VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Mobiliųjų programų testavimas remiantis TMMi modeliu.

TMMi model based mobile application testing.

Kursinis darbas

Atliko: 3 kurso 4 grupės studentas
Ričardas Mikelionis (parašas)

Darbo vadovas: Lekt. dr. Vytautas Valaitis (parašas)

Vilnius – 2017

TURINYS

Įvadas

Nuo 2007, kai buvo pristatytas pirmasis iPhone jau praėjo kiek daugiau nei dešimt metų. Šio mobiliojo įrenginio, išmaniojo telefono (angl. Smartphone), pristatymas iš esmės pakeitė mobiliųjų kompiuterių rinką. Iki tol asmeniniai skaitmeniai asistentai (angl. PDA personal digital asistant) buvo nei prieinami, nei labai naudingi įprastam vartotojui, todėl juos turėjo keletas verslo pasaulio žmonių, o paprastas vartotojas su savimi nešiojosi krūvą skirtingą funkcionalumą atliekančių prietaisų: MP3 grotuvų, Fotoaparatų, nešiojamųjų kompiuterių, telefonų. iPhone žadėjo delne telpantį kompiuterį kiekvienam už prieinamą kainą, ir savo pažadą įvykdė. Vienas po kito ėmė rastis mobiliosios operacinės sistemos, turėjusios tiesiogiai konkuruoti su iPhone naudojama iOS, kaip PalmOS, Symbian, Windows 7 mobile, vėliau tapusi 8 ir 8.1 bei Android. Mobiliųjų kompiuterių bei telefonų rinką užplūdo gausa prieinamų įrenginių, kurie su kiekviena nauja laida savo skaičiavimo sugebėjimais artėja arčiau ir arčiau to ką gali atlikti staliniai kompiuteriai ir, iš pažiūros, 2017 metų išmanusis įrenginys savo specifikacijomis „ant popieriaus“ jau seniai pranoko 2007 metų kompiuterį.

Nors šiandien iš gausaus operacinių sistemų pasirinkimo rinkoje iš esmės išlikę tik dvi: iOS ir Android, tačiau išmaniųjų prietaisų populiarumas toli gražu neblėsta. 2016 metais pasaulyje jau buvo apie 2.1 mlrd. išmaniųjų telefonų naudotojų, o iki 2020 planuojama, jog šis skaičius sieks 2.87 mlrd. [Sta]. Esant didžulei įrenginių paklausai proporcingai kyla poreikis ir programinei įrangai pritaikytai šiems įrenginiams – mobiliosioms programoms.

Šio darbu siekiama pateikti esminius skirtumus tarp įprastų darbastalio (angl. desktop) bei internetinių (angl. Web) programų ir mobiliųjų programų skirtų išmaniesiems įrenginiams kokybės užtikrinimo. Šio darbo tikslas - išanalizavus dabartinės rinkos mobiliųjų programų kūrimo procesų tendencijas, išskirti problemines mobiliųjų programų kokybės užtikrinimo sritis bei pateikti keletą probleminių sričių kokybės užtikrinimo sprendimų remiantis testų brandos modeliu.

Siekiant sėkmingai įgyvendinti užsibrėžtą dabo tikslą darbo metu bus:

1. Atliekama mobiliųjų įrenginių sampratos analizė, siekiant išsiaiškinti esminius skirtumus, kurie gali daryti įtaką mobiliųjų programų veikimui;
2. Remiantis mobiliųjų įrenginių sampratos analize išrenkamos pagrindinės probleminės sritys mobiliųjų programų testavime;
3. Atliekama integruotojo testų brandos modelio - TMMi detali analizė;
4. Remiantis mobiliųjų programų testavimo probleminėmis sritimis bei TMMi analizės rezultatais pateikiamas TMMi paremtas modelis skirtas padengti visas problemines sritis;
5. Remiantis agiliųjų metodikų iteratyvumu pateikiamas TMMi modelis pritaikytas komandoms dirbančioms pagal agiliasias metodikas.
6. Remiantis probleminių sričių bei agiliųjų metodikų modeliais pateikiamas mobiliųjų programų testavimo sprendimas.

1. Mobiliųjų programų kokybės užtikrinimas.

2013 metais atliktas tyrimas rodo, jog 79% išmaniųjų telefonų naudotojų išbando atsisiųstą programėlę tik kartą prieš ištrindami [Com]. Didelė to priežastis yra žemesnė mobiliųjų programų kokybė palyginus su tuo ką vartotojai yra įpratę matyti savo naršyklėse, ar darbastalio programose. Išmaniųjų telefonų naudotojai yra pratę prie nemokamos ar žymiai mažiau kainuojančios programinės įrangos, kuri gauna nuolatinius atnaujinimus. (Pvz.: Adobe photoshop express iOS 0eur., Adobe Photoshop Windows 10 290eur./m. [Ado]). Kyla kainų dilema: norint palaikyti žemas kainas rinkoje reikia mažinti kainas ir kūrimo procese. Tai dažnai daroma taupant laiką, kur ir nukenčia kokybės užtikrinimo procesai.

Į mobiliųjų programų kūrimą vis dar žiūrima taip pat, kaip į interneto programų (angl. WEB application) ar darbastalio programų kūrimą, neįžvelgiant esminių skirtumų tarp šių dviejų programinės įrangos kūrimo procesų. Didžiulė įrenginių variacijų gausa, ir iš esmės išsiskiriantys naudojimosi mobiliąja programine įranga įpročiai turėtų versti kūrėjus į mobiliųjų programų kūrimą žvelgti kitaip.

1.1. Mobiliųjų programų kūrimo proceso probleminės sritys.

M. Satyanarayanan dar 1996-aisiais savo straipsnyje apie mobiliuosius kompiuterius puikiai aibrėžė esmines sritis, kurios išskiria mobiliuosius įrenginius nuo stacionariųjų. Išskirtos keturios sritys [Sat96]:

- Riboti skaičiavimo ištekliai (angl. limited computing resources)
- Apsauga ir pažeidžiamumas
- Galia (angl. performance) ir patikimumas
- Riboti energijos ištekliai.

Iš esmės mobiliosios programos yra tokia programinė įranga, kuri veikia nešiojamame (mobiliame) įrenginyje. Tačiau į mobiliąją kompiuteriją žvelgiant per išmaniųjų telefonų prizmę mobiliųjų programų apibrėžimą galima sukonkretinti. Mobilioji programa (angl. mobile app) - programa suprojektuota veikti išmaniąjame telefone, planšetiniame kompiuteryje ar kitame mobiliame įrenginyje ir/arba, kaip įvestį, priimanti situacija paremtą (angl. context based) informaciją pvz. GPS signalas [KK13]. Situacijos supratimas, naudojimasis situacine informacija ir gebėjimas keisti programos elgseną pagal gautus duomenis yra dar vienas esminis mobiliųjų programų, ypač skirtų išmaniesiems telefonams, skiriamasis bruožas.

Remiantis šiais keturiais techniniais išskirtinumo bruožais, ir esminiu veikimo skirtumu galima pradėti išskirti problemines kokybės užtikrinimo sritis mobiliųjų programų kūrime. Laikant progamas išmaniesiems telefonams tyrimo baziniu objektu išskiriamos šios sritys:

- Ryšio sujungiamumo
- Ribotų išteklių
- Draugiškumo vartotojui
- Situacinio prisitaikymo
- Įrenginių gausos

- Naujų operacinių sistemų versijų
- Liečiamųjų ekranų

1.1.1. Ryšio sujungiamumo probleminė sritis

Sugebėjimas prisijungti prie mobiliojo ryšio stočių bei bevielio interneto taškų dažnai mobiliajai programai yra kritinis reikalavimas be kurio ši nesugebės funkcionuoti. Problema kyla dėl ryšio nepastovumo keliaujant. Keliaujant didelius atstumus vyksta persijungimai tarp mobilaus ryšio stotelių, vartotojas gali pakliūti į neryšio zoną, ar vietą kurioje veikia viso labo GPRS ar EDGE ryšys. Taip pat ir su bevielio interneto stotelėmis, kurių dažna veikia vos 100m. spinduliu nesant jokiems trukdžiams.

