VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Testavimo proceso modeliavimas

Modelling of the testing process

Magistro baigiamasis darbas

Atliko:	Tomas Stončius	(parašas)
Darbo vadovas:	Lekt. Irmantas Naujikas	(parašas)
Recenzentė:	Lekt. Laura Savičienė	(parašas)

Vilnius – 2009

Santrauka

Daugelis testavimo procesų modelių neužtikrina visų programinės įrangos testavimo aspektų. Darbo tikslas – sumodeliuoti testavimo procesą, kuris nurodytų testavimo proceso vietą programų sistemų kūrimo gyvavimo ciklo kontekste, organizacijai, kuriančiai didelės apimties programinę įrangą.

Uždaviniai: ištirti, kaip nuo programų sistemų kūrimo modelių ir metodų („Krioklio“ modelis, iteratyvus gyvavimo ciklo modelis, „V-modelis“, *RAD*, *Scrum*) priklauso testavimo proceso modeliai; Ištirti pasirinkto organizacijos tipo testavimo proceso specifiką, problematiką; Sukurti testavimo proceso modelį organizacijai, kuriančiai programinę įrangą; Įvertinti sukurtą modelį pagal testavimo proceso brandos modelį.

Baigiamajame darbe remtasi literatūros analize, išnagrinėta didelės apimties PS testavimo specifika, taikytini metodai, įrankiai, geriausios praktikos.

Darbo rezultatas – sukurtas ir įvertintas bei optimizuotas programų sistemų testavimo proceso modelis, kuris sujungia geriausias esamų modelių savybes ir yra lengvai plečiamas.

Raktiniai žodžiai: testavimas, procesas, modelis, modeliavimas.

Summary

Many testing process models do not assure software quality in all aspects. The goal of this work is to make a testing process model showing testing process place in the software development lifecycle for the organization, developing large scope software.

Objectives: check how testing process model depends on software development process model (Waterfall model, iterative processes, V-model, RAD, Scrum); Analyze testing process specifics for the chosen organization type; Create testing process model for the organization that creates software; Evaluate new model according to the testing process maturity model.

Work is based on literature analysis: large scope program systems testing process specifics, applicable methods, tools and best practices are analyzed.

The resulting newly created testing process model is evaluated and optimized. The model is based on the best qualities of existing models and is easily adjustable.

Keywords: testing, process, model, modelling.

Turinys

Įvadas

Testavimas – vienas iš pagrindinių programų kūrimo procesų. Bendras testavimo suvokimas yra toks, jog programų sistemos testavimas atliekamas vykdant testus, siekiant išbandyti programinę įrangą. Nebrandus testavimas gali lemti prastos kokybės programinius produktus. Testavimas yra neatsiejama brandaus programų kūrimo proceso dalis ir vyksta lygiagrečiai su programų sistemos kūrimo procesu [Nor01]. Taip patikrinama ir užtikrinama kuriamos programinės įrangos kokybė. Paprastai testavimui skiriama ne mažiau išteklių ir laiko negu programinės įrangos kūrimui, tačiau ne visada pasiekiamas užsibrėžtas tikslas – pagaminti kokybišką ir vartotojo poreikius atitinkančią programinę įrangą.

Nors egzistuoja nemažai testavimo procesų modelių, tačiau visus juos sieja bendri veiksmai. Testavimo veiksmus sudaro: planavimas ir kontrolė, testavimo sąlygų pasirinkimas, testavimo atvejų parinkimas, rezultatų patikrinimas, užbaigimo kriterijaus įvertinimas, ataskaitos, užbaigimas. Testavimo procesas skiriasi organizacijose, bet testavimo ciklas išlieka toks pat (reikalavimų analizė, testavimo planavimas, testų kūrimas, vykdymas, ataskaitų rengimas ir klaidų pakartotinis testavimas).

Pritaikytas testavimo modelis dar neužtikrina kokybiško testavimo. Kyla klausimas, kuo turėtų pasižymėti brandus testavimas. Pirmiausia procesas turi būti valdomas (planuojamas, aprūpinamas ištekliais, kontroliuojamas, turėti organizacinius komponentus), matuojamas, prižiūrimas, efektyvus. Programinės įrangos procesų gerinimo modeliai, tokie kaip CMMI[1] (Gebėjimų brandos integravimo modelis), SPICE[2] padeda pagerinti programinės įrangos brandos lygį. „Žinomi programų kūrimo brandos modeliai (CMM, ISO 15504, ISO 9001), tačiau jie yra bendriniai ir detaliai netyrinėja konkrečių procesų“ [MBE+05]. Taigi jie ir nebus nagrinėjami darbe. Taip pat egzistuoja ir modelių, specialiai sukurtų testavimo procesų brandos lygiui gerinti, pavyzdžiui, TMM[3], TPI[4] [Sta02]. Testavimo proceso brandą galima įvertinti pagal lygius. Kiekviename lygyje atskleidžiamos principinės testavimo proceso problemos. Sistemos testavimo procesų brandos modeliai užtikrina, kad šie tikslai būtų pasiekti.

Testavimas yra artimai susijęs ir su verslo procesais. Pavyzdžiui, sudarant sistemos reikalavimus turi būti kartu ruošiami ir testavimo reikalavimai. Būtent šios problemos testavimo brandos modelis ir neišsprendžia, taigi darbe bus bandoma surasti testavimo proceso vietą

[1] Angl. *Capability Maturity Model Integration*

[2] Angl. *Software Process Improvement and Capability Determination*

[3] Angl. *Test Maturity Model*

[4] Angl. *Test Process Improvement*

bendresnio organizacijos proceso kontekste. Organizacinius ir palaikymo procesus reikia organizuoti taip, kad būtų numatytas testuotojų mokymas, reikiamos testavimo įrangos bei įrankių įsigijimas, valdymo procesai turėtų numatyti testavimo matavimą ir valdymą. Testavimo proceso pasirinkimas priklauso nuo taikomo programų sistemos kūrimo modelio. Kuo sudėtingesnis programavimo proceso valdymas — tuo sunkiau suvaldyti nuolat besikeičiančius programų modulius, testavimo planus. Automatinis testavimas žymiai pagreitina ir supaprastina testavimo procesą, tačiau taip pat gali būti atvejų, kai šis testavimo būdas nerekomenduotinas taikyti. Kai kurie autoriai mano, jog testavimo proceso automatizavimas yra toks brangus, palyginti su gaunama nauda, kad turėtų būti taikomas taupiai. Kiti autoriai rekomenduoja šį metodą taikyti visais atvejais [FM06]. Negana to, lieka neaišku, kokias ir kada testavimo proceso veiklas galima automatizuoti, kad būtų pasiekta maksimali nauda.

Darbo aktualumas ir naujumas

Daugelis testavimo procesų modelių neužtikrina visų programinės įrangos testavimo aspektų [BCS96]. Būta bandymų modeliuoti testavimo proceso veiklas naudojant bendrus parametrus nuo pat 1990-ųjų. Pastarąjį dešimtmetį programinės įrangos industrija dėjo nemažas pastangas gerindama produktų kokybę. Šis procesas buvo sunkus dėl didėjančio programinės įrangos sudėtingumo ir augančių klientų poreikių. Nepaisant padrąsinančių rezultatų taikant įvairius kokybės gerinimo požiūrius, vis dėlto PĮ kūrimo kompanijoms nulio defektų riba nepasiekiama. Tam, kad pagerintų produkto kokybę, programinės įrangos industrija dėmesį telkė į kūrimo procesų tobulinimą. Pavyzdys, kuris buvo plačiai naudojamas kūrimo procesams gerinti, yra gebėjimų brandos modelis.

Gebėjimų brandos modelis (CMM) ir integruotas gebėjimų brandos modelis (CMMI), kuris sujungia skirtingas CMM modifikacijas, dažnai yra laikomi programinės įrangos proceso gerinimo standartu. Nepaisant fakto, kad testavimui tenka 30-40% visų projekto kaštų, CMM (brandos) modelyje testavimo procesui skiriamas ribotas dėmesys [Vee08]. Dėl šios priežasties ir sukurtas TMM – testavimo proceso brandos modelis.

TMM (testavimo proceso brandos modelis) ir TPI (testavimo proceso gerinimo modelis) – pastebimiausi testavimo proceso brandos kontekste. Proceso modelio kokybę galima įvertinti tik pritaikius testavimo proceso brandos modelį. Pastaruoju metu testavimas suvokiamas kaip skaidrus procesas, nes siejamas su plačiai naudojamomis metodikomis ir empiriniais tyrimais, tačiau sunkiai nusakoma testavimo vieta PĮ kūrimo kontekste, o taikomi metodai sunkiai siejami tarpusavyje. Daugeliu atvejų testavimo brandos lygis yra nepakankamas, net jei organizacijos ir inves-

tuoja į kokybės užtikrinimą. Taigi egzistuoja didelė spraga tarp programinės įrangos teorijos ir praktikos.

Testavimo procesas ypač aktualus organizacijoms, kurių veikla nebeįsivaizduojama be programinės įrangos naudojimo. Programų sistemos pirmiausia buvo panaudotos draudimo ir bankininkystės srityse, kuriose apdorojami dideli informacijos kiekiai [SD05]. Bendriausiu atveju didelės apimties programinės įrangos kūrimo projektais laikomi tokie projektai, kurie užtrunka daugiau negu penkerius vieno žmogaus darbo metus. Detaliau didelės apimties programų sistemas apibrėžia [Wu08]. Minėto tipo organizacijoms kuriama programinė įranga pasižymi kompleksiškumu, turi įvairius, skirtingų paskirčių modulius, yra susieta su kitų organizacijų sistemomis, duomenų bazėmis, todėl ir testavimo procesas tampa kritiniu kuriamos sistemos kokybės veiksniu. Vykdančios didelės apimties projektų įgyvendinimą organizacijos šiuo metu pirmauja pagal programinės įrangos užsakymų skaičių [SD05]. Būtent dėl pasirinkto tipo didelės apimties projektų, pasiūlytos testavimo proceso gerinimo praktikos gali būti nesunkiai pritaikytos ir kitose organizacijose, įgyvendinant mažesnius projektus. Mažesnės apimties projektuose, nedidelio prioriteto modelio veiklų vykdymas taptų neprivalomas. Dėl šių priežasčių pasirinkta magistro baigiamojo darbo tema „Testavimo proceso modeliavimas“.

Tikslas

Sumodeliuoti testavimo procesą, kuris nurodytų testavimo proceso vietą programų sistemų kūrimo gyvavimo ciklo kontekste, organizacijai, kuriančiai didelės apimties programinę įrangą.

Uždaviniai

1. Ištirti didelės apimties programinės įrangos testavimo proceso specifiką ir problematiką;
2. Palyginti, kaip nuo programų sistemų kūrimo modelių ir metodų („Krioklio“ modelis, iteratyvus gyvavimo ciklo modelis, „V-modelis“, *RAD*, *Scrum*) priklauso testavimo proceso modeliai;
3. Sukurti testavimo proceso modelį organizacijai, kuriančiai didelės apimties programinę įrangą;
4. Įvertinti sukurtą modelį pagal testavimo proceso brandos modelį.

Pirmoje darbo dalyje apžvelgiamas programų sistemų kokybės užtikrinimo procesas. Čia išaiškinama programinės įrangos defektų svarba produkto kokybei, nagrinėjami bendri metodai, praktikos, kurios padeda užtikrinti kokybę. Taip pat apžvelgiami ir testavimo proceso kokybės užtikrinimo standartai, kurie tarsi sujungia anksčiau nagrinėtus metodus ir priemones naudoja-

mus testavimo procese. Čia aprašoma ir testavimo proceso problematika: kokybės standartų trūkumai, testavimo procesų modelių nebuvimas atskiroms programinės įrangos kūrimo rūšims[5].

Antrojoje dalyje siekiama išsiaiškinti testavimo proceso vietą programų sistemų kūrimo procese, pasiūlomas naujas modelis, parodomi naujo modelio privalumai, jo svarba,.

Galiausiai, trečiojoje dalyje sukurtas modelis tiriamas, detalizuojamas ir tobulinamas. Pridedamos praktikos.

[5] Brandos modelis tam tikrais atvejais yra per daug bendras ir nenagrinėja konkrečių problemų, susijusių su pasirinktu programinės įrangos kūrimo modeliu.

1. Programų sistemų kokybės dalykinės srities ir problemos analizė

Apie kokybę kalbama dažnai, bet apie patį terminą dažnai yra diskutuojama. Visi specialistai sutinka, kad kokybė yra svarbiausias sėkmingo verslo faktorius. Patį terminą sunku tiksliai apibrėžti. Kokybę lengva pastebėti, jei jos nėra [Dap07].

Tinkamos kokybės programinei įrangai sukurti, reikia taikyti kartu ir defektų prevencijos, ir defektų aptikimo priemones. Įprasta defektų prevencijos strategija yra realizuojama modulių integracijos ir sistemos testavimo etapuose. Šie etapai sunaudoja žymią projekto išteklių ir laiko dalį. Siekiant sumažinti projekto vykdymo laiką ir kainą, tuo pačiu didinant programų sistemų kokybę, testavimo procesas tampa pagrindiniu tyrimo objektu [CYT96].

Kokybės užtikrinimas yra procesas, kai apibrėžiama, kaip gali būti pasiekta kokybė ir kada kuriančioji organizacija žino, kad kokybė yra reikiamo lygio. Kokybės užtikrinimas visų pirma yra susijęs su standartų, kurie bus taikomi programinės įrangos kūrimo procesui ir programiniam produktui, apibrėžimu arba parinkimu. Taip pat reikia pasirūpinti standartus palaikančiomis priemonėmis ir metodais [Dap07].

Kuo ilgiau defektai būna nepastebėti programinėje įrangoje, tuo didesnė tikimybė, kad kils su tuo susijusių įvairių problemų. Todėl problemos ar defektai, susiję su reikalavimais produktui, specifikacija, aukšto lygio projektavimu, gali atnešti daug nuostolių ir iš esmės paveikti kitas programinės įrangos gyvavimo ciklo fazes. Kadangi ankstyvose kūrimo fazėse dar nėra programinės įrangos, tai testavimo metodai nėra efektyvūs. Kitos priemonės, kaip inspektavimas ir formalus patikrinimas (verifikavimas), taip pat turi savų trūkumų. Todėl iškyla klausimas, ar įmanoma tokių defektų prevencija.

Klaidų būtų galima išvengti, teisingai apibrėžus reikalingas veiklas ir pakoregavus neteisingas. Toks veiksmas vadinamas klaidų blokavimu. Tai gali būti pasiekta veikiant griežtai pagal apibrėžtus programinės įrangos kūrimo proceso žingsnius, prisilaikant atitinkamų standartų. Tam gali padėti ir tinkamos programinės įrangos kūrimo priemonės – programavimo kalbos, kūrimo metodai [Dap07].

1.1. Testavimo metodai ir priemonės

Testavimo proceso praktikas, kurios vykdomos įdiegus procesą organizacijoje, struktūrizuoti – ne trivialus uždavinys, tačiau jas visas sieja bendras tikslas – užtikrinti kuo aukštesnę programinės įrangos kokybę.

1.1.1. Defektų prevencijos programinės priemonės

Tinkamai naudojamos programinės priemonės taip pat gali sumažinti klaidų tikimybę. Šiam tikslui dažnai naudojamos kūrimo palaikymo priemonės. Jos paprastai palaiko konkrečias kūrimo veiklas, skirtingus aspektus, kūrimo fazes. Dėmesys kreipiamas į palaikymo strategijas, koreguojančias žmogaus veiksmus, susijusius su defektų atsiradimu. Kai kurie tokių priemonių pavyzdžiai:

Programavimo kalbos ir programavimo aplinkos palaikymo priemonės

Pavyzdžiui, sintaksiškai orientuotas redaktorius gali padėti surasti sintaksės klaidas dar programos teksto įvedimo metu. Gali būti palaikomi kodavimo standartai, jie automatiškai tikrinami ir priimama tik tai, kas šiems standartams neprieštarauja.

Pradinio programos kodo ir versijų kontrolės palaikymo priemonės

Šios palaikymo priemonės gali padėti išlaikyti kodo atitikimą projektui, kuris ypač svarbus lygiagrečioms programoms, išskirstytoms didelėms sistemoms, arba atviro kodo sistemoms, kurių kūrimas koordinuojamas internetu. Tokių priemonių naudojimas padeda išvengti nesuderinamumo ir sąsajų problemų tarp skirtingų programinės įrangos arba atskirų komponenčių versijų.

