projekto komandos narių. | Delsimas | Laikas |
| Negalėjimas surasti brėžinių ar dokumentų, kai jie reikalingi. | Delsimas | Laikas |
| Svarbių dokumentų praradimas. | Delsimas | Laikas ir kaina |
| Darbai atliekami su pasenusia informacija. | Ginčai, delsimas, perdarymas | Laikas, kaina ir kokybė |
| Nepilnos specifikacijos ir brėžiniai. | Perdarymas | Laikas, kaina ir kokybė |
| Neatitikimas statybvietės ir darbų inspekcijos reikalavimams dėl nepilnos ar neteisingos dokumentacijos. | Delsimas | Laikas, kaina ir kokybė |

Remiantis 6-a lentele, galima teigti, kad specializuotos DVS nenaudojimas tiesiogiai daro įtaką trims iš keturių projektų valdymo trikampio apribojimų elementų – kokybei, kainai ir laikui. Tai reiškia, kad AEC organizacijai valdyti projektus nenaudojant DVS yra didelis iššūkis, kadangi tokios sistemos nebuvimas lemia neigiamą poveikį projektams bei kelia papildomų rizikų.

Specializuotų DVS diegimas AEC industrijos organizacijoms gali atnešti ir tiesioginės naudos. Štai JAV Konektikuto valstijos Transporto Departamento, kuris yra įgyvendinęs specializuotos *ProjectWise Design Integration* DVS pirmąją diegimo fazę, 2017 m. atliktas tyrimas atskleidė, kad sistemos įdiegimas departamentui per metus vien kontraktų planų, specifikacijų ir papildomų dokumentų valdymui leidžia sutaupyti 315 tūkst. JAV dolerių. Departamentas vien popieriaus mažinimui sutaupo po 770 tūkst. JAV dolerių per metus, o bendra per metus sutaupytų išlaidų suma siekia 1,2 mln. JAV dolerių. (Pratt, Connors, 2017). Atsižvelgiant į tai, kad įdiegta tik pirma sistemos fazė ir ji naudojama daugiausiai teisinei ir techninei dokumentacijai, galima daryti prielaidą, jog įdiegus sistemą ir kitiems departamento procesams palaikyti, sutaupyta suma būtų dar didesnė.

Remiantis šiais pavyzdžiais, galima teigti, jog specializuotos DVS padeda AEC organizacijoms efektyviai valdyti projektus, todėl jų nauda yra reikšminga.

## 3.4. AEC industrijai pritaikytos dokumentų valdymo sistemos reikalavimų analizė

Projektavimo etapuose generuojamų dokumentų valdymas yra specifinis, kadangi projektuojant statinius, naudojama kompleksiška specializuota CAD ir BIM programinė įranga, suteikianti galimybę atvaizduoti bei projektuoti statinių komponentus įvairiais būdais bei kurianti atitinkamų tipų failus – CAD brėžinius, vaizdus, 2D ir 3D modelius. Tai diktuoja poreikį integruoti dokumentų valdymo sistemą ne tik su standartine *Office* programine įranga, bet ir su CAD, BIM bei kitais AEC industrijoje naudojamais sprendimais. Projektavimo metu, statinių struktūriniai komponentai dažniausiai saugomi atskiruose failuose, o tas pats komponentas gali turėti daug jam priklausančių kitų dokumentų. Be to, dokumentai yra tarpusavyje susiję ryšiais, kadangi vienas didesnis komponentas susideda iš keleto mažesnių (pvz., sienoje suprojektuojamos angos (langai, durys), elektros ir kitų komunikacijų taškai, kurie gali būti kaip atskiri dokumentai). Siekiant suvaldyti dokumentus, toks ryšių medis turi aiškiai atsispindėti ir dokumentų valdymo sistemoje. Statinių projektavimo kompleksiškumas dokumentų valdymo sistemai kelia ir daugiau reikalavimų – galimybę su dokumentais dirbti keletui žmonių, specifines ir lengvai konfigūruojamas dokumento darbo eigas. Siekiant išsiaiškinti visus AEC industrijai pritaikytų DVS reikalavimus, atlikta tokių sistemų reikalavimų analizė.

### 3.4.1. Specializuotų DVS lyginamoji analizė

Siekiant apžvelgti rinkoje esančių AEC industrijai pritaikytų DVS sprendimų siūlomą funkcionalumą, buvo atlikta tokių sistemų lyginamoji funkcijų analizė. Pirmiausia, atlikta tokių DVS paieška. Nors iš pirmo žvilgsnio atrodo, kad AEC industrijai pritaikytų DVS pasiūla yra didelė, įsigilinus paaiškėjo, kad dauguma rastų sistemų nėra skirtos vien tik AEC industrijai, t. y. orientuojasi į universalų naudojimą ir kitose industrijose, arba jų funkcionalumas yra gana ribotas, dėl ko jas rekomenduojama naudoti nedidelėms įmonėms dirbant su nesudėtingais projektais.

Galiausiai, analizei atlikti buvo pasirinktos trys sistemos – *Autodesk* kuriama *BIM 360 Docs*, *Bentley Systems* kuriama *ProjectWise Design Integration* bei *Errevi System* kuriama *Engineering Document Management*. 7-oje lentelėje pateikiamas sistemų siūlomų funkcijų palyginimas.

7 lentelė. **DVS funkcijų palyginimas**

(sudaryta autoriaus)

| **Sistemos funkcija** | ***BIM 360 Docs*** | ***ProjectWise Design Integration*** | ***Engineering Document Management*** |
|---|---|---|---|
| Dokumentų peržiūra ir publikavimas | + | + | + |
| Dokumentų palyginimas | + | | |
| Integracija su AEC programine įranga | + | + | |
| Naudotojų valdymas | + | + | + |
| Dokumentų versijų valdymas | + | + | + |
| Projektų, kontraktų ir planų administravimas | + | + | + |
| Brėžinio pagrindinės įrašų lentelės palaikymas | + | + | |
| Optinis brėžinio pagrindinės įrašų lentelės atpažinimas | + | | |
| Automatinės brėžinių žymų nuorodos | | + | |
| Prieigos kontrolė | + | + | + |
| Dokumentų rinkiniai | + | + | |
| Metaduomenys ir failų atributai | + | + | + |

7 lentelės tęsinys. **DVS funkcijų palyginimas**

(sudaryta autoriaus)

| **Sistemos funkcija** | ***BIM 360 Docs*** | ***ProjectWise Design Integration*** | ***Engineering Document Management*** |
|---|---|---|---|
| Versijų kontrolė | + | + | + |
| Patvirtinimo darbo eigos | + | + | + |
| Konfigūruojamos darbo eigos | | + | |
| Ataskaitos ir skydeliai | + | | + |
| Industrijos standartų palaikymas | | + | |
| Pagrindiniai ir nuorodiniai failai | + | + | |
| Dokumentų audito įrašai (pakeitimų istorija) | + | + | + |
| Dokumentų sauga pagal darbo eigas | | + | |
| Dokumentų priskyrimas erdvinėms lokacijoms | | + | |
| Automatinis PDF dokumentų generavimas iš kitų formatų | | + | + |
| Dokumentų kodai | | + | |
| Lankstus dokumentų dalijimasis su susijusiomis šalimis (angl. *transmittal-submittal*) | | + | + |
| Automatiškai atnaujinamas naudotojo „Reikia padaryti“ sąrašas | | | + |
| Automatinis pranešimų siuntimas | | + | + |

Atlikus tokių sistemų viešai publikuojamų funkcijų palyginimo analizę, aiškiai matoma, kad AEC industrijai specializuotos dokumentų valdymo sistemos turi ne tik standartinį anksčiau jau aptartą DVS funkcionalumą, bet atlieka ir nemažai specifinių funkcijų, kurios nėra būdingos universalioms ar į kitas sritis orientuotoms DVS.

Be to, verta paminėti, kad tiek *BIM 360 Docs*, tiek *ProjectWise Design Integration* dokumentų valdymo sprendimai gali būti gerokai išplėsti. Pavyzdžiui, *Autodesk* be bazinės *BIM 360 Docs* versijos siūlo dar tris variantus – *BIM 360 Design*, *BIM 360 Coordinate* ir *BIM 360*

*Build*. Kiekvienas iš šių pasiūlymų padengia atitinkamų sričių organizacijos procesų valdymą ir siūlo tokį papildomą funkcionalumą, kaip projekto kontrolės priemonės, koordinavimas, kokybės ir saugumo valdymas. (*Autodesk,* 2019). *Bentley Systems* savo ruožtu taip pat siūlo platų *ProjectWise* grupės programinės įrangos ir paslaugų spektrą ir pateikia debesų kompiuterijos sprendimus. (*Bentley Systems*, 2019). Šie papildomai siūlomi programinės įrangos paketai ir paslaugos bazinę specializuotą DVS išplečia iki kompleksinės bendradarbiavimo ir darbo pasidalijimo turinio valdymo sistemos. Tokios plačios galimybės išplėsti specializuotas DVS iki sudėtingų ir daug funkcijų padengiančių informacinių sistemų tik patvirtina, kad DVS yra viena svarbiausių sistemų AEC industrijoje, kurią įdiegus vėliau galima įgyvendinti ir šių sistemų papildymą kitu funkcionalumu.

### 3.4.2. Specializuotos DVS funkcijų grupės

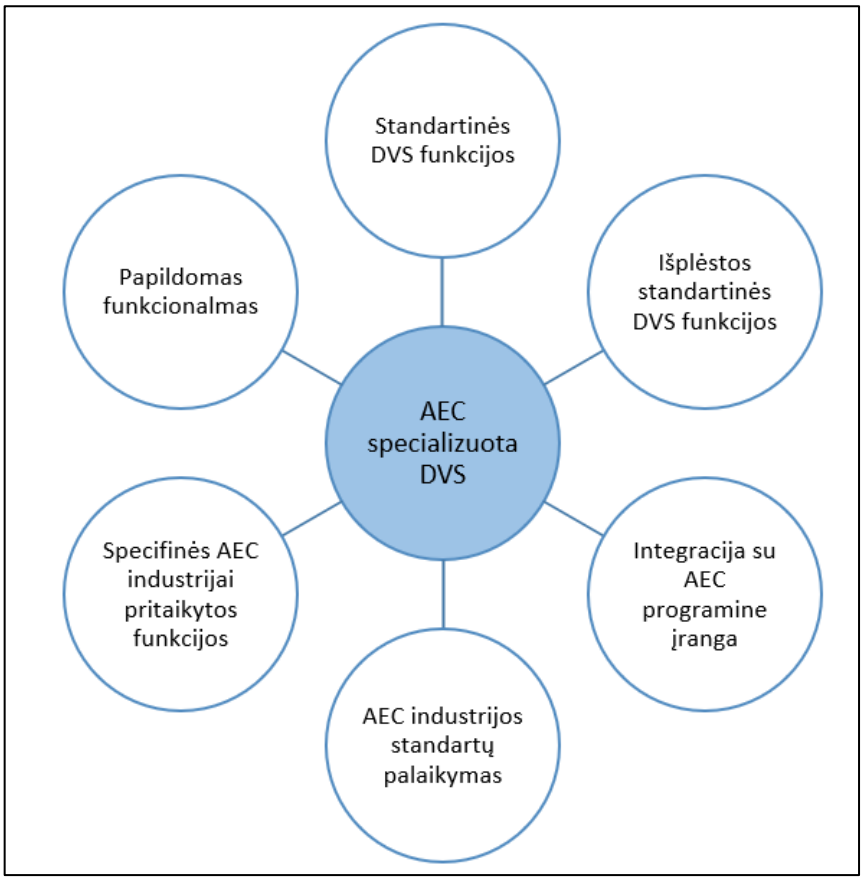
Tęsiant specializuotų DVS funkcionalumo tyrimą, patogu struktūrizuoti nustatytas funkcijas jas sugrupuojant. Specializuotų DVS smulkesnio funkcionalumo sugrupavimas į keletą grupių pateikiamas toliau.

- **Standartinės DVS funkcijos**
- **Išplėstos standartinės DVS funkcijos:**
  - AEC industrijai pritaikytos dokumentų būsenos ir darbo eigos, jų kūrimas bei valdymas;
  - lanksti prieigos kontrolė, suteikianti galimybę valdyti prieigą pagal įvairius kriterijus;
  - AEC industrijai pritaikytas dokumentų metaduomenų valdymas;
  - AEC industrijai pritaikyti dokumentų atributai ir jų sinchronizavimas;
  - išplėstinė paieška (pilno teksto, paieška atributuose, paieška pagal dokumentų ryšius).
- **Integracija su AEC programine įranga:**
  - BIM programine įranga;
  - CAD programine įranga;
  - realybės modeliavo programine įranga;
  - analizės sistemomis.
- **AEC industrijos standartų palaikymas:**
  - standartizuotas AEC projektų administravimas;
  - standartizuotas AEC kontraktų administravimas;
  - automatizuotas dokumentų kodo (failų pavadinimai, sugeneruojami pagal tam tikrus iš anksto apibrėžtus kriterijus) generavimas;

- gerųjų AEC praktikų standartų palaikymas (pvz.: BS 1192).

- **Specifinės AEC industrijai pritaikytos funkcijos:**
  - brėžinių ir modelių peržiūra bei publikavimas;
  - skirtingos naudotojo sąsajos aplinkos;
  - skirtingos aplinkos integruotai AEC programinei įrangai;
  - dokumentų rinkiniai;
  - dokumentų ryšiai, automatinis jų valdymas, ryšių atvaizdavimas;
  - dokumentų žymėjimas ir komentavimas (angl. *markup*);
  - brėžinio lentelių (angl. *title block*) palaikymas;
  - brėžinio žymų (angl. *callout*) nuorodos;
  - dokumentų priskyrimas erdvinėms lokacijoms;
  - dokumentų perdavimas susijusioms šalims ir gavimas.
- **Papildomos funkcijos**:
  - progreso sekimas;
  - ataskaitos ir skydeliai;
  - automatinis pranešimų arba el. laiškų siuntimas;
  - integracijos su kitomis sistemomis, pvz.: verslo valdymo ar klientų valdymo sistemomis.