Testuojant šioje srityje derėtų atsižvelgti į jau egzistuojančius ryšio defektus esančius specifiniame testavimui naudojamame įrenginyje, ar operacinėje sistemoje [And]. Testuojant programas kurios priklausonuo įrenginio sugebėjimo prisijungti prie interneto ar mobiliojo ryšio tinklų atsiranda poreikis atlikti funkcinius testus įvairuose ryšio sujungiamumo scenarijuose. Pagrindinis siekis yra užtikrinti teisingą programos veikimą, net dingus ryšiui ar šiam staiga smarkiai sulėtėjus.

1.1.2. Ribotų išteklių probleminė sritis

Kol mobilieji įrenginiai tampa vis galinesni ir galingesni, jie vis dar negali pasivyti satcionariųjų bei nešiojamųjų kompiuterių. Pavyzdžiui vienas iš galingesnių šiuo metu rinkoje esančių telefonų, Samsung Galaxy s8+ turi 6GB atminties bei 8 branduolių, palaikančių 8 gijas, sistemą mikroschemoje (angl. system on a chip) į kurią integruoti vaizdo procesorius bei pagrindinis procesoriai [Sam]. Šiuo metu didžioji dauguma vartotojams prieinamų motininių plokščių kompiuteriams palaiko iki 64GB atminties, o intel neseniai atskleidė savo naujosios kartos i9 procesorių liniją, kur aukščiausios klasės procesorius sudarytas iš 18 branduolių bei palaiko 36 gijas [Int].

Testuojant šioje srityje būtina atsižvelgti į išteklių įrenginyje panaudojimą paleidžiant bei naudojantis programa. Tai daroma siekiant išvengti staigių našumo sumažėjimų, neteisingo, nenumatyto programos ar įrenginio veikimo staiga imant naudoti 100% prieinamų išteklių. Pagrindinis testavimo siekis užtikrinti teisingą programos bei įrenginio veikimą esant skaičiavimo išteklių trūkumui.

1.1.3. Draugiškumo vartotojui probleminė sritis

Kūrėjai kuriantys programas mobiliosioms operacinėms sistemos (Android, iOS) privalo sekti operacinių sistemų leidėjo nustatytas varotojų sąsajos gaires. Dėl įrenginių gausos, skirtingų dydžių bei raiškos ekranų atsiranda grafinių neatitikimų tikimybė.

Testuojant šioje srityje būtina testavimą atlikti su daug skirtingų įrenginių, veikiančių su skirtingomis operacinės sistemos versijomis bei skirtingų raiškų ir dydžių ekranais.

1.1.4. Situacinio prisitaikymo probleminė sritis

Didelė dalis mobiliųjų programų remiasi duomenimis gautais priklausomai nuo situacijos, iš įvairių sensorių išmaniajame įrenginyje. Dažnai išmanieji įrenginiai keletą skirtingų sensorių,

tokių kaip mikrofonai, šviesos jutikliai, akselerometrai bei giroskopai, vaizdo kameros. Tokie įrenginiai turi ir keletą būdų komunikuoti, susijungti su kitais prietaisais ar prisijungti prie interneto: bluetooth, wifi, 4G/LTE, NFC. Visi šie sensoriai gali suteikti programai daugybę informacijos realiu laiku, atsižvelgiant į tai kokius vartotojas veiksmus atlieka, ar kur jis tuo metu yra.

Testuojant šioje srityje patikrinti visų įrenginių ir įvesčių kombinacijų praktiškai neįmanoma. Todėl testus reikia skirstyti į kontekstinius (angl. context-specific) bei nusistatyti užtektinus testų padengtumo kriterijus. Kuriant kontekstu paremtus testus, bei renkantis įrenginius su kuriais bus vykdomi testavimo atvejai verta atsižvelgti į konkrečiam įrenginiui ar operacinės sistemos versijai būdingus defektus [And].

1.1.5. Įrenginių gausos probleminė sritis

Rinkoje vartotojams prieinami šimtai skirtingų mobiliųjų įrenginių iš skirtingų gamintojų, veikiančių su skirtingomis operacinėmis sistemomis, surinktų iš skirtingų komponentų. 2013 metais atlikto tyrimo metu buvo suskaičiuota, jog rinkoje yra 1800 unikalių vien android operacine sistema paremtų įrenginių [MDFE12].

Testuojant šioje srityje reikia stengtis didinti testų padengtumą stengiantis sumažinti naudojamų įrenginių kiekį. Tą galima daryti grupuojant įrenginius ir testuojant ne konkrečiuose įrenginiuose (mažiau įrenginių) tačiau grupėse. Pavyzdžiui:

- Pirmoji grupė, žemiausio prioriteto testai, silpni, palyginus su dabartiniais pagrindinių gamintojų flagmanais, turintys silpnesnius procesorius, mažiau atminties, žemesnės raiškos ekranus. Paremti senesne operacinės sistemos versija, veikiantys su senesne naršykle. Dar vis populiarūs, tačiau gamintojo jau nebepalaikomi įrenginiai.
- Antroji grupė, vidutinio prioriteto testai, vidutinio galingumo įrenginiai, dažniausiai kelių iteracijų, palyginius su dabartiniais pagrindinių išmaniųjų įrenginių gamintojų flagmanais, senumo pvz: iPhone 5s palyginus su iPhone 7. Jau gan seni, 3-2 metai, tačiau dar vis gamintojo palaikomi įrenginiai.
- Trečioji grupė, patys naujausi nedaugiau, nei vienerių metų senumo įrenginiai. Paremti naujausia operacinės sistemos versija, naujausios kartos komponentais, turintys daugiau atminties, aukštesnės raiškos ekranus, nei praeitos iteracijos modeliai [KK13].

Testuojant pasirinkta įrenginių kombinacija įgalina keletą pasirinkimų, kaip tokį testavimą galima vykdyti: naudoti rankinį testavimą ar automatinį testavimą, testavimui pasitelkti komandas įmonėje ar žmones pasamdytus iš kitų įmonių, testuoti pagal planą ar naudoti tiriamąjį testavimą (angl. exploratory testing) bei ar naudoti emuliatorius, virtualias mašinas ar testuoti ant realių, fizinių įrenginių.

1.1.6. Naujų operacinių sistemų probleminė sritis

Mobiliosios operacinės sistemos, palyginus su operacinėmis sistemomis stacionariesiems kompiuteriams, yra išleidžiamos labai dažnai. Pastaruoju metu tiek iOS tiek Android nauja „numeruotoji“ versija yra išleidžiama kas met. Taip dažnai išleidžiant naują operacinės sistemos

iteraciją gali kilti daugybė nesuderinamumo defektų, turint omenyje tai, kad ne visos versijos yra palaikomos, t.y. naujesnės programų versijos gali veikti tik su keliomis paskutinėmis operacinės sistemos iteracijomis.

Testuojant programas reikėtų atsižvelgti į egzistuojančius konkrečios operacinės sistemos versijos defektus, bei testuoti naudojant keletą skirtingų tos pat operacinės sistemos iteracijų. Taip siekiant įsitiktinti, ar defektai atsiradę mobiliojoje programoje buvo sukelti operacinės sistemos defekto, ar blogo kodo.

1.1.7. Liečiamųjų ekranų probleminė sritis

Liečiamieji ekranai yra pagrindinis, ir dažniausiai vienintelis įvesties pateikimo būdas mobiliuosiuose įrenginiuose. Dažna problema, ypač senesniuose įrenginiuose, yra lėtas prisilietimo prie ekrano atsakas, ar ekrano įvesties atsako laiko padidėjimas imant naudotis sudėtingesnėmis programomis.

Testuojant reikėtų naudoti daug skirtingų įrenginių, kurie iš esmės skiriasi savo specifikacijomis, ypač ekranų dydžiu, raiška. Pagrindinis testavimo objektas yra resursų pasiskirstymas. Tikrinama ar testuojama programinė įranga mobilajame įrenginyje nenaudoja daugiau skaičiavimo ištekilių, nei manoma, kad ji turėtų naudoti, taip pat tikrinama, ar pasisavinti ištekliai nebenaudojant yra teisingai atlaisvinami. Reikėtų stengtis įvykdyti didelės apkrovos scenarijų, kai programinė įranga veikia pilnu pajėgumu atlikdama didžiausių skaičiavimų reikalaujančias užduotis, ir tikrinti mobiliojo įrenginio atsako laiką.

