21-osios Lietuvos jaunųjų mokslininkų konferencijos „Mokslas – Lietuvos ateitis“ teminė konferencija
Proceedings of the 21st Conference for Junior Researchers „Science – Future of Lithuania“

VERSLAS XXI AMŽIUJE / BUSINESS IN XXI CENTURY

2018 m. vasario 8 d. Vilnius

eISSN 2029-7149
8 February, Vilnius, Lithuania Article Number: vvf.2017.008

Šiuolaikinio verslo aktualijos http://jmk.vvf.vgtu.lt
Contemporary Business

TESTAVIMO PROCESO PROBLEMATIKA INFORMACINIAME AMŽIUJE

Justina TOMKIENĖ, Jolanta SABAITYTĖ

Vilniaus Gedimino Technikos Universitetas, Verslo vadybos fakultetas, Verslo technologijų ir verslininkystės katedra, adresas: Saulėtekio al.11, 10223, Vilnius, Lietuva

Vilniaus Gedimino Technikos Universitetas, Verslo vadybos fakultetas, Verslo technologijų ir verslininkystės katedra, adresas: Saulėtekio al.11, 10223, Vilnius, Lietuva
El. paštai: justina.tomkiene@gmail.com; jolanta.sabaityte@vgtu.lt

Santrauka. Straipsnio tikslas nustatyti testavimo proceso valdymo problematiką. Išsikelti uždaviniai: apibrėžti testavimo proceso valdymo sampratą, pagrindinius etapus, identifikuoti ir išanalizuoti pagrindinius informacinių sistemų kūrimo modelius, nustatytant jų įtaką bei pagrindinius trūkumus. Apžvelgti pagrindiniai informacinių sistemų diegimo modeliai, išskiriant testavimo proceso vietą skirtinguose informacinių sistemų diegimo etapuose, leido nustatyti, kad testavimo proceso valdymo problematika turi glaudų sąryšį su vieta diegimo procese. Analizė papildyta testavimo proceso privalumais ir trūkumais, atsižvelgiant į diegimo proceso eigą. Apibrėžta testavimo proceso sąvoka, bei testavimo proceso pagrindiniai valdymo aspektai. Nagrinėjamas testavimo proceso valdymo modelis RATMN, apžvelgiama problematika naudojama žinių valdymo modelyje programinės įrangos testavimo procesui. Identifikuotos pagrindinės testavimo proceso valdymo problemos.

Reikšminiai žodžiai: testavimo procesas, valdymo modelis, informacinių sistemų kūrimo modelis, vadybos funkcijos.

Įvadas

Svarbiausias reikalavimas informacinėms sistemoms – kokybė. Tam kad ši kokybė būtų užtikrinta, kiekviena informacinė sistema prieš pasiekdama galutinį vartotoją turi būti sėkmingai ištestuota. Šio mokslinio tyrimo objektas yra testavimo procesas. Tai viena iš svarbiausių veiklų informacinių sistemų kūrimo ir tobulinimo procese, kadangi testavimas užtikrina jog kritinis sistemos funkcionalumas veiktų tinkamai. Didėjanti susidomėjimą testavimu rodo organizuojamos konferencijos, tokios kaip Lietuvoje jau kasmetine tampanti TestCon Vilnius konferencija (TestCon Vilnius 2017), kurioje dalyvauja testavimo srities specialistai iš Lietuvos ir užsienio šalių (Estijos, Švedijos, Norvegijos, Didžiosios Britanijos, Italijos ir t.t). Šiame renginyje didelis dėmesys skiriamas ir testavimo valdymui. Taip pat labai vertinami Tarptautinės programinės įrangos testavimo kvalifikacijos tarybos (angl. *International Software Testing Qualifications Board*) (ISTQB exam certification 2017) sertifikatai pripažinti visame pasaulyje: nuo 2017 metų birželio mėnesio ISTQB suvaldė daugiau kaip 740 000 egzaminų ir išdavė daugiau kaip 535 000 sertifikatų daugiau nei 120 šalių visame pasaulyje. Toks susidomėjimas įrodo susidomėjimą testavimo procesu ir testavimo valdymu kaip neatsiejama testavimo proceso dalimi. Sparčiai plečiantis informacinių sistemų sudėtingumui, testavimo apimtys tik didėja - to pasekoje sudėtinga suvaldyti didelės apimties testavimus. Ir nors informacinių sistemų kūrimo bei valdymo procesai plačiai nagrinėjami ir yra ne viena tam taikoma metodika, tačiau būtent testavimo proceso valdymas atskirai yra aptariamas gan siaurai ir nepakankamai. Straipsnio problema yra orentuota į tai, kokios yra pagrindinės testavimo proceso valdymo problemos. Atsakant į šį klausimą buvo pasiūlyta hipotezė: testavimo procesas bei testavimo proceso valdymas priklauso nuo informacinių sistemų diegimo modelio. Hipotezės patikimumui patikrinti buvo naudojami lyginimo ir analizės metodai.

Testavimo proceso sąvoka ir reikšmė IS diegime

Myers (Myers 2004) programinės įrangos testavimą apibrėžia kaip procesą ar procesų seriją, sukurtą užtikrinti jog kompiuterinis kodas darytų tai kam jis buvo sukurtas ir nedarytų nieko nenuspėjamo. Programinė įranga turėtų būti nuspėjama ir nuosekli, nepateikianti vartotojams jokių staigmenų. Taip pat Myers teigia, jog testavimas išlieka tarp *„tamsiųjų menų“* informacinių sistemų kūrime.