Atskirų individualių programinės įrangos kūrimo veiklų palaikymo priemonės

Šios priemonės gali padėti automatizuoti įvairius dažnai kartojamus darbus, kuriuos atlikdami žmonės nuolat daro klaidų, išlaisvinti žmogų nuo rutininių darbų. Pavyzdžiui:

- Reikalavimų formulavimo priemonės gali padėti surinkti naudotojo reikalavimus tiksliau.
- Projektavimo automatizavimo priemonės gali padėti įvairiuose projektavimo darbuose.

Kitos programinės priemonės, palaikančios procesą, ir anksčiau aprašytas technologijas

Tai paprastai būna programinių priemonių rinkinys.

Norint parinkti tinkamas programines priemones ir jas specialiai paruošti, reikia nemažai papildomo darbo. Taip pat reikalingas efektyvus valdymas ir tinkama priežiūra, kad pasirinktos priemonės būtų tinkamai naudojamos ir padidintų klaidų išvengimo galimybes [Dap07].

1.1.2. Testavimo duomenų generavimo priemonės

Kompanijos surenka didžiulius duomenų apie klientus kiekius. Atliekant testus reikalingas specifinis testinių duomenų parinkimas.

Nagrinėjant didelės apimties programinę įrangą, pavyzdžiui, draudimo srities, tinkama analizuoti informacija yra tikėtinas žalų skaičius per metus ir mažiausiai vienos žalos tikimybė. Abu

skaičiai priklauso nuo keleto kintamųjų. Pavyzdžiui, vairuotojų civilinės atsakomybės draudimo atveju, tarp šių kintamųjų yra automobilio aprašymas, informacija apie klientą, pagrindinį automobilio vairuotoją, kitus automobilio vairuotojus, ankstesnes žalas, nuvažiuotą atstumą per metus ir geografinį regioną. Pavyzdžiui, tarifo reikšmė didėja, kai didėja nuvažiuotas atstumas per metus. Bendriausiu atveju naudojama sudėtinga daugiamatė duomenų struktūra tarp pradinių įeities ir išeities kintamųjų. Dažnai draudimo tarifams konstruoti naudojami ir apibendrinti linijiniai modeliai. Tačiau net šie apibendrinti linijiniai modeliai gali sukelti problemų modeliuojant sudėtingas struktūras. Šiai problemai spręsti naudojami statistiniai mokymosi algoritmai, nes jie remiasi ne parametriniais metodais ir gali parodyti lankstų elgesį daugelyje programų sistemų [MC04].

1.1.3. Modeliais grįstas testavimas

Modeliu grįstas testavimas yra viena iš paklausiausių, greičiausiai besivystančių testavimo automatizavimo priemonių. Susidomėjimas ir modeliu grįsto testavimo naudojimas nuolat auga. Modelis yra pradžios taškas testams generuoti ir skirtas analizuoti testo rezultatus. Daugelis testavimo atvejų gali būti algoritmiškai ir visiškai automatiškai sugeneruoti iš modelio. Jei modelis yra validus, tai yra tiksliai išreiškia, ką testuojama realizacija turėtų daryti, visi testų rezultatai taip pat yra validūs. Modeliu grįsti testavimo metodai veikia „juodosios dėžės“ principu [FT06].

Modelio patikrinimas (angl. *Model Checking*). Naudojant algoritminį verifikavimą, galima patikrinti, ar modelis atitinka specifikacijos reikalavimus. Tikrinama, ar programinės įrangos projektas tenkina formalią specifikaciją. Specifikacija dažniausiai parašyta temporalinės logikos formulėmis. Palyginus su kitais verifikavimo metodais, modelio patikrinimas suteikia šiuos pranašumus:

- tai yra pilnas verifikavimo metodas, naudotojo dalyvavimas yra nebūtinas (naudojant nepilną verifikavimo metodą (pavyzdžiui, testavimą), nėra užtikrinimo, kad neliks neatitikimų tarp modelio ir specifikacijos);
- verifikavimo nurodymai ruošiami visiškai automatiškai;
- naudotojas neprivalo nusimanyti apie verifikavimo metodus;
- klaidos atveju bus betarpiškai gautos specifikacijos savybės arba sistemos būsena ir nustatytas į klaidą vedantis kelias, klaidos priežastys; klaidos analizė bus optimizuota ir gautas bendras efektyvus kūrimo procesas.

Modelio sudarymas – tai abstraktus procesas, kai bandoma sumažinti sistemos sudėtingumą iki „nugalimo“ lygio. Modelyje turi išlikti esminės sistemos savybės, funkcionalumas. Naudojami du pagrindiniai būdai:

- funkcionalumo dekompozicija,
- objektiškai orientuotos paradigmos.

Naudojant funkcionalumo dekompoziciją, sistemos funkcijos skirstomos į smulkesnes, kol gaunamos tokios, kurios gali būti modeliuojamos elgsenos diagramomis. Paprastai atskiros komponentės yra hierarchinės arba lygiagrečios.

Objektiškai orientuotas modeliavimas naudoja, pavyzdžiui, UML diagramas. Bet gali būti pasirinktos ir kitokios diagramos:

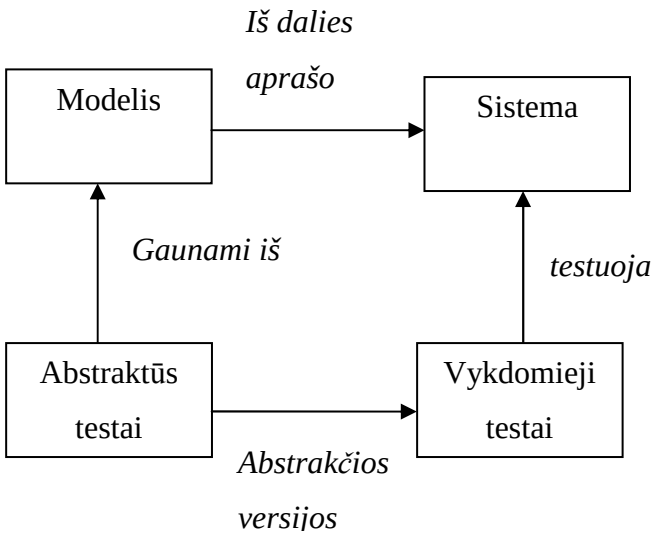
- Panaudos atvejų diagramos skirtos procesams vaizduoti; šiose diagramose matomi įvairūs panaudos atvejai, jų tarpusavio ryšiai ir ryšiai su konkrečiais asmenimis, įvykiais arba procesais;
- Klasių diagramos aprašo objektinio modelio statinę struktūrą, parodo atributus kaip ryšius tarp objektų klasių;
- Elgesio diagramos vaizduoja dinaminius įvykius objektiškai orientuotuose modeliuose;
- Realizacijos diagramos siūlo objektiškai orientuoto modelio realizavimo būdą programos kode.

Modeliais grįstas testavimas tai automatinis testų generavimas panaudojant testuojamos sistemos (*system under test*) modelį.

Testavimo modeliais veiklos:

1. Sistemos modelio kūrimas;
2. Testų generavimas;
3. Testų konkretizavimas;
4. Testų vykdymas;
5. Rezultatų analizė.

1 pav. pateikiama modeliais grįsto testavimo schema.



1. pav. Testavimo modeliais veiklos.

Testavimo modeliais privalumai:

1. Mažesnė testų kūrimo kaina.
2. Sisteminis testavimas.
3. Greitas reagavimas į reikalavimų pakeitimus.
4. Klaidų aptikimas sistemoje.
5. Galimybė automatiškai sekti reikalavimų pakeitimus.

Trūkumai:

1. Modelis gali būti panaudotas netinkamai;
2. Galimas pasenusių reikalavimų panaudojimas;
3. Taikant modeliais grįstą testavimą sugaištamas papildomas laikas klaidoms nustatyti;
4. Modelyje formaliai vertinama daug neaktualių parametrų. Šis procesas užima papildomo laiko.

1.1.4. Regresinis ir automatizuotas testavimas

Regresinis testavimas – tai bet koks testų pakartojimas (dažniausiai po programinės įrangos modifikavimo). Testų kartojimas vykdomas siekiant įrodyti, jog programinės įrangos elgesys nepakito (išskyrus pakeitimus, kurie buvo numatyti modifikuojant PĮ).

Toks testavimo būdas reikalingas tam, kad būtų galima užtikrinti programinės įrangos kokybės valdymą. Turi būti patikrinta, ar naujas programinės įrangos kodas atitinka reikalavimų specifikaciją bei ar programų sistemos modifikavimo metu nebuvo paveiktas senas kodas.

Regresinio testavimo strategijos:

- Regresiniam testavimui yra pakartotinai panaudojami jau sukurti testų rinkiniai.
- Pilnas testavimas (angl. *retest all*)
    - Iš naujo vykdomas kiekvienas testas esantis rinkinyje;
    - Užima daug laiko ir išteklių.
- Dalinis testavimas (angl. *selective retest*)
    - Atrenkami bei pakartotinai panaudojami testai ir modifikuotos programos vietos, kurias reikia testuoti;
    - Testuojant modifikuotą programą stengiamasi sumažinti sunaudojamo laiko ir išteklių kiekį.

Dažniausiai regresinis testavimas užima daug laiko ir pastangų ir daug kainuoja. Norint sumažinti regresinio testavimo kainą galima naudoti automatinį regresinį testavimą.

Automatinio testavimo etapai:

1. Sukurti testavimo atvejį;
2. Vykdyti testavimo atvejį ir patikrinti rezultatą;
3. Jei programa suklysta, pranešti apie defektą ir vėliau bandyti iš naujo;
4. Jei programa įvykdoma sėkmingai, įsiminti rezultatą;
5. Ateities testavimuose palyginti rezultatus su išsaugotais bei pranešti apie klaidą, jei jie nesutampa.

Regresiniai testai yra dokumentuojami ir pakartotinai panaudojami. Kiekvieną kartą kai programa pakeičiama, regresiniai testai turi būti atnaujinami, atsižvelgiant į padarytą pakeitimą. Automatinis regresinis „pertestuoti viską“ metodas nėra brangus laiko atžvilgiu, tačiau daug kainuojantis pradinės analizės ir testų sudarymo metu.

Regresinis testavimas turi būti vykdomas po pakeitimo, netgi patvirtinus, kad viskas atlikta tiksliai ir teisingai. Tokio pobūdžio testų automatizavimas savo naudą atskleidžia tik vėlesniuose projekto etapuose.

1.1.5. Programų sistemos esybių gyvavimo ciklo testavimas

Esybė programų sistemoje yra mažiausias identifikuojamas objektas. Pavyzdžiui, sąskaita arba kliento ID yra esybė bankinėje programų sistemoje. Finansinėje programų sistemoje labai

svarbu identifikuoti esybes ir aiškiai apibrėžti kiekvienos esybės gyvavimo ciklą. Kiekvienam esybės gyvavimo ciklo etapui galima taikyti skirtingus verslo scenarijus.

Gyvavimo ciklo etapų apibrėžimas ir susiejimas su verslo scenarijais yra esybės gyvavimo ciklo testavimo svarbiausias aspektas.

Anot M. Subrahmanyan ir M. Kumar: „Esybės gyvavimo ciklo testavimas yra pati efektyviausia technologija testuojant sudėtingas bankines ir draudimo programų sistemas.“ [SK01].

Didžiausia problema testuojant didelės apimties ir sudėtingą finansų srities programinę įrangą yra nemažas testavimo atvejų ir verslo scenarijų skaičius. Vidinės priklausomybės tarp esybių turi būti aiškiai suprantamos. Kaupiami duomenys privalo būti patikrinti taip, kad rezultatai bet kuriame esybės gyvavimo ciklo etape būtų teisingi. Tai ypač aktualu, kai sistema yra migruojama arba atnaujinama.

Įprastiniai sistemų testavimo metodai negali padengti visų procesų, vykdomų per esybės gyvavimo ciklą [MM01].

Atliekant esybės gyvavimo ciklo testavimą, procesas vykdomas pagal verslo esybės gyvavimo ciklo etapus. Kiekviename gyvavimo ciklo etape esybės būsenos yra aiškiai apibrėžtos, taigi testavimo atvejai taip pat gali būti vienareikšmiškai apibrėžiami. Esybių būsenų persidengimas surandamas automatiškai. Vadinasi, užtikrinamas ir visų testavimo atvejų suradimas.

Kiekviename gyvavimo ciklo etape gali būti skirtingų verslo scenarijų. Taigi kiekvienas verslo scenarijus yra padengiamas „nuo – iki“ gyvavimo ciklo etapais.

Gyvavimo ciklo testavimo etapai:

- Verslo esybių identifikavimas;
- Gyvavimo ciklo etapų identifikavimas;
- Verslo scenarijų kiekviename esybės gyvavimo cikle identifikavimas;
- Verslo scenarijų susiejimas su testavimo atvejais;
- Gyvavimo ciklo testavimo vykdymas pagal iš anksto nustatytus etapus.

1.2. Bendra testavimo proceso modelio kūrimo, vertinimo ir gerinimo praktika

ISO 9000 standartų grupėje aprašomas kokybės valdymas yra vienas žinomiausių. ISO 9001 nustatomi reikalavimai tikrinimui, testavimui, matavimui, testavimo įrangai ir testavimo statuso tikrinimui [HAR04]. Tačiau šis standartas nepritaikytas testavimo modeliams kurti: jis nurodo tik praktikas, skirtas užtikrinti kuriamo produkto kokybę.

Reikalavimai testavimo laboratorijoms pateikti standarte ISO/IEC 17025 (ISO/IEC 17025:2005[6]). Programinio produkto kokybės vertinimui svarbus testavimas. Su testavimu susijęs standartas ISO/IEC 12119 (ISO/IEC 12119:1994[7] panaikintas 2006-03-31, jį pakeitė ISO/IEC 25051:2006). Jame apibrėžti kokybės reikalavimai ir duodamos instrukcijos, kaip pagal šiuos reikalavimus testuoti programinę įrangą. Bet šis standartas taikytinas tik programinės įrangos pasiūlymui ir įsigijimui. Jis nesusijęs su gamybos procesu, specifikacija ir programos kodu. Kokybės apibrėžimui jame vadovaujamasi standartu ISO/IEC 9126[8] [Dap07].

Kiti literatūroje aptariami testavimo proceso brandos bei gerinimo modeliai:

- *The Maturity Model for Automated Software Testing* (MMAST);
- *The Testing Assessment Programme* (TAP);
- *The I.T.B.G. Testing Capability Maturity Model* (TCMM);
- *The Test Improvement Model* (TIM);
- *The Test Organization Maturity Model* (TOM);
- *The Test Process Improvement Model* (TPI);
- *The Testability Support Model* (TSM) [Swi00].

Pastebėtina, jog labiausiai pritaikytas testavimo kokybei vertinti yra TMM modelis, taip pat ir TPI – testavimo proceso gerinimo modelis. Ron Sinkels savo darbe „A comparison of TMM and other Test Process Improvement models“ modelius lygina su TMM. Jis daro išvadą, jog TMM yra „de facto“ pats populiariausias modelis. TPI ir ypač TMM modelis bus naudojami kaip abstraktūs modeliai sukurtajam testavimo proceso modeliui detalizuoti kitose darbo dalyse.

Testavimo brandos modeliai (TMM – *Testing Maturity Model*) sukurti remiantis programinės įrangos gebėjimo brandos modeliais ir susideda iš keleto dalių, kurie nurodo brandos lygį. TMM aprašo procesus, kurie yra pagrindiniai, norint užtikrinti gerai suplanuotą ir kontroliuojamą testavimo procesą. TMM sukurtas Ilinojaus Technologijos Institute. Burnstein (1996), Krause (1994), Olsen ir Vinje (1998), Pol ir Koomen (1999) nagrinėjo TMM [Har04].

Testavimo brandos modeliai nėra nauji. Daugelis jų sukurti apie 1996 m., bet nė vienas plačiai nepaplito. Viena iš priežasčių ta, jog šie modeliai buvo neišsamiai dokumentuoti. Straipsniuose, knygose apie testavimo brandos modelius rašoma labai teoriškai [Sta02].

Testavimo proceso organizacijoje viena didžiausių problemų yra ta, kad testavimas nesiejamas su kitais procesais. Organizacijų darbuotojai labai skirtingai vertina testavimo procesą [Sta02]. Testavimo vertinimui naudojant testavimo proceso brandos modelį ne tik galima doku-

---

[6] *General requirements for the competence of testing and calibration laboratories*

[7] *Information technology – Software packages – Quality requirements and testing*

[8] *International standard for the evaluation of software quality*

mentuoti esamą brandos lygį, bet ir objektyviai įvertinti situaciją. Tik tada, kai proceso branda žinoma, galima gerinti patį procesą [Sta02].