1.2. Integracinis testų brandos modelis - TMMi

Testų brandos modelis buvo sukurtas, tam, kad įvairios organizacijos galėtų įvertinti bei įvertinę pagerinti savo testavimo procesus. Tai taip pat teikia naudą, kaip modelis, tiesiogine to žodžio prasme, kuris nurodo, kaip testavimo procesas turėtų palaipsniui gerėti savo efektyvumu bei kokybe [Bur10]. Žinoma, jog TMMi nėra vienintelis procesų vertinimo bei gerinimo modelis ar standartas, per paskutiniuosius du dešimtmečius buvo sukurti keletas tokių standartų. Vienas iš jų yra CMM - Gebėjimo Brandos Modelis (angl. Capability Maturity Model) bei jo įpėdinis Integruotas Gebėjimo Brandos Modelis (angl. Capablitiy Maturity Model Integration - CMMi). Tačiau palyginus su TMM bei TMMi daugelis šių modelių neskiria pakankamai dėmesio testavimo procesų gerinimui.

TMMi modelis sudarytas iš penkių brandos lygių, kurie kiekvienas rodo testavimo proceso brandos būseną. Šie modelio nustatyti lygiai padeda struktūrizuotai siekti aukštesnio proceso našumo, taigi jų reikia siekti iš eilės t.y. negalima iš antrojo brandos lygio iš karto pasiekti ketvirtojo ar penktojo. Taip pat struktūrizuotas išskirstymas lygiais užtikrina, jog pasiekus aukštesnyjį lygį, žemesniajame lygyje pradėtos naudoti praktikos bus naudojamos ir toliau.

Kiekvienam testų brandos lygiui pasiekti užsibrėžiamos veiklos, užduotys bei pareigos (angl. Activities, Tasks, Responsibilities - ATR). Šie trys kriterijai nurodo esmines sritis kurios turi būti įtrauktos į testavimo procesą, tam, kad būtų pasiektas norimas brandos lygis. Šie brandos tiksliai turi tapti siekiamybe visiems kūrimo procese dalyvaujantiems žmonėms, ne tik kokybės

užtikrinimo srities specialistams. Taip užtikrinimas visapusiškas testavimo proceso palaikymas bei įvertinimas siekiant pagerėjimo.

Testų brandos lygis nustatomas naudojantis TMMi įvertinimo modeliu (angl. TMMi assessment model) sudatyto iš trijų komponentų.

- Klausimyno, susijusio su brandos tikslų siekimu. Skirtu nustatyti dabartinę testavimo proceso būsesną.
- Gairių rinkino, skirto brandos lygių įvertinimu užsiimančiai komandai.
- Įvertinimo procedūra išdėstyta pažingsniui, skirta vesti įvertinimo komandą per testų procesų įvertinimą bei gerinimą.

1.2.1. Testų Brandos Modelio testų brandos lygiai

Testų brandos modelį apibrėžia penki brandos lygiai [Bur10]:

1. Pradinis (angl. Initial)
2. Fazių aiprėžimo (angl. Phase definition)
3. Integracijos (angl. Integration)
4. Valdymo bei vertinimo (angl. Management and Measurement)
5. Optimizacijos arba Defektų prevencijos bei kokybės valdymo (angl. Optimisation/Defect Prevention and Quality Control)

Kiekvienas iš šių lygių turi ir vidinę struktūrą Priedas nr. 1. Pirmasis brandos lygis savo struktūros neturi, nes tiesiog nurodo bazinio proceso egzistavimą, o jo pasiekimui užtenka chaotiško proceso. Kiekvieno brandos lygio brandos tikslai atspindi testavimo proceso pagerinimo tikslus, kuriuos reikia įvykdyti norint pasiekti norimą brandos lygį. Taip pat norint pasiekti aukštesnį lygį būtina įgyvendinti ir visus tikslus reikalingus pasiekti žemesnius lygius tam, kad nevyktų proceso brandos regresija ir tikrai būtų pasiektas testavimo proceso pagerinimas [Tmm].

1.2.1.1. Pirmasis brandos lygis

Šiame brandos lygyje operuoja visos programinės įrangos kūrėjų komandos, kurios turi bet kokį testavimo procesą. Pasiekti šį lygį užtenka chaotiško, neefektyvaus, tačiau vykdomo testavimo proceso. Šiame lygyje testai dažniausiai būna pritaikyti konkrečiai situacijai, nėra kartotini. Testavimo procesas iš esmės neturi kokybės standarto.

1.2.1.2. Antrasis brandos lygis

Šiame brandos lygyje iš chaotiško proceso imamas apibrėžti konkretus testavimo metodas, bei testavimo procesas imamas valdyti. Imami apibrėžti testavimo scenarijai, kuriami testavimo planai, dažniausiai iš reikalavimų yra keliami testavimo atvejai. TMM bei TMMi paremti krioklio (angl. waterfall) metodologija, todėl testavimas dažniausiai atliekamas kūrimo proceso gale, lyginant turimą produktą, su kūrimo pradžioje apibrėžtais reikalavimais. Pasiekę šį lygį dar, savaime aišku, neturime visiškai efektyvaus testavimo proceso, tačiau turime konkrečiai apibrėžtą testavimo proceso kryptį, kurią bus galima toliau plėtoti siekiant aukštesnio testavimo proceso brandos lygmens.

1.2.1.3. Trečiasis brandos lygis

Šiame brandos lygyje apibrėžiamas standartizuotas procesas bei procedūros, kurios bus naudojamos, visoje organizacijoje ar komandoje, kurios testavimo procesą stengiamasi pagerinti. Pasiekus antrąjį brandos lygį testavimo procesas yra struktūrizuotas čia jis imamas integruoti į programinės įrangos kūrimo gyvavimo ciklą (angl. Developement life cycle). Pasiekus trečiąjį testavimo proceso bandos lygį, antrąjame apibrėžtas procesas bus integruotas ir pradėtas naudoti visuose naujuose projektuose nuo pat projekto pradžios. Šiame lygyje dar dažnai planuojamas bei vykdomas nefunkcinis testavimas kiekviename projekte įtrauktame į testavimo procesą.

1.2.1.4. Ketvirtasis brandos lygis

Pasiekus šį brandos lygmenį vykdomas visų kūrimo veiklų bei rezultatų detalus vertinimas. Procesų vertinimas turėtų būti pradėtas vykdyti nuo pačios anksčiausios kiekvieno projekto stadijos. Taip siekiama kiekvieną projektą įgyvendinti su kuo įmanoma mažiau defektų. Šiame lygyje vykdomos išsamios projekto apžvalgos, bei ankstyvose projektų stadijose vykdomas statinis testavimas derinamas su dinaminėmis testavimo technikomis apibrėžtomis antrąjame testų proceso brandos lygyje. Pasiekus šį brandos lygį testavimo procesas vykdomas visuose programinės įrangos kūrimo gyvavimo ciklo etapuose, taip pat įtraukiant produktų dokumentacijos peržiūrą Priedas nr. 2. Pakankamos kokybės kriterijai apibrėžiami visiems produktams toje organizacijoje.

1.2.1.5. Penktasis brandos lygis

Šiame brandos lygyje visos testavimo veiklos bei rezultatai yra vertinami siekiant juos optimizuoti. Tai daroma norint nuolatos tobulinti testavimo procesą siekiant visiškos defektų prevencijos, o ne defektų taisymo.

1.2.2. Testų Brandos Modelis TMM bei Gebėjimo Brandos Modelis CMM

Gebėjimo brandos modelis iš pradžių buvo sudarytas, kaip įrankis objektyviai bei tiksliai įvertinti valstybinių projektų vykdytojų darbo procesų kokybę įgyvendinant užsakytą programinės įrangos projektą. Šis modelis smarkiai paremtas procesų brandos modeliu pirmą kartą aprašytu W.Humphrey knygoje „Managing the software Process“ [PCCW93].