Remiantis tokiu grupavimu, AEC industrijai pritaikytai DVS galima sudaryti sistemos funkcinių grupių modelį. Struktūrizuotas AEC industrijai skirtos DVS funkcionalumo modelis pateikiamas 6-ame pav.




6 pav. **AEC industrijai specializuotos DVS funkcijų grupės**

(sudaryta autoriaus)

6-ame paveiksle pavaizduotos DVS funkcijos suskirstytos į 6-ias grupes. Standartinės DVS funkcijos – tai visos anksčiau aptartos universalioms DVS būdingos funkcijos, kaip dokumentų saugykla, saugumo kontrolė ar dokumentų operacijos. Išplėstos standartinės DVS funkcijos – tai funkcijos, kurias turi ir įprastos DVS, tačiau šis funkcionalumas yra pakankamai išplėstas bei pritaikytas AEC industrijai. Integracija su AEC programine įranga – tai integracijos taškai su kitomis AEC naudojamomis sistemomis bei PĮ, kaip CAD ar BIM įrankiai. AEC industrijos standartų palaikymas – tai funkcionalumas, leidžiantis dokumentų valdymui pritaikyti tam tikrus AEC industrijoje naudojamus standartus ar gerąsias praktikas, pvz.: BS 1192 standartą. Specifinės AEC industrijai pritaikytos funkcijos – tai visas specializuotos DVS funkcionalumas, kuris būdingas tik AEC industrijai naudojamose DVS. Tai gali būti dokumentų priskyrimas pagal erdvinę lokaciją, nuorodos į CAD brėžinių žymas ir pan. Papildomas funkcionalumas – tai visos kitos papildomos funkcijos, kurios nėra privalomos DVS, tačiau suteikia šiai sistemai pridėtinės vertės. Dažnu atveju tokias specializuotas DVS galima integruoti su projektų valdymo sistemomis, verslo analitikos įrankiais ar kitomis sistemomis, praplečiančiomis DVS ar panaudojančiomis jų duomenis.

## 3.5. AEC industrijai pritaikytos dokumentų valdymo sistemos ypatumų analizė

AEC industrijai skirta specializuota DVS gerokai skiriasi nuo standartinės DVS ir turi nemažai išplėsto arba papildomo funkcionalumo. Norint sukurti tinkamą specializuotos DVS testavimo strategiją, reikalinga detaliau aptarti kai kurias specializuotos DVS funkcijas, kadangi nuo jų priklausys ir strategijos kūrimas.

**Integracija su AEC programine įranga**

Vienas pagrindinių specializuotos DVS reikalavimų – galimybė integruoti sistemą su specializuota AEC programine įranga – CAD, BIM, realybės modeliavimo įrankiais bei kita PĮ. Pirmiausia, turi būti palaikomos bazinės operacijos su failais, tai reiškia, kad AEC programinėje įrangoje naudojamos standartinės failo operacijos – *Atidaryti*, *Išsaugoti*, *Pridėti nuorodinį dokumentą* ir pan., automatiškai nukreiptų ne į kompiuterio failų sistemą, o į dokumentų valdymo sistemą. Tokiu būdu visi saugomi dokumentai yra saugomi centralizuotoje DVS saugykloje, o ne atskirų kompiuterių diskuose ar kitose laikmenose. DVS integracija taip pat gali suteikti galimybę siųsti DVS saugomus dokumentus tiesiai iš AEC programinės įrangos, o ne iš pačios DVS.

Dėl to, kad dokumentai AEC industrijoje privalo būti peržiūrėti, patvirtinti ir publikuoti, AEC programinėje įrangoje naudojamos žymėjimo, komentavimo ir pastabų pateikimo funkcijos. Kad šiuos papildomus duomenis būtų galima peržiūrėti bei surasti neatidarius dokumentu su specialia PĮ, jie sinchronizuojami kaip dokumento atributai ar papildoma informacija, prisegama prie dokumento. Tokį sinchronizavimą leidžia ne kas kitas, o integracija su AEC programine įranga.

Be to, DVS integracija palaiko duomenų apsikeitimą su AEC programine įranga. Duomenų apsikeitimo principas taikomas naudojant skirtingas darbines sritis (angl. *workspace*). Darbinė sritis – tai AEC programinės įrangos parametrų rinkinys, kuris nustato atitinkamus PĮ parametrus, atvaizdavimą ir kitus konfigūruojamus objektus, bei yra saugomas DVS. Pavyzdžiui, tas pats CAD įrankis gali būti naudojamas tiek struktūrinio inžinieriaus, tiek architekto, tačiau jų naudojama įrankio konfigūracija skiriasi dėl jų rolių. Tokiu atveju, programinės įrangos darbinė sritis, t. y. konfigūracija ir parametrai, yra skirtingi priklausomai nuo naudotojo rolės. Kad naudotojui kiekvieną kartą nereikėtų konfigūruoti integruotos AEC programinės įrangos pagal savo poreikius, šiuos parametrus galima saugoti kaip DVS projekto ar aplanko nustatymus, kuriuos paveldės visi objekto dokumentai. Įgyvendinus tokią integracijos konfigūraciją, iš DVS atidarant dokumentus su AEC programine įranga, DVS

iškart perduos parametrų rinkinį, nustatytą pagal naudotojo, kuris atidaro failą, rolę, dokumento būseną ar kitus kriterijus.

**Naudotojų rolės ir prieiga**

Kadangi AEC projektuose dalyvauja ir kartu dirba daugybė skirtingų specialistų, DVS turi palaikyti skirtingas naudotojų roles. Remiantis BS 1192 standartu, tipinės projekto dalyvių rolės yra:

- architektas;
- pastatų inspektorius;
- civilinis inžinierius;
- drenažo ir greitkelių inžinierius;
- elektros inžinierius;
- geografijos ir krašto inžinierius;
- šildymo ir ventiliavimo projektuotojas;
- visuomenės sveikatos inžinierius;
- medžiagų ir darbų kiekybės ekspertas;
- interjero dizaineris;
- klientas;
- kraštovaizdžio architektas;
- mechanikos inžinierius;
- struktūrinis inžinierius;
- miesto ir šalies planuotojas;
- rangovas;
- subrangovas;
- projektuotojas specialistas;
- bendras naudotojas (ne industrijos specialistas).

Tokia didelė naudotojų rolių įvairovė reikalauja lanksčių prieigos pagal roles konfigūravimo galimybių, t. y. turi būti užtikrinta, kad naudotojas turi prieigą tik prie jo rolei priskirtos informacijos bei su ja gali atlikti tik jam numatytus veiksmus (pvz.: tik peržiūrėti, atidaryti, redaguoti, patvirtinti, atmesti, kurti versijas ir t. t.). Be to, kadangi DVS naudojasi ir išoriniai naudotojai (pvz. subrangovai, klientai, laisvai samdomi specialistai), turi būti užtikrinamas organizacijos informacijos bei dokumentų saugumas. Tokiu atveju prieigos ir teisių kontrolė pagal naudotojų roles netinka, kadangi tos pačios rolės naudotojai gali būti tiek organizacijos vidiniai darbuotojai, tiek išoriniai specialistai, o jų prieigos prie informacijos lygis gali būti skirtingas. Tokiu atveju prieiga ir teisės priskiriamos konkretiems naudotojams arba sistema turi palaikyti prieigos kontrolę naudojant tinklo apribojimus – kad naudotojai, kurie jungiasi ne iš organizacijos vidinio tinklo, automatiškai gautų tik tam tikro lygio prieigos ir teisių rinkinį.

**Dokumentų atributai ir DVS aplinkos**

Didelis naudotojų rolių skaičius kelia ir kitų specifinių reikalavimų DVS sistemai. Skirtingų rolių naudotojai turi matyti skirtingą informaciją apie dokumentus. Pavyzdžiui, elektros inžinieriui aktualūs visiškai kiti dokumento atributai negu kraštovaizdžio architektui.

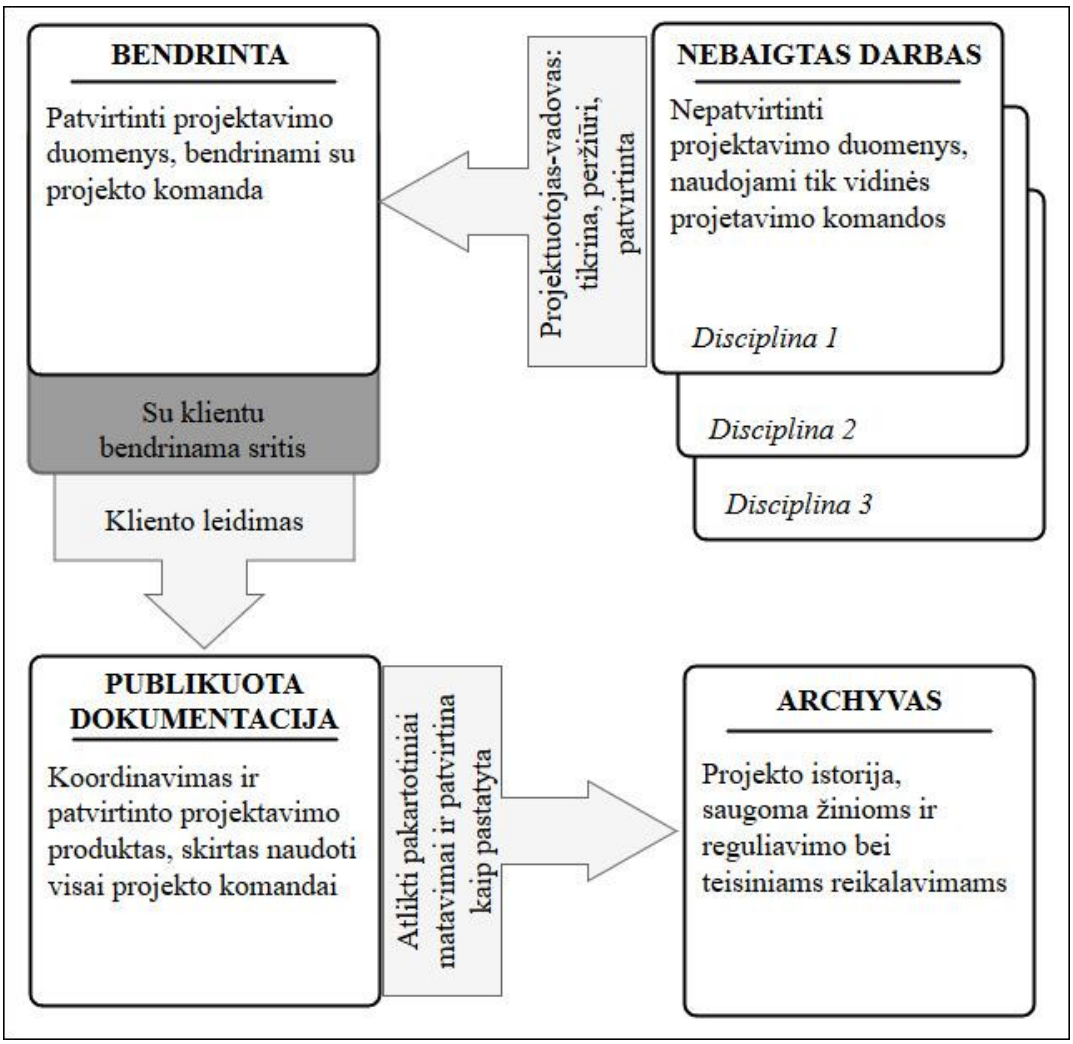
Taigi, pirmiausia reikalinga galimybė projektui ar aplankui sukonfigūruoti didelį skaičių skirtingų dokumento atributų laukų, kuriuose bus saugoma kiekvienam specialistui aktuali informacija.

Kadangi laukų atvaizdavimas priklauso nuo naudotojo rolės, dokumentų metaduomenų ir atributų informacijos diferencijavimui reikalingos skirtingos pagal naudotojų grupes konfigūruojamos DVS aplinkos. Kiekvienam naudotojui galima priskirti arba leisti pasirinkti skirtingas DVS aplinkas, kurios sukonfigūruoja tam tikrus DVS nustatymus – informacijos pateikimo kiekį ir lygį, atvaizdavimo būdą, reikalingas įrankių juostas ir pan.

**Dokumentų būsenos ir automatizuotos darbo eigos**

Dokumentai per savo gyvavimo ciklą kinta ir pereina kelias būsenas. Dokumentų būsenos priklauso nuo organizacijos verslo procesų. AEC industrijoje dokumentų būsenos taip pat gali būti skirtingos, priklausomai nuo projekto reikalavimų. BS 1192 standartas rekomenduoja naudoti skirtingų lygių dokumentų ir duomenų valdymo saugyklą. Jos modelis pateikiamas 7-ame pav.