1.2.1. Modernaus testavimo praktikų panaudojimas

Programinės įrangos testavimo procesai, sukurti remiantis programinės įrangos kūrimo modeliais, labai skiriasi. Kai kurie autoriai išskiria praktikas, kurios galėtų tikti bet kuriam testavimo procesui.

Testavimas gali būti sunkiai integruojamas į programinės įrangos kūrimo procesą, tačiau sugebėjimas žvelgti į testavimą iš atitinkamos perspektyvos yra vienas iš būdų valdyti testavimo procesą [PRI03]. M. Pyhajarvi, K. Rautiainen ir J. Itkonen identifikavo geriausias modernaus testavimo praktikas, susiedami testavimą su testavimo kontekstu. Iš viso išskirtos penkios geriausios praktikos, kurias rekomenduotina taikyti projektuose.

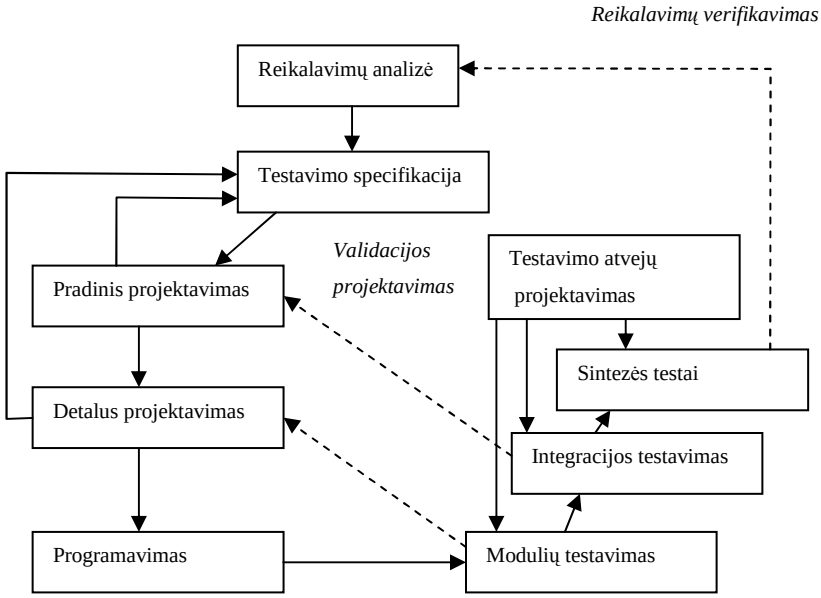
1-oje lentelėje pateikiamos modernaus testavimo praktikos.

1 lentelė. Testavimo praktikos [PRI03].

Praktika	**Aprašymas**
Testavimo pagrindu laikyti produkto ir verslo riziką	Suprasti produkto keitimo ir verslo rizikos ryšį Suskirstyti testavimo atvejus pagal svarbą Suskirstyti testavimo aplinkas Testuoti pagal prioritetą
Destruktyvus požiūris	Tikslu laikyti defektų aptikimą Reikalauti nepriklausomumo
Visų testavimo lygių ankstyvas vykdymas	Peržiūrėti ankstyvo defektų radimo priemones Pabrėžti verifikaciją ir validaciją Visi testavimo lygiai atliekami lygiagrečiai ir nuolat Apsvarstyti, ar reikalingas dokumentacija grįstas testavimas
Pakopiniai testavimo veiksmai	Priklausomumas nuo kitų veiksmų Pakartotinas testavimas ištaisius defektus Testavimo atvejų išskaidymas Kasdienių testuotojų darbų nustatymas Kontroliuojama testavimo aplinka
Trasuojamumas ir prižiūrimumas	Testų grupavimas Trasuojamumo matricos panaudojimas „Lengvo svorio“ testavimo atvejai Apibrėžtų vidinių testų versijų ataskaitos Testavimo grupių tvarkaraštis skirtingoms testavimo aplinkoms

Geram testavimo proceso modeliui sukurti, šios praktikos privalo būti įdiegtos į testavimo proceso modelio veiklas.

1.2.2. Elgsena grįsto testavimo proceso modelis



2 pav. Elgsena grįstas testavimo proceso modelis. [YSJ08].

Geras testavimo modelis turėtų turėti šias charakteristikas:

1) Modelis efektyviai sujungia testavimo planavimo, testavimo atvejų kūrimo, vykdymo ir testų rezultatų analizės veiklas;

2) Būti lengvai pritaikomas prie bet kokios rūšies programinės įrangos, atliekančios skirtingas funkcijas;

3) Modelis suteikia galimybę rasti klaidas kaip galima anksčiau.

Siekiant sujungti testavimo procesą su programinės įrangos kūrimo procesu ir rasti klaidas kaip įmanoma anksčiau, pristatomas pagerintas programinės įrangos testavimo procesas 4 pav.

Šiame modelyje testų specifikacijos gaunamos iš reikalavimų analizės. Testų specifikacijos yra testavimo atvejų projektavimo pagrindas. Modelis iš pagrindų analizuoja ir paaiškina programinės įrangos reikalavimų operacijas ir funkcijas. Kartu su programavimo veikla, sudaromi ir testavimo atvejai. Vykdant procesą, reikalavimų specifikacija nuolat tobulinama. Testavimo scenarijus ir atvejai naudojami atlikti modulių testus tam, kad būtų validuojamas patikslintas projektas, vykdomi integracijos testai bei vykdomi sisteminiai testai reikalavimams verifikuoti.

Modelis pateikia elgsena grįstą struktūrinį testavimo procesą. Vykdomas testavimo planavimas, testavimo atvejų kūrimas, atliekami testai, peržiūrimi testavimo rezultatai, vykdoma jų analizė, regresiniai testai. Išvardintos veiklos įdiegtos į visą programų sistemų kūrimo ciklą. Toks požiūris gali pagerinti tradicinį programinės įrangos testavimo procesą. Šiame modelyje PĮ testavimas yra apibrėžtas ir organizuotas procesas. Projektų vadovai ir testavimo komanda įtrau-

kiama į programinės įrangos kūrimo procesą nuo pat pradžios. Sutvarkomi testavimo tikslai ir reikalavimai, suprojektuojami testavimo atvejai. Negana to, šios veiklos yra įvykdomos prieš programavimo etapą ir testavimo vykdymą. Pagrindinės modelio charakteristikos:

1) Testavimo specifikacija pagrįsta reikalavimų analize. Specifikacija naudojama patikrinti reikalavimus.

2) Testavimo specifikacija yra sistemos ir testo projekto pagrindas, įgalinanti programuotojus kartu su testuotojais suprasti programinės įrangos kūrimo procesą ir testuoti paprasčiau.

3) Nuolatinė komunikacija su programinės įrangos projektuotoju gali pagerinti testavimo specifikaciją. Testavimo specifikacijos gerinimas turi didelės įtakos ir programavimui, ir testavimo atvejų kūrimui. Šis procesas iteratyvus.

4) Testavimo specifikacija yra elgesiu grįsto testavimo pagrindas testavimo vykdymui. Specifikacija naudojama tuo pačiu metu moduliams ir integracijai testuoti.

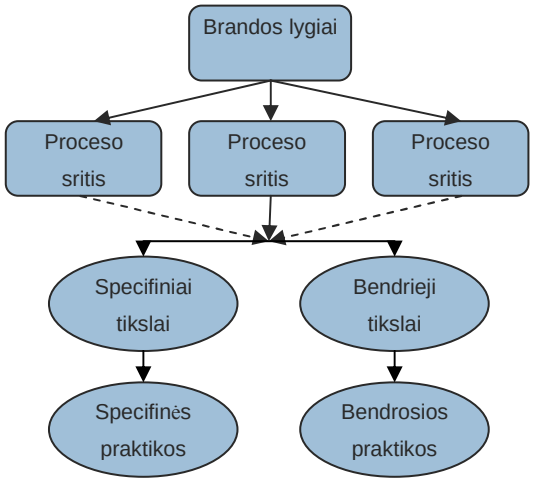
5) Testavimo specifikacija svarbi programinės įrangos kūrimo ir testavimo procesui. Visi reikalavimų pokyčiai keičia ir testavimo specifikaciją. Taigi patogu analizuoti ir kontroliuoti pokyčius kiekviename etape.

6) Reikalavimų specifikacija, testavimo scenarijus ir testavimo atvejai visame testavimo procese gali būti panaudojami ir kuriant kitus produktus [YSJ08].

Lengva pastebėti, jog apibrėžia ne visus PĮ kūrimo etapus, taip pat pasigendama tam tikrų testavimo veiklų, pavyzdžiui, priėmimo testavimo veiklos. Kita vertus, elgsena grįsto testavimo modelis nubrėžia tinkamas gaires naujam testavimo modeliui kurti ir yra pakankamai universalus.

1.2.3. Testavimo brandos (TMM) ir gerinimo (TPI) modeliai

TMM privalumas yra jo suderinamumas su CMM (TMM modelis paremtas tokia pačia struktūra). Testavimo brandos modelis, kaip ir CMM, turi penkis brandos lygius.



3 *pav. Testavimo brandos modelio schema [Vee08].*

TMM Pradinis lygis

Testavimas suvokiamas kaip chaotiškas procesas. Procesas nėra apibrėžtas ir nėra atskirtas nuo programos derinimo. Testavimas atliekamas iš karto po kodavimo be papildomo pasiruošimo. Testavimas ir programos derinimas persidengia: jie abu naudojami programos klaidoms pašalinti. Pagrindinis testavimo tikslas šiame lygyje – parodyti, kad programa veikia. Produktai atiduodami užsakovui be kokybės užtikrinimo. Taip pat labai trūksta išteklių, įrankių ir gerų specialistų. Jokie brandos tikslai šiame lygyje nėra keliami [Nau03].

TMM 1 lygio proceso sričių nėra.

Taigi akivaizdu, kad kiekviena organizacija tenkina šį brandos lygį, nes jis paprasčiausiai neturi reikalavimų.

TMM Apibrėžimo lygis

Testavimas yra atskirtas nuo programos derinimo, apibrėžtas kaip nepriklausoma fazė, kuri eina iš karto po kodavimo. Pastebimos testavimo planavimo užuomazgos – testavimas planuojamas po kodavimo, atsižvelgiant į sukurtą kodą. Šiame lygyje pagrindinis tikslas – parodyti, kad programa atitinka specifikaciją. Naudojamos įvairios testavimo technologijos ir metodai, tačiau kokybės problemos apsiriboja tuo, kad testavimo procesas planuojamas gana vėlai ir negali užtikrinti, jog bus pastebėtos projektavimo ar analizės klaidos, nėra jokių peržiūrų. Pagrindinė testavimo veikla – programos veikimo tikrinimas [Nau03].

TMM 2 lygio proceso sritys yra:

2.1 Testavimo politika ir strategija

2.2 Testavimo planavimas

2.2 Testavimo monitoringas ir kontrolė

2.4 Testavimo projektavimas ir vykdymas

2.5 Testavimo aplinka [Vee08]

Pagrindinis šio lygio išskirtinumas – tai testavimo atskyrimas nuo programavimo.

TMM Integravimo lygis

Šiame lygyje testavimas nėra tik fazė, kuri eina po kodavimo: testavimas integruotas į visą gyvavimo ciklą – jis prasideda reikalavimų sudarymo faze. Testavimo tikslai ir siekiai nustatomi remiantis vartotojų ir užsakovų poreikiais, pagal juos sudaromi ir testavimo scenarijai bei vertinimo kriterijai. Šiame lygyje testavimas suprantamas kaip profesionali veikla ir tam yra specialus organizacinis vienetas. Skiriami reikiami ištekliai ir atliekami mokymai, tačiau testavimo procesas nėra matuojamas, peržiūros nedaromos [Nau03].

TMM 3 lygio proceso sritys yra:

3.1 Testavimo organizavimas

3.2 Testavimo mokymo programa

3.3 Testavimo gyvavimo ciklas ir integravimas

3.4 Nefunkcinis testavimas

3.5 Įdėmios peržiūros [Vee08]

TMM Valdymo lygis

Testavimo procesas yra matuojamas. Peržiūros gyvavimo ciklo metu suprantamos kaip testavimo ir kokybės užtikrinimo veiklos. Tikrinami programinių produktų patikimumo, patogumo ir eksploatavimo parametrai. Testavimo scenarijai renkami ir saugomi duomenų bazėje, tai leidžia pakartotinai juos naudoti ar atlikti regresinius testavimus. Registruojami defektai ir jiems suteikiami svarbumo prioritetai. Paprastai šiame lygyje testavimo procesui trūksta defektų prevencijos, automatizuoto su testavimo procesu susijusių duomenų rinkimo (pagal metrikas), analizės.

TMM 4 lygio proceso sritys yra:

4.1 Testavimo matavimai

4.4. Programinės įrangos kokybė ir vertinimas

4.4. Detaliosios peržiūros [Vee08]

TMM Optimizavimo lygis

Šio lygio testavimas turi numatyti proceso gerinimo mechanizmą. Defektų prevencija ir kokybės užtikrinimas turi būti svarbūs dalykai. Testavimo procesą reikia keisti atsižvelgiant į statistinius parametrus, organizacinius tikslus, užsibrėžtą patikimumo lygį. Turi būti apibrėžta testavimo įrankių ir įrangos įsigijimo procedūra. Naudojami automatizuoti įrankiai testavimo scenarijams tikrinti, jiems kurti, duomenims rinkti, analizuoti ir saugoti.

TMM 5 lygio proceso sritys yra:

5.1 Defektų prevencija

5.2 Testavimo proceso optimizavimas

5.3 Kokybės kontrolė [Vee08]

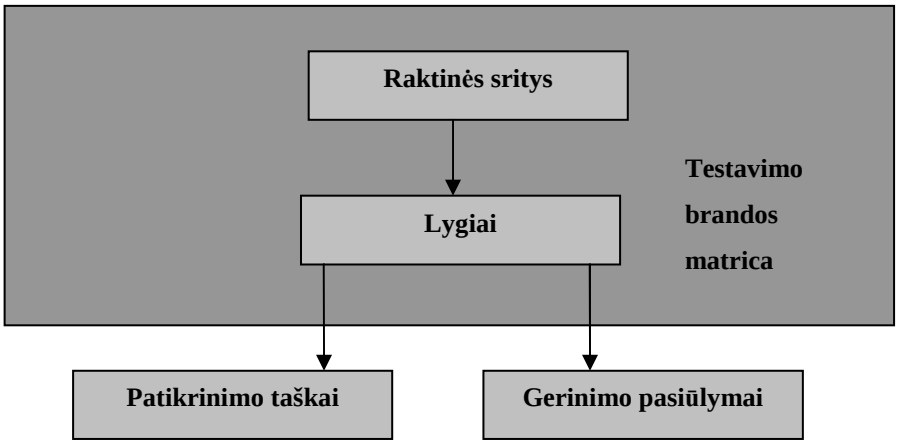
Testavimo proceso gerinimo modelis TPI

Grįstas profesionalių ir patyrusių testuotojų žiniomis testavimo proceso gerinimo modelis (TPI) buvo sukurtas Sogeti, Olandijoje. TPI modelis palaiko testavimo proceso gerinimą. Modelis organizacijoje įskiepija testavimo proceso brandos sampratą. Laikantis šios sampratos, mode-

lis padeda kontroliuojamai ir palaipsniui gerinti procesą. TPI yra labiau į tęstinumą orientuotas modelis, kuris artimas savo struktūriniu požiūriu *TMap* modeliui [Vee08].

Vertinant testavimo proceso brandą TPI modelis nurodo praktinius patarimus, kaip pagerinti testavimo procesą pažingsniui. Modelis susideda iš 20 raktinių sričių, iš kurių kiekviena yra skirtingo brandos lygio. Visi raktinių sričių lygiai yra atvaizduojami brandos matricoje. Kiekvienas lygis aprašomas keliais kontroliniais punktais. Gerinimo pasiūlymai, kurie padeda pasiekti pageidaujamą lygį, yra modelio dalis [KP99].

Testavimo proceso gerinimo modulio schema pateikiama 6 pav.



4 pav. Testavimo proceso gerinimo modelio schema.