Jussi (Kasurinen 2012) savo straipsnyje teigia, kad galima išskirti tokias testavimo fazes: testavimo planavimas ir kontrole, testavimo analizė ir dizainas, testavimo įgyvendinimas ir vykdymas, rezultatų vertinimą ir ataskaitas, testavimo uždarymo veiklos.

Testavimo procesas turi būti nuoseklus, proceso veiklos priklausomai nuo taikomo programų sistemos gyvavimo ciklo ir testavimo modelio gali persidengti viena su kita, vykti lygiagrečiai ar netgi vykti pagal konkretaus projekto specifiką. Tačiau svarbiausia yra jog visas testavimo procesas būtų tinkamai suplanuotas: projekto plane esantys terminai būtų realūs, nepamiršta įtraukti jokių testavimo procesui įtakos galinčių turėti veiklų, įtraukiami tinkamą kompetenciją turintis komandos nariai. Siekiant užtikrinti testavimo sėkmę kone svarbiasias yra testavimo proceso valdymas.

Testavimo proceso valdymo samprata

Testavimo procesas sudarytas iš skirtingų veiklų, kurios padeda pagerinti produkto kokybę, yra vadinamas programinės įrangos testavimo ciklu (angl. *Software Testing Life Cycle*) (Software testing class 2017).

Standartinis programinės įrangos testavimo gyvavimo ciklas apima: reikalavimų analizę, testavimo planavimą, testavimo atvejų parengimą, testinių aplinkų parengimą, testavimo vykdymą, testavimo ciklo užbaigimą.

Visų šių testavimo gyvavimo ciklo veiklų planavimas, suvaldymas ir darnaus veikimo užtikrinimas ir yra vadinamas testavimo proceso valdymu. Testavimo proceso valdymas – viena kertinių projekto veiklų, be kurios sėkmingo vykdymo projektas gali būti pasmerktas nesėkmei. Tam kad taip nenutiktų šiai veiklai didelėse įmonėse dažnai numatoma ir atskira testavimo vadovo (angl. *Test manager*) pareigybė.

Kūriant vis sudetingesnes sistemas didėja ir testavimo apimtys, ko pasekoje suvaldyti didelius testavimus pasidaro sudėtinga. Efektyviai sukoordinuoti visą testavimo gyvavimo ciklą reikia patirties ir žinių. Atsižvelgiant į tai kokią problematiką mokslininkai įžvelgia ir siekiant užtikrinti jog testavimo valdymo procesas būtų kiek įmanoma sklandesnis yra kūriami testavimo valdymo modeliai.

Informacinių sistemų diegimo teoriniai aspektai

Testavimas yra neatskiriama informacinių sistemų kūrimo dalis, todėl pirmiausia apžvelgiami pagrindiniai programų sistemos gyvavimo ciklo modeliai (angl. *Software development life cycle models* (SDLC)), išskiriami dažniausiai aptariami jų privalumai ir trūkumai. Kiekviename modelyje ypatingas dėmesys skiriamas testavimui: išskiriama kokiuose skirtinguose programų sistemų gyvavimo ciklo modelių etapuose vykdomas testavimas ir pagrindinė testavimo proceso problematika būdinga kūriant informacines sistemas pagal kiekvieną iš šių modelių

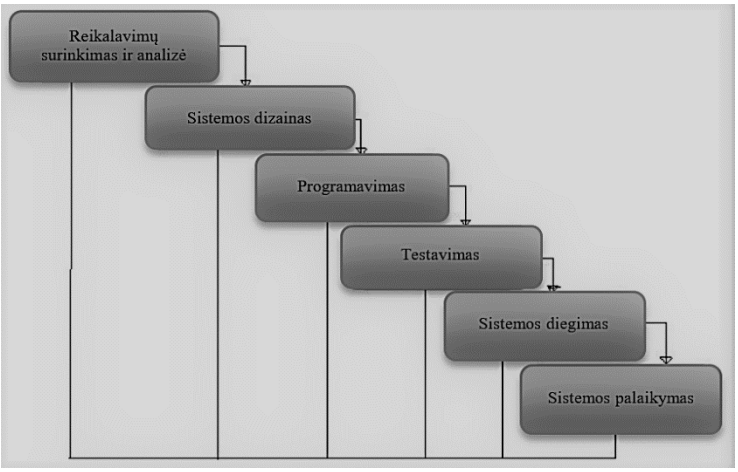
Krioklio modelis

Krioklio (angl. *Waterfall*) modelis yra vienas seniausių ir geriausiai žinomų informacinių sistemų diegimo modelių. Kartu tai yra vienas paprasčiausių modelių, pagal kurį visas projektas yra dalinamas į etapus, kur kievienas etapas seka vienas paskui kitą ir turi apibrėžtus darbus kurie turi būti padaryti tame etape. Taip pat kiekvienas šio modelio etapas yra visiškai priklausomas nuo prieš tai esančio, todėl suklydus kažkuriame iš pradinių etapų ir apie klaidingus sprendimus paaiškėjus vėlesniuose atapuose gali tekti grįžti į tą etapą kur buvo padaryta klaida.

Techbeacon, savo internetinėje svetainėje (Techbeacon 2017) pabrėžia jog taikant Krioklio modelį daroma prielaida kad visų suinteresuotų šalių reikalavimai juos pasirašant nepakis, nepaisant to jog programinė įranga bus pagaminta po metų ar net kelių. Tačiau dauguma projektų parodė jog tam kad kūrimo komandos gautų puikių rezultatų paprastai būna per daug kintančių dalių, įskaitant ir pačius reikalavimus.