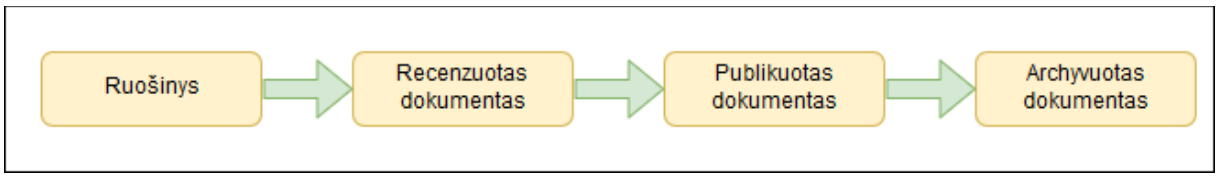
Pagal šį modelį, visi dokumentai ir duomenys organizacijoje turėtų būti saugomi 4-ių lygių saugykloje. Pirmas lygis – skirtingų disciplinų (pvz.: struktūrinė inžinerija, elektros inžinerija) nebaigti darbai. Šiame etape saugomi visi nepatvirtinti projektavimo duomenys, naudojami tik komandos viduje. Projektuotojas-vadovas šiuos dokumentus gali peržiūrėti, patikrinti bei patvirtinti. Jei dokumentai atmetami, jie lieka pirmajame lygyje ir specialistai turi juos koreguoti tol, kol projektuotojas-vadovas patvirtins jų perkėlimą į antrąjį lygį. Antras lygis – bendrinta saugykla, kur saugomi patvirtinti projektavimo duomenys, bendrinami su projekto komanda. Bendrinta saugykla taip pati gali turėti ne tik su projekto vidine komanda bendrinamą sritį, bet ir bendrinamą su klientu. Klientui peržiūrėjus ir patvirtinus dokumentaciją, ji gali būti publikuojama į trečiąjį saugyklos lygmenį, kur dokumentai jau tik koordinuojami ir laikomi patvirtintu projektavimo produktu, skirtu naudotis visai projekto komandai. Laikui bėgant, įgyvendinus projekto statybas ir užbaigimą, gali būti atliekami papildomi matavimai ir dokumentai koreguojami pagal faktinę statinio pastatymo būseną. Laikui bėgant, dokumentai perkeliami į archyvą, kur saugomi kaip projekto istorija organizacijos žinioms bei teisinių reikalavimų atitikimui.




7 pav. **Dokumentų ir duomenų valdymo saugyklos modelis pagal BS 1192 standartą**

(sudaryta autoriaus, remiantis BS 1192:2007)

Specializuotoje DVS toks keturių lygių saugyklos modelis gali būti įgyvendinamas naudojant skirtingas dokumentų būsenas. Pavyzdinis dokumento būsenų kitimo procesas pateikiamas 8-ame pav.




8 pav. **Dokumento būsenų kitimas**

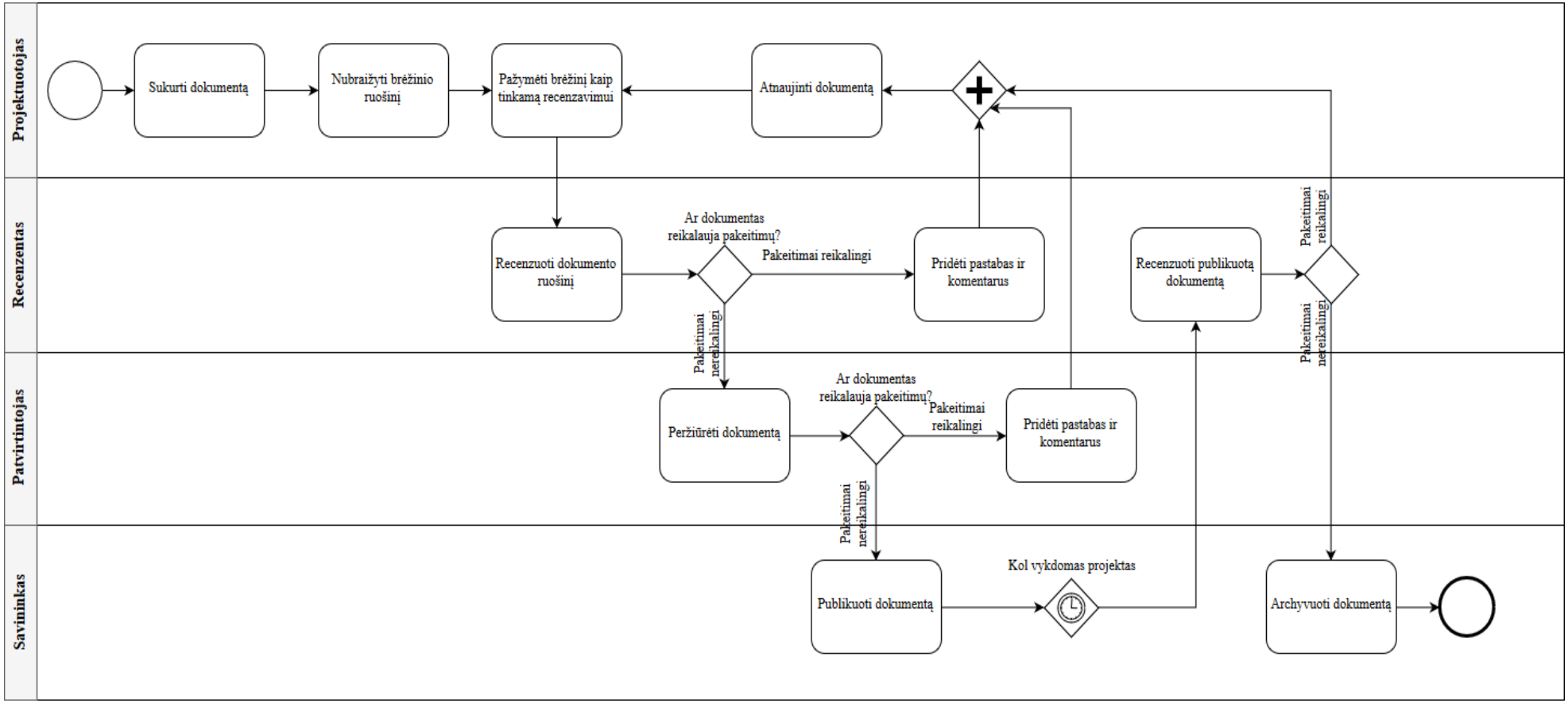
(sudaryta autoriaus)

Praktikoje dokumento būsenos kitimas yra sudėtingesnis ir jo gyvavimo metu gali kisti priklausomai nuo organizacijos verslo procesų. Pavyzdžiui, recenzuotas dokumentas gali būti atmestas ir nepublikuotas bei taip grąžinamas į būseną „ruošinys“. Tai reiškia, kad sistema

privalo palaikyti ne tik dokumento perkėlimą į sekančią būseną, bet ir grąžinimą į ankstesniąją. Be to, keičiant būseną sistema turėtų palaikyti tokias galimybes, kaip komentavimas keičiant būseną, automatinis pranešimų siuntimas suinteresuotoms šalims, dokumento atributų keitimas bei atnaujinimas ir pan.

Taip pat verta paminėti, kad tokio funkcionalumo įgyvendinimas reikalauja papildomų galimybių konfigūruoti naudotojų prieigą pagal dokumentų būsenas. Tokiu atveju užtikrinamas reikalingas skirtingų lygių prieinamumas prie dokumentų ne tik pagal naudotojų roles, bet ir pagal dokumentų būsenas.

Dokumentų būsenų kitimas įgyvendinamas naudojant iš anksto sukonfigūruotas darbo eigas. Pavyzdinis dokumento darbo eigos procesas pateikiamas 9-ame pav.




9 pav. **Dokumento darbo eigos proceso diagrama**

(sudaryta autoriaus)

9-ame pav. pateiktame dokumento gyvavimo ciklo procese dalyvauja keturių skirtingų rolių naudotojai. Jų atliekami veiksmai pavaizduoti skirtinguose „plaukimo takeliuose“. Projektuotojas gali sukurti dokumentą, atlikti savo projektavimo darbus ir pažymėti jį kaip tinkamą recenzavimui. Recenzentas peržiūri ir patikrina dokumento turinį. Jeigu dokumentas reikalauja pakeitimų, recenzentas pažymi pastabas ir komentarus bei grąžina dokumento būseną į pradinę. Projektuotojas savo ruožtu turi atnaujinti dokumentą pagal pateiktas pastabas ir iš naujo pažymėti kaip recenzuotiną. Jeigu recenzento manymu dokumentas pakeitimų nereikalauja, jis pakeičia dokumento būseną į recenzuotą (patvirtintą). Recenzuotą dokumentą dar turi peržiūrėti ir patvirtinti patvirtintojas. Patvirtintojas vėlgi gali dokumentą atmesti palikdamas savo pastabas, jeigu jis reikalauja pakeitimų, ir grąžinti projektuotojui. Jeigu pakeitimai nereikalingi, dokumento būsena pakeičiama į patvirtintą bei tinkamą publikavimui. Dokumento savininkas, pavyzdžiui – projekto vadovas, tada turi teisę publikuoti dokumentą taip padarant jį prieinamą visai projekto komandai. Kol vykdomas projektas, dokumento būsena išlieka „Publikuotas“. Projektui pasibaigus, pagal įgyvendinto projekto dokumentaciją recenzentas atlieka papildomą publikuoto dokumento recenziją. Jeigu pagal faktinius įgyvendinto projekto duomenis dokumentas reikalauja pakeitimų, jis vėl grąžinamas į pirmąją fazę projektuotojui. Jeigu pakeitimai nereikalingi, recenzentas jį pažymi kaip tinkamą archyvavimui. Tuomet projekto savininkas perkelia dokumentą į archyvą nustatytam laiko tarpui.

Specializuotoje DVS didelė dalis tokių darbo eigų procesų yra automatizuojama. Taigi, sistema turi leisti iš anksto sukonfigūruoti darbo eigas, nustatyti atitinkamus parametrus bei atliekamus veiksmus. Vėliau, įgyvendinant darbo eigą, tokie veiksmai, kaip automatinis dokumentų perkėlimas, atributų atnaujinimas, perdavimas kitiems sistemos naudotojams, pranešimų išsiuntimas ir pan., yra atliekami automatiškai.

**Automatinis dokumentų pavadinimų valdymas**

Kadangi AEC industrija yra griežtai reglamentuota įstatymais, dokumentų kūrimas ir valdymas reikalauja atitikimo nustatytiems standartams. AEC industrijos dokumentai priklausomai nuo teisės aktų privalo būti archyvuojami ir saugomi apibrėžtą laiko tarpą. Dėl sugeneruojamo itin didelio kiekio dokumentų, projekto dokumentacijos hierarchinis medis tampa labai didelis ir sunkiai administruojamas. Siekiant suvaldyti šiuos sunkumus, naudojama standartinė dokumentų pavadinimų schema. Šis būdas dar vadinamas dokumento kodu. Lietuvoje tokio dokumento pavadinimą sudaro: projekto numeris, statybų numeris, fazė, suprojektuota dalis ir bylos numeris. (Gabrielaitis, Baušys, 2006).

Taigi, pavyzdinis standartizuotas dokumento pavadinimas galėtų atrodyti taip:

*1234-05-WP-HV-P-02*

Laikantis apibrėžtos dokumentų pavadinimų schemos, lengvai standartizuojama ir struktūrizuojama visa AEC projekto dokumentacija. Tačiau, tai neišsprendžia kitos problemos – kuriant kiekvieną naują dokumentą, tokį unikalų dokumento pavadinimą reikėtų rašyti ranka. Be to, dokumentų pavadinimai gali kisti, priklausomai nuo dokumento būsenos ar bylos numerio.

Šias problemas padeda spręsti specializuota DVS, kuri suteikia galimybę automatizuoti šį procesą visam projekto medžiui nustatant standartinį pavadinimo šabloną, kuris bus pritaikomas projekto dokumentacijai ar tam tikrai jos daliai (pvz. aplankams). Tai reiškia, kad visų naujai įkeliamų ar sukuriamų dokumentų pavadinimai sugeneruojami automatiškai, o kintančios pavadinimo dalys gali būti automatiškai atnaujinamos pasitelkiant apibrėžtas darbo eigos taisykles. Toks DVS funkcionalumas labai palengvina dokumentų administravimą bei archyvavimą, padeda atitikti teisinius reglamentus bei ženkliai sumažina žmogiškųjų klaidų riziką.

**Dokumentų ryšiai ir jų valdymas**

Dėl AEC industrijos specifikos, naudojami dokumentai dažnai tarpusavyje yra susiję ryšiais. Tai gali būti tiek paprasti ryšiai, kai vienas dokumentas yra prisegamas prie kito (pavyzdžiui, susijusios ataskaitos), tiek ir CAD ar BIM failų sąryšiai, kurie gerokai sudėtingesni. Paprastai, projektuojant statinius yra sukuriamas pagrindinis (angl. *master*) failas, kur projektuojamas bendras objekto vaizdas. Tačiau, šis aukšto lygio brėžinys susideda iš daugiau žemesnio lygio komponentų, kurie gali turėti dar žemesnio lygio komponentų ir t. t. Tokie dokumentai vadinami nuorodiniais (angl. *reference*). Priklausomai nuo brėžinio detalumo, vienas objektas gali paveldėti daugybę mažesnių komponentų, iki pat mažiausių statinio dalių (Banerjee *et al.*, 2019). Paprastas tokio scenarijaus pavyzdys būtų toks:

*Statinys > Statinio dalis > Siena > Langas*

Taip atsiranda dokumentų tarpusavio ryšiai ir hierarchija, kai kiekvienas komponentas yra susietas su kitu. Taip pat tai dažniausiai reiškia, kad kiekvienas komponentas yra rezultatas, projektuojamas atskirame dokumente. Tarpusavyje jie yra susiję loginiais ir paveldimumo ryšiais, kurie turi būti valdomi – sukūrus dokumentą atsiranda nauji ryšiai, ištrynus dokumentą, jo ryšiai paveldimi ir pan. Jeigu dokumentai saugomi DVS, šiuos ryšius automatiškai valdo ir atvaizduoja DVS modulis.