TMM modelio struktūra yra paremta tuo kas buvo sukurta CMM modeliui, todėl abiejuose modeliuose randamos tokios pat esminės sudedamosios dalys, kaip pavyzdžiui brandos lygiai su panašiais, o kai kur ir identiškais tikslais tam lygiui pasiekti. Pažvelgus į CMM modelį kyla klausimas ar juo vertėtų naudotis šiandien, gerinant testavimo procesų kokybę nuosavame projekte? CMM ir kiti panašūs modeliai (ISO 9001, BOOT-STRAP, SPICE) tinkamai neatsižvelgia į testavimo keliamas problemas, todėl nėra tinkami konkrečių testavimo procesų gerinimui, tačiau verta pažvelgti giliau į tai, kaip CMM modelis buvo pakeistas kuriant TMM modelį.

Pagrindiniai CMM bei TMM skirtumai:

- Gebėjimo brandos modelyje testavimo proceso branda nėra apibrėžta.
- Pagrindinės testavimo probleminės sritys nėra apibrėžtos esminiuose procesuose.

- Nedaug dėmesio skiriama aukštos kokybės testavimui, kaip procesų bei produktų gerinimo strategijai.
- Nepilnai apibrėžiami, ne daug minimos su kokybe susijusios problemos kaip testuojamumas, testų pakankamumo kriterijai, testavimo planavimas bei programinės įrangos sertifikavimas.
- Pažangesni testavimo procesai, kaip vartojimo statistikos surinkimas, statistinis testavimas bei kiekybinė testavimo proceso kontrolė yra nepakankamai aprašyti.

Šiandien testavimas programinės įrangos kūrimo procesuose atlieka labai svarbų vaidmenį, todėl jam teikiamas toks pat, jeigu ne didesnis dėmesys, kaip programinės įrangos kūrimui. Turint tai omenyje, CMM bei jo įpėdinis CMMi tampa nepriimtini dėl dėmesio testavimo procesų gerinimui stokos.

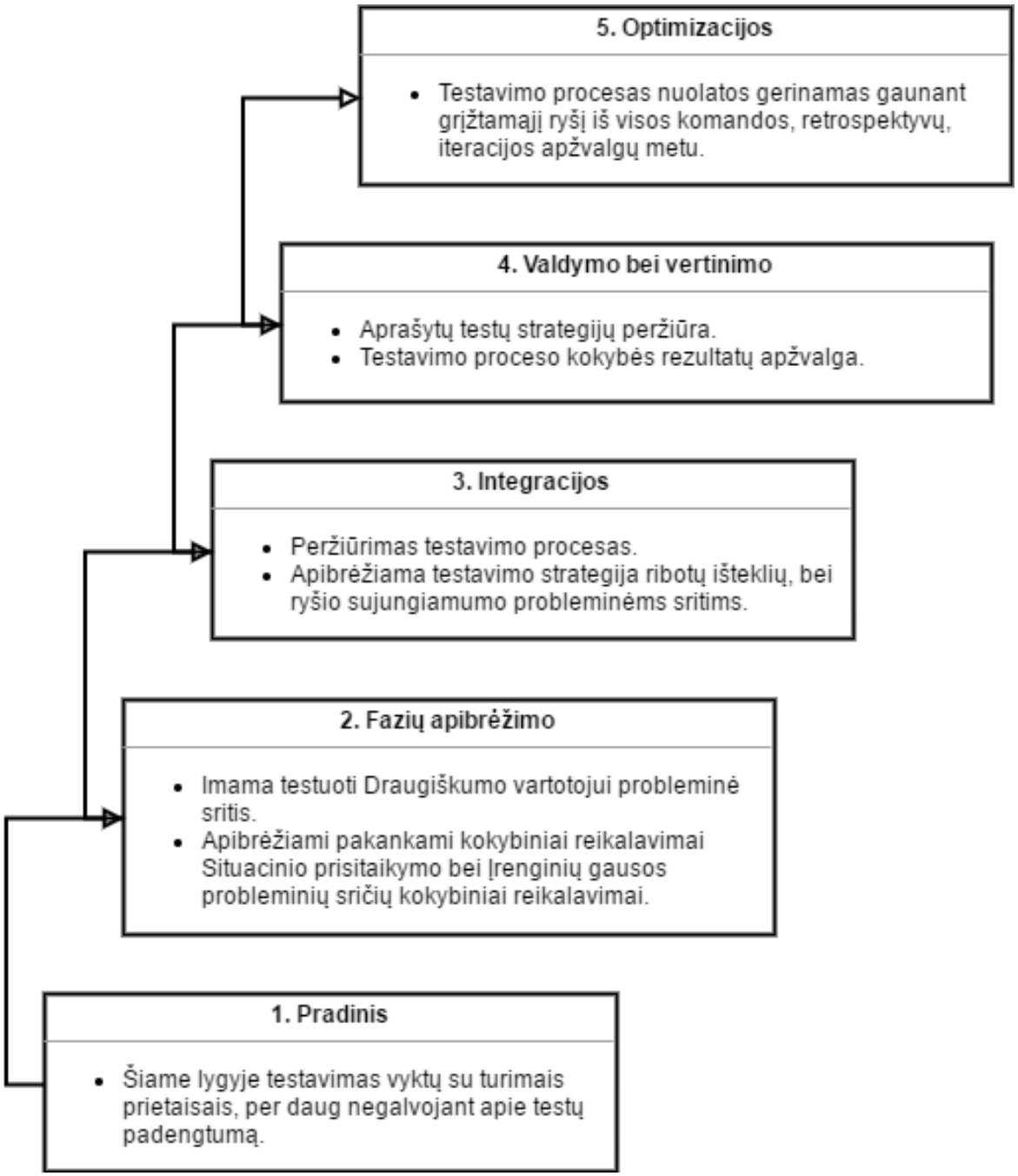
1.3. Probleminių sričių pasiskirstymas TMMi modelyje.

TMMi savo lygmenimis išskirsto ir funkcinį bei nefunkcinį testavimą, pvz. Antrąjame lygyje užtenka sutvarkyti funkcinio testavimo procesus, o trečiojo lygio nebus galima pasiekti neįgyvendinus nefunkcinio testavimo proceso tvarkos. Mobiliųjų programų kūrimo analizėje išskirtas mobiliųjų programų testavimo problemines sritis taip pat paprastai galima išskirstyti į funkcinio testavimo bei nefunkcinio testavimo rūšis:

- Funkciniai testai: Draugiškumo vartotojui probleminė sritis, Situacinio prisitaikymo probleminė sritis, Liečiamųjų ekranų bei Įrenginių gausos probleminės sritys.
- Nefunkciniai testai: Ribotų išteklių probleminė sritis, Ryšio sujungiamumo probleminė sritis bei Operacinių sistemų probleminės sritys.

Kadangi probleminių sričių testavimas lengvai išskirstomas į funkcinį bei nefunkcinį, problemines sritis nesunkiai galima išdėstyti brandos modelyje. Taip ne tik pasiekiant pilną probleminių sričių padengtumą testais, tačiau ir bendro testavimo proceso pagerinimą. Funkciniai testai, ir sritys kuriose pagrindinis dėmesys yra jiems teikiamas turėtų būti apgalvojami antrąjame brandos lygyje, o nefunkciniai testai ir sritys, kuriose pagrindinis demesys į nefunkcinius testus, patektų į trečiąjį brandos lygį.

Taip atrodytų Integraciniu Testų Brandos modelių paremtas probleminių sričių testavimo modelis:



1 pav. Mobiliojo testavimo probleminės sritys sudėtos į TMMi brandos modelį.

2. Agile metodologijos mobiliųjų programų kūrime.

Agilieji kūrimo metodai (angl. agile) tai lanksčios programinės įrangos kūrimo metodikos. Tokiose metodikose viskas kūrimo procesas yra suskaldomas į eilę ciklų, dar vadinamų iteracijomis. Dažnai, kuriant mobiliasias programas remiantis agiliaisiais metodais, visos užduotys padalijamos į keletą mažesnių, skirtų įvykdyti didesniąjai, kurių kiekviena tampa kone savo atskiru mikroprojektu iš kurio pro iteracijos komanda gauna veikiančio produkto dalį.