1.3. Testavimo proceso problematika

Apžvelgus literatūros šaltinius pastebėta, jog testavimo proceso metodai ir praktikos yra gana detaliai išnagrinėti. Kita vertus, pasigendama sąryšio tarp siūlomų metodų. Taip pat retai arba išvis tik užsimenama apie testavimo proceso vietą programų sistemų kūrimo kontekste.

Pagrindinis tolimesnis uždavinys yra nustatyti, kaip sudaromi konkretūs testavimo proceso modeliai. Surasti bendri metodai pritaikomi visiems testavimo modeliams. Taip pat kokybės standartai bei brandos ir gerinimo modeliai užtikrina bendrą proceso kokybę. Kita vertus, nors testavimas, remiantis antruoju TMM brandos lygiu, ir turi būti atskiras procesas, tačiau jis negali egzistuoti be programų sistemų kūrimo proceso ir privalo būti integruotas bei susietas su PS kūrimo veiklomis.

Testavimo proceso integravimas į bendresnius organizacijos procesus ir yra tema, kuri toliau aptariama nagrinėjant testavimo proceso kūrimo niuansus.

2. Darbo metodai. Testavimo proceso modelio sudarymas

Egzistuoja keletas modelių programinei įrangai testuoti. Šie modeliai parodo, kaip planuoti testavimą, kokia testavimo prasmė. Tačiau testavimas dažniausiai pradedamas tik tada, kai produktas sukuriamas, o tai nėra optimalu. Tuo atveju, kai testavimas vykdomas tik tada, kai produktas jau sukurtas, jis nėra efektyvus, taigi testavimas turi būti vykdomas ir atliekant kitus procesus [Har04]. Kita vertus, Norman Hines teigia, kad: „yra fiziškai neįmanoma ištestuoti programinės įrangos komponento, kol jis yra nesukurtas" [Nor01]. Taigi, kuriant naują testavimo proceso modelį reikia atsižvelgti į šį prieštaravimą ir rasti sprendimą.

2.1. Testavimo procesas programų sistemų kūrimo metodų ir modelių kontekste

Siekiant sudaryti optimalų programinės įrangos testavimo proceso modelį, šioje dalyje nagrinėjama keletas žinomiausių programų sistemų kūrimo modelių ir metodikų.

Kiekvienas modelis vertinamas pagal šiuos kriterijus:

- Šiuolaikiškumas. Pažymima, ar modelis taikomas vykdant didelės apimties projektus.
- Ar egzistuoja testavimo modelių, kurie būtų tinkami integruoti į nagrinėjamą PS kūrimo gyvavimo ciklą.

2.1.1. „Krioklio" modelis

Krioklio modelis yra pakopinis procesas, kuriame į kūrimo procesą žvelgiama kaip į etapus einančius vieną po kito: reikalavimų analizė, projektavimas, programavimas, testavimas.

Modelis kritikuojamas dėl blogų praktikų, daugiausia dėl to, kad neįmanoma, jog reikalavimai nekistų sudėtingame programų sistemų kūrimo procese. Dr. Winston W. Royce savo straipsnyje nurodo, kad Krioklio modelis yra „rizikingas ir vedantis į nesėkmę" [Roy03].

Praktiškai „Krioklio" modelis yra pasenęs todėl, kad yra labai nelankstus ir taikomas labai retais atvejais. Dažniausiai jo dalys naudojamos integravus į kitus modelius ir, pavyzdžiui, nagrinėjamas modelis modifikuojamas į iteratyvų.

„Krioklio" modelis yra pakopinis, todėl testavimas vykdomas užbaigus tam tikrus PS kūrimo etapus. Šitoks sprendimas iš esmės netinkamas, nes prieštarauja TMM reikalavimams ir modernioms testavimo praktikoms, kurios siūlo, jog testavimas turi būti vykdomas lygiagrečiai kūrimui.

Dėl išvardintų priežasčių modelis detaliau nenagrinėjamas.

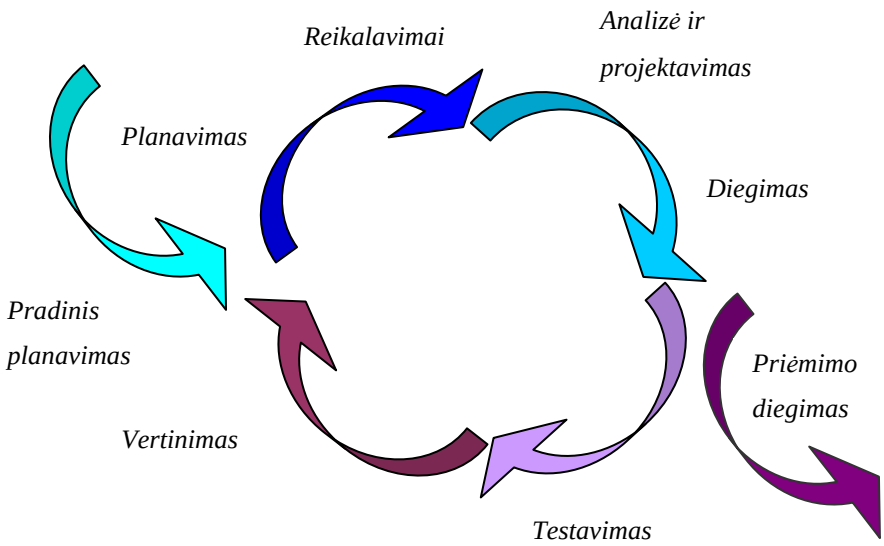
2.1.2. Iteratyvus modelis

Iteratyvus programų sistemų kūrimo procesas yra ciklišKas, sukurtas atsižvelgus į Krioklio modelio neigiamas savybes ir jas pašalinus. Šis procesas pradedamas planavimu ir baigiamas diegimu, kurie susieti ciklišku sąryšiu.

Pagrindinė iteratyvaus proceso nauda yra pažingsninis kūrimo procesas, kuris suteikia galimybę programuotojui pritaikyti ankstesniuose etapuose įgytas žinias.

Nagrinėjamas modelis dažnai taikomas PS kūrimo ir diegimo didelės apimties projektuose, nes yra pakankamai lankstus.

Iteratyvus modelis tinkamas didelės apimties programinei įrangai kurti, tačiau egzistuoja problema, jog toks iteratyvus testavimas literatūroje neapžvelgtas arba aprašytas nepakankamai ir labai abstrakčiai.



5. pav. Iteratyvus programų sistemų kūrimo modelis.

2.1.3. RAD – greitas programinės įrangos kūrimas

Šiuolaikiniame pasaulyje programinė įranga turi būti kuriama kuo greičiau. Būtent *RAD* (greitas programinės įrangos kūrimas) ir pateikia metodiką, kuri gali įgyvendinti šiuos reikalavimus. RAD – tai metodas, kuri siūlo programinę įrangą kurti iteratyviai, skubiai konstruoti prototipus ir naudoti CASE[9] priemones.

[9] *Computer-Aided Software Engineering* – įrankiai ir metodai programinei įrangai kurti.

Remiantis RAD metodika, programinės įrangos kūrimas trunka nuo 30 iki 90 dienų. Tačiau greitas PĮ kūrimas turi ir savų minusų:

- Mažas plečiamumas;
- Maža apimtis;
- Mažas patikimumas didelės apimties ir kritinio tikslumo PĮ [Cat07].

RAD modeliui apibrėžiamos tokios pačios veiklos kaip ir iteratyviame modelyje. Didelės apimties programų sistemai RAD metodika kurti yra netinkama todėl tolimesnė analizė neturi prasmės.

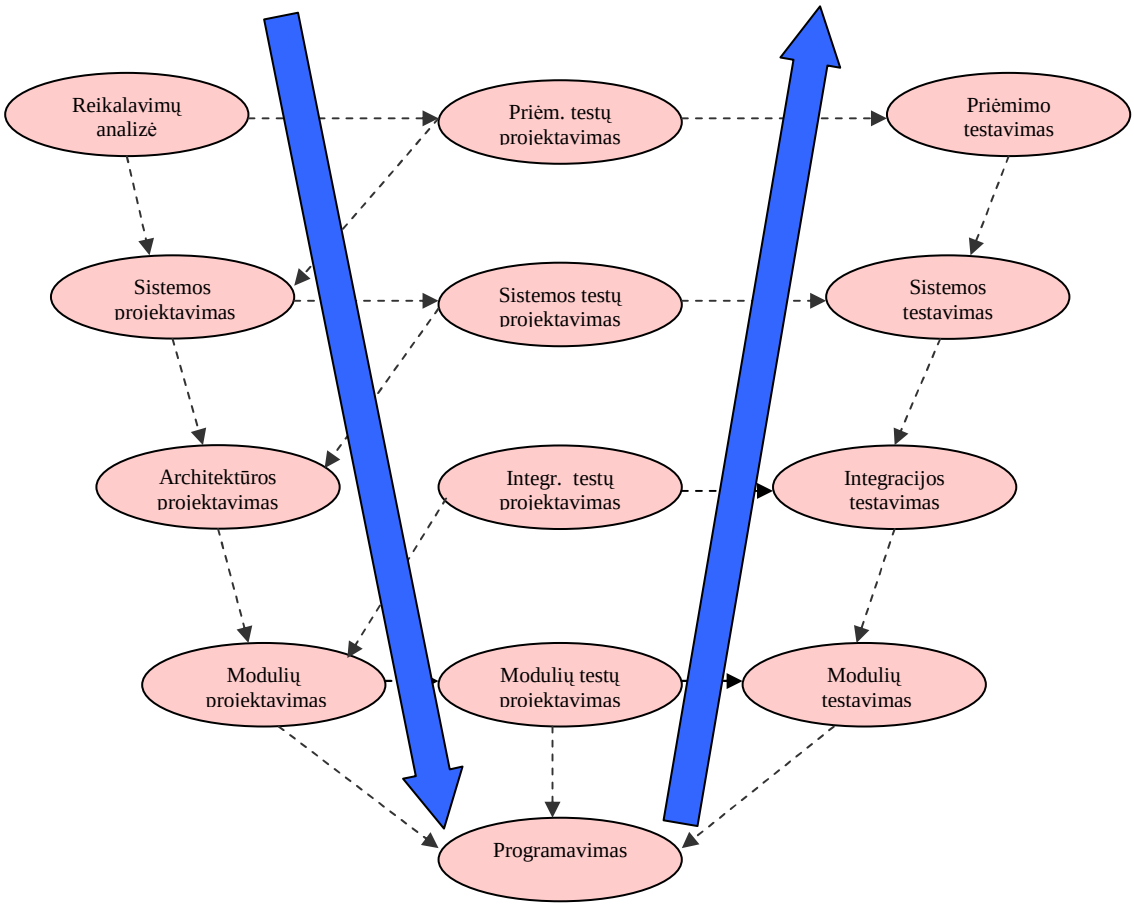
2.1.4. Testavimas ir “Scrum”

Scrum yra projekto valdymo pagrindas, kuris nesusijęs nei su PS kūrimo, nei su testavimo praktikomis. Daugelyje kompanijų *Scrum* naudojamas kartu su *XP* projektams valdyti, o *XP* praktikos taikomos programų sistemų kūrimo procese [Roo07]. Darbe šis modelis pažymimas todėl, kad jis yra kartais taikomas ir didesnės apimties projektuose.

Kita vertus, *Scrum* neaprašomos testavimo praktikos, o *XP* nurodo programinės įrangos kūrimo veiklas. Taigi organizacijos, taikančios *Scrum*, pačios kuria testavimo modelius.

2.1.5. „V-modelis“

Populiarus „V-modelis“, aptariamas literatūroje, susieja testavimo procesą su programinės įrangos kūrimu. Nors „V-modelis“ buvo sukurtas programinei įrangai kurti, jis vėliau apibendrintas ir išplėstas iki testavimo veiklų įtraukimo.



6 pav. Modifikuotas V-modelis [BCS96].

Kairioji „V-modelio“ pusė nurodo reikalavimus, projektavimo ir programavimo etapus bei su testavimu susijusias veiklas, kurios turi būti atliekamos po programavimo etapų. Pavyzdžiui, po projektavimo fazės gali būti vykdoma projektavimo peržiūra bei atliekami integravimo testai. Dešinioji modelio pusė parodo testavimo veiklas, kurios turi būti vykdomos po to, kai programavimas užbaigiamas [BCS96].

„V-modelis“ nustato testavimo veiklas kiekviename gyvavimo ciklo etape nuo pradinių reikalavimų iki galutinio testavimo etapo. Šiame modelyje modulių testavimas patikrina, ar kodas atitinka reikalavimus. Integracijos testavimas patikrina, ar anksčiau testuoti komponentai dera tarpusavyje. Sistemos testavimas patvirtina, ar visiškai integruotas produktas atitinka specifikacijas, o galiausiai „priėmimo testavimas“ nustato, ar galutinis produktas atitinka naudotojo reikalavimus.

M. Pyhajarvi, K. Rautiainen ir J. Itkonen teigimu: pagrįstas tradiciniu „Krioklio“ modeliu, „V-modelis“ yra tradicinis testavimo sprendimas ir negali būti iteratyvus, kad būtų galima taikyti judriąsias metodikas. „V-modelis“ prasideda nuo smulkių dalių ir plečiasi iki stambesnių. Tai

leidžia efektyviau paskirstyti testavimo išteklius. Tokia modelio savybė suteikia privalumą planuojant, tačiau dažnai testavimo procesas tampa neefektyvus.

Testavimo proceso „V-modelis“ smarkiai kritikuojamas, nes jis sukurtas remiantis anksčiau sudarytu programinės įrangos kūrimo „V-modeliu“. Šis modelis ignoruoja faktą, kad programinė įranga yra dažnai keičiama. Tyrimai parodė, kad didelės apimties ir sudėtinga programinė įranga yra dažnai keičiama per programinės įrangos kūrimo gyvavimo ciklą [FM06]. „V-modelis“ pasikliauja tuo, jog programinė įranga yra užbaigta ir gerai dokumentuota. Šis modelis aprašo „tvarkingą procesą“, kurio komunikacija prasta. Taip yra dėl to, jog remiamasi „Krioklio“ principu. Mažiau patyręs testuotojas gali manyti, kad su diegimu susijusi dokumentacija yra užbaigta, nors ji dar nėra patvirtinta tuo metu, kai pradedamas testavimo planavimas. Gali būti nepakankamai aišku, kurios dalys, pavyzdžiui, projektavimas, yra užbaigtos. Tai lemia neproduktyvų darbą, nes testuotojai paprasčiausiai išsigąsta dėl nesuderinamumų.

„V-modelis“ akcentuoja verifikavimą (ar produktas yra kuriamas teisingai?), tačiau verslo pusei svarbesnis validavimas (ar yra kuriamas tinkamas produktas?). Testavimo vadovui „V-modelis“ per griežtas paskirstyti testavimo išteklius tinkamai. Šio modelio naudojimas gali sukurti nelankstų testavimo procesą organizacijoje [PRI03]. Taigi reikalingas dinamiškesnis sprendimas. Pastebima, kad „V-modelis“ panašus į elgsena grįstą testavimo procesą, tačiau yra išsamesnis, todėl darbe tinkamesnis plačiau taikyti.

Nagrinėjamo modelio modifikacijos plačiai taikomos ir net egzistuoja sukurtas testavimo modelis, kuriame veiklos susietos su PĮ kūrimo veiklomis. Vis dėlto, modelis nepakankamai lankstus, per mažai detalus ir formalizuotas.

Didelės apimties, pavyzdžiui, bankinės ir draudimo programų sistemų kūrimo modelis dažniausiai yra iteratyvus. „Krioklio“ modelis tokiai sistemai kurti nėra tinkamas, nes reikalavimai dažnai keičiasi, o kuriant didelės apimties sistemą, klientai suvokia naujo funkcionalumo naudą bei pakeitimų svarbą verslo procese tik įdiegus naują funkcionalumą. Dėl šios priežasties dideli programų sistemų projektai skaidomi į etapus pagal veiklos sritis, kurios viena nuo kitos priklauso tik iš dalies.

Kita vertus, detaliau išnagrinėtas yra tik „V-modelis“, kuris yra kritikuojamas būtent dėl to, kad jis sukurtas „Krioklio“ modelio principu ir nėra iteratyvus.

2.2. Naujas programų sistemų testavimo proceso modelis

Apžvelgus taikomas testavimo proceso modelio kūrimo praktikas, galima pasiūlyti modelį didelės apimties programų sistemai testuoti.