Be to, dokumentai yra ne tik susieti loginiais ryšiais, kad būtų paprasčiau juos valdyti, bet ir vienas nuo kito priklausomi. Jeigu atnaujinamas mažesnis brėžinio komponentas, pavyzdžiui – langas, išsaugojus dokumentą šie pakeitimai turi atsispindėti ir pagrindiniame, statinio, dokumente. Taigi, pakeitus ir išsaugojus vieną dokumentą, DVS iškviečia integracijos su AEC programine įranga suteikiamą galimybę automatiškai atnaujinti susijusius dokumentus. Tai įgalina ne tik vieno tiesos šaltinio principą, bet ir leidžia su to paties projekto dokumentais vienu metu dirbti bendradarbiaujant keletui žmonių.

**Brėžinių informacija ir jos sinchronizavimas su DVS**

Tiek skaitmeninis, tiek popierinis techninis brėžinys privalo atitikti tam tikras taisykles ir rekomendacijas. Brėžinių formatas turi reikalavimus dydžiui, kraštinėms ir pan. Kiekvienas brėžinys turi turėti pagrindinę įrašų lentelę (angl. *title block*) arba kitaip – brėžinio legendą, kuri leidžia identifikuoti brėžinį duomenų bazėje. Pagal ISO 7200 standartą, pagrindinių įrašų lentelė yra lentelės forma, susidedanti iš įvairių stačiakampių laukų. (Najman *et al.*, 2001).

Pagal ISO 7200 standartą, pagrindinė įrašų lentelė turi standartinius privalomus bei pasirenkamus laukus, jų dydį bei simbolių skaičiaus limitą. Pavyzdinė pagrindinė įrašų lentelė remiantis ISO 7200 standartu pateikiama 10-ame pav.

<table>
<tr><td>Responsible dept.<br>ABC 2</td><td>Technical reference<br>Patricia Johnson</td><td>Created by<br>Jane Smith</td><td colspan="2">Approved by<br>David Brown</td><td></td></tr>
<tr><td colspan="2" rowspan="3">Legal owner</td><td colspan="2">Document type<br>Sub-assembly drawing</td><td colspan="2">Document status<br>Released</td></tr>
<tr><td colspan="2" rowspan="2">Title, Supplementary title<br>Apparatus plate<br>Complete with brackets</td><td colspan="2">AB123 456-7</td></tr>
<tr><td>Rev.<br>A</td><td>Date of Issue<br>2002-05-14</td><td>Lang.<br>en</td><td>Sheet<br>1/5</td></tr>
</table>

10 pav. **Pagrindinė įrašų lentelė pagal ISO 7200:2004 standartą**
(šaltinis: ISO 7200:2004)

Remiantis ISO 7200 standartu, pagrindinė įrašų lentelė gali turėti iki 20 laukų, iš kurių 8 laukai yra privalomi (tokie kaip *Pavadinimas, Teisinis savininkas, Identifikacijos numeris, Data...*). Likusieji laukai, pvz.: *Papildomas pavadinimas, Atsakingas departamentas* ar *Puslapių skaičius*) nėra privalomi. Taigi, kiekvienas brėžinys savo viduje turi svarbiausią jam charakteringą informaciją, kuri pateikiama specialioje lentelėje.

Žvelgiant iš DVS pusės, pagal gerąsias praktikas pagrindinėje įrašų lentelėje saugomą informaciją turėtų būti galima peržiūrėti neatsidarius paties dokumento, nes tai reikalauja specializuotos PĮ. Tai reiškia, kad šią informaciją reikėtų suindeksuoti ir saugoti duomenų bazėje bei atvaizduoti DVS naudotojui dokumento atributuose. Be to, indeksavimas kartu įgalina ir kitą reikalavimą – brėžinių paiešką pagal svarbiausias jų charakteristikas. Tačiau, neužtenka lentelės suindeksuoti vieną kartą, kadangi informacija jos laukuose kinta (pvz.

laukuose „Patvirtino“ (angl. *Approved By*), „Dokumento statusas“ (angl. *Document status*). Vadinasi, lentelės duomenys dokumento (brėžinio) viduje ir dokumento atributai dokumentų valdymo sistemoje turėtų būti sinchronizuojami. Praktikoje, tai gali būti daroma atliekant operacijas su failais – atidarant, saugant, uždarant ir pan. Pavyzdžiui, jeigu pagrindinės lentelės informacija buvo pakeista iš DVS pusės dokumento atributuose, kitą kartą atidarant failą, suveiks viena iš integracijos su specializuota PĮ funkcijų, iškviečianti automatinį veiksmą atnaujinti brėžinio lentelės informaciją pagal DVS esančio dokumento atributų laukus. Jeigu vykdomas atvirkštinis veiksmas, t. y. informacija pakeičiama pačiame brėžinyje, DVS esančio dokumento atributuose ji atsinaujins, kai naudotojas išsaugos dokumentą.

Kadangi AEC industrijoje vis dar naudojami ir popieriniai brėžiniai, ypač kai naudojamasi senais, paveldėtais brėžiniais, jie dažniausiai skaitmenizuojami (skenuojami) bei įkeliami į DVS. Tuomet informacija iš pagrindinės įrašų lentelės į dokumento atributus automatiškai nepatenka, todėl kai kurios specializuotos DVS siūlo optinį pagrindinės įrašų lentelės atpažinimą (angl. *optical character recognition, OCR*) automatiškai juos perkeliant į dokumento atributus. Žinoma, optinis atpažinimas gali būti ne visada tikslus, todėl žmogaus įsikišimas vis tiek reikalingas, tačiau tokia specializuotos DVS funkcija žymiai supaprastina skaitmenizuotų brėžinių informacijos valdymą.

**Paieška**

AEC industrijos specifika kelia papildomų reikalavimų ir dokumentų paieškos sistemai. Standartinės DVS paieškos sistemos galimybių, tokių kaip dokumentų ar aplankų paieška pagal pavadinimus, failų formatus ar datą, AEC industrijoje nepakanka. Kaip jau minėta, DVS turi palaikyti paiešką pagal brėžiniuose naudojamų pagrindinių brėžinio lentelių informaciją, kuri gali būti sinchronizuojama su dokumento atributais. DVS turėtų palaikyti ir paiešką pagal dokumentų metaduomenis bei paiešką pagal konfigūruojamus dokumentų atributus: projekto kriterijus, klientą, sistemos naudotojus (pvz.: kas sukūrė ar paskutinis redagavo dokumentą), aprašymus, komentarus, pastabas ir t. t. Be to, AEC industrijoje praverčia ir pilno teksto paieška, t. y., kai DVS paieškos sistema suteikia galimybę ieškoti ne tik pagal išorinius dokumento kriterijus, bet ir pagal tekstinių dokumentų viduje esančią informaciją. Tokios išplėstos paieškos sistemos galimybės padaro dokumentų paiešką žymiai lankstesnę.

Kadangi AEC industrijoje naudojami dokumentai yra susiję ryšiais, reikalinga, kad paieškos sistema gebėtų surasti ir susijusius dokumentus bei tinkamai atvaizduoti jų tarpusavio sąryšius. Tai ypač praverčia, kai surandamas pagrindinis failas ir ieškoma visų su juo susijusių nuorodinių failų. Paieškos sistema taip pat privalo leisti ieškoti ir filtruoti dokumentus pagal jų būseną, recenzijos numerį ir kitus darbo eigoje apibrėžiamus atributus.

Kitas svarbus paieškos sistemos iššūkis – dokumentų paieška pagal vietą. Remiantis BS 1192 standartu, dokumentus reikalinga priskirti erdvinėms lokacijoms, t. y. konkrečioms koordinatėms iš anksto įkeltame žemėlapyje (BS 1192:2007). Paieškos sistema turi leisti ieškoti dokumentų pagal pasirinktą lokaciją. Paieška turi būti lanksti ir leisti naudotojui pasirinkti tiek konkretų plotą žemėlapyje, tiek spindulio dydį, apibrėžiantį paiešką apskritime nuo tam tikro pasirinkto taško. Tokia erdvinė paieška leidžia lengviau surasti visus DVS saugomus dokumentus pagal konkrečių projektų, objektų vietą. Tai ypač naudinga, kai organizacija įgyvendina projektą toje pačioje ar netolimoje vietoje nuo jau anksčiau įgyvendinto projekto. Tokiu būdu galima nesunkiai surasti projektavimui ar statyboms aktualią informaciją ir panaudoti ją dar kartą.

# 4. KOKYBĖS KRITERIJAI IR PROJEKTO APLINKA

Be jau aptartų testavimo strategijos elementų grupių, dar svarbu apžvelgti kitas dvi grupes – sistemai keliamus kokybės kriterijus bei testavimo projekto aplinką. Šios elementų grupės taip pat svarbios ir tiesiogiai daro įtaką testavimo strategijos kūrimo metu parenkamoms technikoms, įrankiams ir kitiems sprendimams. Taigi, plačiau šios elementų grupės ir aptariamos šiame skyriuje.

## 4.1. Specializuotos DVS kokybės kriterijai

Kaip nustatyta 1.4 skyriuje, testavimo strategijos kūrimo kontekste kokybės kriterijai atitinka nefunkcinius sistemos reikalavimus. Išanalizavus specializuotos DVS, skirtos AEC industrijai, funkcionalumą, paaiškėjo, kad tokios sistemos turi ir specifinius arba aukštesnio lygmens nei įprasta standartinei DVS nefunkcinius reikalavimus. Tokie reikalavimai kyla iš industrijos specifikos bei naudojimosi sistema poreikių. Pavyzdžiui, kadangi specializuotos DVS naudojamos ne palaikančiosioms organizacijos veikloms, kur sistemos naudotojų skaičius yra gana ribotas, o pagrindinėms veikloms – tokias sistemas vienu metu dažnai turi naudoti daug specialistų. Tai reikalauja, kad sistema palaikytų didelį naudotojų skaičių vienu metu. Šiame skyriuje apibrėžti pagrindiniai kokybės reikalavimai, svarbūs specializuotos DVS testavimo strategijos kūrimui.

**Suderinamumas**

Specializuota DVS privalo tinkamai veikti su nustatyta technine įranga ir minimaliais jos parametrais. Kadangi sistema yra kliento – serverio architektūros ir skirta *Windows* operacinėms sistemoms bei *Microsoft SQL Server* duomenų bazių valdymo sistemoms (DBVS), sistemos diegimas turėtų būti ištestuotas su palaikomomis *Windows* operacinėmis sistemomis ir DBVS, pvz. *Windows 10* kliento aplikacijai, *Windows Server 2019* DVS serveriui ir failų saugyklai, *Microsoft SQL Server 2019* sistemos duomenų bazei. Taip pat turi būti užtikrintas atgalinis sistemos suderinamumas, pavyzdžiui kai kliento aplikacijos yra atnaujinamos į naujausią versiją, o DVS serveris paliekamas senesnės versijos. Be to, sistema turi būti suderinama su kitomis aplikacijomis ar sistemomis, jeigu tai reikalinga pagal funkcinius reikalavimus – palaikyti integraciją su šiomis sistemomis, tačiau netrukdyti jų įprastam darbui. Šiems reikalavimams užtikrinti turi būti sudaroma suderinamumo matrica, o testavimui naudojamas suderinamumo testavimo tipas.

**Diegimas ir plečiamumas**

Sistemos diegimo funkcija klientų organizacijose turėtų būti patikėta konsultantams arba sistemų administratoriams. Diegimo procesas turi būti aiškus, dokumentuotas ir lankstus, t. y. palaikoma keletas skirtingų sistemos diegimo būdų: tiek naudotojo sąsajos pagalba, tiek komandinės eilutės pagalba. Kad diegimo procesas būtų efektyvesnis, sistema turi palaikyti scenarijus (angl. *script*), kurie pagreitintų sistemos diegimą ir konfigūravimą naudojant iš anksto parašytus scenarijus ar jų rinkinius. DVS kliento dalis turėtų palaikyti masinio diegimo į daug atskirų kompiuterių galimybes. Sistemos dieglys taip pat turėtų aptikti ir, jeigu techniškai įmanoma, automatiškai įdiegti trūkstamus privalomus komponentus (pvz. naujausi *Windows* atnaujinimai, reikalingi PĮ karkasai ir pan.). Sistema turi palaikyti ir išdiegimo galimybę, kuria pasinaudojus švariai ištrinami visi failai ir operacinė sistema atstatoma į pradinę būseną.

Sistema turi palaikyti atnaujinimų ir pataisų diegimą. Jeigu atnaujinimai ar pataisos automatiniai, jie neturėtų neigiamai paveikti operacinės sistemos ar aplikacijų veikimo. Jeigu atnaujinimai diegiami rankiniu būdu, jų diegimas turi būti kaip įmanoma paprastesnis. Sistema taip pat turi palaikyti papildomų modulių valdymą: jų diegimą, pakeitimą ir atnaujinimą. Aprašytiems reikalavimams užtikrinti, reikalinga naudoti diegimo galimybių bei plečiamumo testavimo tipus.

**Panaudojamumas ir patrauklumas**

Kadangi analizuojama DVS yra specifinė sistema, skirta AEC industrijai, tai daro įtaką panaudojamumo ir patrauklumo kokybės kriterijams. Žinoma, tokiai DVS galioja ir įprasti panaudojamumo reikalavimai, tokie kaip aiškus sistemos statusas, naudotojo kontrolė, vientisumas, klaidų prevencija, pateikiama pagalba ir dokumentacija. Nepaisant to, tokios DVS panaudojamumas turi būti pritaikytas industrijos specialistams, kurie naudosis sistema. Pavyzdžiui, naudotojams turėtų būti suteikiama galimybė konfigūruoti įvairius parametrus ir objektus (dokumentų atributus, aplinkas, paieškos kriterijus ir kt.) pagal organizacijos ar projekto procesus. Tai reiškia, kad be įprastinių panaudojamumo reikalavimų, sistema turi būti lanksti įvairiems parametrų konfigūravimams ir pritaikymams.