Iteracinis išskirstymas leidžia projektą tvarkingai išskirstyti nuosekliai, viena po kitos einančiomis dalimis, taip pat išskirstant komandos resursus atitinkamai dirbti mažesnėse komandose, taip sumažinant kūrimo rizikas, tokias, kaip defektai.

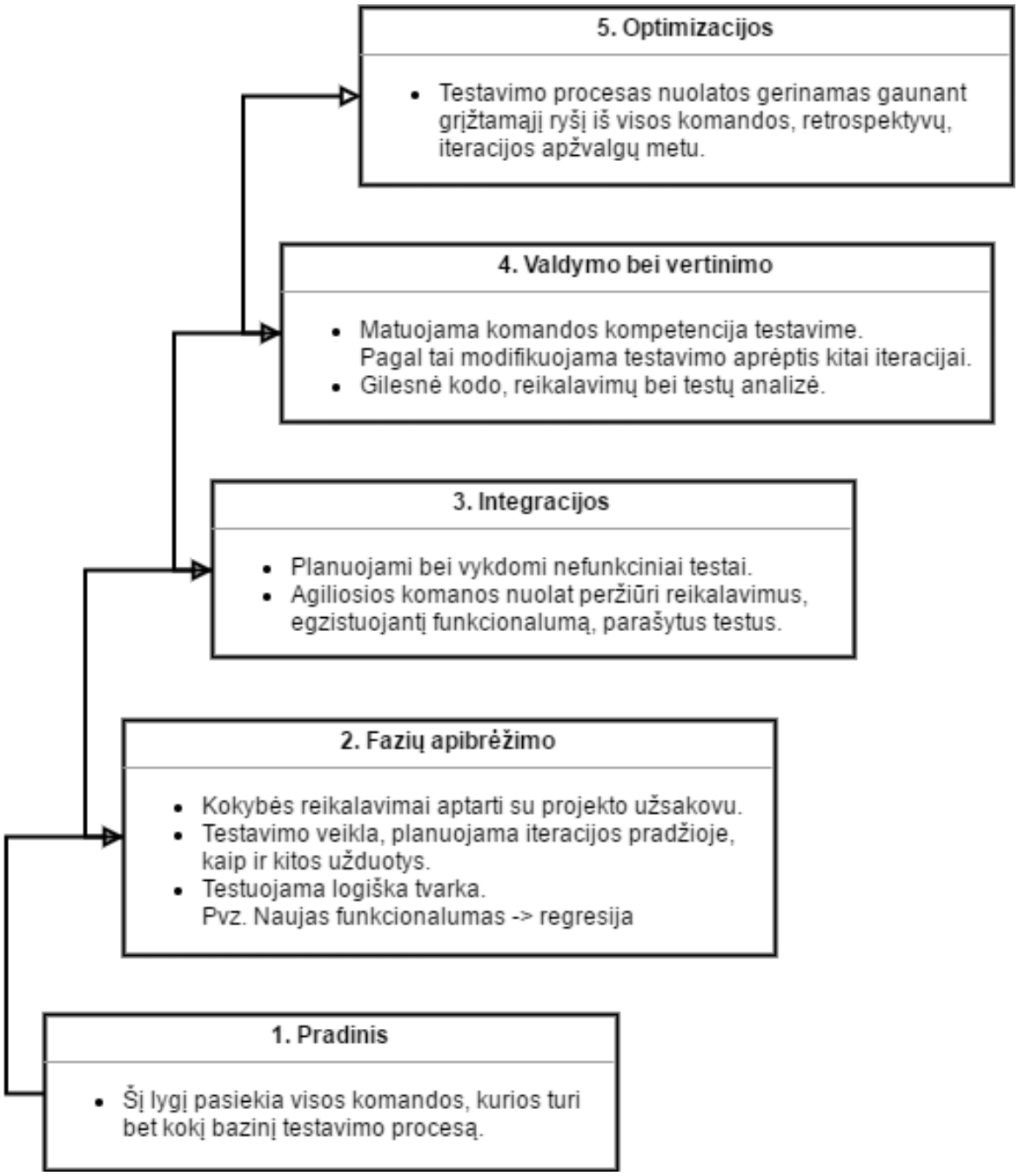
2.1. TMMi ir Agile

Testų brandos modelis bei agiliosios kūrimo metodologijos, dvi praktikos, kurios siekia to paties, tačiau iš pirmo žvilgsnio nėra suderinamos. TMMi, kaip ir daugelis kitų brandos modelių, yra paremtas V-modeliu Priedas nr. 2, tai tokia programinės įrangos kūrimo metodologija, kai visi planavimo, kūrimo bei testavimo procesai yra vykdomi atskirai vienas nuo kito, paeiliui, o kūrimo metu tik planuojant testavimą. Kol tuo tarpu agiliųjų metodologijų manifeste teigiama, jog žmonės ir bendravimas vertinamas labiau, nei procesai ar įrankiai, veikianti programinė įranga vertinama labiau, nei išsami dokumentacija, bendradarbiavimas su klientu vertinamas labiau, nei derybos dėl kontraktų ir reagavimas į pokyčius vertinamas labiau, nei plano įvykdyamas [Agi]. Iš pirmo žvilgsio abi metodologijos atrodo visiškomis priešingybėmis. Ar galime daryti išvadą, jog agiliųjų metodologijų testavimo procesas pasmerktas ir nepagerinamas?

Kuriant programinę įrangą agiliosiomis metodologijomis, nėra rašoma palti dokumentacija, jei ji rašoma iš vis, tačiau kiekvienos iteracijos pradžioje imama dalis reikalavimų, užduočių, ir detaliai išnagrinėjiama bei įvertinama valandų kiekiu ar sudėtingumą, o kiekvienos iteracijos pabaigoje vyksta apžvalga, aptarimas skirtas pagerinti procesus kitai iteracijai bei aptarti tai, kas buvo nuveikta šią iteraciją.

Naudojantis agiliosiomis metodologijomis nėra kuriamas detalus testavimo planas visam projektui, komanda pati neapibrėžia kokybės reikalavimų kiekvienai užduočiai. Testai kuriami universalūs, pritaikomi kiekvienai iteracijai, papildomi testais naujam funkcionalumui atitinkamai nuo to, kas yra atlikta, ar planuojama atlikti. Kokybiniai reikalavimai kinta, tačiau yra lengvai nustatomi glaudžiai bendradarbiaujant su projekto užsakovu, ką agiliosios metodologijos ir skatina.

Žvelgiant į kiekvieną agiliąją iteraciją, kaip atskirą projektą kiekvienai iš jų galima pritaikyti V-modelį taip kiekvieną agiliojo proceso iteraciją padarant suderinamą su TMMi testų brandos modeliu. Remiantis išskirtais TMMi lygiais, aglijiuoju manifestu bei tokio kūrimo procesu agilųjį testavimo procesą TMMi modelyje butų galima išdėstyti taip:

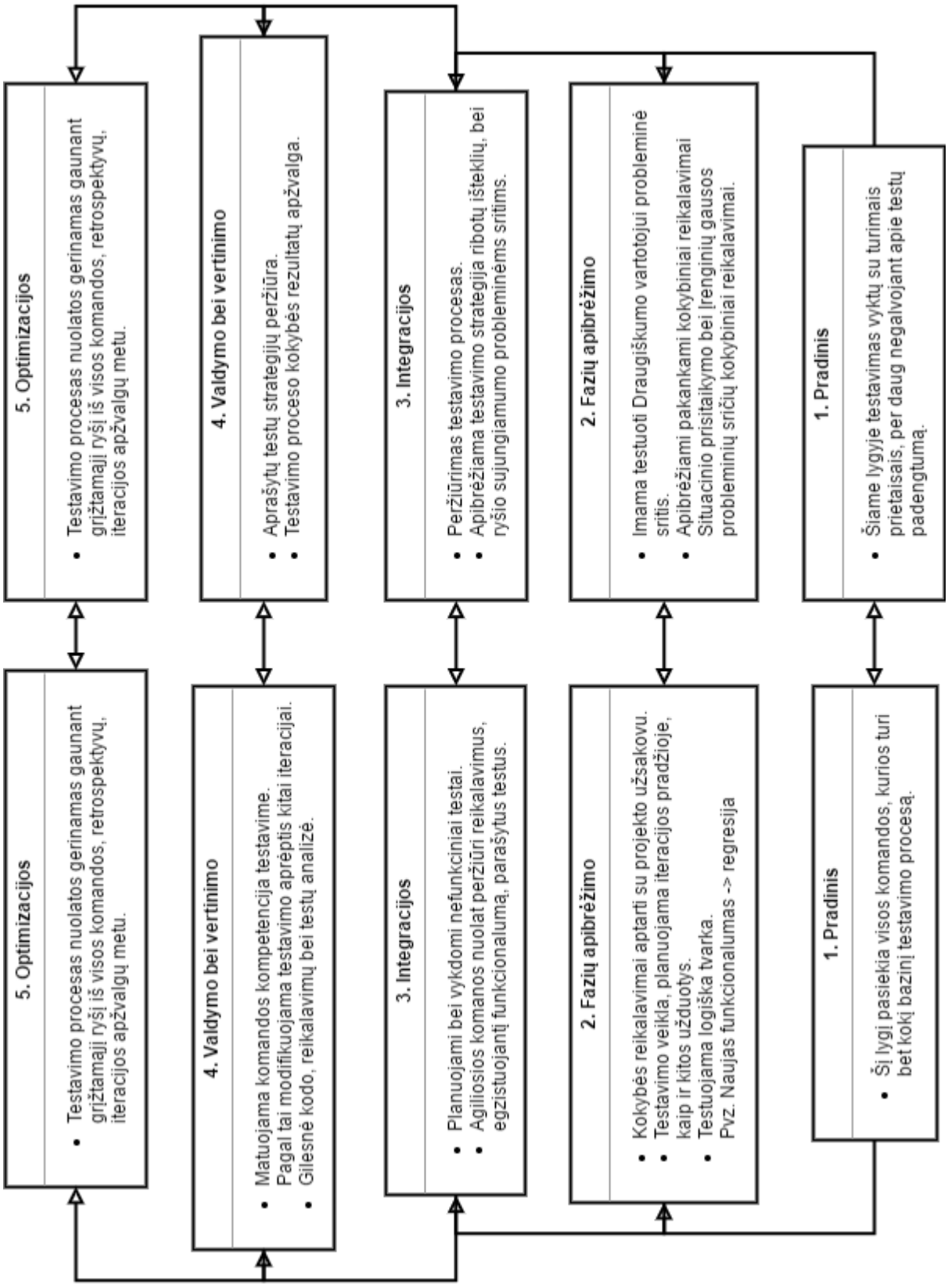


2 pav. Agile metodologijos įpatumai sudėti į TMMi brandos modelį.

3. TMMi modeliu paremtas mobiliųjų programų testavimas

Mobiliųjų programų, ypač kuriamų remiantis agiliosiomis metodologijomis, testavimo procesų gerinimas yra gan problematiškas. Programų kūrimas vykdomas vos kelių savaičių iteracijomis, kurių metu reikia aprėpti visas problemines sritis, taip pat bazinį funkcionalumą bei regresinius testus. Kyla problema, kaip suderinti V-modeliu paremtą probleminių sričių testavimo brandos modelį, su brandos modeliu skirtu gerinti agiliuosius procesus.

Modeliai iš esmės nėra per daug skirtingi, nes yra paremti integraciniu testų brandos modeliu, kas ir leistų juos sudėti vienas į kitą. Nors ir probleminių sričių modelis, buvo sudarytas turint omenyje visą projektą, tačiau, agiliuosius projektus skaidomus iteracijomis, galima laikyti projektų rinkiniu, kur kiekvienam iš jų taikomas agiliųjų metodologijų brandos modelis, bei paraleliai galima taikyti probleminių sričių padengtumo testais brandos modelį.



3 pav. Agile TMMi modelio, bei mobiliojo testavimo problemonių sričių TMMi modelio apjungimas.

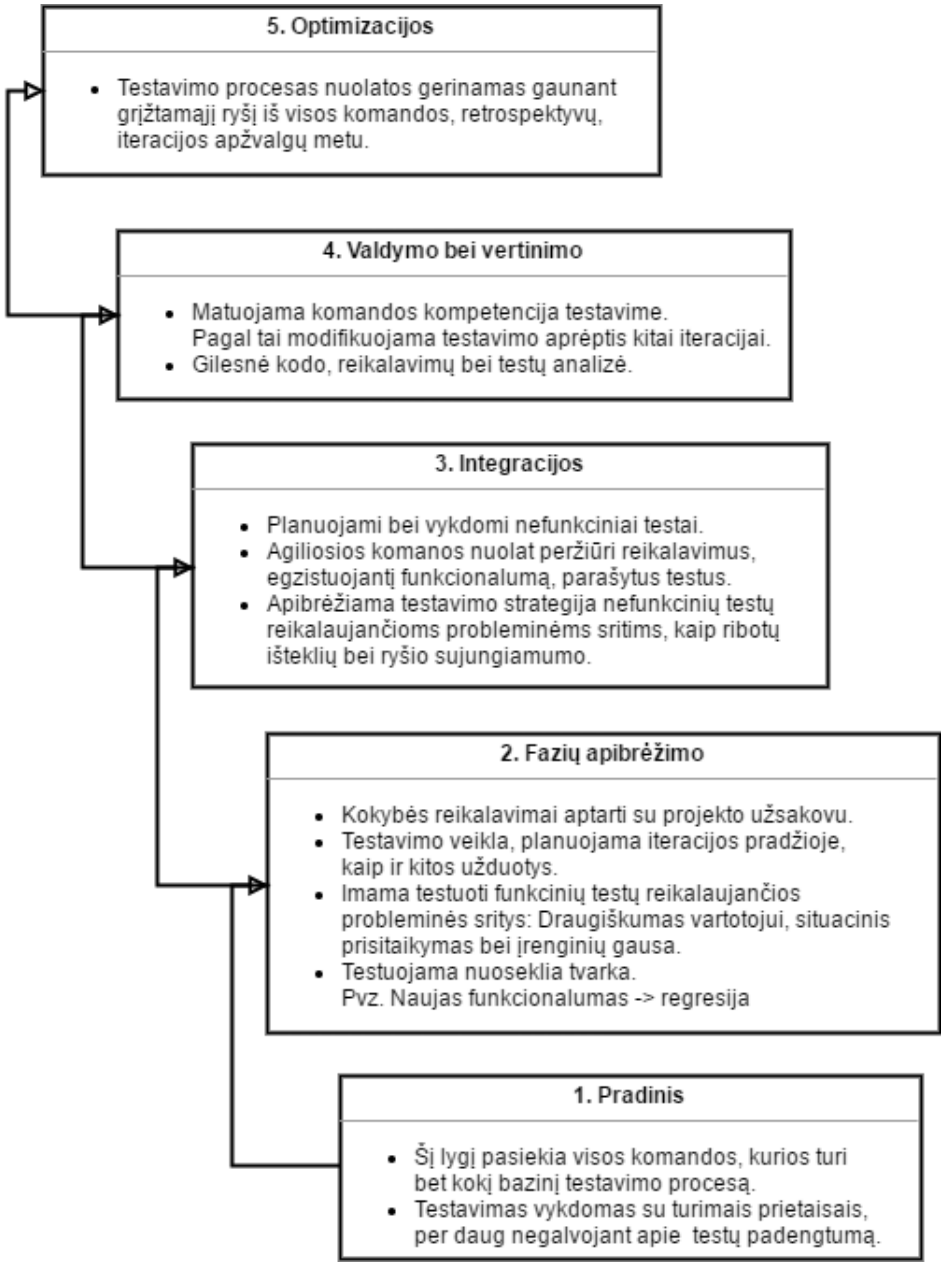
Rezultatai ir išvados

Šiame darbe darbe buvo išnagrinėtas mobiliosos programinė įrangos kūrimas remiantis mobiliojo įrenginio apibrėžimu, taip išskiriant esminius skirtumus nuo programinės įrangos stacionariesiems įrenginiams bei išskiriant problemines sritis. Didžiausias dėmesys darbe skiriamas išmaniųjų įrenginių, tokių kaip išmanieji telefonai bei planšetiniai kompiuteriai programinės įrangos kūrimo procesams, ieškant priežasčių, kodėl tokių programų kūrimo bei testavimo procesa daugelyje komandų gali būti neefektyvūs. Ištyrus tokį programinės įrangos kūrimo konteksta buvo ieškota programinės įrangos kūrėjų komandos efektyvumo bei darbo struktūrizavimo sprendimo remiantis integruotuoju testų brandos modeliu.