Egzistuojantys modeliai netinkami taikyti organizacijoje, nes:

- „V-modelis“ nors ir plačiausiai taikomas, tačiau pritaikomas savo reikmėms dėl savo netobulumo, nelankstumo;
- Elgsena grįsto testavimo modelis nepadengia visų „V-modelio“ ir TMM siūlomų proceso sričių;
- TMM yra per daug bendras, kad galėtų būti taikomas organizacijoje. Be to, TMM nepasiūlo konkretaus integravimo į PS gyvavimo ciklą sprendimo, o tik užsimena, kad tokio reikia;
- Kiti modeliai sukurti TMM pagrindu ir skirti proceso brandai vertinti arba gerinti, todėl jie taip pat yra per daug bendri.

Dėl šių priežasčių reikalingas naujas modelis, kuris sujungtų pastarųjų modelių geriausias savybes.

Naujo testavimo proceso modelio konstravimo žingsniai:

1. Apibrėžiama modelio notacija remiantis BPMN, kuris modifikuojamas siekiant optimizuoti darbo rezultatą. Parodomi modelyje naudojami elementai.
2. Aprašomas testavimo modelio metamodelis, skirtas paaiškinti vartojamas sąvokas ir naudojamus elementus, jų hierarchiją.
3. Pasirenkamas modelio veiklų sąrašas remiantis iteratyviu ir „V-modeliu“.
4. Apibrėžiami sąryšiai tarp modelio vykdomų veiklų.
5. Trečiojoje darbo dalyje brandos ir gerinimo modelių praktikos pritaikomos siūlomame modelyje.

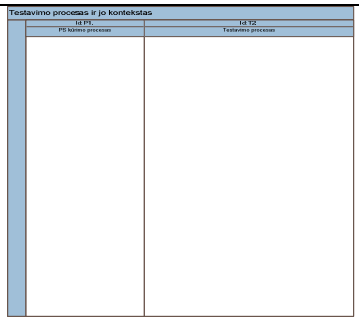
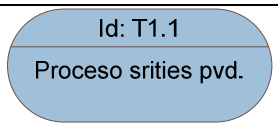
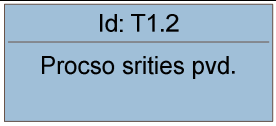
2.2.1. Modelio notacija

Darbo tiksle nurodoma, jog kuriamas modelis privalo susieti testavimo procesą su programų sistemų kūrimo procesu. Modeliui aprašyti reikalinga pasirinkti notaciją. Sąryšiams tarp procesų sričių pavaizduoti tinkamiausias grafinis pavidalas. Verslo proceso notacija BPMN (angl. *Business Process Modelling Notation)* pasirinkta, todėl, kad „svarbiausias BPMN tikslas yra pasiūlyti notaciją, kuri yra lengvai suprantama visų naudotojų: nuo analitikų, kurie kuria procesus,

iki programuotojų ir kitų verslo atstovų [Whi09].“ BPMN apibrėžia naudojamus grafinius elementus ir leidžia pagal poreikius pakeisti esamus bei pridėti naujus.

2-oje lentelėje pateikiamas modelio elementų aprašas. Naudojamos notacijos pagrindas [Whi09]. Iš esmės modifikuotas tik inicializacijos ir pabaigos elementai. Šiuo pakeitimu siekta aiškiau pavaizduoti modelį grafiškai atsisakant standartiškai siūlomų pradžios ir pabaigos įvykių elementų, kurie praktikoje naudojami tik formaliai aprašant įvykius, o ne procesus. Abstraktumo dėlei atsisakyta ir sąlygos elementų, kurie naudojami ne proceso, o įvykių lygyje aprašant praktikas. Pagrindinis elementas – proceso sritis, kuri turi pavadinimą ir identifikatorių.

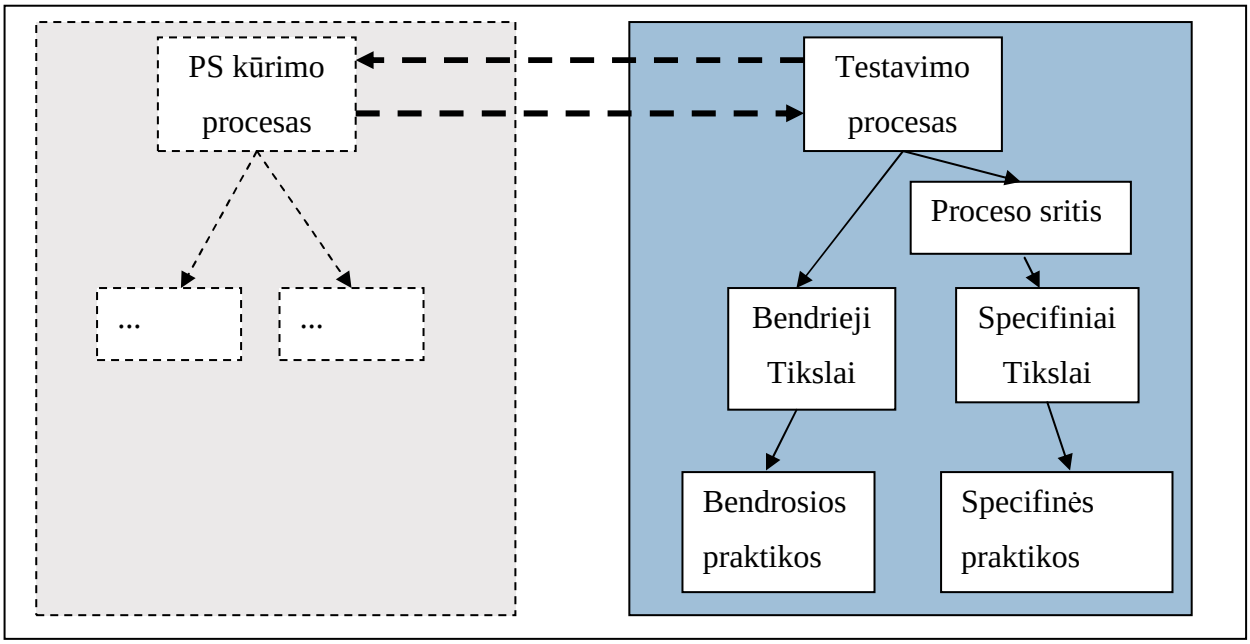
2 lentelė. Modelyje taikyti modifikuoti BPMN elementai.

Elementas	**Aprašymas**	**Notacija**
Dvigubas proceso sričių konteineris	Sugrupuoti vienodos paskirties proceso sritis pagal subprocesą. Procesas bei subprocesai turi savo Id bei pavadinimą.	
Inicializacijos arba pabaigos proceso sritis	Vaizduojamas elementas žymi proceso pradžią arba pabaigą.	
Proceso sritis	Elementu parodoma atitinkama proceso sritis.	
Srautas	Ištisinė rodyklė nurodo duomenų srauto ryšį tarp skirtingų vieno proceso sričių.	
Asociacija	Asociacijos punktyrinė rodyklė nurodo sąsajas tarp dviejų subprocesų.	

2.2.2. Testavimo proceso metamodelis

Verslo proceso modeliavimo notacija nenurodo, tačiau ir neuždraudžia naudoti metamodelį. Pastebima, kad autoriai tiems patiems procesams analizuoti naudoja vis kitokias sąvokas. Natūralu, jog problema ypač išryškėja nagrinėjant skirtingų kalbų literatūros šaltinius. Iškilusiai problemai išspręsti testavimo proceso modelyje vartojamas sąvokas paaiškina 8 pav. pateikiamas proceso metamodelis.

Pagrindinis kuriamo modelio procesas yra testavimo procesas. Metamodelis parodo detalią modelio sandarą. Modelyje procesas vaizduojamas iki proceso sričių lygio. Metamodelis nurodo, kad proceso sritys susideda iš tikslų ir praktikų. Toks detalumo lygis pasirinktas dėl modelio vaizdumo tam, kad būtų įmanoma vienoje schemoje pakankamai išsamiai pateikti sąryšius tarp procesų sričių, kurie yra svarbiausia modelio dalis įgyvendinanti darbo tikslą. Pateikiami jų aprašymai, o 2.2.3. darbo dalyje pateikiamas grafinis pavaizdavimas.



7 pav. Testavimo proceso metamodelis.

2.2.3. Siūlomas testavimo proceso modelis

Siūlomas testavimo proceso modelis turi būti iteratyvus, kad išspręstų anksčiau iškeltą problemą, jog testavimas privalo būti vykdomas lygiagrečiai su PS kūrimo procesu ir, kita vertus, testavimo neįmanoma vykdyti, kai komponentas dar nesukurtas. Tai pasiekiama suvokiant testavimą ne vien siaurąja prasme, kaip testų vykdymą, o įtraukiant planavimą, projektavimą ir kitas proceso sritis.

Iteratyviu testavimo procesu laikomas toks procesas, kurio kiekviena iteratyvaus PS kūrimo modelio veikla atitinka testavimo proceso veiklą.

Testavimo proceso modelį iš esmės sudaro du pagrindiniai subprocesai, kurie priklauso vienas nuo kito. Dėl šios priežasties pagrindinis procesas ir pavadintas „Testavimo procesas ir jo kontekstas“.

Darbe išsamiau pasiūlomos testavimo proceso sritys, kurių iš viso yra vienuolika. Modelio dalys pasirinktos atsižvelgus į bendrai siūlomas praktikas, elgsena grįstą testavimo proceso modelį, „V-modelį“. Modelyje akcentuojami sąryšiai tarp veiklų. Testavimo procesas turi bendrinių

praktikų, kurios naudojamos visame procese. Kiekviena proceso sritis turi savo specifinių praktikų, kurias, kaip minėta, sieja atitinkami tikslai. Taigi siūlomas modelis remiasi TMM modeliu ir BPMN notacija.

Toliau bendrai apibrėžiamos proceso sritys, kurios vaizduojamos modelyje, tikslai ir praktikos bus pasiūlyti trečiojoje darbo dalyje.

Id: P1.) PS kūrimo procesas

PS kūrimo proceso sritys:

(Id: P1.1) Reikalavimų analizė

(Id: P1.2) Sistemos projektavimas

(Id: P1.3) Architektūros projektavimas

(Id: P1.4) Modulių projektavimas

PS kūrimo proceso sritys detaliau nepateikiamos, nes darbo tikslas yra testavimo proceso kūrimas.

(Id: T2.) Testavimo procesas

Šiame lygmenyje bus apibrėžtos bendros proceso praktikos, kurios turėtų būti taikomos visoms testavimo proceso sritims.

Testavimo proceso modelio sritys:

(Id: T2.1) Testavimo planavimas

Testavimo planavimo proceso sritis inicializuoja visą testavimo procesą, nes yra vykdoma anksčiausiai. Tokią išvadą galima daryti išanalizavus bendrąsias visų testavimo procesų veiklas (planavimas ir kontrolė, testavimo sąlygų pasirinkimas ir kt.).

(***Id: T2.2) Priėmimo testavimo projektavimas***

Veikla vykdoma iškart po testavimo planavimo ir jau vykdoma kaip iteratyvaus testavimo proceso ciklo dalis. Priėmimo testavimo projektavimo veikla sujungta asociatyviu sąryšiu su PS kūrimo reikalavimų analize ir su sistemos projektavimo veikla.

(***Id: T2.3) Sistemos testavimo projektavimas***

Testavimo sritis vykdoma iškart po sistemos projektavimo. Sistemos testavimo projektavimas, ką keisti projektuojant sistemos architektūrą, jei atliekamas nebe pirmas iteracijos žingsnis.

(***Id: T2.4) Integracijos testavimo projektavimas***

Integracijos testavimo projektavimas susietas asociatyviu sąryšiu su sistemos architektūros projektavimu. T.2.4 pateikia siūlymus, ką keisti, kai projektuojami moduliai.

(***Id: T2.5) Modulių testavimo projektavimas***

Modulių testavimo projektavimas keičia programavimo veiklos rezultatus, nes susietas srauto sąryšiu. Jis vykdomas atsižvelgus į modulių projektavimą.

(Id: T2.6) Programavimas.

Programavimo sritis yra sritis tarp PS kūrimo veiklų ir testavimo. Vis dėlto, siūlomame modelyje programavimas pateikiamas ir kaip testavimo proceso dalis todėl, kad programuotojai taip pat vykdo testus ir taiko įvairius testavimo metodus.

(Id: T2.7) Priėmimo testavimas

Priėmimo testavimo proceso sritis yra vykdomoji. Vykdomosiomis veiklomis vadinamos visos ne projektavimo veiklos, o veiklos, kurių rezultatas – aptikti defektai. Vykdant T.2.7. veiklą atliekamas reikalavimų verifikavimas. Dėl šios priežasties proceso sritis jungiama asociatyviu sąryšiu su PS kūrimo proceso reikalavimų analizės veikla.

(Id: T2.8) Sistemos testavimas

Nagrinėjama proceso sritis taip pat yra vykdomoji. Sistemos testavimas vykdomas remiantis sistemos testavimo projektu.

(Id: T2.9) Integracijos testavimas

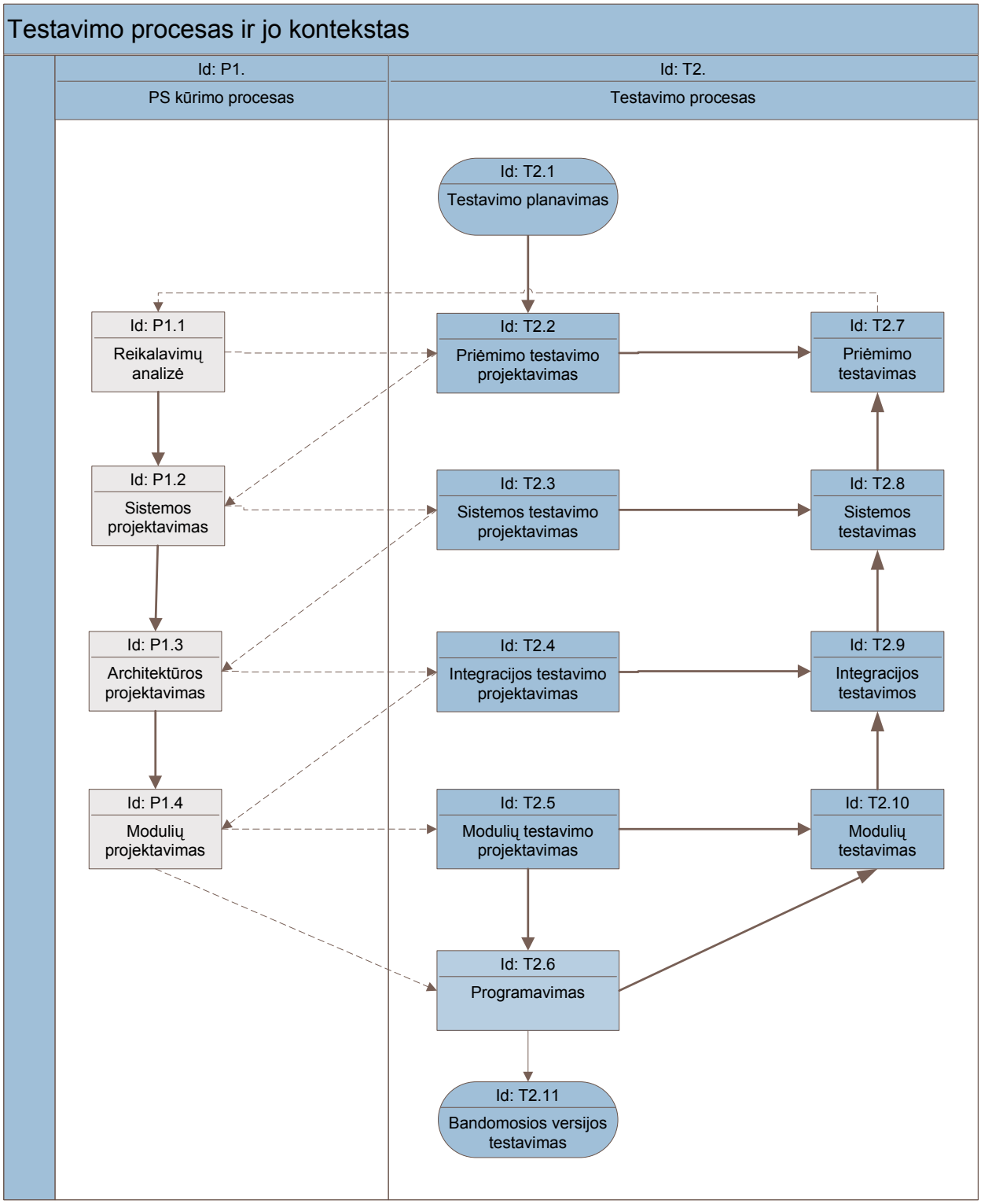
Integracijos testavimas vykdomas po modulių testavimo ir integracijos testavimo projektavimo. Proceso sritis susieta srauto sąryšiu su sistemos testavimu.