Pasak S. Balaji (Balaji 2012) išskiriami tokie Krioklio modelio privalumai ir trūkumai. Privalumai: aiškūs reikalavimai dar prieš pradedant plėtrą; kiekvienas etapas pradedamas ir pabaigiamas nustatytu laiku, vėliau jis pereina į kitą etapą; kadangi tai linijinis modelis – jis yra lengvai įgyvendinamas; šis modelis reikalauja minimalių žmogiškųjų resursų, kadangi vienu metu vykdomas tik vienas etapas; kiekvieno etapo tinkamas dokumentavimas užtikrina tinkamą kitų fazių vykdymą. Trūkumai: atsiradus problemoms dažniausiai jos negali būti išsprendžiamos esamame etape; pasikeitus vartotojo reikalavimams jie dažniausiai nebegali būti įgyvendinti dabartiniame vystymosi procese.

Nepaisant trūkumų, daugelis šio modelio pranašumų užtikrina, kad jis vis dar yra vienas iš populiariausių modelių, naudojamų programinės įrangos kūrimo srityje. Jis tinkamas naudoti tuomet kai reikalavimai yra aiškūs ir nekis.



1 pav. Krioklio modelis (ISTQB exam certification 2017)
Fig. 1. Waterfall model (ISTQB exam certification 2017)

Testavimo apimtis yra pilnai apibrėžta, galima gan tiksliai suplanuoti pradžios ir pabaigos terminus. Taikant klasikinį krioklio modelį, testavimas pradedamas vykdyti tik po programavimo darbų testavimo etape. Todėl galima problema jog testavimo etapas gali užtrukti ilgiau, nei būtų tuo atveju jei jam būtų pradėta ruoštis nuo ankstesnių etapų. Taip pat nėra galimybės keisti reikalavimus ir pertestuoti, jei testavimo metu išaiškinamas naujas poreikis.

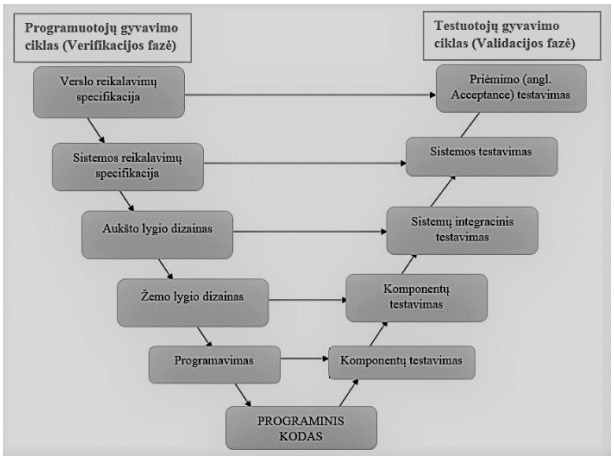
V modelis

V modelis (angl. *Validation and Verification model*) yra pakeistas Krioklio modelis. Ties kodavimo etapu V modelis pasisuka ašimi į viršų. Šiame modelyje programuotojas ir testuotojas dirba lygiagrečiai. V modelyje yra tiesioginis ryšys tarp kiekvieno programavimo etapo ir kiekvieno testavimo etapo.

Pasak S. Balaji (Balaji 2012) išskiriami tokie Krioklio modelio privalumai ir trūkumai.

Privalumai: paprastas ir lengvai panaudojamas; testuotojas įtraukiamas jau į reikalavimų rinkimo stadiją; testavimo planavimo darbai atliekami dar prieš programavimą, todėl taip sutaupoma laiko; reikalavimų pakeitimai galimi bet kuriame etape.

Trūkumai: V modelis mažiausiai lankstus; pusiaukelėje vykstant pakeitimams turi būti atnaujinti netik reikalavimų bet ir testavimo dokumentacija; reikalinga daugiau resursų, todėl dažniausiai taikoma didesnėse organizacijose.



2 pav. V modelis (ISTQB exam certification 2017)
Fig 2. Validation and Verification model) (ISTQB exam certification 2017)

Produkto testavimas planuojamas kartu su atitinkamu vystimosi etapu V modelyje. Kadangi testuotojai įtraukiami jau į reikalavimų rinkimo stadiją, o testavimo planavimo darbai atliekami dar prieš programavimą – sutaupoma daug laiko. Modelis labiau tinkamas naudoti mažesniuose ar vidutinio dydžio projektuose.

Spiralinis modelis

Spiralinis modelis jungia idėją apie pakartotinį vystimą su sistemingai kontroliuojamais krioklio modelio aspektais. Tai leidžia pakartotinai išleisti produktą ar papildomą tobulinimą per kiekvieną iteraciją aplink spiralę. Spiralinis modelis turi keturis etapus: planavimą, rizikos analizę, inžineriją ir vertinimą. Programinės įrangos projektai pakartotinai praeina per šiuos etapus iteracijomis, kurios šiame modelyje vadinamos spiralėmis. Pradinė spiralė prasideda nuo planavimo etapo. Kiekviena paskesnė spiralė kyla iš pagrindinės linijos spiralės. Reikalavimai renkami planavimo etapo metu. Rizikos analizės etape identifikuojamos rizikos ir nustatomi alternatyvūs sprendimai. Prototipas turimas pasibaigus rizikos analizės etapui. Programinė įranga gaminama inžinerijos etape, kartu su testavimo etapo pabaiga. Vertinimo etape klientas gali įvertinti projekto rezultatus prieš tęsiant projektą su nauja spirale. Spiraliniame modelyje kampinis komponentas žymi progresą, o spiralės spindulys – kainą. Indų mokslininkai (Mohammed *et al*. 2010) išskira tokius Spiralinio modelio privalumus ir trūkumus. Privalumai: didelis dėmesys skiriamas rizikos analizei; tinkamas dideliems ir kritinės misijos projektams; programinė įranga gaminama ankstyvame programinės įrangos gyvavimo cikle. Trūkumai: gali būti brangus; rizikos analizė reikalauja labai specifinių žinių; projekto analizė labai priklauso nuo rizikos analizės etapo; modelis netinkamas mažiems projektams. Testavimas yra pradedamas planavimo etape ir vykdomas inžinerijos etape. Modelis netinka mažiems projektams.