Kadangi tokia sistema nėra skirta plačiajai naudotojų auditorijai, sistemos patrauklumas nėra didelis prioritetas. AEC industrijai aktualus sistemos funkcionalumas, suteikiamos lanksčios konfigūravimo galimybės, tačiau jos išvaizda, dizaino naujumas ar inovatyvumas nėra tiek svarbus. Šiuo atveju, netgi kyla priešingas reikalavimas – dėl industrijos konservatyvumo, sistemų kompleksiškumo bei specialistų (sistemos naudotojų) įpročių, DVS dizainas neturėtų dažnai kisti, ypač tokių objektų kaip meniu, mygtukų ar nustatymų išdėstymas, nes tai tik apsunkintų naudojimąsi ir taip kompleksiška sistema. Tai reiškia, kad

sistemos patrauklumui, dizainui ir naudotojo sąsajai turėtų būti taikomi tęstinumo, panašumo į kitas industrijoje naudojamas sistemas principai, o ne naujausios ir tuo metu populiarios PĮ naudotojo patirties gerinimo rekomendacijos. Šiems reikalavimams užtikrinti naudojamas panaudojamumo testavimas.

**Lokalizavimas**

Kadangi sistema skirta globaliai rinkai, ji turi palaikyti lokalizavimo galimybes. Tai reiškia, kad sistema turi palaikyti papildomų kalbų paketų diegimą ir suteikti galimybę pakeisti sistemos kalbą. Turi būti užtikrinta, kad sistemos kalbos pakeitimas neigiamai nepaveiks sistemos darbo, funkcionalumo ar panaudojamumo. Siekiant užtikrinti lokalizavimo reikalavimus, naudojamas lokalizacijos testavimas.

**Saugumas**

Specializuota DVS reikalauja aukšto saugumo lygio. Toks reikalavimas kyla, nes sistemoje be įprastos jautrios informacijos saugoma ir kita labai svarbi informacija, pvz. statinių, infrastruktūros objektų dokumentai ir brėžiniai. Tai ypač aktualu, kai projektuojami valstybių strateginiai, kariniai, valdžios, infrastruktūros objektai. Taigi, reikalinga naudoti pažangų ir saugų sistemos naudotojų autentifikacijos metodą. Be to, globalus sistemos naudojimas organizacijose reiškia, kad prie DVS prieigą gali turėti labai didelis kiekis naudotojų, kurių teisės matyti tam tikrą informaciją nėra vienodos. Tai kelia poreikį sistemai užtikrinti ir leisti efektyviai valdyti skirtingus naudotojų prieigos lygius. Kad sistemos administratorius kilus poreikiui galėtų peržiūrėti naudotojų atliktus veiksmus, reikalingas detalus sistemos naudojimo veiksmų žurnalizavimas (angl. *audit trail*). Kadangi DVS veikia tinkle, privaloma užtikrinti saugų ir šifruotą duomenų perdavimą, apsaugą nuo įsilaužimų ir kitas panašioms sistemoms taikomas saugumo priemones. Saugumo reikalavimus užtikrina to paties pavadinimo testavimo tipas – saugumo testavimas.

**Pasiekiamumas ir greitaveika**

Projektai AEC industrijoje neretai yra globalūs, o projekto komanda – tarptautinė. Tokia praktika ypač dažnai taikoma didelių infrastruktūros objektų projektų valdyme. Todėl užsakovas, statinio vieta, kurioje dirba statytojai, už projektavimą atsakingos įmonės ir kitos suinteresuotos šalys gali būti išsidėsčiusios visame pasaulyje. Kadangi visoms suinteresuotoms šalims reikalinga prieiga prie dokumentų, tai kelia papildomų reikalavimų ir dokumentų valdymo sistemai. Pirmiausia, ji turi užtikrinti galimybę naudotojams bendradarbiauti ir dirbti su dokumentais nepriklausomai nuo jų geografinės vietos. Tai reiškia, kad sistema turi būti

prieinama ne tik organizacijos vidiniame tinkle, bet ir iš išorės. Sistema taip pat privalo užtikrinti skirtingų laiko juostų palaikymą.

Kadangi sistema naudojama pagrindinėms organizacijos veikloms ir nuo jos yra tiesiogiai priklausomas didelis naudotojų skaičius, reikalingas aukštas sistemos pasiekiamumas. Ne mažiau svarbi ir sistemos greitaveika bei stabilumas. Prie to dar labiau prisideda ir AEC industrijos dokumentų specifika – dideli dokumentų dydžiai ir kiekiai. Pavyzdžiui, standartinis CAD failas užima nuo 1 iki 100 MB ir tokio failo atidarymas iš lokalios failų sistemos trunka 5 – 10 sekundžių (*Panzura*, 2017). Tokių failų kiekis projekte gali būti labai didelis. Modelių failai, žinoma, dar didesni ir gali užimti nuo keleto šimtų megabaitų iki keleto gigabaitų, o jų atidarymas iš lokalaus kompiuterio trukti daugiau nei 10 min. Kai šie dokumentai saugomi nutolusioje DVS saugykloje, jų atidarymo, išsaugojimo ir kitų operacijų laikas gali pailgėti dar kelis kartus. Šios problemos reikalauja, kad DVS galėtų užtikrinti pakankamai aukštą greitaveikos ir pasiekiamumo lygį bei būtų stabili.

Praktiškai pasiekiamumo ir stabilumo reikalavimus įgyvendinti gali padėti specializuotos DVS serverių dubliavimas ir jungimas į klasterius, taip pat pasitelkiant apkrovos išlyginimo (angl. *load balancing*) technologiją. Sistemos greitaveikos problemą spręsti padėtų DVS galimybė glaudinti failus (angl. *compression*). Tai reiškia, kad naudojami dokumentai būtų suglaudinami ir užimtų mažiau vietos, o jų operacijos galėtų vykti greičiau. Kita DVS galimybė užtikrinti greitaveiką – naudoti papildomus spartinančios atmintinės (angl. *caching*) serverius failų saugojimui. Pavyzdinis tokios DVS architektūros modelis pateikiamas 2-ame priede.

Remiantis 2-ame priede pateiktu modeliu, DVS klientų aplikacijos kreipiasi ne tiesiogiai į DVS serverį, o į apkrovos išlyginimo serverį, kuris užklausas paskirsto keletui DVS serverių. Tai užtikrina sistemos stabilumą ir pasiekiamumą – užklausų srautas yra paskirstomas taip neapkraunant vieno serverio, o jeigu vienas iš serverių nustoja veikti, darbą tęsia kitas. Taip pat DVS turi ne tik duomenų bazę ir pagrindinę saugyklą, bet ir papildomą arčiau esančioje geografinėje vietoje esantį spartinančiosios atmintinės DVS serverį, kuriame laikinai saugomos dokumentų kopijos greitam jų pasiekimui. Todėl naudotojui atliekant veiksmus su dokumentu, DVS serveris gali kreiptis ne į nutolusią DVS saugyklą, o į sparčiosios atmintinės serverį. Tik tuo atveju jeigu dokumento sparčiosios atmintinės serverio saugykloje nėra, jis būtų siunčiamas iš pagrindinės DVS saugyklos. Jeigu dokumentas sparčiosios atmintinės serveryje pasenęs ir technologiškai tai įmanoma – siunčiamas ne visas dokumentas, o tik pasikeitusios jo dalys. Be to, spartinančiosios atmintinės serveriai gali būti keli, juos taip pat galima paskirstyti geografiniams regionams. Tokiu atveju sistemos greitaveika dar padidėja.

Taigi, norint pasiekti AEC industrijoje reikalaujamą greitaveikos, stabilumo ir pasiekiamumo lygį, specializuota DVS turėtų palaikyti tokias technologijas ir galimybes, kaip

serverių jungimas į klasterius, apkrovos išlyginimo serveris bei spartinančios atminties serveriai.

Šiems specializuotos DVS reikalavimams užtikrinti reikalinga naudoti efektyvumo, apimties ir streso testavimo tipus bei svarbu testuoti visas palaikomas klientų – serverių konfigūracijas.

## 4.2. Specializuotos DVS kūrimo projekto aplinka

Kaip nustatyta, testavimo strategijos kūrimui įtaką daro ir projekto aplinkos faktoriai. Projekto aplinka gali turėti tam tikrų apribojimų, įpareigoti laikytis standartų ar teisinių reikalavimų arba nulemti testavimo aplinkos kūrimo aplinkybes. Projekto aplinkos faktorių sąrašas nėra baigtinis, nes priklauso nuo kiekvieno projekto specifikos. Taigi, kuriant testavimo strategiją konkrečiai specializuotai DVS, reikėtų apžvelgti ir daug platesnį projekto aplinkos faktorių spektrą, tačiau dėl 1.4 skyriuje aprašytų apribojimų, šio darbo kontekste buvo nustatyti pagrindiniai projekto aplinkos aspektai (suinteresuotos šalys, rizikos, testų vykdymo aplinka ir testavimo duomenys, defektų ir testų valdymas bei industrijos reikalavimai), kurie ir aptariami toliau.

Suinteresuotų šalių rolės ir atsakomybės nustatytos 5 skyriaus 1-oje lentelėje ir šiame skyriuje plačiau aptariamos nebus.

**Rizikos**

Dėl to, kad viena iš specializuotos DVS funkcijų yra palaikyti integraciją su AEC industrijoje naudojama programine įranga (pvz. braižymo, modeliavimo PĮ) bei kita standartine programine įranga (pvz. *Office* paketas), viena pagrindinių rizikų testuojant tokią DVS yra testavimui skirto laiko trūkumas dėl integracijų su kitomis sistemomis testavimo. Norint nepralaimėti konkurencijos, specializuota DVS privalo užtikrinti integracijas su PĮ, kurios naujos versijos yra išleidžiamos kas metus ar netgi dažniau (pvz. *AutoCAD)*. Tai reiškia, kad specializuotos DVS atnaujinimai turės būti leidžiami periodiškai po susijusios PĮ išleidimo. Tai lemia, jog testavimo tvarkaraštis yra tiesiogiai priklausomas nuo kitos PĮ išleidimo laiko ir dažnumo, o tai apsunkina testavimo proceso planavimą. Be to, svarbus ne tik kitos PĮ išleidimo laikas ar dažnumas, bet ir šių išorinių sistemų pakeitimai (naujas ar pakeistas funkcionalumas, architektūriniai pakeitimai ir pan.). Norint suvaldyti šias rizikas, svarbu sudaryti tikslų palaikomos integruotos PĮ versijų sąrašą bei bendradarbiauti su tokias sistemas kuriančiomis organizacijomis tam, kad būtų galima iš anksto sužinoti apie planuojamus pakeitimus bei testavimui gauti dar neišleistas PĮ versijas.

Tikrojo testavimo proceso metu kyla žymiai daugiau rizikų, tačiau kadangi strategija kuriama ne konkrečiai organizacijai ir projektui, į šias rizikas nėra atsižvelgiama, tačiau diegiant testavimo strategiją organizacijoje, svarbu aiškiai apibrėžti visas testavimo procesui įtaką darančias rizikas ir jų valdymą.

**Testų vykdymo aplinka ir testavimo duomenys**

Siekiant užtikrinti efektyvų specializuotos DVS testavimą bei kuo didesnį sistemos padengimą testais, reikalinga turėti keletą testavimo aplinkų. Pirmiausia, būtina testavimą atlikti ne tik imituotoje testavimo aplinkoje, tačiau ir realią produkcinę aplinką atitinkančioje aplinkoje. Testavimas turi būti atliekamas izoliuotose aplinkose, o ne asmeniniuose testuotojų kompiuteriuose. Iš techninės pusės, reikalinga užtikrinti tinkamus techninius resursus, leidžiančius sukurti keletą skirtingų parametrų testavimo aplinkų. Sistema turi būti testuojama pagal parengtą testavimo matricą, taip padengiant visus palaikomų konfigūracijų (duomenų bazė, serverio operacinė sistema, kliento operacinė sistema, integruotų aplikacijų versijos) variantus. Testavimo aplinkų kūrimas turėtų būti automatizuotas, t. y. naudojami virtualizacijos sprendimai su iš anksto paruoštais ir sukonfigūruotais virtualių mašinų šablonais. Sistemos diegimo procesas testavimo aplinkoje turėtų būti automatizuotas (tai negalioja, kai vykdomi sistemos diegimo testavimo atvejai).

Sistemos testavimui turi būti paruošti testavimo duomenys – konfigūraciniai failai, šablonai, dokumentai ir kiti failai bei testavimui reikalingi duomenys. Tokie duomenys papildomai gali būti generuojami automatinio testavimo metu ar naudojant specialius tam skirtus įrankius. Sukurti testavimo duomenys turi būti periodiškai atnaujinami ir papildomi.

**Defektų ir testų valdymas**

Testavimo metu aptikti defektai turi būti tinkamai dokumentuojami bei sekami. Tam naudojamos defektų sekimo sistemos. Testuojant specializuotą DVS, reikalinga pildyti ir sekti aptiktus defektus organizacijos naudojamoje defektų sekimo sistemoje. Panašus principas galioja ir testavimo planų ir atvejų kūrimui, valdymui ir vykdymui. Šiai užduočiai reikalinga pasitelkti organizacijos naudojamus testų valdymo įrankius ir jų naudojimo standartus. Naudojant šiuos organizacijos pateikiamus įrankius, galima tinkamai valdyti defektus ir testus.