(Id: T2.10) Modulių testavimas

Siūloma veikla vykdoma iškart po programavimo. Čia išryškėja pirmieji PS defektai.

(Id: T2.11) Bandomosios versijos testavimas

Bandomosios versijos testavimas yra testavimo proceso pabaigą žyminti veikla, kuri nurodo, kad ji vykdoma vieną kartą projekte Čia realizuotos praktikos, kurios gali būti naudingos kitam projektui.



8 pav. Siūlomas programų sistemų testavimo modelis.

Naujo modelio privalumai:

- Testavimo procesas iteratyvus („V-modelio“ pakeitimas);
- Apibrėžiamas regresinis testavimas (vykdomas pakartotinis testavimas);
- Sujungiamos geriausias tinkamiausių „V-modelio“ ir iteratyvaus modelio savybės;
- Testavimo veiklos vykdomas lygiagrečiai, visais PS kūrimo etapais;

- Kiekviena proceso sritis gali būti atnaujinama įtrakus naujas praktikas (praktikos pridedamos trečiojoje darbo dalyje).

Šiame modelyje PS kūrimo subprocese išdėstytos pagrindinės veiklos, kurios, kaip ir reikalauja modernaus testavimo praktikos, visos yra susietos su testavimo subproceso veiklomis.

Modelis yra iteratyvus – kiekviena iteratyvaus PS kūrimo modelio dalis turi atitikmenį testavimo procese.

Modelyje realizuojamas validavimas ir verifikavimas. Validumas patikrinamas priėmimo testavimo veikloje, o verifikavimą įgyvendina asociatyvūs sąryšiai.

Geriausia šio modelio savybe būtų galima įvardinti lengvą plečiamumą, į proceso sritis įtraukiant reikalingas praktikas, kurios sugrupuotos į tikslus ir priskirtos atitinkamai modelio proceso sričiai (pateikiamos darbo prieduose).

Darbe nagrinėjami programų sistemų ir testavimo procesai. PS kūrimo procesas reikalingas parodyti testavimo proceso kontekstą, jo vietą organizacijoje. Vykdomas praktikas sieja vienas tikslas. Taigi tikslai suvokiami kaip tam tikros praktikų grupės. Procesas skaidomas į proceso sritis, kurių sąryšiai parodyti 8 pav.

Praktinė testavimo proceso modelio paskirtis yra apibrėžti įmonės veiklas tam, kad organizacijoje vykdomos veiklos būtų brandesnės. Formalizavus procesą, įmanoma jį valdyti ir kontroliuoti. Taigi modelis padėtų organizacijai įvykdyti projektus laiku ir kokybiškai.

Modelis gali būti naudojamas organizacijų, kurios kuria didelės apimties programinę įrangą. Mažos apimties programinei įrangai modelis nesiūlytinas naudoti, nes tokiuose projektuose ne visada būtinas veiksmų formalizavimas. Taikančiai modelį organizacijai būtų paprasčiau vykdyti kokybišką testavimo procesą.

Kitoje darbo dalyje modelis detalizuojamas ir optimizuojamas. Atsižvelgiant į TMI ir TPI aprašomos bendrosios proceso ir specifinės proceso sričių praktikos. Taigi modelis detalizuojamas iki praktikų lygio. Toliau siekiama detaliau aprašyti kiekvieną proceso sritį.

3. Testavimo proceso modelio tyrimas

3.1. Tyrimo apimtis

TMMi modelis sujungia skirtingas TMM modifikacijas, todėl yra bendresnis. Detaliau galima panagrinėti, kaip 8 paveikslėlyje pateikiamas naujas programų sistemų testavimo modelis padengia TMMi brandos lygių proceso sritis[10], kurios apibrėžtos [Vee08].

Reikėtų atkreipti dėmesį, kad TMMi neturi specifinių proceso sričių testavimo ir testavimo automatizavimo įrankiams vertinti. TMMi į įrankius žiūrima kaip į palaikomuosius išteklius (praktikas). Pavyzdžiui, testavimo įrankių taikymas yra testavimo projektavimo ir vykdymo proceso srities praktika, 2-ame brandos lygyje [Vee08]. Pastaroji savybė įrodo, jog TMMi yra per daug bendras ir papildomų praktikų naudojimas yra rekomenduotinas, todėl toliau atsivelgiama ir į TPI modelį.

3.2. Testavimo proceso modelio tyrimas pagal TMM ir TPI

Testavimo brandos modelio integracijos modelis nurodo gaires, kaip lengvai patobulinti testavimo proceso modelį, kad jis tenkintų norimą testavimo proceso brandos lygį pagal TMM. Kiekviena brandos lygio proceso sritis turi savo tikslus bei praktikas. Svarbu tai, kad modelyje būtų sudarytos galimybės jį plėsti ir modelis neribotų reikiamų praktikų diegimo testavimo procese.

Kaip matoma iš testavimo brandos modelio, tikslai apibūdina tam tikras praktikų grupes. Abstraktumo dėlei darbe naudojamos praktikų grupės – testavimo brandos modelio tikslai. Ankstesnėse darbo dalyse apibrėžėme TMM brandos lygius, dabar brandos lygiai nagrinėjami detaliau iki proceso sričių tikslų. Tikslų apibrėžimai aprašyti remiantis E. Van Veenendaal *Test Maturity Model Integration*. Papildomi tikslai įterpti ir iš TPI modelio.

Brandos tikslai prasideda nuo antro lygio, nes pirmas lygis yra neapibrėžtas ir chaotiškas.

(2.1.) Sukurti testavimo ir klaidų taisymo tikslus

Šio tikslo paskirtis yra aiškiai atskirti testavimą nuo klaidų derinimo. Tikslai, užduotys, veiklos ir įrankiai privalo būti identifikuoti. Taip pat turi būti sukuriama valdymo strategija tam, kad pritaikytų ir įtvirtintų abu procesus.

[10] Angl. *Process area*

2.1.1. Procedūros ir testavimo tikslai sukuriami, patvirtinami, vykdomi ir reguliariai peržiūrimi.

2.1.2. Naudojama pagrindinė defektų klasifikacijos schema, saugykla.

2.1.3. Vykdomi pagrindiniai testavimo ir klaidų taisymo matavimai.

Iš tikslo aprašymo matome, kad jis yra bendrinis, nes privalo būti įdiegtas į visas proceso sritis. Testavimo atskyrimas nuo klaidų derinimo nurodo proceso lygio atskyrimą: PS kūrimo proceso dalis (derinimas, kuris susijęs su kodo keitimu) privalo būti atskirtas nuo testavimo. Taigi tikslas gali būti integruojamas kaip bendrinis testavimo proceso tikslas.

(2.2.) Testavimo planavimo proceso inicijavimas

Šio tikslo paskirtis yra įtvirtinti testavimo planavimą organizaciniame lygyje. Testavimo planavimas įtraukia statinio testavimo tikslus, rizikos analizę, strategijų išgryninimą ir testavimo specifikacijos bei testavimo atvejų kūrimą. Testavimo plane privalomas išteklių, kainų ir atsakomybių aptarimas modulių, integracijos ir priėmimo testavimo lygiuose.

2.2.1. Testavimui planuoti skirtos procedūros, tikslai ir strategija yra sukurti, patvirtinti ir reguliariai naudojami bei peržiūrimi.

2.2.2. Kiekvienam testavimo lygiui sukuriami testavimo šablonai.

2.2.3. Paruošiamas kursas apie šablonų naudojimą.

2.2.4. Naudotojo reikalavimai įkeliami kaip įvestis į testavimo planą.

2.2.5. Taikomi pagrindiniai planavimo ir matavimo įrankiai.

Jau vien iš pavadinimo aišku, jog šis tikslas turėtų būti realizuotas testavimo proceso planavimo srityje.

(2.3.) Oficialiai įtvirtinti pagrindines testavimo technikas ir metodus

Tikslo paskirtis yra pagerinti testavimo proceso brandą taikant pagrindines testavimo technikas ir metodus. Kaip ir kada šios pagrindinės technikos, metodai ir pagrindiniai įrankiai jiems palaikyti taikomi, turėtų būti aiškiai aprašyta testavimo strategijoje ir planuose. Skirtingos pagrindinės testavimo technikos ir metodai, kurie yra dažnai naudojami, yra „juodos ir baltos dėžės“ testavimo strategijos, reikalavimų validacijų matricos panaudojimas ir testavimo vykdymo suskaidymas į subfazes, tokias kaip integracijos, sisteminę ir priėmimo testavimą.

2.3.1. Apibrėžiama pagrindinių technikų ir metodų aibė.

2.3.2. Paruošta techninio mokymo medžiaga bei įrankiai pagrindinėms technikoms ir metodams naudoti.

2.3.3. Testavimas planuojamas ir įdiegiamas skirtinguose testavimo lygmenyse.

2.3.4. Pagrindinės „baltos ir juodos dėžės“ strategijos, technikos ir metodai naudojami projektuojant testavimo atvejus. Bendravimas tarp programuotojų ir testuotojų testavimo atžvilgiu yra aukštesnio lygio.

Tikslas pakankamai bendrai apibrėžia testavimo technikų ir metodų panaudojimą, taigi siūlytinas būti bendrine testavimo proceso praktika.

(2.4.) Testavimo infrastruktūra planuojama ir gerinama (TPI)

Šio tikslo paskirtis pagerinti testavimo proceso kokybę įtvirtinant gerai valdomą testavimo infrastruktūrą. Pagrindiniai šios infrastruktūros komponentai yra: testavimo aplinka, testavimo įrankiai bei biuro aplinka.

2.4.1. Testavimo aplinka yra valdoma ir kontroliuojama.

2.4.2. Naudojami testavimo įrankiai.

2.4.3. Kontroliuojama biuro aplinka ir laikas.

Testavimo infrastruktūra susijusi su visomis testavimo veiklomis. Nagrinėjamas tikslas siūlytinas naudoti kaip bendrinė testavimo proceso modelio praktika, nes sujungia visas proceso sritis.

(3.1.) Įtvirtinti programinės įrangos testavimo organizavimą

Tikslo paskirtis yra identifikuoti ir organizuoti kvalifikuotų žmonių komandą, kuri būtų atsakinga už testavimą. Testavimo komanda turėtų būti atsakinga už testavimo planavimą, vykdymą ir įrašymą, su testavimu susijusius standartus, testavimo metrikas, testų duomenų bazę, testų daugkartinį panaudojamumą, testavimo sekimą ir vertinimą. Komanda taip pat atsakinga už defektų saugyklos priežiūrą.

3.1.1. Testavimo komanda atsakinga už testavimo metodų apibrėžimą ir priežiūrą organizacijoje (TPI).

3.1.2.Aukšto lygio testai atliekami paskirtų testavimo grupių, susidedančių iš motyvuotų narių (TPI).

3.1.3. Jeigu reikia techninės ekspertizės, galimi mokymai.

Testavimo organizavimas galėtų būti realizuotas testavimo planavimo srityje, nes yra pakankamai specifinis.

(3.2.) Įtvirtinti techninio mokymo programą

Pagrindinis tikslo siekimas iš testavimo komandos yra apsidrausti, kad tik kvalifikuoti asmenys atliktų užduotis. Formali testavimo programa įgyvendinama mokymo strategijoje. Joje aprašomi mokymo tikslai ir vystomi mokymo planai bei parengiama medžiaga. Mokymo klasės paruošiamos visiems komandos nariams. Mokymo programos paskirtis yra išmokti testuotojams naujausių testavimo technikų, metodų ir įrankių. Programa taip pat paruošia testuotojus planuoti testavimo procesą, integracijos į programų kūrimo gyvavimo ciklą, reikalingą peržiūroms, bei identifikuoti ir prioritetizuoti su testavimu susijusią riziką. Aukštesniuose TMM lygiuose mokymo programa moko testuotojus proceso kontrolės, testavimo metrikų, statistinio testavimo, testavimo proceso veiksmų planavimo, patikimumo modelio ir kitų aukšto lygio testavimo veiklų.

3.2.1. Sukuriamas organizacinio mokymo strateginis dokumentas.

3.2.2. Sukuriama vidinė mokymo grupė, kuri ruošia strategiją ir medžiagą.

Nagrinėjamas tikslas rekomenduotinas realizuoti testavimo planavimo veikloje, nes yra vieną kartą specifiškai taikomas ir ne bendrinis.

(3.3.) Integruoti testavimą į programų sistemų gyvavimo ciklą

Šio tikslo paskirtis yra pagerinti testavimo veiklų našumą lygiagrečiai su kitomis PS kūrimo veiklomis pradedant nuo pat gyvavimo ciklo pradžios. Brandi organizacija privalo neuždelsti testavimo veiklų iki kodo rašymo pabaigos. Gerų praktikų pavyzdžiai yra testavimų veiklų pradžia nuo pat reikalavimų rinkimo etapo. Dažniausiai naudojamos „V-modelio“ modifikacijos.

3.3.1. Sukuriamos procedūros, tikslai ir strategijos testavimui integruoti, jos patvirtinamos, vykdomos ir reguliariai peržiūrimos.

3.3.2. Testavimo veiklos integruojamos į programinės įrangos gyvavimo ciklą naudojant pasirinktą gyvavimo ciklo modelį. Peržiūrima projekto ir testavimo strategija.

3.3.3. Paskiriami ištekliai ir vykdomi mokymai tam, kad testavimas būtų integruotas į PS kūrimo gyvavimo ciklą.

Šis TMM tikslas naujai sukurtam modeliui aktualus tik iš dalies, nes naujai sukurtas modelis pakankamai vaizdžiai parodo testavimo proceso integraciją į PS kūrimo gyvavimo ciklą. Aktualia praktika galime laikyti 3.3.3.

(3.4.) Testavimo proceso kontrolė ir monitoringas

Tikslo paskirtis yra kontroliuoti ir stebėti testavimo procesą, kad nukrypimai būtų pastebėti kaip galima anksčiau. Įvykdžius šį tikslą – didesnė tikimybė, jog projektas bus atliktas laiku ir neviršijant biudžeto.

3.4.1. Sukuriamos ir reguliariai peržvelgiamos procedūros, tikslai, strategijos, matavimai tam, kad būtų galima kontroliuoti ir stebėti testavimą.

3.4.2. Atliktų su testavimu susijusių matavimų rezultatai panaudojami kontrolei ir stebėjimui kituose projektuose. Testavimo statusų ataskaitos pateikiamos reguliariai.

3.4.3. Mokymo įrankiai bei kiti ištekliai turi būti prieinami tam, kad būtų galima kontroliuoti ir stebėti testavimą.

Nagrinėjami tikslai ir subtikslai, kuriuos paprastumo dėlei galime laikyti praktikomis yra bendriniai ir siūlytini taikyti testavimo proceso bendrųjų praktikų aspektu, nes jos skirtos visoms proceso sritims todėl, kad kontrolė ir monitoringas turi būti atliekamas vykdant visas veiklas, nei vienos nepraleidžiant.

(4.1.) Sukurti organizacijos lygio peržiūros programą

Peržiūra yra testavimo technika, kuri gali būti naudojama siekiant pašalinti defektus iš programinės įrangos. Pasiekus šį brandos lygio tikslą organizacija gali identifikuoti, surūšiuoti ir pašalinti defektus efektyviai ir anksti. Į peržiūras taip pat įtraukiami reikalavimų dokumentai, testavimų planai, testavimo atvejų specifikacijos ir kiti dokumentai.

4.1.1. Sukuriamos procedūros, tikslai, strategijos ir matavimai visų programinės įrangos produktų peržiūrai, jie patvirtinami, vykdomi ir reguliariai peržvelgiami.

4.1.2. Personalas apmokomas, kad mokėtų tvarkingai peržiūrėti strategijas ir praktikas bei rinktų peržiūrų duomenis.

4.1.3. Programinė įranga peržiūrima kiekviename projekte, matavimai renkami bei jais remiantis gerinama produkto ir proceso kokybė.

Nagrinėjamas praktikas siūloma taikyti visame testavimo procese. Kiekviena proceso sritis privalo būti peržiūrima, tam kad būtų įmanoma ją tobulinti

(4.2.) Įtvirtinti testavimo matavimo programą.