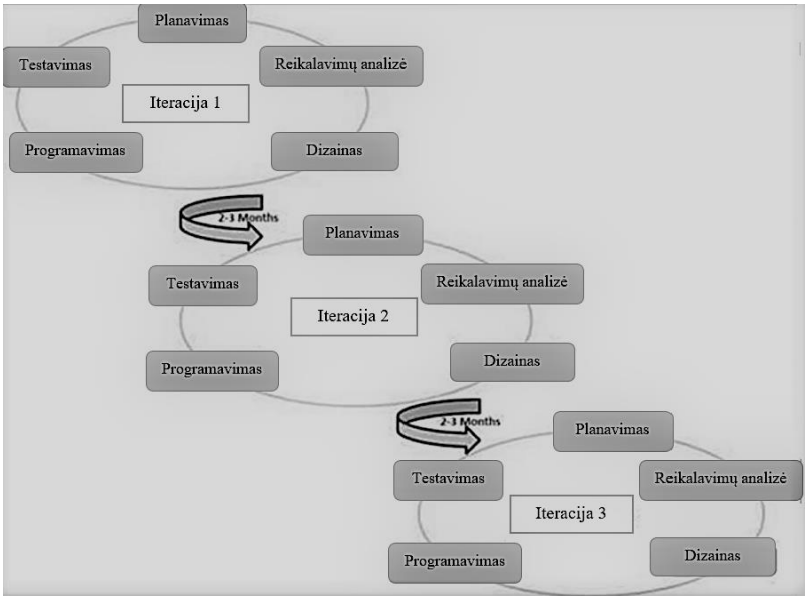
Agile metodologija

Agile metodologija vieną naujausių informacinių sistemų kūrimo procese. Ši metodologija reiškia greitą judėjimą. Agile metodologija turi prisitaikančią komandą, galinčią operatyviai reaguoti į kintančius reikalavimus. Nauji reikalavimai yra priimtini bet kuriame programinės įrangos kūrimo etape.

S. Balaji savo straipsnyje išskiria tokius Agile motodologijos privalumus ir trūkumus (Balaji 2012):

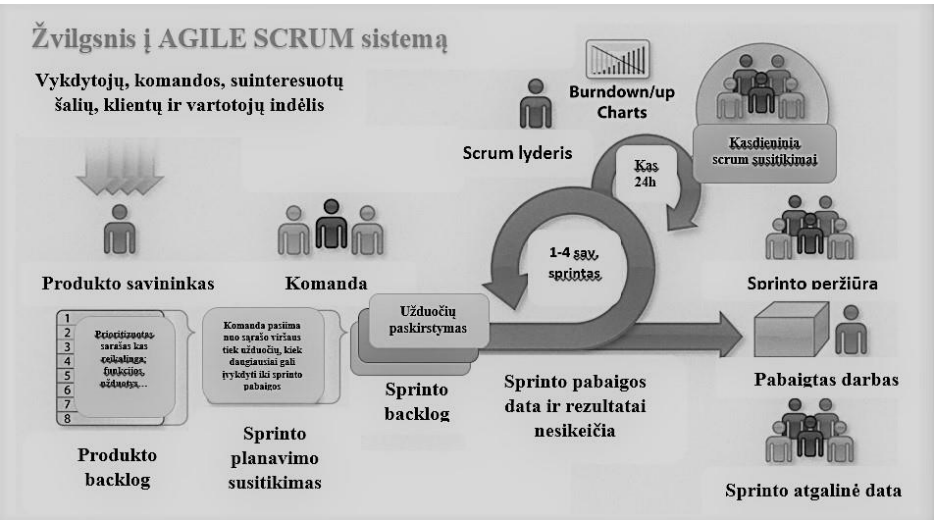
Privalumai: svarbiausias Agile modelio privalumas yra gebėjimas reaguoti į kintančius reikalavimus; nuolatinis bendradarbiavimas su užsakovu;

Trūkumai: dideliuose projektuose sunku įvertinti laiką, reikalingą projektui programinės įrangos kūrimo cikle; tik vyresni (labiau patyrę) specialistai turi galimybes priimti sprendimus, reikalingus tokio tipo greitam programinės įrangos kūrimui.