1. Apibrėžta konkreti mobiliojo įrenginio sąvoka, taip išskiriant keletą esminių išskirtinumo sričių, kuriomis nereikia rūpintis kuriant programinę įrangą staliniams kompiuteriams [Sat96].
2. Remiantis šiais išskirtinumo bruožais, bei keletu tyrimų apie mobiliųjų programų kūrimą buvo išskritos esminės probleminės mobiliųjų programų testavimo sritys į kurias reikėtų kreipti daugiausiai dėmesio norint programinės įrangos kūrimo porceso pabaigoje turėti kokybišką produktą [KK13; MDFE12]. Kiekvienai iš išskirtų sričių išskirti pagrindiniai testavimo objektai bei iššūkiai galintys kilti testavimo metu.
3. Nagrinėjama konkretus testavimpo proceso gerinimo sprendimas – TMMi, testų brandos integracinis modelis. Buvo plačiau išnagrinėtas kiekvienas iš modelyje apibrėžtų lygių, taip siekiant atrasti sąsają, tarp mobiliųjų programų kūrimo probleminių sričių, bei testavimo proceso struktūrizavimo metodų. Taip pat išnagrinėjamas CMM – gebėjimo brandos modelis, kuris buvo sukurtas dar prieš TMM, tačiau išsiaiškinta, jog CMM didelio dėmesio testavimui neskiria, todėl jis toliau darbe nenagrinėtas.
4. Probleminės mobiliųjų programų testavimo sritys buvo išskirtytos į aprėpiančias funkcinius bei nefunkcinius reikalavimus, taip problemines sritis adaptuojant TMMi brandos lygiams atitinkantiems funkcinius bei nefunkcinius testus. Pateiktas TMMi paremtas modelis skirtas mobiliųjų programų probleminių sričių testavimui.
5. Kuriant TMMi paremtą modelį probleminėms sritims iškilo dar viena problema – metodologijų nesuderinamumas. Daugelis mobiliųjų programų yra kuriamos agiliosiomis metodologijomis paremtu kūrimo procesu, tačiau TMMi glaudžiai paremtas V-modeliu. Buvo priimtas sprendimas vieną agiliąją programos kūrimo iteraciją laikyti atskiru projektu, paremtu V-modeliu. Ir kiekvienai agiliajai iteracijai būdingus bruožus pritaikyti jų V-modelio atitikmenims. Gautas iteracinisį testų brandos modelis, kuris testavimo procesą gerina kiekvienai iteracijai atskirai, bei visam projektui po kiekvienos iteracijos.
6. Išskirti du esminiai testavimo procesui gerinti skirti modeliai: vienas modelis konkrečioms probleminėms sritims, kitas, agiliojo proceso adaptacija į testų brandos modelį. Abu modeliai yra paremti integruotuoju testų brandos modeliu todėl tarpusavyje suderinami. Juos galima įgyvendinti tarpusavyje, paraleliai siekiant geresnio agiliojo proceso bei pilno probleminio sričių padengtumo testais.

Ištyrus mobiliųjų programų kūrimo sritį aiškiomis tampa keletą problemų: vartotojai tikisi dažnų programinės įrangos atnaujinimų, tačiau nori, jog programinė įrangą mobiliajame įrenginyje kainuotų dešimteriopai mažiau ar iš viso nemokamai, negu tokios programinės įrangos atitikmuo staliniame kompiuteryje. Tokie vartotojų lūkesčiai verčia mobiliųjų programų kūrėjus taupyti laiką mažinant su kodo rašyimu nesisijusių procesų sąskaitą arba renkantis dirbti pagal agiliasias metodologijas.

Darbe struktūrizuotas pagrindinių probleminių sričių testavimas, bei struktūrizuotas programinės įrangos kūrimas remiantis agiliosiomis metodologijomis viską pateikiant TMMi paremtame brandos lygių modelyje, kuris leis komandai struktūrizuotai gerinti testavimo, bei kitus procesus iteracija, po iteracijos.



4 pav. Galutinis TMMi paremtas modelis aprėpiantis pagrindines mobiliųjų programų kūrimo problemas.

Literatūra

[Ado] Adobe photoshop products. http://www.adobe.com/lt/creativecloud/photography.html, 2017.

[Agi] Agile manifestas. http://agilemanifesto.org/iso/lt/manifesto.html.

[And] Googlecode android open issues. https://source.android.com/source/report-bugs, 2017.

[Bur10] Ilene Burnstein. *Practical Software Testing: A Process-Oriented Approach*. Springer Publishing Company, Incorporated, 1st leid., 2010. ISBN: 1441928855, 9781441928856.

[Com] Statistics on user app usage. https://www.compuware.com, 2013.

[Int] Intel core i9-7980xe specifications. https://www.intel.com/content/www/us/en/products/processors/core/x-series/i9-7980xe.html, 2017.

[KK13] B. Kirubakaran ir V. Karthikeyani. Mobile application testing 2014; challenges and solution approach through automation. *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*, p. 79–84, 2013. DOI: 10.1109/ICPRIME.2013.6496451.

[MDFE12] Henry Muccini, Antonio Di Francesco ir Patrizio Esposito. Software testing of mobile applications: challenges and future research directions. *Proceedings of the 7th International Workshop on Automation of Software Test*, AST '12, p. 29–35, Zurich, Switzerland. IEEE Press, 2012. ISBN: 978-1-4673-1822-8. URL: http://dl.acm.org/citation.cfm?id=2663608.2663615.

[PCCW93] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis ir Charles V. Weber. Capability maturity model, version 1.1. *IEEE Softw.*, 10(4):18–27, 1993-07. ISSN: 0740-7459. DOI: 10.1109/52.219617. URL: http://dx.doi.org/10.1109/52.219617.

[Sam] Samsung galaxy s8+ specs. http://www.gsmarena.com/samsung_galaxy_s8+-8523.php, 2017.

[Sat96] M. Satyanarayanan. Fundamental challenges in mobile computing. *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '96, p. 1–7, Philadelphia, Pennsylvania, USA. ACM, 1996. ISBN: 0-89791-800-2. DOI: 10.1145/248052.248053. URL: http://doi.acm.org/10.1145/248052.248053.

[Sta] Number of smartphone users worldwide from 2014 to 2020 (in billions). https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, 2017.

[Tmm] Tmmi model. levels. https://www.tmmi.org/tmmi-model/, 2017.

Žodynas

Agiliosios metodologijos (angl. Agile) - Tokios programų sistemų kūrimo metodologijos, kai reikalavimai ir sprendimai keičiasi, proceso eigoje prisitaikant prie situacijos.

Atsako laikas (angl. Response Time) - Laiko vienetas per kurį prietaisas ar sistema sureaguoja į įvestį.

Dinaminis testavimas (angl. Dynamic Testing) - Tai tokia testavimo rūšis, kai paleidus kodą analizuojamas jo eglesys.

EDGE (angl. Enhanced Data rates for GSM Evolution) - Nauja GPRS tinklo iteracija, kuri teoriškai galėjo pasiekti 120Kbps-384Kbps interneto greičius.

GPRS (angl. General Packet Radio Services) - Paketais paremta bevielio komunikavimo paslauga, kuri teoriškai gali pasiekti 56Kbps-114Kbps atsiuntimo greičius ir nuolatinę prieigą prie interneto mobiliųjų įrenginių, bei kompiuterių naudotojams.

NFC (angl. Near Field Communication) - Bevielio duomenų perdavimo technologija, paremta mažo atstumo tarp įrenginių sujungiamumu, ir nereikalaujanti interneto.