Testavimo matavimo programos tikslas yra identifikuoti, surinkti, analizuoti ir pritaikyti matavimus organizacijoje apibrėžiant testavimo procesą tam, kad būtų galima įvertinti efekty-

vumą ir kokybę. Pavyzdžiais galėtų būti laikomi su testavimu susiję matavimai: kaina, produktyvumas, įvykdytų testavimo atvejų ir aptiktų defektų skaičius.

4.2.1. Sukuriami, patvirtinami, vykdomi ir reguliariai peržiūrimi: procedūros, tikslai, strategijos bei matavimai.

4.2.2. Sukuriama testavimo matavimo programa kartu su matavimo ataskaitų sistema. Rezultatai plačiai naudojami organizacijoje.

Testavimo matavimo programos įtvirtinimo realizavimas rekomenduotinas testavimo proceso modelio priėmimo testavimo srityje.

(4.3.) Programinės įrangos kokybės vertinimas

Tikslo paskirtis yra susieti programinės įrangos kokybę su testavimo procesu, apibrėžti ir naudoti pamatuojamus programinės įrangos kokybės vertinimo atributus bei apibrėžti kokybės tikslus, vertinti PĮ produktus.

Pavyzdiniai kokybės atributai galėtų būti tikslumas, našumas, integralumas, panaudojamumas, prižiūrimumas, lankstumas, testuojamumas, perkeliamumas ir t.t.

4.3.1. Sukuriamos, patvirtinamos ir naudojamos procedūros, tikslai, strategijos bei matavimai programinės įrangos kokybei vertinti.

4.3.2. Vykstantys apmokymai, naudojami įrankiai bei kiti ištekliai užtikrina programinės įrangos kokybės vertinimo prieinamumą.

4.3.3. Sukuriami tikslai kiekvienam projektui pagal strategiją. Testavimo procesas yra struktūrizuotas tam, kad būtų galima užtikrinti kokybės tikslų pasiekimą. Sveikintinas ir kliento indėlis į kokybės tikslų sukūrimą.

Siūloma testavimo proceso modelio sritis, į kurią būtų galima integruoti išdėstytas praktikas yra „Priėmimo testavimas“.

(5.1.) Defektų prevencija

Defektų prevencijos paskirtis yra paskatinti organizaciją formaliai klasifikuoti, kaupti ir analizuoti defektus. Organizacija taip pat skatinama naudoti kombinuotą defektų priežasčių analizę ir veiksmų planavimą tam, kad procesas būtų tobulinamas ir defektai pašalinami iš būsimų produktų.

5.1.1. Sukuriamos procedūros, strategijos ir matavimai defektų prevencijai, sukurtos procedūros patvirtinamos, vykdomos ir reguliariai peržiūrimos.

5.1.2. Mokymai, įrankiai ir kiti ištekliai yra prieinami, tam kad defektų prevencija būtų remiama.

5.1.3. Sukuriamos defektų prevencijos komandos, identifikuoti defektai pašalinami iš kiekvienos gyvavimo ciklo fazės, atliekat priežasčių analizę ir sukuriant planus apsaugoti nuo defekto pasikartojimo.

Defektų prevencijos praktika siūloma įdiegti į testavimo proceso projektavimo veiklas, nes jomis galima užtikrinti kokybišką tolimesnio proceso vykdymą iš anksto.

(5.2.) Kokybės kontrolė

Kokybės kontrolės tikslas yra sukurti suprantamas kokybės kontrolės procedūras ir praktikas, kurios padeda sukurti aukštos kokybės programinę įrangą, kuri visiškai atitinka kliento reikalavimus. Pasiekus šį brandos tikslą leidžia organizacijai vykdyti pažangius matavimus, kurie pagerina testavimo proceso našumą. Kokybės kontrolės veiklos susijusios su automatizuotų įrankių naudojimu. Naudojami statistiniai metodai leidžia padidinti tikimybę, kad PĮ neturi defektų. Kiekybiniai kriterijai naudojami nustatyti, kada pakanka testuoti ir reikia testavimo procesą nutraukti.

5.2.1. Sukuriamos, patvirtinamos ir vykdomos procedūros, tikslai ir strategijos, siekiant užtikrinti kokybės kontrolę.

5.2.2. Testavimo ir PĮ kokybės užtikrinimo komandos nustato atributus ir įtvirtina tikslus produkto kokybei užtikrinti. Testavimo įrankiai ir technikos naudojami nustatyti, ar tenkinami kokybės tikslai pasiekiami. Kiekybinis kriterijus naudojamas nustatyti testavimo pabaigą.

5.2.3. Testavimo komanda paruošiama naudoti statistinius metodus ir kitas su kokybe susijusias veiklas, pavyzdžiui, panaudojamumo testavimą.

Kokybės kontrolės tikslas turėtų būti realizuotas vykdomosiose testavimo proceso srityse

(5.3.) Testavimo proceso optimizavimas

Tikslo paskirtis yra skatinti nuolatinį testavimo proceso gerinimą ir daugkartinį panaudojamumą. Organizacija drąsinama nustatyti testavimo praktikas, kurios turi būti pagerintos ir įdiegti bei sekti pakeitimus. Negana to, organizacija skatinama taikyti proceso kontrolės veiklas, nuolat vertinti naujus įrankius bei technologijas ir jas pritaikyti. Galiausiai organizacijai rekomenduojama nustatyti aukštos kokybės testavimo subprocesus, išsaugoti juos „proceso bibliotekoje“ ir pritaikyti kituose projektuose.

5.3.1. Organizacijoje suformuojama testavimo proceso gerinimo grupė, atsakinga už: daugkartinį panaudojamumą, kontrolę, vertinimą, gerinimą.

5.3.4. Aukštos kokybės testavimo proceso komponentai yra pripažįstami ir nuolat naudojami visoje organizacijoje.

Testavimo proceso optimizavimą siūloma realizuoti kaip bendrinę praktiką taikytiną visoms proceso sritims.

1 darbo priede pateikiama TMM ir TPI tikslų[11] ir praktikų integracija į naują testavimo proceso modelį. Visos praktikos, kurios aprašomos testavimo proceso lygyje yra bendrinės, kitos – specifinės.

3.3. Naujo modelio sąsajos su brandos ir gerinimo modeliais

Kaip matoma 3 lentelėje, naujas testavimo proceso modelis gali būti lengvai plečiamas įtraukiant trūkstamas testavimo proceso brandos modelio siūlomas praktikas į esamas testavimo modelio sritis.

Tokia modelio savybė įgyvendinama, todėl, kad testavimo proceso brandos modelio schema yra labai panaši į naujai sukurto proceso metamodelį. Ankstesnėje darbo dalyje apibrėžtą brandos tikslų panaudojimą naujai sukurtame testavimo proceso modelyje galime apibendrintai išdėstyti lentelėje.

3 lentelė. Testavimo modelio proceso sričių sąsaja su brandos modeliu.

Brandos lygis	**Brandos modelio tikslai pagal proceso sritis**	**Modelio proceso srities Id**	**Pastabos**
1 lygis Pradinis	-	T.2	Testavimo procesas chaotiškas ir jis neturi privalomų veiklų. Taigi tinka bet kuris testavimo modelis.
2 lygis Apibrėžimo	2.1. Sukurti testavimo ir klaidų taisymo tikslus.	T.2.1.	TMM veikla įtraukiama į testavimo modelio planavimo proceso sritį, nes ši veikla yra specifinė ir taikoma tik vieną kartą.

[11] Toliau vadinama vien tik praktikomis.

	2.2. Testavimo planavimo proceso inicijavimas.	T.2.1.	Testavimui planuoti skirta atskira testavimo planavimo proceso sritis, kuri atliekama vieną kartą. Todėl šios brandos modelio procesų sričių praktikų grupės priskiriamos testavimo proceso planavimo sričiai.
	2.3. Oficialiai įtvirtinti pagrindines testavimo technikas ir metodus.	T.2	Kiekviena testavimo veikla privalo vykdyti pagrindines testavimo technikų ir metodų praktikas. Šią TMM proceso srities praktikų grupę galime laikyti bendrine testavimo proceso modelio praktika.
	2.4. Testavimo infrastruktūra planuojama ir gerinama (TPI).	T.2.7. T.2.8. T.2.9. T.2.10.	Testavimo aplinka priklauso nuo testavimo lygio (priėmimo, sistemos, integracijos, modulių) ir realizuoja rekomenduojamas specifines praktikas.
3 lygis **Integravimo**	3.1. Įtvirtinti programinės įrangos testavimo organizavimą.	T.2.1.	Testavimo organizavimas vykdomas per testavimo planavimą. Taigi yra specifinis ir nekartojamas daug kartų, o atliekamas tik kartą.
	3.2. Įtvirtinti techninio mokymo programą.	*T.2.1.*	Proceso sritį galima įtraukti į testavimo planavimą ir vykdyti papildomą nepriklausomą veiklą, nes ši proceso sritis nepriklauso nuo konkretaus projekto.
	3.3. Integruoti testavimą į programų sistemų gyvavimo ciklą.	T.2	Visas modelis ir aprašo testavimo gyvavimo ciklą. Taigi ši proceso sritis papildomai nenagrinėjama. TMM modelyje ši proceso sritis aptariama labai siaurai.
	3.4. Testavimo proceso kontrolė ir monitoringas.	T.2	Testavimo modelyje esantys ciklai įgyvendina peržiūrų vykdymą.
4 lygis **Valdymo**	4.1. Sukurti organizacijos lygio peržiūros programą.	T.2.	Testavimo modelyje esantys ciklai įgyvendina peržiūras taip pat kaip 3 brandos lygio peržiūros proceso sritis, tik detaliau. Ši brandos modelio proceso sritis yra bendrinis testavimo proceso modelio praktikų rinkinys.
	4.2. Įtvirtinti testavimo matavimo programą.	T.2.7.	Testavimo matavimas vykdomas atlikus priėmimo testus.

	4.3. Vertinti programinės įrangos kokybę.	T.2.7.	Kokybė vertinama baigiantis vienam iš projekto etapų.
5 lygis Optimizavimo	5.1. Defektų prevencija.	T.2.2. T.2.3. T.2.4. T.2.5.	Defektų prevencija gali būti realizuojama skirtingose testavimo projektavimo lygiuose, nes būtent jie sieja kūrimo proceso veiklas su testavimo proceso veiklomis. Būtent projektavimo metu apsisaugoma nuo pakartotinio defektų pasirodymo projekte.
	5.2. Kokybės kontrolė.	T.2.7 T.2.8. T.2.9. T.2.10.	Testavimo kokybė kontroliuojama testavimo vykdymo metu.
	5.3. Testavimo proceso optimizavimas.	T.2.7 T.2.8. T.2.9. T.2.10.	Testavimo procesas optimizuojamas testavimo projektavimo, o ne vykdymo metu. Optimizavimas suvokiamas kaip geros defektų prevencijos rezultatas.

Rezultatai ir išvados

Darbe struktūrizuojama programų sistemų kokybės tema. Aptarti programinės įrangos defektų šalinimo mechanizmai, testavimo metodai ir priemonės, jų taikymo atvejai, nauda. Rasti bendri aspektai, siejantys visus testavimo proceso modelius: standartai, metodai, priemonės, brandos ir gerinimo modeliai. Detaliau išnagrinėtos bendros testavimo praktikos, rastos jų teigiamos ir neigiamos savybės, rekomenduojami taikymo atvejai. Pagrindu testavimo modeliui kurti pasirinktas brandos modelis, kuris tinkamas vertinti sukurto testavimo proceso brandai. Pastarasis modelis tinka bet kokiam testavimo procesui tirti. Kita vertus, TMM, testavimo proceso brandos modelis, pernelyg bendras ir detaliau nenurodantis testavimo proceso vietos programų sistemų kūrimo proceso kontekste. Todėl, išsiaiškinus darbo problematiką, ieškota sąsajų tarp programų sistemų kūrimo ir testavimo procesų. Kitaip, nutarta ieškoti taisyklių, kaip sukurti darbo tikslą atitinkantį modelį.

Išnagrinėjus populiariausius programų sistemų kūrimo modelius, nustatytas tinkamiausias modelis šiuolaikinei programinei įrangai kurti. Aptartos kiekvieno modelio savybės, jų tinkamumas. Taip pat rasti sąryšiai siejantys visus modelius, geriausios taikytinos praktikos. Apžvelgtas ir elgsena grįsto testavimo modelis, rasti jo privalumai ir trūkumai. Svarbiausia, pirmojoje ir antrojoje darbo dalyje išnagrinėta, kokiu principu naujas modelis turėtų būti kuriamas ir šis principas darbe pritaikytas. Išsiaiškinta, jog bet kokio testavimo proceso modelio testavimo ciklas išlieka toks pats (vykdoma reikalavimų analizė, testavimo planavimas, testų kūrimas, vykdymas, ataskaitų rengimas ir klaidų pakartotinis testavimas). Modelis kuriamas remiantis programų sistemų kūrimo modeliu, o visų modelių kokybės kriterijus apibrėžia TMM, TPI bei kiti brandos ir gerinimo modeliai. Brandos modelis nurodo, jog testavimo procesas turi būti integruotas į PS kūrimo gyvavimo ciklą. Atsižvelgus į literatūroje nurodytas testavimo proceso sąsajas su PS kūrimo modeliais, įvertinus modelio plečiamumą bei galimybę lengvai pritaikyti modernaus testavimo praktikas, pasiūlytas naujas testavimo proceso modelis naudojant modifikuotą verslo proceso notaciją. Taigi antros darbo dalies rezultatas – pasiūlytas naujas programinės įrangos testavimo modelis.

Galiausiai modelis detalizuotas naudojant testavimo proceso brandos modelį. Išsiaiškinta, jog modelis gali būti lengvai atnaujinamas, nes naujai sukurto modelio metamodelis (7 pav.) neprieštarauja TMM schemai (5 pav.), į atitinkamas jo proceso sritis įtraukus testavimo brandos modelio siūlomas praktikas. Toks sprendimo būdas natūraliai pritaikytas tyrinėjant testavimo proceso brandos modelį. Galiausiai modelis detalizuotas pagal TMM (Testavimo brandos) bei TPI (Testavimo proceso gerinimo) modelius. Pastebėta, jog testavimo brandos modelis ir testa-

vimo proceso gerinimo modeliai ganėtinai panašūs. Šis panašumas buvo pasitelktas: bendros praktikos panaudotos naujai sukurtame testavimo proceso modelyje.

Iškyla klausimas, kuo naujas modelis skiriasi nuo nagrinėtųjų „V-modelio“ ir TMM, TPI. Visų pirma „V-modelis“ turi trūkumų, kurie pateikti 2.1.5 darbo dalyje ir remiantis nagrinėtais literatūros šaltiniais [FM06] yra pasenęs. Nagrinėjant brandos modelį, suprasta, jog jis yra pernelyg bendras. Ypač išryškėjo jo sąsajos su PS kūrimo procesu trūkumas. Pastebėtina, jog tik 3 brandos lygyje užsimenama (ne analizuojama ir siūloma kaip susieti), jog reikia integruoti testavimą į PS gyvavimo ciklą. Darbe pasiūlytas modelis akcentuoja sąsają su PS kūrimo procesu. TMM yra būtent brandos modelis, skirtas vertinti, gerinti procesą (TPI), bet jis nurodo „ką“, o ne „kaip“ reikia atlikti. TPI yra gerinimo modelis, tačiau T. Koomen, M. Pol [KP99] jį priskiria brandos modelių grupei, vadinasi, jis toks pat bendras kaip ir TMM. Kaip pavyzdys, įrodantis TMM bendrumą, būtų jo schemos panašumas į naujo modelio metamodelį. TMM atsako į klausimą, *ką* vykdyti, kad testavimo procesas būtų brandus. Taigi naujai sukurtas modelis nurodo, *kaip* integruoti testavimo procesą į programų sistemų kūrimo gyvavimo ciklą (8 pav.) ir *kada* bei *kokias* praktikas vykdyti.