3 pav. Agile modelis (Tutorials point 2017)
Fig. 3. Agile model (Tutorials point 2017)

Svarbu paminėti jog Agile metodologijai priskiriama ir programinės įrangos kūrimo valdymo sistema, vadinama scrum. Šios sistemos pagrindu darbas organizuojamas komandomis. Sudaromas darbų sąrašas, prioritizuojamas. Kiekvieną dieną organizuojami trumpi apie 15 minučių susitikimai, kurių metu iš darbų sąrašo viršaus yra paimamas nustatytas darbų kiekis, šiems darbams numatomas terminas iki kada jie turi būti atlikti ir paskirstoma komandai. Toks numatytų darbų per numatytą laiko tarpą planavimas vadinamas sprintu. Sprintai paprastai organizuojami 1-4 savaičių terminui. Vėliau kasdieninių susitikimų metų sekamas progresas, perplanuojama jei dėl nenumatytų priežasčių keičiasi pabaigos terminai ar rezultatai.



4 pav. Agile scrum sistema (Agile for all 2017)
Fig. 4 Agile scrum system (Agile for all 2017)

Teisingai pastebi Kasurinen (Kasurinen 2012), jog esant sparčiam sistemos vystime Agile metodas leidžia daugiau laiko skirti testavimui, kadangi testavimo procesas gali būti pradėtas anksčiau nei taikant tradicinį Krioklio modelį.

Suburta kompetetinga komanda sprinto metu įsipareigoja atlikti prioritetinius darbus per trumpus terminus, o kasdieninių susitikimų metu sekamas progresas užtikrina darbų atlikimo tinkamą kontroliavimą (4 pav.).

Testavimo vieta skirtinguose IS diegimo modeliuose

Pagal atliktą programų sistemos gyvavimo ciklų apžvalgą nustatyta kokiuose programinės įrangos kūrimo etapuose vyksta testavimo veiklos. Pagal testavimo praktiką galima daryti išvadą jog labiausiai tikėtina kad testavimas bus sėkmingas tuo atveju, kai jo veiklos prasideda kuo ankstesniame IS diegimo etape. Šiame palyginime nebuvo įtrauktas spiralinis modelis, kadangi jame išskiriami kitokie etapai nei kituose informacinių sistemų gyvavimo ciklo modeliuose, o testavimo procesas yra vykdomas inžinerijos etape.

1 lentelė. Testavimo vieta skirtinguose IS diegimo modelių etapuose (sudaryta autorių remiantis: (Balaji 2012; Mohammed *et al.* 2010 ; Kasurinen 2012)
Table 1. Testing in different IS development life cycle models (created by authors according (Balaji 2012; Mohammed *et al.* 2010; Kasurinen 2012)

Programinės įrangos kūrimo etapai / Programų sistemų gyvavimo ciklo modelis	Reikalavimų rinkimas	Analizė	Dizaino	Programavimo/ kūrimo	Testavimas	Sistemos palaikymas
Krioklio					X	
V modelis	X	X	X		X	
Agile	X	X	X	X	X	

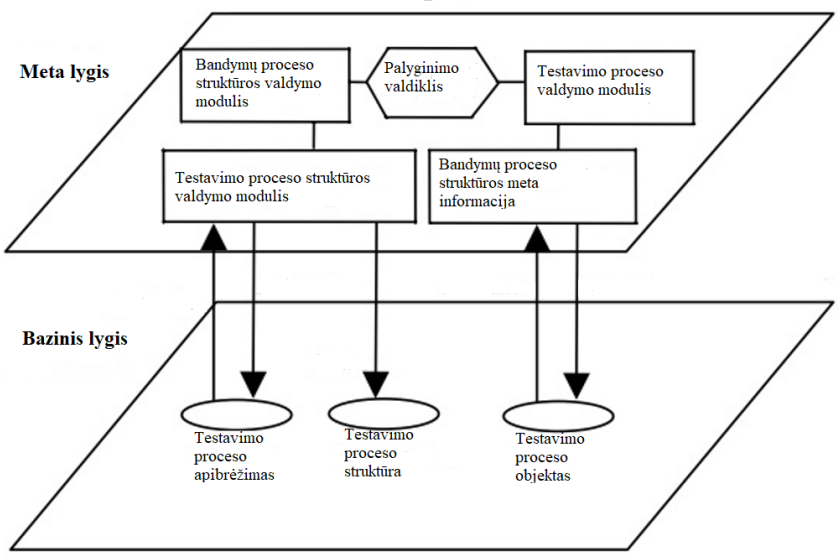
Pagal atliktą analizę galima išskirti Agile metodologiją, kuri užtikrina jog testavimo veiklos bus pradedamos projekto pradžioje ir esant pakeitimams jie galės būti įtraukti į projekto apimtį bet kuriame etape.

RATMM testavimo proceso valdymo modelis

Kiniečių mokslininkai (Jun-feng *et al.* 2006) teigia, jog žmonėms tampa vis svarbesni testavimo valdymo metodikos tyrimai, o programinės įrangos kūrimo procesas turėtų iš uždaro ir statinio virsti atviru ir dinamišku. Mokslininkai išskiria ir tai jog pagrindinė problema programinės įrangos testavimo valdymo tyrimuose yra kaip efektyviai valdyti visą programinės įrangos testavimo procesą. Ir siūlo Testavimo valdymo modelį RATMM (angl. *Reflective Architecture Based SoftwareTesting Management Model*). RATMM (4 paveikslas) yra logiškai suskaidytas į meta ir bazinį lygmenis. Šie lygmenys tarpusavyje yra susiję. Meta lygmuo fiksuoja būseną ir elgiasį baziniame lygmenyje,

o bazinis lygmuo atlieka skaičiavimus. Taip pat Meta-lygis atspindi meta informacijos pokyčius baziniame lygmenyje po skaičiavimo. Remiantis savo paties būklės aprašymu ir elgesiu RATMM gali kooreguoti testavimo procesą, gauti informaciją apie testavimo proceso struktūrą ir elgseną bei sukurti programinės įrangos testavimo valdymo modelį atspindinčią architektūrą. Modelis leidžia vartotojui keisti testavimo proceso sąlygas neatliekant programavimo darbų. RATMM egzistuoja visame testavimo proceso gyvavimo cikle, taip pat patikrina ir patvirtina visą sistemos procesą.