**Industrijos reikalavimai**

AEC industrijoje veikiančios organizacijos yra reguliuojamos prižiūrinčių institucijų bei įstatymų, todėl specializuota DVS turi užtikrinti jų laikymąsi. Tai reiškia, jog sistema privalo užtikrinti tokių teisinių reguliavimų vykdymą, kaip dokumentų saugojimo laikas, sistemos saugumo parametrai, standartų laikymasis ir pan. Testavimo proceso metu turi būti testuojamas

ne tik sistemos funkcionalumas ir veikimas, bet ir teisinių reguliavimų paisymas: galimybė valdyti dokumentus remiantis standartais ar gerųjų praktikų rinkiniais (pvz. BS 1192), konfigūruoti sistemą pagal teisės aktuose apibrėžtus reikalavimus ir t. t.

# 5. SPECIALIZUOTOS DOKUMENTŲ VALDYMO SISTEMOS TESTAVIMO STRATEGIJA

Remiantis sudarytu testavimo strategijos kūrimo modeliu ir ankstesniuose skyriuose išanalizuotomis modelio elementų grupėmis, šiame skyriuje pateikiama sukurta struktūrizuota specializuotos DVS testavimo strategija. Kadangi vieno standarto, apibrėžiančio kaip turi atrodyti strategijos dokumentas, nėra, šiam darbui pasirinkta laisva darbo autoriaus sudaryta testavimo strategijos struktūra.

## TESTAVIMO STRATEGIJOS DOKUMENTAS

### 1. Testavimo strategijos tikslas

Testavimo strategijos, kaip organizacinio testavimo proceso, tikslas yra pateikti aukšto lygio rekomendacijas, atliktinus testavimo lygius, metodus bei tipus, galimus testavimo įrankius bei kitus testavimo proceso reikalavimus, skirtus AEC industrijai specializuotos dokumentų valdymo sistemos (toliau šiame dokumente vadinama „DVS“ arba „sistema“) produktui. Testavimo strategija skirta ne tik apibrėžti testavimo proceso strategiją, bet ir daryti testavimo procesą efektyviu, pagerinti testuojamo produkto kokybę.

### 2. Testuojamas objektas ir apimtis

Testavimo strategija skirta AEC industrijai specializuotai DVS. Testuojamas objektas apima visą specializuotą DVS, kuri yra kliento – serverio architektūros programinė įranga, skirta *Windows* (kliento aplikacija) ir *Windows Server* (DVS serveriui ir failų serveriui) operacinėms sistemoms ir naudoja *Microsoft SQL Server* duomenų bazių valdymo sistemą. Testavimo strategijoje atsižvelgiama į sistemos funkcinius ir nefunkcinius reikalavimus bei visą testavimo procesą.

### 3. Rolės ir atsakomybės

Testavimo procese dalyvaujančių suinteresuotų šalių rolės ir pagrindinės atsakomybės (testavimo proceso kontekste) pateikiamos 8-oje lentelėje.

8 lentelė. **Suinteresuotų šalių rolės ir atsakomybės**

| Rolė | Atsakomybės |
|---|---|
| **Testavimo vadovas** | Testavimo proceso priežiūra ir valdymas, planavimas, resursų skyrimas, sprendimų priėmimas, ataskaitų analizė. |
| **Sistemos testuotojas** | Testavimo vykdymas, priežiūra, defektų ir ataskaitų teikimas, pataisytų defektų verifikavimas, naujų testavimo planų ir atvejų kūrimas bei automatizavimas. |
| **Produkto vadovas** | Naudotojų reikalavimų pateikimas ir patikra, funkcionalumo prioritetų ir tvarkaraščio pateikimas. |
| **Sistemos kūrimo vadovas** | Nenutrūkstamo sistemos kūrimo proceso užtikrinimas bei valdymas, naujų sistemos versijų pateikimas testavimo komandai, užduočių skyrimas programuotojams. |
| **Programuotojas** | Sistemos funkcionalumo implementavimas, vieneto ir integracijos testų kūrimas ir vykdymas, aptiktų defektų taisymas pagal gautus testavimo rezultatus. |
| **Saugumo ekspertas** | Tinkamo sistemos saugumo lygio užtikrinimas ir testavimas, saugumo spragų paieška, saugumo ataskaitų pateikimas. |
| **Naudotojo patirties specialistas** | Sistemos grafinės sąsajos projektavimas, dalyvavimas panaudojamumo testavime. |
| **Produkto palaikymo ir dokumentacijos vadovas** | Dokumentacijos kūrimas ir atnaujinimas, dokumentacijos pateikimo užtikrinimas, dokumentacijos defektų taisymas. |
| **Sistemos naudotojas** | Verslo poreikių pateikimas, sistemos priėmimo testavimas, dalyvavimas beta testavime. |

### 4. Suinteresuotų šalių komunikacija

Pagrindinės suinteresuotos šalys pagal poreikį dalyvauja periodiniuose susitikimuose, kur kiekviena šalis pateikia savo tuometinį statusą, informuoja bei aptaria iškilusias problemas, apribojimus, nestandartines situacijas, tvarkaraščio pakeitimus ir kitas temas, kurioms reikalinga komunikacija. Kilus kritinėms situacijoms, atsiradus darbų delsimams ar pan., suinteresuotos šalys kaip įmanoma greičiau informuoja viena kitą apie esamą situaciją el. paštu ar kitais organizacijos komunikacijos kanalais.

## 5. Testavimo matrica ir tvarkaraštis

Prieš kiekvienos sistemos versijos testavimo pradžią, turi būti sudaroma testavimo matrica, kurioje nurodomos visos palaikomos testuotinos sistemos konfigūracijos (pvz.: techninės įrangos parametrai, operacinės sistemos, valdikliai, susijusios informacinės sistemos, integruota programinė įranga). Sudarant testavimo tvarkaraštį, atsižvelgiama į susijusios integruotos programinės įrangos išleidimo datas, komunikuojama su tokias sistemas kuriančiomis organizacijomis, siekiant iš anksto sužinoti naują integruotų sistemų funkcionalumą bei kuo anksčiau gauti dar neišleistas PĮ versijas, kurias būtų galima naudoti DVS testavimui.

## 6. Testavimo technikos ir įrankiai

### 6.1. Testavimo lygiai ir metodai

Sistemos testavimas vykdomas keturiais lygiais, o kiekvieno lygio testavimui vykdyti pasirenkami skirtingi metodai. Kiekvienas iš šių lygių su jiems priskirtais metodais plačiau aptariamas 9-oje lentelėje.

9 lentelė. **Testavimo lygiai ir metodai**

| **Vieneto (angl. *unit*) testavimas** | |
|---|---|
| **Apibūdinimas** | Vieneto testavimo metu testuojami sistemos vienetai (mažiausi įmanomi programinio kodo eilučių rinkiniai). |
| **Tikslas** | Užtikrinti, kad atskiri sistemos vienetai veikia be defektų ir kaip tikėtasi. |
| **Testuojami objektai** | Moduliai, metodai ir kiti programinio kodo komponentai. |
| **Testavimą vykdo** | Sistemos kūrimo komanda. |
| **Testavimo metodas** | Baltosios dėžės testavimas. |
| **Įvesties kriterijai** | Užbaigtas programinis kodas. |
| **Išvesties kriterijai** | • Visi vieneto testai yra sukurti ir įvykdyti.<br>• Nėra neištaisytų defektų. |

9 lentelės tęsinys. **Testavimo lygiai ir metodai**

| **Integracijos (angl. *integration*) testavimas** | |
|---|---|
| **Apibūdinimas** | Integracijos testavimo metu į vieną grupę sujungiama du ar daugiau sistemos vienetų ir vykdomas vienetų grupės testavimas, tikrinant jų tarpusavio integraciją. |
| **Tikslas** | Užtikrinti, kad sistemos vienetų sąsaja veikia be defektų. |
| **Testuojami objektai** | Procedūros, klasės, funkcijos. |
| **Testavimą vykdo** | Sistemos kūrimo komanda. |
| **Testavimo metodai** | • Baltosios dėžės testavimas.<br>• Pilkosios dėžės testavimas. |
| **Įvesties kriterijai** | Užbaigtas vieneto testavimas. |
| **Išvesties kriterijai** | • Visi integracijų testai yra įvykdyti.<br>• Nėra neištaisytų defektų. |
| **Sistemos (angl. *system*) testavimas** | |
| **Apibūdinimas** | Sistemos testavimo metu testuojama visa vientisa sistema. |
| **Tikslas** | Užtikrinti sistemos kokybę ir veikimą pagal reikalavimus. |
| **Testuojami objektai** | Sistema, sistemos dokumentacija, sistemos diegimas ir konfigūracija. |
| **Testavimą vykdo** | Sistemos testavimo komanda. |
| **Testavimo metodai** | • Juodosios dėžės testavimas.<br>• Pilkosios dėžės testavimas.<br>• Baltosios dėžės testavimas (retais atvejais). |
| **Įvesties kriterijai** | Užbaigtas integracijos testavimas. |
| **Išvesties kriterijai** | • Įgyvendinti visi suplanuoti sistemos reikalavimai.<br>• Nėra neištaisytų defektų. |

9 lentelės tęsinys. **Testavimo lygiai ir metodai**

| **Priėmimo (angl. *acceptance*) testavimas** | |
|---|---|
| **Apibūdinimas** | Priėmimo testavimo metu testuojama, ar sukurta sistema atitinka naudotojo lūkesčius, reikalavimus ir verslo procesus. |
| **Tikslas** | Užtikrinti, kad kliento aplinkoje įdiegta sistema iš jo perspektyvos veikia taip, kaip tikėtasi bei atitinka verslo procesus ir sistemos naudotojų poreikius. |
| **Testuojami objektai** | Verslo procesai įdiegtoje sistemoje, operaciniai procesai, konfigūravimas, naudotojų veiklos. |
| **Testavimą vykdo** | Sistemos naudotojai. |
| **Testavimo metodas** | Juodosios dėžės testavimas. |
| **Įvesties kriterijai** | Užbaigtas sistemos testavimas. |
| **Išvesties kriterijai** | Naudotojo patvirtintas sistemos priėmimas. |

### 6.2. Testavimo tipai ir būdai

Remiantis testuojamos sistemos specifika ir funkciniais reikalavimais, funkcinio sistemos testavimo proceso metu naudojami testavimo tipai ir būdai naujam ir esamam funkcionalumui testuoti pateikti 10-oje ir 11-oje lentelėse.

10 lentelė. **Naujo funkcionalumo funkcinio testavimo tipai ir būdai**

| **Testavimo tipas** | **Testavimo būdai** |
|---|---|
| **Paviršinis testavimas** | Rankinis testavimas, rečiau – automatinis testavimas. |
| **Priėmimo testavimas** | Rankinis testavimas, rečiau – automatinis testavimas. |
| **Pilnas funkcinis testavimas** | Rankinis ir automatinis testavimas. |
| **Beta versijos testavimas** | Rankinis testavimas. |

11 lentelė. **Esamo funkcionalumo funkcinio testavimo tipai ir būdai**

| Testavimo tipas | Testuojami funkciniai reikalavimai | Testavimo būdai |
|---|---|---|
| **Dūmų testavimas** | Visi sistemos funkciniai reikalavimai (dūmų testavimas atliekamas kiekvienai sistemos versijai). | Automatinis testavimas, retais atvejais – rankinis testavimas. |
| **Regresijos testavimas** | Standartinės DVS funkcijos. | Automatinis testavimas, retais atvejais – rankinis testavimas. |
| | Išplėstos standartinės DVS funkcijos. | Automatinis testavimas, rankinis testavimas. |
| | Integracija su AEC PĮ. | Automatinis testavimas, retais atvejais – rankinis testavimas. |
| | AEC industrijos standartų palaikymas. | Automatinis testavimas, retais atvejais – rankinis testavimas. |
| | Specifinės AEC industrijai pritaikytos funkcijos. | Automatinis testavimas, rankinis testavimas. |
| | Papildomas funkcionalumas. | Automatinis testavimas, rankinis testavimas. |

Sistemos kokybės kriterijams užtikrinti, atliekamas nefunkcinis testavimas, kurio tipai ir būdai plačiau aptarti 12-oje lentelėje.

12 lentelė. **Nefunkcinių reikalavimų testavimo tipai ir būdai**

| Testavimo tipas | Testuojami nefunkciniai reikalavimai | Testavimo būdai |
|---|---|---|
| **Suderinamumo testavimas** | Suderinamumas ir sistemos konfigūravimo galimybės. | Automatinis testavimas, pagal poreikį – rankinis testavimas. |
| **Konfigūracijos testavimas** | | |
| **Diegimo galimybių testavimas** | Sistemos diegimo procesas. | Rankinis testavimas, rečiau – automatinis testavimas. |

12 lentelės tęsinys. **Nefunkcinių reikalavimų testavimo tipai ir būdai**

| Testavimo tipas | Testuojami nefunkciniai reikalavimai | Testavimo būdai |
|---|---|---|
| **Plečiamumo testavimas** | Plečiamumas. | Rankinis testavimas, rečiau – automatinis testavimas. |
| **Panaudojamumo testavimas** | Panaudojamumas ir patrauklumas. | Rankinis testavimas, pagal poreikį – automatinis testavimas. |
| **Saugumo testavimas** | Saugumas. | Automatinis testavimas, pagal poreikį – rankinis testavimas. |
| **Dokumentacijos testavimas** | Dokumentacija, sistemos pagalba. | Rankinis testavimas. Nuorodų į dokumentaciją tikrinimui – automatinis testavimas. |
| **Lokalizacijos testavimas** | Sistemos pritaikymas kitoms kalboms. | Rankinis testavimas. Sistemos teksto išvertimo padengimui – automatinis testavimas. |
| **Efektyvumo testavimas** | Sistemos pasiekiamumas, stabilumas ir greitaveika. | Automatinis testavimas, rečiau – rankinis testavimas. |
| **Apimties testavimas** | | |
| **Streso testavimas** | | Automatinis testavimas. |

### 6.3. Testavimo įrankiai

Sistemos testavimo procese naudojami toliau pateikti įrankiai.