Rezultatai

1. Nustatyta didelės apimties PS testavimo specifika, taikytini metodai, įrankiai, geriausios praktikos. Visi šie elementai įdiegti naujame testavimo procese.
2. Palyginti modeliai skirti programinei įrangai kurti. Išsiaiškinta, kad literatūroje iteratyvus testavimo modelis tik labai abstrakčiai aptartas nors plačiai taikomas. Visos 8 pav. pateikiamos modelio procesų sritys, išskyrus *testavimo planavimą*, kuris žymi testavimo proceso pradžią ir *bandomosios versijos testavimą,* žymintį testavimo pabaigą, yra iteratyvios, susietos ir sudaro ciklus, kurie naudojami validacijai, verifikacijai ir regresiniam testavimui realizuoti. Rasti sąryšiai, jungiantys PS kūrimo ir testavimo procesus, paaiškėjo, jog pagrindinės PS kūrimo veiklos turi savo atitikmenis testavimo procese, nes testavimas turi vykti lygiagrečiai su PĮ kūrimu.
3. Pasiūlytas testavimo proceso modelis organizacijai, kuriančiai didelės apimties programinę įrangą (2.2. darbo dalis, 7 pav., 8 pav., 1 priedas). Modelis sujungia geriausias „V-modelio“ ir iteratyvaus modelio savybes bei yra lengvai plečiamas ir apima modernaus testavimo praktikas. Remtasi testavimo brandos ir gerinimo modeliais, iš kurių panaudotos geriausios praktikos, aptartos 3.2 skyriuje.
4. Pasiūlyti testavimo modelio patobulinimo sprendimai, kurie įdiegiami į testavimo proceso modelio sritis, panaudojus testavimo brandos modelio ir testavimo proceso gerinimo modelio siūlomas praktikas. Siekiant įgyvendinti šį uždavinį, 3-iojoje darbo

dalyje, apžvelgus TMM ir TPI modelius, kiekviena brandos ir gerinimo modelių praktikų grupė išanalizuota ir pateiktas siūlymas, į kurią šiame darbe pateikto testavimo proceso modelio proceso sritį ji turėtų būti integruota. Integruotos praktikų grupės, susietos su proceso sritimis, pateikiamos 1 priede.

Šaltinių sąrašas

[BCS96] I. Burnstein, C. Carlson R., T. Suwannasart. Developing a Testing Maturity Model: Part I. 1996. CrossTalk.
[žiūrėta 2008-04-16]. Prieiga per internetą:
<http://www.stsc.hill.af.mil/crosstalk/1996/08/developi.asp>

[Cat07] A. Catherine. Rapid Application Development: A Quick Methodology. 2007.
[žiūrėta 2008-05-20]. Prieiga per internetą:
<http://ezinearticles.com/?Rapid-Application-Development:-A-Quick-Methodology&id=414949>

[CYT96] J. S. Collofello, Zhen Yang, J. D. T. Tvedt. Modelling Software Testing Processes. 1996.
[žiūrėta 2009-04-02]. Prieiga per internetą:
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=493647&isnumber=1063>

[Dap07] S. Dapkūnas. Programų sistemų kokybė. Vilniaus universitetas. 2007.

[FM06] H. Fujita , M. Mejri. New trends in software methodologies, tools and techniques. IOS Press, 2006, p. 410.

[FT06] L. Frantzen, J. Tretmans. Formal Methods for Components and Objects. 2006.
[žiūrėta 2008-06-12]. Prieiga per internetą:
<http://books.google.com/books?id=QfZrkJvpVLoC&hl=lt>

[Har04] J. Harkonen. Testing Body of Knowledge. University of Oulu. 2004.

[YSJ08] L. Youngzhong, D. Simeng, Yang J. Research on Behaviour – Based Test Process Model. 2008.
[žiūrėta 2009-04-02]. Prieiga per internetą:
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4731584&isnumber=4731550>

[KP99] T. Koomen, M. Pol. Test Process Improvement, Addison-Wesley. 1999.

[MBE+05] T. Müller, R. Black, S. Eldh, D. Graham, K. Olsen, M. Pyhäjärvi, M. Thompson Veendendal. Certified Tester Foundation Level Syllabus. 2005, p.13.

[MC04] M. Marin-Galiano, A. Christman. Insurance: an R-Program to Model Insurance Data. Department of Statistics, University of Dortmund, Germany. 2004.

[Nau03] I. Naujikas. Testavimo brandos modeliavimas. Informacijos mokslai. Vilnius. 2003. P. 167-171.

[Nor01] H. Norman. The Problem With Testing. 2001. CrossTalk.

[PRI03] M. Pyhajarvi, K. Rautiainen, J. Itkonen. Increasing understanding of the modern testing perspective in software development projects. 2003. [žiūrėta 2008-05-20]. Prieiga per internetą: <http://csdl2.computer.org/comp/proceedings/hicss/2003/1874/08/187480250b.pdf>

[Roy03] W. W. Roice. Managing the Development of large Software Systems. 2003.

[Roo07] R. van Roosmalen. Scrum and Testing. 2007. [žiūrėta 2008-06-08]. Prieiga per internetą: <http://www.scrumalliance.org/resources/270>

[SD05] E. G. Swedin, L. David. The Life Story Of A Technology. Greenwood Press. 2005, p.166.

[SK01] M. Subrahmanyan, M. Kumar. Entity Life Cycle Testing. 2001. [žiūrėta 2008-06-14]. Prieiga per internetą: <http://www.tarrani.net/EntityLifeCycleTesting.pdf>

[Sta02] C. Staab. Using SW-TMM to Improve the Testing Process. 2002. [žiūrėta 2008-06-14]. Prieiga per internetą: <http://www.improveqs.nl/pdf/Using%20SW%20TMM.pdf>

[Swi00] R. Swinkels. A comparison of TMM and other Test Process Improvement Models. Frits Philips Institute. 2000. [žiūrėta 2009-04-20]. Prieiga per internetą: <http://wwwbruegge.informatik.tu-muenchen.de/static/contribute/Lehrstuhl/documents/12-4-1-FPdef.pdf >

[Vee08] E. Van Veenendaal. Test Maturity Model Integration (TMMi). 2008. [žiūrėta 2009-01-05]. Prieiga per internetą: < http://www.tmmifoundation.org/downloads/tmmi/TMMi%20Framework.pdf>

[Whi09] S. A. White. Business Process Modeling Notation (BPMN). 2009. [žiūrėta 2009-04-05]. Prieiga per internetą: <http://www.omg.org/docs/formal/09-01-03.pdf>

[Wu08] Jun Wu. A Software Size Measurement Model for Large-Scale Business Applications. 2008. [žiūrėta 2009-04-05]. Prieiga per internetą: <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4721996>

Santrumpos

BPMN – angl. *Business Process Modelling Notation* – verslo proceso modeliavimo notacija.

CASE – angl. *Computer Aided Software Engineering* – automatizuotas kompiuterinis programinės įrangos projektavimas.

CMM – angl. *Capability Maturity Model* – gebėjimų brandos modelis.

CMMI - angl. *Capability Maturity Model Integration* – gebėjimų brandos integracijos modelis.

MMAST - angl. *The Maturity Model for Automated Software Testing* – automatizuoto testavimo brandos modelis.

PĮ – programinė įranga.

PS – programų sistemos.

RAD – angl. *Rapid Application Development* – greitas programinės įrangos kūrimas.

SPICE - angl. *Software Process Improvement and Capability Determination* – programinės įrangos proceso gerinimo ir gebėjimo nustatymas.

TAP - angl. *The Testing Assessment Programme* – testavimo vertinimo programa.

TCMM - angl. *The I.T.B.G. Testing Capability Maturity Model* – testavimo gebėjimų brandos modelis.

TIM - angl. *The Test Improvement Model* – testavimo gerinimo modelis.

TMap – angl. *Test Management Approach* – testavimo valdymo požiūris.

TMM - angl. *Test Maturity Model* – testavimo brandos modelis.

TMMi - angl. *Test Maturity Model integration* – testavimo brandos integracijos modelis.

TOM - angl. *The Test Organization Maturity Model* – testavimo organizavimo brandos modelis.

TPI - angl. *Test Process Improvement* – testavimo proceso gerinimas.

TSM - angl. *The Testability Support Model* – testavimo palaikymo modelis.

UML – angl. *Unified Modelling Language* – vieninga modeliavimo kalba.

Priedai

1 Priedas. Testavimo proceso modelio sričių sandara

Proceso sričių aprašyme pirmiausia aprašomi tikslai ir po to siūlomos praktikos. Pavyzdžiui:

1.1. Tikslas

1.1.1. Praktika

Pastaba. Išlaikyta TMM atitinkanti tikslų ir praktikų numeracija.

Id: P1. PS kūrimo procesas.
PS kūrimo praktikos nedetalizuojamos.

Id: P1.1 Reikalavimų analizė.
PS kūrimo praktikos nedetalizuojamos.

Id: P1.2 Sistemos projektavimas.
PS kūrimo praktikos nedetalizuojamos.

Id: P1.3 Architektūros projektavimas.
PS kūrimo praktikos nedetalizuojamos.

Id: P1.4 Modulių projektavimas.
PS kūrimo praktikos nedetalizuojamos.

Id: T2. Testavimo procesas.
2.1. Sukurti testavimo ir klaidų taisymo tikslus. 2.1.1. Procedūros ir testavimo tikslai sukuriami, patvirtinami, vykdomi ir reguliariai peržiūrimi. 2.1.2. Naudojama pagrindinė defektų klasifikacijos schema, saugykla. 2.1.3. Vykdomi pagrindiniai testavimo ir klaidų taisymo matavimai. 2.2.5. Taikomi pagrindiniai planavimo ir matavimo įrankiai. 2.3. Oficialiai įtvirtinti pagrindines testavimo technikas ir metodus. 2.3.1. Apibrėžiama pagrindinių technikų ir metodų aibė. 2.3.2. Paruošta techninio mokymo medžiaga bei įrankiai pagrindinėms technikoms ir metodams naudoti. 2.3.3. Testavimas planuojamas ir įdiegiamas skirtinguose testavimo lygmenyse. 2.3.4. Pagrindinės baltos ir juodos dėžės strategijos, technikos ir metodai naudojami projektuojant testavimo atvejus. Bendravimas tarp programuotojų ir testuotojų testavimo atžvilgiu yra aukštesnio lygio. 2.4. Testavimo infrastruktūra planuojama ir gerinama (TPI). 2.4.1. Testavimo aplinka yra valdoma ir kontroliuojama. 2.4.2. Naudojami testavimo įrankiai. 2.4.3. Kontroliuojama biuro aplinka ir laikas. 3.3.3. Paskiriami ištekliai ir vykdomi mokymai tam, kad testavimas būtų integruotas į PS kūrimo gyvavimo ciklą. 3.4. Testavimo proceso kontrolė ir monitoringas.

3.4.1. Sukuriamos ir reguliariai peržvelgiamos procedūros, tikslai, strategijos, matavimai tam, kad būtų galima kontroliuoti ir stebėti testavimą.

3.4.2. Atliktų su testavimu susijusių matavimų rezultatai panaudojami kontrolei ir stebėjimui kituose projektuose. Testavimo statusų ataskaitos pateikiamos reguliariai.

3.4.3. Mokymo įrankiai bei kiti ištekliai turi būti prieinami tam, kad būtų galima kontroliuoti ir stebėti testavimą.

4.1. Sukurti organizacijos lygio peržiūros programą.

4.1.1. Sukuriamos procedūros, tikslai, strategijos ir matavimai visų programinės įrangos produktų peržiūrai, sukurti elementai patvirtinami, vykdomi ir reguliariai peržvelgiami.

4.1.2. Personalas apmokomas, kad sugebėtų tvarkingai peržiūrėti strategijas ir praktikas bei rinktų peržiūrų duomenis.

4.1.3. Kiekviename projekte peržiūrima programinė įranga, atliekami matavimai bei jais remiantis gerinama produkto ir proceso kokybė.

5.3. Testavimo proceso optimizavimas.

5.3.1. Organizacijoje suformuojama testavimo proceso gerinimo grupė, atsakinga už daugkartinį panaudojamumą, kontrolę, vertinimą, gerinimą.

5.3.4. Aukštos kokybės testavimo proceso komponentai yra pripažįstami ir nuolat naudojami visoje organizacijoje.

Id: T2.1 Testavimo planavimas.

2.2. Testavimo planavimo proceso inicijavimas.

2.2.1. Testavimui planuoti skirtos procedūros, tikslai ir strategija yra sukurti, patvirtinti ir reguliariai naudojami bei peržiūrimi.

2.2.2. Kiekvienam testavimo lygiui sukuriami testavimo šablonai.

2.2.3. Paruošiamas kursas apie šablonų naudojimą.

3.1. Įtvirtinti programinės įrangos testavimo organizavimą.

3.1.1. Testavimo komanda atsakinga už testavimo metodų apibrėžimą ir priežiūrą organizacijoje (TPI).

3.1.2.Aukšto lygio testai atliekami paskirtų testavimo grupių, susidedančių iš motyvuotų narių (TPI).

3.1.3. Jeigu reikia techninės ekspertizės, galimi mokymai.

3.2. Įtvirtinti techninio mokymo programą.

3.2.1. Sukuriamas organizacinio mokymo strateginis dokumentas.

3.2.2. Sukuriama vidinė mokymo grupė, kuri ruošia strategiją ir medžiagą.

Id: T2.2 Priėmimo testavimo projektavimas.

2.2.4. Naudotojo reikalavimai įkeliami kaip įvestis į testavimo planą.

5.1. Defektų prevencija.

5.1.1. Sukuriamos procedūros, strategijos ir matavimai defektų prevencijai, sukurtieji elementai patvirtinami, vykdomi ir reguliariai peržiūrimi.

5.1.2. Mokymai, įrankiai ir kiti ištekliai yra prieinami, tam kad defektų prevencija būtų remiama.

5.1.3. Sukuriamos defektų prevencijos komandos, identifikuoti defektai pašalinami iš kiekvienos gyvavimo ciklo fazės, atliekat priežasčių analizę ir sukuriant planus apsaugoti nuo defekto pasikartojimo.

Id: T2.3 Sistemos testavimo projektavimas.

5.1. Defektų prevencija (Praktikos išvardintos proceso srityje T2.2).

Id: T2.4 Integracijos testavimo projektavimas.

5.1. Defektų prevencija (Praktikos išvardintos proceso srityje T2.2).

Id: T2.5 Modulių testavimo projektavimas.
5.1. Defektų prevencija (Praktikos išvardintos proceso srityje T2.2).

Id: T2.6 Programavimas.
Proceso srityje taikytinos praktikos, kurios aptartos literatūros apžvalgoje ir tinkamos programuotojams: metodai ir įrankiai.

Id: T2.7 Priėmimo testavimas.
4.2. Įtvirtinti testavimo matavimo programą. 4.2.1. Procedūros, tikslai, strategijos bei matavimai sukuriami, patvirtinami, vykdomi ir reguliariai peržiūrimi. 4.2.2. Sukuriama testavimo matavimo programa kartu su matavimo ataskaitų sistema. Rezultatai plačiai naudojami organizacijoje. 4.3. Programinės įrangos kokybės vertinimas. 4.3.1. Sukuriamos, patvirtinamos bei naudojamos procedūros, tikslai, strategijos bei matavimai programinės įrangos kokybei vertinti. 4.3.2. Vykstantys apmokymai, naudojami įrankiai bei kiti ištekliai užtikrina programinės įrangos kokybės vertinimo prieinamumą. 4.3.3. Sukuriami tikslai kiekvienam projektui pagal strategiją. Testavimo procesas yra struktūrizuotas tam, kad būtų galima užtikrinti kokybės tikslų pasiekimą. Sveikintinas ir kliento indėlis į kokybės tikslų sukūrimą. 5.2. Kokybės kontrolė (Praktikos išvardintos proceso srityje T2.10).

Id: T2.8 Sistemos testavimas.
5.2. Kokybės kontrolė (Praktikos išvardintos proceso srityje T2.10).

Id: T2.9 Integracijos testavimas.
5.2. Kokybės kontrolė (Praktikos išvardintos proceso srityje T2.10).

Id: T2.10 Modulių testavimas.
5.2. Kokybės kontrolė. 5.2.1. Siekiant užtikrinti kokybės kontrolę sukuriamos, patvirtinamos ir vykdomos procedūros, tikslai ir strategijos. 5.2.2. Testavimo ir PĮ kokybės užtikrinimo komandos nustato atributus ir įtvirtina tikslus produkto kokybei užtikrinti. Testavimo įrankiai ir technikos naudojami nustatyti, ar tenkinami kokybės tikslai pasiekiami. Kiekybinis kriterijus naudojamas nustatyti testavimo pabaigą. 5.2.3. Testavimo komanda paruošiama naudoti statistinius metodus ir kitas su kokybe susijusias veiklas, pavyzdžiui, panaudojamumo testavimą.

Id: T2.11 Bandomosios versijos testavimas.
Projektas perduodamas klientui.