RATMM pagrįstas dviejų lygių architektūra: baziniame lygmenyje objektai kuriuos turime keisti yra testavimo proceso apibrėžimas ir testavimo proceso objektas. Meta lygmenyje pagrindiniai objektai turi būti visiškai suprantami, įskaitant palyginimo valdiklį, testavimo proceso struktūros valdymo modulį, bandymų proceso valdymo modulį, bandymų proceso struktūros meta informacija. Palyginimo valdiklis naudojamas valdyti programinės įrangos testavimo srautą, testavimo proceso struktūros valdymo modulis daugiausia naudojamas užbaigti kontroliuoti programinės įrangos testavimo proceso struktūrą.



5 pav. Testavimo valdymo modelis RATMN (Jun-feng *et al.* 2006)
Fig. 5 Reflective Architecture Based SoftwareTesting Management Model) (Jun-feng *et al.* 2006)

Apžvelgtas RATMN modelis yra realiai sunkiai pritaikomas praktikoje, kadangi tai galima pavadinti tik pirminiu prototipu, kuris reikalauja išsamesnio tyrimo ir įgyvendinimo, kad galėtų būti realiai išbandytas praktikoje. Tačiau RATMN kūrėjai įžvelgia vieną iš pagrindinių testavimo proceso problemų, t.y. bando atsakyti į klausimą kaip efektyviai suvaldyti programinės įrangos testavimo procesą.

Žinių valdymo modelis programinės įrangos testavimo procesui

Mokslininkai (Xuemei *et al.* 2008) siūlo ir žinių valdymo modelį programinės įrangos testavimo procesui. Jie teigia, jog efektyvus žinių valdymas testavimo procese yra raktas kaip pagerinti informacinių sistemų testavimo kokybę.

Autorius analizavo pagrindinius programinės įrangos testavimo procesus su pagrindiniais žinių valdymo principais ir išskyrė penkias pagrindines testavimo proceso problemas: žemas programinės įrangos testavimo žinių pernaudojimas - nėra kaupiamas žinių bagažas; sudėtinga perduoti žinių bagažą mažiau patyrusiems darbuotojams; nėra plačiai paplitusi aplinka, kurioje būtų galima dalintis programinės įrangos testavimo žiniomis; didelis programinės įrangos testavimo žinių praradimas, kuomet didelė patirtis bei įgūdžiai yra sukoncentruoti tik pas kelis specialistus ir netampa viešai prieinami visiems; žinių valdymas yra integracija tarp žmonių, procesų ir technologijų, kur svarbiausias yra žmogus; neįmanoma optimaliai greitai paskirstyti žmogiškųjų išteklių.

Esminis žinių valdymo objektas yra skatinti žinių pernaudojamumą, o ne tik žinių kaupimą ar laikymą. Šiuo tikslu turi būti užtikrintas žinių atpažinimas, kurio pagrindinė funkcija yra pasiūlyti darbuotojams informaciją reikalingą sprendimų palaikymui jų veikloje.

Kitos testavimo proceso problemos

Pasibaigus projektui įprasta praktika yra peržvelgti gerąsias ir blogąsias praktikas, tai taip pat leidžia išsigryninti kokios buvo gerosios projekto praktikos ir taip pat pastebėti su kokiomis problemomis susidurta projekto metu. Konsultantų komanda „Guru99“ (žr. Konsultantų įmonė Guru99, 2017) išskiria tokias dažniausiai pasitaikančias testavimo proceso problemas projektuose: testavimas trunka ilgiau nei numatyta; žema testavimo kokybė; blogai suvaldyta projekto testavimo komanda. Tuo tarpu IBM specialistas Chip Davis (žr. IBM 2017) be minėtų problemų, taip pat išskiria tas pačias ir dar daugiau problemų. Nepakankamai laiko testavimui: išskyrus tam tikras specializuotas arba ypatingai misijai svarbias problemas, labai nedaug programinės įrangos projektų pakankamai laiko gyvavimo cikle skiria vystymui, kad būtų pasiektas aukštas kokybės lygis. Labai dažnai projekto uždelsimo priežastimi nurodomas trumpas testavimo ciklas. Net geriausiuose projektuose per labai trumpa laiką gali nepavykti visko ištestuoti. Laiko trūkumas įtakoja testavimo valdymą – to pasekoje nuolat keičiasi prioritetai ir perkeliamos užduotys, sumažėja testavimo apimtys ir taip nukenčia kokybė. Nepakankamai resursų testavimui: papildomai prie laiko stygiaus dar prisideda ir tai, jog sunku rasti tinkamų specialistų testavimams atlikti. Tie patys žmonės gali būti dedikuoti kitiems projektams ar užduotims atlikti. Ir nors projekto eigoje gali būti ir kitų problemų, tokiu kaip pavyzdžiui neparengta techninė įranga, tačiau žmogiškųjų išteklių trūkumas gali būti žymiai sudėtingiau išsprendžiamas. Šios problemos įtaka testavimui panaši kaip ir laiko trūkumo atveju. Testavimo komanda ne visuomet yra vienoje vietoje: šiame technologijų amžiuje tapo įprasta taupant lėšas ieškoti specialistų svetur, tačiau dažnai tai sukelia daug techninių kliūčių. Taip pat dalijimasis projekto medžiaga, įvairių klausimų sprendimas tampa dideliu iššūkiu kuomet komandos nariai yra tarkim skirtinguose žemynuose. Nekokybiški reikalavimai: tam kad galima būtų parengti tinkamus ir ištestuojamus testavimo atvejus, visų pirma turi būti patvirtinti išsamūs, nedviprasmiški ir testuotini reikalavimai. Taip pat tam kad testavimo valdymas būtų efektyvus – turi būti užtikrinta galimybė naudotis besikeičiančiais sistemos ir verslo reikalavimais. Svarbi ne tik reikalavimų formuluotė, tačiau ir jų statusas bei prioritetas ar kiti požymiai. Be to tai skatina dar labiau bendradarbiauti ir palaikyti ryšius tarp komandų rengiančių reikalavimus ir atliekančių testavimus. Siekiant užtikrinti kokybę toks bendradarbiavimas turi būti abipusis.