PDA (angl. Personal Digital Assistant) - dar vadinami rankiniais kompiuteriais, tai mobilieji įrenginiai, skirti asmeninės informacijos tvarkymui.

Programinės įrangos kūrimo gyvavimo ciklas (angl. Developement lice cycle)- tai sąvoka apibrėžianti planavimo, programinės įrangos kūrimo, testavimo bei paleidimo procesus.

Statinis testavimas (angl. Static Testing) - Tai testavimo metodas, kurio metu nėra paleidžiamas programos kodas. Atliekamos kodo bei dokumentacijos peržiūros.

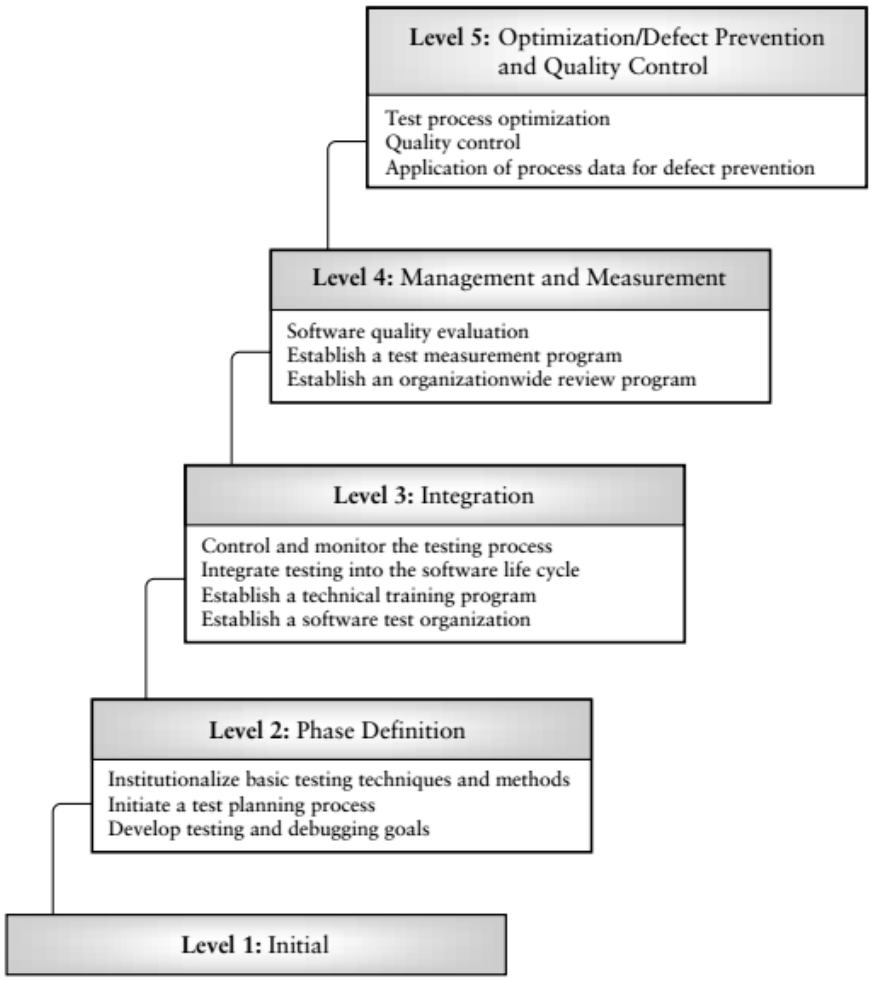
Tiriamasis testavimas (angl. Exploratory testing) - Nestruktūrizuotas testavimo metodas, kuomet testuotojas turi pilną laisvę testuoti remiantis tik savo patirtimi.

V-modelis - tai tokia programų kūrimo metodologija, galima vadinti krioklio išplėtimu, kur iš pradžių apibrėžiami visi reikalavimai projektui, vykdomas planavimas, tuomet programavimas, ir galiausiai projekto pabaigoje - testavimas. Kiekviename iš kūrimo žingsnių planuojant skirtingą testavimo sritį.

„Krioklio“ metodologija (angl. Waterfall) - Tai nuosekli (ne iteratyvi) programų kūrimo metodologija, kuomet programų kūrimo procesas „teka žemyn“, t.y. kiekviena sritis vykdoma paeiliui. Idėjos sukūrimas, projekto pradžia, analizė, planavimas, kūrimas, testavimas, produkto išleidimas ir galiausiai palaikymas.

Priedas nr. 1

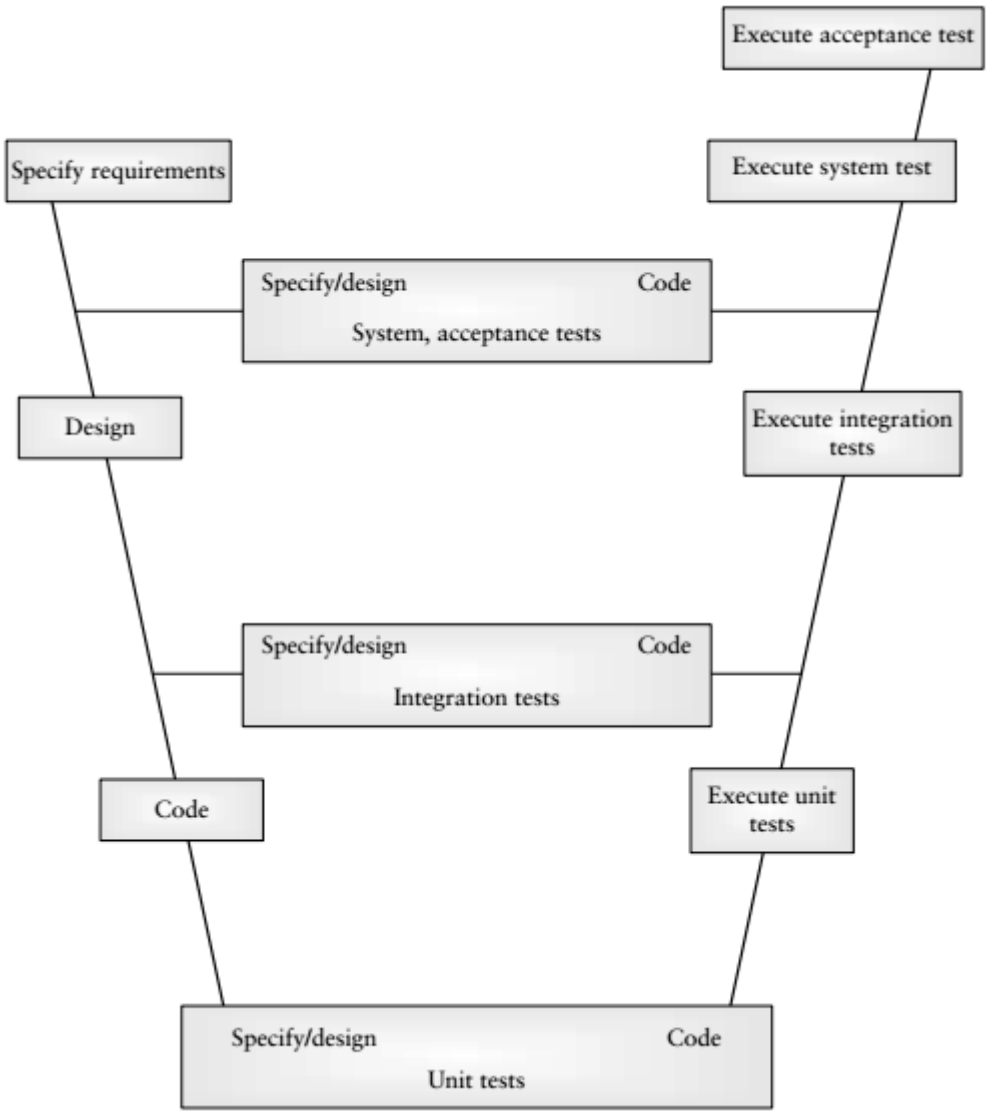
TMMi modelio struktūra



5 pav. TMMi modelio brandos lygiai [Bur10]

Priedas nr. 2

Programų sistemų kūrimo V-modeliu struktūra



6 pav. V-modelis [Bur10]