- **Testų valdymo ir vykdymo įrankis** – testavimo planų ir atvejų kūrimui ir vykdymui.
- **Defektų sekimo sistema** – defektų pildymui, sekimui ir valdymui.
- **Testavimo stebėsenos ir ataskaitų rengimo įrankiai** – stebėti testavimo vykdymo eigą ir progresą, rengti testavimo ataskaitas.
- **Konfigūracijos valdymo įrankiai** – testavimo aplinkoms bei reikalingai PĮ paruošti ir sukonfigūruoti.
- **Testavimo duomenų paruošimo įrankiai** – testavimo duomenų generavimui.
- **Aprėpties matavimo įrankis** – programinio kodo padengimo testais aprėpties nustatymui.
- **Programinio kodo testavimo įrankiai** – atlikti testavimą vieneto ir integracijos testavimo lygiuose.

- **Automatiniai funkcinio ir nefunkcinio testavimo įrankiai** – automatizuoti sistemos testavimo vykdymą.
- **Našumo matavimo įrankiai** – atlikti sistemos efektyvumo, apimties ir streso testavimams.
- **Lokalizavimo testavimo įrankis** – sistemos lokalizacijos testavimui vykdyti.
- **Saugumo testavimo įrankiai** – aptikti potencialias sistemos saugumo spragas.

## 7. Testavimo aplinka ir testavimo duomenys

Jeigu nenurodyta kitaip, testavimas vykdomas izoliuotose, automatizuoto kūrimo testavimo aplinkose, atitinkančiose sistemos palaikomus minimalius techninius reikalavimus. Testavimo aplinkos parinkimas vykdomas pagal iš anksto sudarytą testavimo aplinkų matricą, kuri privalo padengti visas palaikomas sistemos konfigūracijas. Sistema privalo būti testuojama ir produkcinę aplinką atitinkančioje testavimo aplinkoje.

Sistemos testavimui naudojami tiek iš anksto paruošti statiniai testavimo duomenys, tiek dinaminiai testavimo duomenys, generuojami testavimo vykdymo metu (gali būti generuojami automatinių testų metų ar naudojant specialius įrankius).

## 8. Defektų sekimas ir valdymas

Defektų sekimas ir valdymas vykdomas organizacijos naudojamoje defektų sekimo sistemoje. Defektai sistemoje pildomi pagal organizacijos apibrėžtus standartus ir gaires, pateikiant visą reikalingą informaciją: sistemos versiją, testavimo atvejį(-us) testavimo duomenis, naudotą aplinką ir t. t. Tam skirto periodinio susitikimo metu, defektų peržiūrą, prioretizavimą ir patvirtinimą taisymui vykdo produkto vadovas, sistemos kūrimo vadovas ir testavimo vadovas, pagal poreikį įtraukdami ir kitas suinteresuotas šalis. Kritinius sistemos defektus taisymui be specialaus susitikimo, tačiau apie tai informavę atitinkamas suinteresuotas šalis, gali patvirtinti ir sistemos testuotojai, saugumo ekspertai ar kitos suinteresuotos šalys.

Defektų būsenos kitimas sistemoje yra automatizuotas. Kai defektas yra pataisytas naujoje produkto versijoje, defekto verifikavimą vykdo sistemos testuotojas. Jeigu defektas yra pataisytas tinkamai, defekto būsena keičiama į „Uždarytas“, jeigu ne – defekto pataisa atmetama ir defektas grąžinamas į prieš tai buvusią būseną tolimesniam taisymui.

Siekiant užtikrinti testavimo proceso efektyvumą, naudojamas atitinkamas defektų sekimo sistemos funkcionalumas, kaip automatinis pranešimų siuntimas apie naujus defektus, komentarus defektuose, pasikeitusią defektų būseną ir pan.

## 9. Testavimo užbaigimas ir rezultatai

Testavimas laikomas užbaigtu, kai atlikti visi suplanuoti testavimo darbai, testavimo planai laikomi įvykdytais, o visi einamos sistemos versijos defektai yra pašalinti ir verifikuoti. Užbaigiant testavimo procesą turi būti atnaujinama reikalinga informacija: testavimo planai ir atvejai, defektų sąrašas defektų sekimo sistemoje, reikalavimai, testavimo duomenys, vidinė dokumentacija ir t. t., taip pat likviduojamos nereikalingos testavimo aplinkos. Testavimo rezultatais laikomos testavimo rezultatų ataskaitos. Testavimo proceso užbaigimo dokumentą pasirašo testavimo vadovas. Užbaigtas testavimas leidžia atlikti tolimesnius sistemos išleidimo žingsnius.

# IŠVADOS IR PASIŪLYMAI

Darbo išvados pateikiamos toliau.

1. Išanalizavus testavimo proceso poreikį, nustatyta, kad efektyvus sistemų testavimas padeda ženkliai sumažinti programinės įrangos kūrimo kaštus. Kuo ankstesnėje testavimo fazėje aptinkami sistemos defektai, tuo mažesnių kaštų reikalauja jų taisymas.
2. Atlikus testavimo strategijos kūrimo modelių analizę, paaiškėjo, kad sistemų testavimo strategijos kūrimo modeliai skiriasi savo detalumu ir apimtimi, tačiau jų padengiamus elementus galima suskirstyti į keturias pagrindines grupes: testuojamas produktas, testavimo technikos ir įrankiai, kokybės kriterijai bei projekto aplinka. Dėl analizuojamos sistemos specifiškumo ir darbo apribojimų, remiantis gerosiomis analizuotų modelių praktikomis, sukurtas testavimo strategijos kūrimo modelis, skirtas specializuotai DVS.
3. Atlikus teorinę testavimo technikų ir įrankių analizę, struktūrizuotai aptarti testavimo lygiai, tipai, būdai ir metodai. Palyginus rankinį ir automatinį testavimo būdus, pastebėta, kad nepaisant automatinio testavimo pranašumų, automatizavimas dar negali visiškai pakeisti rankinio testavimo. Taip pat nustatyta, kad testavimo procese naudojami įrankiai padeda tinkamai valdyti, pagreitinti ir supaprastinti testavimo procesą.
4. Apibrėžus teorinius DVS aspektus ir naudą organizacijai, nustatytos pagrindinės standartinės DVS funkcijos ir pranašumai. Paaiškėjo, kad AEC organizacijai DVS yra viena svarbiausių IT technologijų, nes padeda valdyti kritinį procesą – dokumentų valdymą, kuris tiesiogiai daro įtaką vykdomų projektų kaštams, kokybei ir įgyvendinimo laikui.
5. Atlikus AEC industrijai pritaikytų specializuotų DVS palyginimo analizę, apibrėžti svarbiausi tokių sistemų funkciniai reikalavimai bei pastebėta, kad funkcinius reikalavimus pagal funkcionalumo sritis galima suskirstyti į šias grupes: standartinės DVS funkcijos, išplėstos standartinės DVS funkcijos, integracija su AEC programine įranga, AEC industrijos standartų palaikymas, specifinės AEC industrijai pritaikytos funkcijos, papildomas funkcionalumas. Ši reikalavimų analizė padėjo atskleisti dokumentų valdymo aspektus AEC organizacijose bei sistemos funkcinius reikalavimus, į kuriuos reikalinga atsižvelgti projektuojant testavimo strategiją.

6. Atlikus specializuotos DVS ypatumų ir gerųjų praktikų analizę, detaliau atskleisti specifiniai DVS funkcionalumo aspektai: integracija su AEC programine įranga, naudotojų rolės ir prieiga, dokumentų atributai ir DVS aplinkos, dokumentų būsenos ir automatizuotos darbo eigos, automatinis dokumentų pavadinimų valdymas, dokumentų ryšiai ir jų valdymas, brėžinio informacija ir jos sinchronizavimas su DVS, paieška. Ši analizė padėjo įsigilinti ir geriau suprasti nestandartinius AEC industrijos organizacijų dokumentų valdymo procesus bei specifinio funkcionalumo keliamus reikalavimus testavimui.
7. Įvykdžius specializuotos DVS kokybės kriterijų analizę, pastebėta, kad svarbiausi testavimo strategijai įtaką darantys sistemos nefunkciniai reikalavimai yra: suderinamumas, diegimas ir plečiamumas, panaudojamumas, lokalizavimas, saugumas bei pasiekiamumas ir greitaveika. Šiems reikalavimams užtikrinti, nustatyti atitinkami testavimo tipai.
8. Atlikus projekto aplinkos analizę, paaiškėjo, kad kuriant testavimo strategiją specializuotai DVS, svarbu atsižvelgti į tokius projekto aplinkos veiksnius: rizikas, suinteresuotas šalis, testų vykdymo aplinką bei testavimo duomenis, defektų bei testų valdymą ir industrijos keliamus reikalavimus.
9. Pritaikius sukurtą testavimo strategijos kūrimo modelį ir išanalizavus jame apibrėžtus strategijos kūrimo elementus, sukurta specializuotos DVS testavimo strategija (pateikiama 5 skyriuje), kurią, remiantis pateiktomis rekomendacijomis, galima įgyvendinti specializuotas DVS kuriančiose organizacijose.

Galimos tolimesnės darbo temos vystymo ir pritaikymo kryptys bei rekomendacijos nurodytos toliau.

1. Testavimo strategijos pritaikymas konkrečiam rinkos produktui (specializuotai DVS) išplečiant panaudotą testavimo strategijos kūrimo modelį ir papildant testavimo strategiją.
2. Specializuotai DVS sukurtos testavimo strategijos įgyvendinimas organizacijoje. Testavimo strategijos įgyvendinimui galima remtis analizuotu „Testavimo brandos modelio integracijos“ dokumentu, kurį pateikia *TMMi Foundation* organizacija.
3. Organizacijos testavimo brandos vertinimas. Šiai krypčiai įgyvendinti, galima remtis *TMMi Foundation* teikiamu „Testavimo brandos modelio integracijos“ dokumente pateikiamu brandos vertinimo modeliu.
4. Organizacijoje įgyvendintos testavimo strategijos efektyvumo tyrimas ir strategijos tobulinimas remiantis atlikto tyrimo rezultatais.

# LITERATŪROS SĄRAŠAS


[1] Acharya, S., Pandya, V. (2013). Bridge between Black Box and White Box - Gray Box Testing Technique. *International Journal of Electronics and Computer Science Engineering*, Vol. 2, No. 1, p. 175–185. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.303.4479&rep=rep1&type=pdf (žiūrėta 2018 m. gruodžio 21 d.).

[2] Agarwal, R., Chandrasekaran, S., Sridhar, M. (2016). *Imagining construction's digital future*. Prieiga per internetą: https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/imagining-constructions-digital-future (žiūrėta 2019 m. lapkričio 23 d.).

[3] Austerberry, D. (2006). *Digital Asset Management*. 2nd ed. Burlington, MA, USA: Focal Press.

[4] Autodesk (2019). *BIM 360 Docs*. Prieiga per internetą: https://www.autodesk.com/bim-360/ (žiūrėta 2019 m. gegužės 18 d.).

[5] Autodesk (2019). *BIM 360. Document Management Guide.* Prieiga per internetą: https://bim360resources.autodesk.com/document-management/document-management (žiūrėta 2019 m. gegužės 25 d.).

[6] Bach, J. (2019). *Heuristic Test Strategy Model*. Version 5.7.2. Prieiga per internetą: https://www.satisfice.com/download/heuristic-test-strategy-model (žiūrėta 2019 m. spalio 22 d.).

[7] Banerjee, P., Mansoor, S., Das, S., Seraogi, B., Patel, A., Majumder, H., … Chaudhuri, B. B. (2019). Automatic Creation of Hyperlinks in AEC Documents by Extracting the Sheet Numbers Using LSTM Model. *TENCON, IEEE Region 10 International Conference*, p. 1667–1672. doi: https://doi.org/10.1109/TENCON.2018.8650296

[8] Bentley Systems (2019). *ProjectWise Design Integration*. Prieiga per internetą: https://www.bentley.com/en/products/product-line/project-delivery-software/projectwise-design-integration (žiūrėta 2019 m. gegužės 20 d.).

[9] Bjork, B.-C. (2001). Document management - a key IT technology for the construction industry. *European Council of Civil Engineers (ECCE) Symposium*, p. 1001–1009. Prieiga per internetą: http://itc.scix.net/data/works/att/ecce-2001-2.content.pdf (žiūrėta 2019 m. kovo 23 d.)

[10] Briski, K. A., Chitale, P., Hamilton, V., Pratt, A., Starr, B., Veroulis, J., & Villard, B. (2008). Minimizing code defects to improve software quality and lower development costs. *Development Solutions. White Paper*. IBM, USA. Prieiga per internetą: ftp://ftp.software.ibm.com/software/rational/info/do-more/RAW14109USEN.pdf (žiūrėta 2018 m. gruodžio 16 d.).

[11] British Standards Institution (BSI) (2007). *Collaborative production of architectural, engineering and construction information – Code of practice* (BS 1192:2007).

[12] Devanand, C. S. C. (2015). Importance of Electronic Document / Information Management Systems in Modern Architectural, Engineering and Construction Projects. *International Research Journal of Engineering and Technology - IRJET*, Vol. 2, No. 3, p. 2197–2204. Prieiga per internetą: https://www.irjet.net/archives/V2/i3/Irjet-v2i3356.pdf (žiūrėta 2019 m. balandžio 8 d.).

[13] *Electronic Records Management Guidelines. Electronic Document Management Systems* (2012). Minnesota State Archives, Minnesota Historical Society. March 2012, Version 5. Prieiga per internetą: http://www.mnhs.org/preserve/records/electronicrecords/docs_pdfs/DocumentMgmt-v5-march2012.pdf (žiūrėta 2019 m. sausio 5 d.).