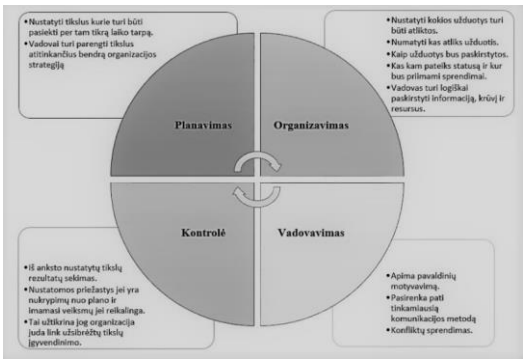
Valdymo funkcijų samprata

Priklausomai nuo to pagal kokį informacinių sistemų diegimo ciklą yra vykdomas sistemos diegimo projektas ir kaip organizuojamas testavimo procesas, galima pastebėti, jog vadybos funkcijos gali būti vykdomos viena iteracija, arba kartotis iki projekto testavimo proceso pabaigos net iki keliasdešim iteracijų. Nepaisant to pagal kokį informacinės sistemos kūrimo modelį vykdomas testavimo procesas ir kiek iteracijų atliekama – kiekviena vadybos funkcija yra labai svarbi ir padeda testavimo procesui būti sėkmingai atliktam ir užtikrinti programinės įrangos kokybę.

Valdymo procesas apibrėžiamas keturiomis valdymo funkcijomis (Stripeikis 2017), kurias atlieka kiekvienas vadovas, nepriklausomai nuo to kokioje sritije dirba. Tai yra: planavimas, organizavimas, vadovavimas ir kontrolė.

Planavimo etape nustatomi tikslai, kurie turi būti pasiekti per tam tikrą laiko tarpą, o vadovai parengia tikslus atitinkančius bendrą organizacijos strategiją. Organizavimo funkcija yra: nustatyti kokios užduotys turi būti atliktos, numatyti kas jas atliks ir kaip jos bus paskirstytos, taip pat kas kam pateiks statusą ir kur bus priimami sprendimai. Taip pat vadovas turi logiškai paskirstyti informaciją, krūvį bei resursus. Vadovavimas apima pavaldinių motyvavimą, vadovas turi pasirinkti patį tinkamiausią komunikacijos metodą ir taip pat yra atsakingas už konfliktinių situacijų sprendimą. Kontrolės etape sekama kaip yra vykdomi iš anksto nustatyti tikslai, nustatomos priežastys jei yra nukrypimų nuo plano ir imamasi veiksmų jei reikalinga. Kontrolės funkcija užtikrina jog organizacija arba projektas juda link užsibrėžtų tikslų įgyvendinimo.

Visos keturios valdymo funkcijos reikalingos ir norint užtikrinti sklandų testavimo proceso valdymą.



6 pav. Valdymo funkcijos (sudaryta autorių remiantis (Business Consi 2016)
Fig.6 Management functions (created by authors according (Business Consi 2016)

Išvados ir tolesnės testavimo proceso valdymo tyrimų kryptys

Atlikus literatūros apžvalgą nustatyta pagrindinė testavimo proceso problematika. Prieita išvada, jog testavimo proceso sėkmė priklauso nuo testavimo valdymo proceso, o problemos gali atsirasti pradedant netinkamai pasirinktu informacinės sistemos diegimo ciklu ir baigiant žmogiškųjų išteklių stoka.

Atlikta pagrindinių informacinių sistemų kūrio modelių analizė leido nustatyti, jog palankiausia testavimo proceso valdymui yra Agile metodologija, kuri: leidžia testavimo veiklas pradėti pačiuose pirminiuose sistemos diegimo etapuose; yra lanksti pakeitimams, kuriuos galime įtraukti bet kuriame vėlesniame sistemos diegimo etape; didelį dėmesį skiria valdymo ir kontrolės funkcijoms – scrum susitikimų metų organizuojant darbus.

Nustatyta jog viena iš kertinių kokybiško produkto kūrimo proceso ašių yra efektyvus testavimo proceso valdymo gyvavimo ciklo sukoordinavimas.

Atlikta testavimo valdymo modelių analizė, leido nustatyti kad išanalizuotas RATMM testavimo proceso valdymo modelio pagrindinė problematika programinės įrangos testavimo valdymo tyrimuose yra kaip efektyviai valdyti visą programinės įrangos testavimo procesą. Tuo tarpu išskirta žinių pernaudojamumo problematika iš žinių valdymo modelio akcentuoja tik žinių pernaudojamumo problemą. Abu modeliai identifikuoja svarbias testavimo valdymui problemas, tačiau nėra bendro testavimo valdymo modelio kuris apimtų visą problematiką. Taip pat literatūros analizė leido identifikuoti tokias problemas kaip: laiko trūkūmas, žema testavimo kokybė, žmogiškųjų resursų stoka, testavimo komandos lokacija ne vienoje vietoje bei nekokybiški reikalavimai.

Išskirta žinių problematika iš žinių valdymo modelio, akcentuojanti jog turi būti kaupiamas žinių bagažas ateičiai. Taip pat iš literatūros analizės identifikuotos problemos yra: laiko trūkūmas, žema testavimo kokybė, blogai suvaldyta projekto testavimo komanda, žmogiškųjų resursų stoka, testavimo komandos lokacija ne vienoje vietoje bei nekokybiški reikalavimai.

Atlikta analizė leido nustatyti, kad mokslas skiria nepakankamą dėmesį testavimo proceso valdymui ir šiuo metu stokojama įrankio, kuris leis užtikrinti maksimaliai kokybišką testavimo procesą. Nustatyta, kad toks įrankis turėtų apimti visas valdymo funkcijas, atsižvelgti į identifikuotas problemas ir atverti galimybes siekiant kuo efektyvesnių rezultatų testavimo proceso valdyme.

Literatūra

Agile for all. 2017. [Žiūrėta: 2017-10-10] Prieiga per internetą: http://agileforall.com/resources/introduction-to-agile/

Anon 2013. Mii_Dis_2013_Serikoviene.Pdf.

Balaji, S.; Murugaiyan, M. S. 2012. Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management* 2(1); 26-30.

Business Consi. 2016. A subsidiary guide to study with business [Žiūrėta: 2017-12-14] http://bconsi.blogspot.lt/2016/03/7-functions-of-management.html

IBM 2006. Test management best practices. [Žiūrėta: 2018-01-05]. Prieiga per internetą: https://www.ibm.com/developerworks/rational/library/06/1107_davis/

ISTQB exam certification 2017 [Žiūrėta: 2017-10-09]. Prieiga per internetą: http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/

Jun-feng, Y.; Shi, Y.; Ju-bo, L.; Dan X.; Xiang-yang J. 2006. Reflective Architecture Based Software Testing Management Model, *2006 IEEE International Conference on Management of Innovation and Technology* 2: 821–825. DOI: 10.1109/ICMIT.2006.262335 .

Kasurinen, J. 2012. Software Organizations and Test Process, *Development Advances in computers* 85: 1-63 DOI: 10.1016/B978-0-12-396526-4.00001-1 .

Konsultantų įmonė Guru99. [Žiūrėta: 2017-12-20]. Prieiga per internetą: https://www.guru99.com/a-complete-guide-to-test-process-improvement.htmlc

Mohammed, N.; Munassar, A.; Govardhan, A. 2010. A Comparison Between Five Models Of Software Engineering, *International Journal of Computer Science* 7(5): 94–101. DOI: 10.1.1.403.3201 .

Myers, G. 2004. *The Art of Software Testing, Second edition*. DOI: 10.1002/stvr.322 .

Software Testing Class. [Žiūrėta: 2017-10-11]. Prieiga per internetą: http://www.softwaretestingclass.com/software-testing-life-cycle-stlc/

Stripeikis, O.; Verslo valdymas VDU [Žiūrėta: 2018-01-05] Prieiga per internetą: https://fcis.vdu.lt/~o.stripeikis@evf.vdu.lt/FOV1-00094DBE/FOV1-00095038/9%20Tema%20-%20Verslo%20valdymas.pdf

TatvaSoft Software Development Company. 2000-2017 [Žiūrėta: 2017-10-09]. Prieiga per internetą: https://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/

TechBeacon. A digital hub for dev and tech professionals. [Žiūrėta: 2017-10-10]. Prieiga per internetą: https://techbeacon.com/waterfall-birth-agile-what-your-manager-needs-know.

TestCon Vilnius 2017 testavimo konferencija. [Žiūrėta: 2018-01-22]. Prieiga per internetą: http://www.testcon.lt/#conference-day

Tutorials point. [Žiūrėta: 2017-10-11]. Prieiga per internetą: https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm

Xuemei, L.; Guochang, G.; Yongpo, L.; Ji, W. 2008. Research and Application of Knowledge Management Model Oriented Software Testing Process : 1–8

TESTING PROCESS PROBLEMS IN INFORMATIVE AGE

Justina TOMKIENĖ, Jolanta SABAITYTĖ

Summary

Test management process is considered narrowly and insufficiently. The aim of the article is to identify the problems of test management process. The main goals is to identify main stages and to define the concept of test management process, to identify and analyse the main models of information systems development, determining their influence and main disadvantages. An overview of the main models of implementation of information systems is done in this article, distinguishing the location of the testing process in different stages of the implementation of information systems, has made it possible to determine that the problem of the test management process has a close relationship with the place of the installation process. The analysis is complemented by the advantages and disadvantages of the testing process, taking into account the progress of the implementation process. The concept of the testing process and the main aspects of test management process are defined. Two test management models considered: the RATMN and the knowledge management model for the software testing process. From the reviewed models and the literature analysis, the main problems of the testing process management were identified. Also the core management functions are reviewed as each model of test management process management. Methods of comparison and analysis are used in this article.

Keywords: testing process, management model, software development life cycle models and management functions.