[14] Errevi System (2019). *Engineering Document Management*. Prieiga per internetą: https://www.engineeringsoftware.eu/EDM (žiūrėta 2019 m. gegužės 20 d.).

[15] European Committee for Standardization (CES). (2004). *Technical product documentation - Data fields in title blocks and document headers* (ISO 7200:2004).

[16] Gabrielaitis, L., Baušys, R. (2006). Electronic document management in building design. *Journal of Civil Engineering and Management*, Vol. 12, No. 2, p. 103–108. doi: https://doi.org/10.1080/13923730.2006.9636381

[17] Garousi, V., Elberzhager, F. (2017). Test Automation: Not Just for Test Execution. *IEEE Software*, Vol. 34, No. 2, p. 90–96. doi: 10.1109/MS.2017.34.

[18] Gelperin, D., Hetzel, B. (1998). The Growth of Software Testing. *Communications of the ACM*, Vol. 31, No. 6, p. 687–695. doi: 10.1145/62959.62965.

[19] International Software Testing Qualifications Board (ISTQB) (2018). *Certified Tester: Foundation Level Syllabus*. Version 2018. Prieiga per internetą: https://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html (žiūrėta 2019 m. sausio 9 d.).

[20] International Software Testing Qualifications Board (ISTQB) (2019). *Standard Glossary of Terms used in Software Testing*. Version 3.3. Prieiga per internetą: https://www.istqb.org/downloads/category/20-istqb-glossary.html (žiūrėta 2019 m. gruodžio 7 d.).

[21] Jensen, C. (2016). Nissan Recalls 3.5 Million Vehicles for Airbag Problems. *The New York Times.* April 29, 2016. Prieiga per internetą: https://www.nytimes.com/2016/04/30/business/nissan-recalls-3-5-million-vehicles-for-airbag-problems.html (žiūrėta 2018 m. gruodžio 20 d.).

[22] Karhu, K., Repo, T., Taipale, O., Smolander, K. (2009). Empirical Observations on Software Testing Automation. *Proceedings of the 2nd International Conference on Software Testing, Verification, and Validation*, p. 201–209. Denver, CO, 1-4 April 2009. doi: 10.1109/ICST.2009.16.

[23] Kasurinen, J. (2011). *Software test process development: PhD dissertation*. Lappeenranta: Lappeenranta University of Technology.

[24] Khan, M. E. (2010). Different Forms of Software Testing Techniques for Finding Errors. *International Journal of Computer Science Issues*, Vol. 7, No. 3, p. 11–16. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.364.9841&rep=rep1&type=pdf (žiūrėta 2019 m. sausio 3 d.).

[25] Khan, S., Rani, U., Prasad, B. V. N., Srivastava, A. K., Selvi, S., Gautam, D. K. (2015). Document management system: An explicit knowledge management system. *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, p. 402–405. New Delhi, 11-13 March 2015. Prieiga per internetą: https://ieeexplore.ieee.org/document/7100281 (žiūrėta 2019 m. sausio 6 d.).

[26] Koekemoer, S., Von Solms, R. (2018). Document management systems-legislative compliance, good governance and municipal practice. *2018 IST-Africa Week Conference (IST-Africa)*, p. 1–10. Gaborone, 9-11 May 2018. Prieiga per internetą: https://ieeexplore.ieee.org/document/8417356 (žiūrėta 2019 m. sausio 10 d.).

[27] Kumar, V. (2012). Comparison of Manual and Automation Testing. *International Journal of Research in Science and Technology*, Vol. 1, No. 1. Prieiga per internetą: http://www.ijrst.com/images/short_pdf/Jun_2012_VIVEK%20KUMAR%201.pdf (žiūrėta 2019 m. sausio 2 d.).

[28] Leitner, A., Ciupa, H., Meyer, B., Howard, M. (2007). Reconciling manual and automated testing: The AutoTest experience. *Proceedings of the Annual Hawaii International*

*Conference on System Sciences*, p. 1–10. Waikoloa, HI, 3-6 Jan. 2007 doi: 10.1109/HICSS.2007.462.

[29] Mateika, D. (2008). *Specialios paskirties dokumentų bendro kūrimo ir naudojimo elektroninės sistemos tyrimas: daktaro disertacija.* Vilnius: Vilniaus Gedimino Technikos Universiteto leidykla „Technika“. Prieiga per internetą: https://vb.vgtu.lt/object/elaba:2045942/2045942.pdf (žiūrėta 2019 m. balandžio 26 d.).

[30] Maurya, V. N., Rajender Kumar, E. (2012). Analytical Study on Manual vs. Automated Testing Using with Simplistic Cost Model. *International Journal of Electronics and Electrical Engineering*, Vol. 2, No. 1, p. 23–35. Prieiga per internetą: https://pdfs.semanticscholar.org/4e64/bc43f4f58244d3d0ddf71164a5c3746bf556.pdf (žiūrėta 2019 m. sausio 3 d.).

[31] McConnell, S. (2004). *Code Complete: A Practical Handbook of Software Construction*. 2nd ed. Redmond, USA: Microsoft Press.

[32] Myers, G. J. (2004). *The Art of Software Testing.* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc.

[33] Naik, K., Tripathy, P. (2008). *Software Testing and Quality Assurance*. Hoboken, NJ, USA: John Wiley & Sons, Inc.

[34] Najman, L., Gibot, O., & Berche, S. (2001). Indexing technical drawings using title block structure recognition. *Proceedings of Sixth International Conference on Document Analysis and Recognition*, p. 587–591. Seattle, WA, USA, 13 Sept. 2001. doi: https://doi.org/10.1109/ICDAR.2001.953857

[35] Panzura (2017). *CAD Theory of Operation & Best Practices. Panzura White Paper*. Prieiga per internetą: https://get.vizion.ai/hubfs/content/white-papers/Panzura-wp-CAD-Theory-of-Operation.pdf (žiūrėta 2019 m. birželio 9 d.).

[36] Pratt, W. S., Connors, M. J. (2017). *Development of the Digital Design Environment ProjectWise ™ – Phase 1*. Prieiga per internetą: https://rosap.ntl.bts.gov/view/dot/32253 (žiūrėta 2019 m. gegužės 15 d.).

[37] Rafi, D. M., Moses K. R. K., Petersen, K., Mäntylä, M. (2012). Benefits and Limitations of Automated Software Testing: Systematic Literature Review and Practitioner Survey. *2012 7th International Workshop on Automation of Software Test (AST)*, p. 36–42. Zurich, 2-3 June 2012. doi: 10.1109/IWAST.2012.6228988.

[38] Royal Institute of British Architects (RIBA) (2013). *RIBA Plan of Work 2013 – Overview*. London: RIBA. Prieiga per internetą: https://www.ribaplanofwork.com/Download.aspx (žiūrėta 2019 m. birželio 1 d.).

[39] Sharma, R. M. (2014). Quantitative Analysis of Automation and Manual Testing. *International Journal of Engineering and Innovative Technology (IJEIT)*, Vol. 4, No. 1, p. 252–257. Prieiga per internetą: http://www.ijeit.com/Vol%204/Issue%201/IJEIT1412201407_46.pdf (žiūrėta 2019 m. sausio 4 d.).

[40] Singh, Y. (2012). *Software Testing*. New York, USA: Cambridge University Press.

[41] Sneha, K., Malle, G. M. (2017). Research on Software Testing Techniques and Software Automation Testing Tools. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, p. 77–81. Chennai, 1-2 Aug. 2017. doi: 10.1109/ICECDS.2017.8389562.

[42] Sommerville, I. (2011). *Software Engineering*. 9th ed. Boston, USA: Addison-Wesley.

[43] Sommerville, J., Craig, N. (2006). *Implementing IT in Construction*. New York, USA: Taylor & Francis.

[44] Tassey, G. (2002). *The Economic Impacts of Inadequate Infrastructure for Software Testing*. National Institute of Standards and Technology, RTI Project Number 7007.011. Prieiga per internetą: https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf (žiūrėta 2018 m. gruodžio 16 d.).

[45] TMMi Foundation (2018). *Test Maturity Model integration (TMMi®)*. Release 1.2. Prieiga per internetą: https://www.tmmi.org/tmmi-documents/ (žiūrėta 2019 m. gruodžio 1 d.).

[46] Tricentis GmbH (2018). *Software Fail Watch.* 5th ed. Prieiga per internetą: https://www.tricentis.com/resources/software-fail-watch-5th-edition/ (žiūrėta 2018 m. gruodžio 21 d.).

[47] Wyher, T. (2016). *5 Reasons Why Manual Testing Is Still Very Important*. Prieiga per internetą: https://dzone.com/articles/5-reasons-why-manual-testing-is-still-very-importa (žiūrėta 2018 m. gruodžio 29 d.).

[48] Zhivich, M., Cunningham, R. K. (2009). The Real Cost of Software Errors. *IEEE Security & Privacy Magazine*, Vol. 7, No. 2, p. 87–90. doi: 10.1109/MSP.2009.56.

**CREATION OF TESTING STRATEGY FOR SPECIALIZED DOCUMENT MANAGEMENT SYSTEM**

**Tadas GERMANAVIČIUS**

**Paper for the Master's degree**

***Strategic Management of Information Systems Program***

Vilnius University, Faculty of Economics and Business Administration,
Department of Economic Informatics

Supervisor – doc. M. Krutinis

Vilnius, 2020

## SUMMARY

83 pages, 12 tables, 10 figures, 48 references, 2 annexes

The *main purpose* of this Master thesis is to create the testing strategy for specialized document management system (DMS) used in Architecture, Engineering & Construction (AEC) industry.

The work consists of five main chapters: Test Process and Testing Strategy, Testing Techniques and Tools, Specialized Document Management System, Quality Criteria and Project Environment, Testing Strategy for Specialized Document Management System.

In the first chapter, Test Process and Testing Strategy, author has justified the demand and importance of software testing and reviewed software testing process. In addition, testing strategy has been outlined in this section, followed by analysis and comparison of three software testing strategy creation approaches: ISO 29119 standard, TMMi (Test Maturity Model Integration) model by TMMi Foundation and Heuristic Test Strategy Model by J. Bach. After reviewing these three approaches, it can be said that all approaches are different in their scope and granularity, but all testing strategy components can be combined into 4 groups: testable product, testing techniques and tools, quality criteria and project environment. Considering complexity of specialized DMS and limitations of this thesis, author has developed a testing strategy creation model dedicated for this specific type of system that is based on good practices of analyzed models.

Second chapter covers one of the testing strategy component groups – testing techniques and tools. By performing literature review, author has reviewed testing approaches, levels, types and methods. A comparison of manual and automated testing has revealed that despite the advantages of automation, automated testing still cannot fully replace manual testing. Also, the

review of testing tools has showed that tools are important part of testing process and help to execute tests, manage testing process and make it more simple.

In the third chapter, author performs a research on the testable product – specialized DMS. The research covers analysis of DMS and its standard functionality, demand and advantages of DMS in AEC industry, requirement analysis (including comparison of three specialized DMS) and more detailed feature analysis of such system. It has been revealed that DMS is one of the main IT systems in the AEC organization. In addition, this research helped to define functional requirements, specific industry processes and key product components that have to be taken into account when creating testing strategy.

Fourth chapter covers two remaining element groups of the strategy creation model – quality criteria and project environment. Quality criteria in this work are understood as non-functional system requirements. Therefore, the author reviews main non-functional requirements (compatibility, usability, security, etc.) and their impact on the testing strategy. As for the project environment, few elements, such as risks, testing environment and test data, defect tracking, test management and industry standards are defined in this section, too.

Fifth chapter is dedicated for the testing strategy for specialized DMS. Based on the created testing strategy model along with the performed analysis and research of its required component groups, author has created the testing strategy document for specialized DMS used in AEC industry.

The conclusions and recommendations part of the thesis summarize the main aspects of literature review together with the results of performed research. The author believes that the developed testing strategy is a useful framework for the implementation of the testing strategy in organizations developing specialized DMS for AEC industry and can be considered as an important tool to make testing process effective.

# PRIEDAI

1 priedas. **Testavimo metodų palyginimas pagal „dėžės“ principą** (šaltinis: Acharya, Pandya, 2013)

| Juodosios dėžės testavimas | Pilkosios dėžės testavimas | Baltosios dėžės testavimas |
|---|---|---|
| Žinios apie taikomosios programos vidinį darbą nėra būtinos. | Turima šiek tiek žinių apie vidinį darbą. | Testuotojas turi visas žinias apie vidinį programos darbą. |
| Atliekamas galutinių naudotojų bei testuotojų ir programuotojų. | Atliekamas galutinių naudotojų bei testuotojų ir programuotojų. | Paprastai atliekamas testuotojų ir programuotojų. |
| Testavimas pagrįstas išorės lūkesčiais – vidinis programos elgesys yra nežinomas. | Testavimas atliekamas remiantis aukšto lygio duomenų bazės ir duomenų srauto diagramomis. | Vidinis darbas yra visiškai žinomas, remiantis juo testuotojas gali projektuoti testavimo duomenis. |
| Užimantis mažiausiai laiko ir mažiausiai išsamus. | Reikalaujantis daugiau laiko ir iš dalies išsamus. | Labiausiai išsamus ir reikalaujantis daugiausiai laiko. |
| Netinkamas algoritmų testavimui. | Netinkamas algoritmų testavimui. | Tinkamas algoritmų testavimui. |
| Gali būti atliekamas tik bandymų ir klaidų metodu. | Gali būti testuojamos duomenų sritys ir vidinės ribos (jei žinomos). | Duomenų sritys ir vidinės ribos gali būti testuojamos geriau. |

2 priedas. **DVS modelis naudojant spartinimo technologijas** (sudaryta autoriaus)